japi

Reference Manual

Merten Joost

Inhaltsverzeichnis

I Reference **5**

1 Components **7**

Button . . . 7
Borderpanel . . . 10
Canvas . . . 15
Checkbox . . . 20
Checkmenuitem . . . 23
Choice . . . 24
Dialog . . . 27
Focuslistener . . . 33
Frame . . . 34
Helpmenu . . . 40
Hscrollbar . . . 41
Graphicbutton . . . 44
Graphiclabel . . . 47
Image . . . 50
Keylistener . . . 53
Label . . . 54
Led . . . 57
List . . . 60
Menu . . . 64
Menuitem . . . 66
Meter . . . 67
Mouselistener . . . 70
Panel . . . 71
Popupmenu . . . 76
Printer . . . 78
Progressbar . . . 80
Radiobutton . . . 83
Sevensegment . . . 86
Scrollpane . . . 89
Textarea . . . 92
Textfield . . . 96
Vscrollbar . . . 100
Window . . . 103

2 Functions 109
additem . . . 109
add . . . 110
alertbox . . . 111
appendtext . . . 112
beep . . . 113
borderpanel . . . 114
button . . . 115
canvas . . . 116
checkbox . . . 117
checkmenuitem . . . 118
choicebox2 . . . 119
choicebox3 . . . 120
choice . . . 121
cliprect . . . 122
componentlistener . . . 123
connect . . . 124
delete . . . 125
deselect . . . 126
dialog . . . 127
disable . . . 128
dispose . . . 129
drawarc . . . 130
drawcircle . . . 131
drawimagesource . . . 132
drawimage . . . 133
drawline . . . 134
drawoval . . . 135
drawpixel . . . 136
drawpolygon . . . 137
drawpolyline . . . 138
drawrect . . . 139
drawroundrect . . . 140
drawscaleddimage . . . 141
drawstring . . . 142
enable . . . 143
filedialog . . . 144
fileselect . . . 145
fillarc . . . 146
fillcircle . . . 147
filloval . . . 148
fillpolygon . . . 149
fillrect . . . 150
fillroundrect . . . 151
focuslistener . . . 152
frame . . . 153
getaction . . . 154

getcolumns . . . 155
getcurpos . . . 156
getdanger . . . 157
getfontascent . . . 158
getfontheight . . . 159
getheight . . . 160
getimagesource . . . 161
getimage . . . 162
getinheight . . . 163
getinsets . . . 164
getinwidth . . . 165
getitemcount . . . 166
getitem . . . 167
getkeychar . . . 168
getkeycode . . . 169
getlayoutid . . . 170
getlength . . . 171
getmousebutton . . . 172
getmousex . . . 173
getmousey . . . 174
getparentid . . . 175
getparent . . . 176
getrows . . . 177
getscaledimage . . . 178
getscreenheight . . . 179
getscreenwidth . . . 180
getselect . . . 181
getselend . . . 182
getselstart . . . 183
getseltext . . . 184
getstate . . . 185
getstringwidth . . . 186
gettext . . . 187
getvalue . . . 188
getviewportheight . . . 189
getviewportwidth . . . 190
getwidth . . . 191
getxpos . . . 192
getypos . . . 193
graphicbutton . . . 194
graphiclabel . . . 195
hasfocus . . . 196
helpmenu . . . 197
hide . . . 198
hscrollbar . . . 199
image . . . 200
insert . . . 201

inserttext 202
isparent 203
isresizable 204
isselect 205
isvisible 206
keylistener 207
label 208
led 209
line 210
list 211
loadimage 212
menubar 213
menuitem 214
menu 215
messagebox 216
meter 217
mouselistener 218
multiplemode 219
nextaction 220
pack 221
panel 222
popupmenu 223
printer 224
print 225
progressbar 226
quit 227
radiobutton 228
radiogroup 229
random 230
releaseall 231
release 232
removeall 233
removeitem 234
remove 235
replacetext 236
saveimage 237
scrollpane 238
selectall 239
select 240
selecttext 241
seperator 242
setalign 243
setblockinc 244
setborderlayout 245
setborderpos 246
setcolorbg 247
setcolor 248

setcolumns . . . 249
setcurpos . . . 250
setcursor . . . 251
setdanger . . . 252
setdebug . . . 253
setechochar . . . 254
seteditable . . . 255
setfixlayout . . . 256
setflowfill . . . 257
setflowlayout . . . 258
setfocus . . . 259
setfontname . . . 260
setfontsize . . . 261
setfontstyle . . . 262
setfont . . . 263
setgridlayout . . . 264
sethgap . . . 265
seticon . . . 266
setimage . . . 267
setinsets . . . 268
setmax . . . 269
setmin . . . 270
setnamedcolorbg . . . 271
setnamedcolor . . . 272
setnolayout . . . 273
start . . . 274
setpos . . . 275
setradiogroup . . . 276
setresizable . . . 277
setrows . . . 278
setshortcut . . . 279
setsize . . . 280
setslidesize . . . 281
setstate . . . 282
settext . . . 283
setunitinc . . . 284
setvalue . . . 285
setvgap . . . 286
setxor . . . 287
sevensegment . . . 288
showpopup . . . 289
show . . . 290
sleep . . . 291
start . . . 292
sync . . . 293
textarea . . . 294
textfield . . . 295

translate . . . 296
vscrollbar . . . 297
windowlistener . . . 298
window . . . 299

Teil I

Reference

Kapitel 1

Components

Button

j_button	*int j_button (int obj , char* label);* Creates a new button component with the specified **label** and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds button **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to button **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables button **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the button **obj**.
j_enable	*void j_enable (int obj);* enables the button **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to button **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of button **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of button **obj**.
j_getheight	*int j_getheight (int obj);*

Returns the height of button **obj**.

j_getlength *int j_getlength (int obj);*
Returns the length of button 's label or text.

j_getparentid *int j_getparentid (int obj);*
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getparent *int j_getparent (int obj);*
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getstringwidth *int j_getstringwidth (int obj , char* str);*
Returns the length of **str** of the actual font of button **obj**.

j_gettext *char* j_gettext (int obj , char* str);*
returns the button 's text or label.

j_getwidth *int j_getwidth (int obj);*
Returns the width of button **obj**.

j_getxpos *int j_getxpos (int obj);*
Returns the current horizontal position of button **obj** in its parent's coordinate space.

j_getypos *int j_getypos (int obj);*
Returns the current vertical position of button **obj** in its parent's coordinate space.

j_hide *void j_hide (int obj);*
Hides the button **obj**.

j_isparent *int j_isparent (int obj , int cont);*
Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.

j_isvisible *int j_isvisible (int obj);*
Returns J_TRUE if **obj** is visible, J_FALSE otherwise.

j_keylistener *int j_keylistener (int obj);*
Adds a new key listener to button **obj**, and returns its event number.

j_mouselistener *int j_mouselistener (int obj , int kind);*
Adds a new mouse listener to button **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

j_popupmenu *int j_popupmenu (int obj , char* label);*
Creates a new popupmenu with the specified **label** and returns its event number.

j_print *void j_print (int obj);*
prints the button .

j_release	*void j_release (int obj);* Releases button **obj** from its parent component (container).
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves button **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the button 's **obj** cursor to the specified **cursor**.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to button **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the button **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes button **obj** to specified **width** and **height**.
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the button **obj** to **str**.
j_show	*void j_show (int obj);* Shows the button **obj**.

Borderpanel

j_borderpanel *int j_borderpanel (int obj , int type);*
Creates a new borderpanel component with the style **type** and returns its event number.

j_add *void j_add (int obj , int cont);*
Adds borderpanel **obj** to container **cont**

j_borderpanel *int j_borderpanel (int obj , int type);*
Creates a new borderpanel component with the style **type** and returns its event number.

j_button *int j_button (int obj , char* label);*
Creates a new button component with the specified **label** and returns its event number.

j_canvas *int j_canvas (int obj , int width , int height);*
Creates a new canvas component with the given **width** and **height** and returns its event number. A canvas can be used for general drawing functions. A canvas generates an event, if its size changes. On error -1 will be returned.

j_checkbox *int j_checkbox (int obj , char* label);*
Creates a new checkbox component with the specified **label** and returns its event number.

j_choice *int j_choice (int obj);*
Creates a new choice component and returns its event number.

j_componentlistener *int j_componentlistener (int obj , int kind);*
Adds a new componentlistener to borderpanel **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

j_disable *void j_disable (int obj);*
Disables borderpanel **obj** so that it is unresponsive to user interactions

j_dispose *void j_dispose (int obj);*
Releases the resources of the borderpanel **obj**.

j_enable *void j_enable (int obj);*
enables the borderpanel **obj**.

j_focuslistener *int j_focuslistener (int obj);*
Adds a new focus listener to borderpanel **obj**, and returns its event number.

j_getfontascent *int j_getfontascent (int obj);*
Returns the ascent (space above the baseline) of the actual font of borderpanel **obj**.

j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of borderpanel **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of borderpanel **obj**.
j_getinheight	*int j_getinheight (int cont);* Returns the height of the client size.
j_getinsets	*int j_getinsets (int obj , int side);* Returns the width of the specified inset.
j_getinwidth	*int j_getinwidth (int cont);* Returns the width of the client size.
j_getlayoutid	*int j_getlayoutid (int obj);* Returns the event number of the layoutmanager for containers **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of borderpanel **obj**.
j_getwidth	*int j_getwidth (int obj);* Returns the width of borderpanel **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of borderpanel **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of borderpanel **obj** in its parent's coordinate space.
j_graphicbutton	*int j_graphicbutton (int obj , char* filename);* Creates a new graphicbutton component with the image loaded from **filename** and returns its event number.
j_graphiclabel	*int j_graphiclabel (int obj , char* str);* Creates a new graphiclabel component with the image loaded from **filename** and returns its event number.
j_hide	*void j_hide (int obj);* Hides the borderpanel **obj**.
j_hscrollbar	*int j_hscrollbar (int obj);* Creates a new horizontal scrollbar and returns its event number.

j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to borderpanel **obj**, and returns its event number.
j_label	*int j_label (int obj , char* label);* Creates a new label component with the specified **label** and returns its event number.
j_led	*int j_led (int obj , int style , int color);* Creates a new led component with the specified **style** and the specified color **color**.
j_line	*int j_line (int obj , int orient , int style , int length);* Creates a new line component with the specified **length** and returns its event number.
j_list	*int j_list (int obj , int rows);* Creates a new list component with the specified number of **rows** and returns its event number.
j_meter	*int j_meter (int obj , char* title);* Creates a new pointer–intrument with the specified label **titel**.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to borderpanel **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_pack	*void j_pack (int obj);* Resizes borderpanel to the minimal size of contained components.
j_panel	*int j_panel (int obj);* Creates a new panel component and returns its event number.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the borderpanel .
j_progressbar	*int j_progressbar (int obj , int orient);* Creates a new progressbar with the specified **orient**ation.
j_radiogroup	*int j_radiogroup (int obj);* Creates a new radiogroup and returns its event number.
j_releaseall	*void j_releaseall (int obj);* Releases all components from borderpanel **obj**.

j_release	*void j_release (int obj);* Releases borderpanel **obj** from its parent component (container).
j_scrollpane	*int j_scrollpane (int obj);* Creates a new scrollpane component and returns its event number.
j_setalign	*void j_setalign (int obj , int align);* Sets the alignment in borderpanel **obj** to **align**. Needs a flowlayout Manager.
j_setborderlayout	*void j_setborderlayout (int obj);* Adds a borderlayout manager to borderpanel **obj**.
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves borderpanel **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the borderpanel 's **obj** cursor to the specified **cursor**.
j_setfixlayout	*void j_setfixlayout (int obj);* Adds a fixlayout manager to borderpanel **obj** (default layout manager).
j_setflowfill	*void j_setflowfill (int obj , int bool);* Resizes all containing component to the height (width) of borderpanel **obj**. Needs a flowlayout manager.
j_setflowlayout	*void j_setflowlayout (int obj , int align);* Adds a flowlayout manager to borderpanel **obj** with the specified **align**ment.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to borderpanel **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setgridlayout	*void j_setgridlayout (int obj , int row , int col);*

Adds a gridlayout manager to borderpanel **obj** with the specified **row**s and **col**umns.

j_sethgap
void j_sethgap (int obj , int hgap);
Sets the horizontal gap between components to **hgap** Pixel.

j_setinsets
void j_setinsets (int obj , int top , int bottom , int left , int right);
Set the insets to the specified values.

j_setnamedcolorbg
void j_setnamedcolorbg (int obj , int color);
Sets the background color to a predefined **color**.

j_setnamedcolor
void j_setnamedcolor (int obj , int color);
Sets the foreground color to a predefined **color**.

j_setnolayout
void j_setnolayout (int obj);
Removes the current layout manager from borderpanel **obj** .

j_setpos
void j_setpos (int obj , int xpos , int ypos);
Relocates the borderpanel **obj** to the specified Position (**xpos**,**ypos**).

j_setsize
void j_setsize (int obj , int width , int height);
Resizes borderpanel **obj** to specified **width** and **height**.

j_setvgap
void j_setvgap (int obj , int vgap);
Sets the vertical gap between components to **hgap** Pixel.

j_sevensegment
int j_sevensegment (int obj , int color);
Creates a new sevensegment display with the specified color **color**.

j_show
void j_show (int obj);
Shows the borderpanel **obj**.

j_textarea
int j_textarea (int obj , int rows , int columns);
Creates a new textarea component with the specified number of **rows columns** and returns its event number.

j_textfield
int j_textfield (int obj , int columns);
Creates a new textfield component with the specified number of **columns** and returns its event number.

j_vscrollbar
int j_vscrollbar (int obj);
Creates a new vertical scrollbar and returns its event number.

Canvas

j_canvas	*int j_canvas (int obj , int width , int height);* Creates a new canvas component with the given **width** and **height** and returns its event number. A canvas can be used for general drawing functions. A canvas generates an event, if its size changes. On error −1 will be returned.
j_add	*void j_add (int obj , int cont);* Adds canvas **obj** to container **cont**
j_cliprect	*void j_cliprect (int obj , int x , int y , int width , int height);* Changes current clipping region to the specified rectangle (**x, y, width, height**).
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to canvas **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables canvas **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the canvas **obj**.
j_drawarc	*void j_drawarc (int obj , int x , int y , int rx , int ry , int arc1 , int arc2);* Draws an unfilled arc from angle **arc1** to angle **arc2** with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.
j_drawcircle	*void j_drawcircle (int obj , int x , int y , int r);* Draws an unfilled circle with center (**x, y**) and radius **x**.
j_drawimage	*void j_drawimage (int obj , int image , int x , int y);* Copies the image, given by its eventnumber **image**, to position (**x, y**).
j_drawimagesource	*void j_drawimagesource (int obj , int x , int y , int w , int h , int* r , int* g , int* b);* Paints an image at Position (**x, y,**) with **w**idth and **h**eight. The red, green and blue values of each pixel are given by the arrays **r, g, b**.
j_drawline	*void j_drawline (int obj , int x1 , int y1 , int x2 , int y2);* Draws a line connecting **(x1,y1)** and **(x2,y2)**.
j_drawoval	*void j_drawoval (int obj , int x , int y , int rx , int ry);* Draws an unfilled oval with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.
j_drawpixel	*void j_drawpixel (int obj , int x , int y);* Draws a pixel at **(x,y)**.

j_drawpolygon *void j_drawpolygon (int obj , int len , int* x , int* y);*
Draws an unfilled polygon based on first **len** elements in **x** and **y**.

j_drawpolyline *void j_drawpolyline (int obj , int len , int* x , int* y);*
Draws a series of line segments based on first **len** elements in **x** and **y**.

j_drawrect *void j_drawrect (int obj , int x , int y , int width , int height);*
Draws an unfilled rectangle from **(x,y)** of size **width** x **height**.

j_drawroundrect *void j_drawroundrect (int obj , int x , int y , int width , int height , int arcx , int arcy);*
Draws an unfilled rectangle from **(x,y)** of size **width** x **height** with rounded corners. **arcx** and **arcy** specify the radius of rectangle corners.

j_drawscaleddimage *void j_drawscaleddimage (int obj , int image , int sx , int sy , int sw , int sh , int tx , int ty , int tw , int th);*
Copy the contents of the rectangular area defined by **x, y,**) width **sw**, and height **sh** of the **image** to position (**tx, ty**. The area will be scaled to target width **th** and target height **th**.

j_drawstring *void j_drawstring (int obj , int x , int y , char* str);*
Draws text on screen at position (**x,y**).

j_enable *void j_enable (int obj);*
enables the canvas **obj**.

j_fillarc *void j_fillarc (int obj , int x , int y , int rx , int ry , int arc1 , int arc2);*
Draws an filled arc from angle **arc1** to angle **arc2** with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.

j_fillcircle *void j_fillcircle (int obj , int x , int y , int r);*
Draws an filled circle with center (**x, y**) and radius **x**.

j_filloval *void j_filloval (int obj , int x , int y , int rx , int ry);*
Draws an filled oval with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.

j_fillpolygon *void j_fillpolygon (int obj , int len , int* x , int* y);*
Draws an filled polygon based on first **len** elements in **x** and **y**.

j_fillrect *void j_fillrect (int obj , int x , int y , int width , int height);*
Draws an filled rectangle from **(x,y)** of size **width** x **height**.

j_fillroundrect *void j_fillroundrect (int obj , int x , int y , int width , int height , int arcx , int arcy);*
Draws an filled rectangle from **(x,y)** of size **width** x **height** with rounded corners. **arcx** and **arcy** specify the radius of rectangle corners.

j_focuslistener *int j_focuslistener (int obj);*
Adds a new focus listener to canvas **obj**, and returns its event number.

j_getfontascent *int j_getfontascent (int obj);*

Returns the ascent (space above the baseline) of the actual font of canvas **obj**.

j_getfontheight *int j_getfontheight (int obj);*
Returns the total pixel height of the actual font of canvas **obj**.

j_getheight *int j_getheight (int obj);*
Returns the height of canvas **obj**.

j_getimage *int j_getimage (int obj);*
Copy the contents of canvas **obj** into an image and return its eventnumber.

j_getimagesource *int j_getimagesource (int obj , int x , int y , int w , int h , int* r , int* g , int* b);*
Returns an image of the specified size (**x, y, w**idth, **h**eight) of canvas . The red, green and blue values of each pixel will be stored in **r, g, b**

j_getparentid *int j_getparentid (int obj);*
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getparent *int j_getparent (int obj);*
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getscaledimage *int j_getscaledimage (int obj , int x , int y , int sw , int sh , int tw , int th);*
Copy the contents of the rectangular area defined by **x, y,** width **sw**, and height **sh** into an image and return its eventnumber. The image will be scaled to target width **th** and target height **th**.

j_getstringwidth *int j_getstringwidth (int obj , char* str);*
Returns the length of **str** of the actual font of canvas **obj**.

j_getwidth *int j_getwidth (int obj);*
Returns the width of canvas **obj**.

j_getxpos *int j_getxpos (int obj);*
Returns the current horizontal position of canvas **obj** in its parent's coordinate space.

j_getypos *int j_getypos (int obj);*
Returns the current vertical position of canvas **obj** in its parent's coordinate space.

j_hide *void j_hide (int obj);*
Hides the canvas **obj**.

j_isparent *int j_isparent (int obj , int cont);*
Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.

j_isvisible *int j_isvisible (int obj);*
Returns J_TRUE if **obj** is visible, J_FALSE otherwise.

j_keylistener *int j_keylistener (int obj);*

Adds a new key listener to canvas **obj**, and returns its event number.

j_mouselistener *int j_mouselistener (int obj , int kind);*
Adds a new mouse listener to canvas **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

j_popupmenu *int j_popupmenu (int obj , char* label);*
Creates a new popupmenu with the specified **label** and returns its event number.

j_print *void j_print (int obj);*
prints the canvas .

j_release *void j_release (int obj);*
Releases canvas **obj** from its parent component (container).

j_setborderpos *void j_setborderpos (int obj , int pos);*
Moves canvas **obj** at a certain position. The outer container needs a border layout manager.

j_setcolorbg *void j_setcolorbg (int obj , int r , int g, , int b);*
Sets the background color to the (**r, g, b**) values.

j_setcolor *void j_setcolor (int obj , int r , int g, , int b);*
Sets the foreground color to the (**r, g, b**) values.

j_setcursor *int j_setcursor (int obj , int cursor);*
Changes the canvas 's **obj** cursor to the specified **cursor**.

j_setfocus *int j_setfocus (int obj);*
Directs the input focus to canvas **obj**.

j_setfontname *void j_setfontname (int obj , int name);*
Changes the font to the given **name**.

j_setfont *void j_setfont (int obj , int name , int style , int size);*
Changes the font to the given characteristics **name**, **style** and **size**.

j_setfontsize *void j_setfontsize (int obj , int size);*
Changes the font to the given **size**.

j_setfontstyle *void j_setfontstyle (int obj , int style);*
Changes the font to the given **style**.

j_setnamedcolorbg *void j_setnamedcolorbg (int obj , int color);*
Sets the background color to a predefined **color**.

j_setnamedcolor *void j_setnamedcolor (int obj , int color);*
Sets the foreground color to a predefined **color**.

j_setpos *void j_setpos (int obj , int xpos , int ypos);*
Relocates the canvas **obj** to the specified Position (**xpos**,**ypos**).

j_setsize	*void j_setsize (int obj , int width , int height);* Resizes canvas **obj** to specified **width** and **height**.
j_setxor	*void j_setxor (int obj , int bool);* Changes painting mode to XOR mode, if bool = J_TRUE . In this mode, drawing the same object in the same color at the same location twice has no net effect.
j_show	*void j_show (int obj);* Shows the canvas **obj**.
j_translate	*void j_translate (int obj , int x , int y);* Moves the origin of drawing operations to (**x, y**).

Checkbox

j_checkbox	*int j_checkbox (int obj , char* label);* Creates a new checkbox component with the specified **label** and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds checkbox **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to checkbox **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables checkbox **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the checkbox **obj**.
j_enable	*void j_enable (int obj);* enables the checkbox **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to checkbox **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of checkbox **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of checkbox **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of checkbox **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstate	*int j_getstate (int obj);* Returns J_TRUE , if checkbox is selected, J_FALSE otherwise.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of checkbox **obj**.

j_gettext	char* j_gettext (int obj , char* str); returns the checkbox 's text or label.
j_getwidth	int j_getwidth (int obj); Returns the width of checkbox **obj**.
j_getxpos	int j_getxpos (int obj); Returns the current horizontal position of checkbox **obj** in its parent's coordinate space.
j_getypos	int j_getypos (int obj); Returns the current vertical position of checkbox **obj** in its parent's coordinate space.
j_hide	void j_hide (int obj); Hides the checkbox **obj**.
j_isparent	int j_isparent (int obj , int cont); Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	int j_isvisible (int obj); Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	int j_keylistener (int obj); Adds a new key listener to checkbox **obj**, and returns its event number.
j_mouselistener	int j_mouselistener (int obj , int kind); Adds a new mouse listener to checkbox **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	int j_popupmenu (int obj , char* label); Creates a new popupmenu with the specified **label** and returns its event number.
j_print	void j_print (int obj); prints the checkbox .
j_release	void j_release (int obj); Releases checkbox **obj** from its parent component (container).
j_setborderpos	void j_setborderpos (int obj , int pos); Moves checkbox **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	void j_setcolorbg (int obj , int r , int g, , int b); Sets the background color to the (**r, g, b**) values.
j_setcolor	void j_setcolor (int obj , int r , int g, , int b); Sets the foreground color to the (**r, g, b**) values.
j_setcursor	int j_setcursor (int obj , int cursor); Changes the checkbox 's **obj** cursor to the specified **cursor**.

j_setfocus	*int j_setfocus (int obj);* Directs the input focus to checkbox **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the checkbox **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes checkbox **obj** to specified **width** and **height**.
j_setstate	*void j_setstate (int obj , int bool);* The checkbox becomes selected, if **bool** is J_TRUE .
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the checkbox **obj** to **str**.
j_show	*void j_show (int obj);* Shows the checkbox **obj**.

Checkmenuitem

j_checkmenuitem	*int j_checkmenuitem (int obj , char* label);* creates a new checkmenuitem with the specified **label** and returns its event number.
j_disable	*void j_disable (int obj);* Disables checkmenuitem **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the checkmenuitem **obj**.
j_enable	*void j_enable (int obj);* enables the checkmenuitem **obj**.
j_getlength	*int j_getlength (int obj);* Returns the length of checkmenuitem 's label or text.
j_getstate	*int j_getstate (int obj);* Returns J_TRUE , if checkmenuitem is selected, J_FALSE otherwise.
j_gettext	*char* j_gettext (int obj , char* str);* returns the checkmenuitem 's text or label.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setshortcut	*void j_setshortcut (int obj , char chr);* Changes the shortcut **chr** of the checkmenuitem .
j_setstate	*void j_setstate (int obj , int bool);* The checkmenuitem becomes selected, if **bool** is J_TRUE .
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the checkmenuitem **obj** to **str**.

Choice

j_choice	*int j_choice (int obj);* Creates a new choice component and returns its event number.
j_additem	*void j_additem (int obj , char* str);* adds a new item containing **str** to choice **obj**.
j_add	*void j_add (int obj , int cont);* Adds choice **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to choice **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables choice **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the choice **obj**.
j_enable	*void j_enable (int obj);* enables the choice **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to choice **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of choice **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of choice **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of choice **obj**.
j_getitemcount	*int j_getitemcount (int obj);* Returns the number of items of choice **obj**.
j_getitem	*char* j_getitem (int obj , int item , char* str);* returns the label of the given **item**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);*

Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getselect *int j_getselect (int obj);*
Returns the position of currently selected item.

j_getstringwidth *int j_getstringwidth (int obj , char* str);*
Returns the length of **str** of the actual font of choice **obj**.

j_getwidth *int j_getwidth (int obj);*
Returns the width of choice **obj**.

j_getxpos *int j_getxpos (int obj);*
Returns the current horizontal position of choice **obj** in its parent's coordinate space.

j_getypos *int j_getypos (int obj);*
Returns the current vertical position of choice **obj** in its parent's coordinate space.

j_hide *void j_hide (int obj);*
Hides the choice **obj**.

j_insert *int j_insert (int obj , int pos , char* label);*
inserts a new item to choice **obj** at position **pos** with the specified **label**.

j_isparent *int j_isparent (int obj , int cont);*
Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.

j_isvisible *int j_isvisible (int obj);*
Returns J_TRUE if **obj** is visible, J_FALSE otherwise.

j_keylistener *int j_keylistener (int obj);*
Adds a new key listener to choice **obj**, and returns its event number.

j_mouselistener *int j_mouselistener (int obj , int kind);*
Adds a new mouse listener to choice **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

j_popupmenu *int j_popupmenu (int obj , char* label);*
Creates a new popupmenu with the specified **label** and returns its event number.

j_print *void j_print (int obj);*
prints the choice .

j_release *void j_release (int obj);*
Releases choice **obj** from its parent component (container).

j_removeall *int j_removeall (int obj);*
Removes all items from the choice .

j_removeitem *int j_removeitem (int obj , char* item);*

	remove the first occurrence of **item** from the choice .
j_remove	*int j_remove (int obj , int item);* removes the Item with the Index **item** from the choice .
j_select	*int j_select (int obj , int item);* Makes the given **item** the selected one for the choice .
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves choice **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the choice 's **obj** cursor to the specified **cursor**.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to choice **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the choice **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes choice **obj** to specified **width** and **height**.
j_show	*void j_show (int obj);* Shows the choice **obj**.

Dialog

j_dialog	*int j_dialog (int obj , char* label);* Creates a new dialog window with the specified **label** and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds dialog **obj** to container **cont**
j_borderpanel	*int j_borderpanel (int obj , int type);* Creates a new borderpanel component with the style **type** and returns its event number.
j_button	*int j_button (int obj , char* label);* Creates a new button component with the specified **label** and returns its event number.
j_canvas	*int j_canvas (int obj , int width , int height);* Creates a new canvas component with the given **width** and **height** and returns its event number. A canvas can be used for general drawing functions. A canvas generates an event, if its size changes. On error -1 will be returned.
j_checkbox	*int j_checkbox (int obj , char* label);* Creates a new checkbox component with the specified **label** and returns its event number.
j_choice	*int j_choice (int obj);* Creates a new choice component and returns its event number.
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to dialog **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables dialog **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the dialog **obj**.
j_enable	*void j_enable (int obj);* enables the dialog **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to dialog **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of dialog **obj**.
j_getfontheight	*int j_getfontheight (int obj);*

	Returns the total pixel height of the actual font of dialog **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of dialog **obj**.
j_getinheight	*int j_getinheight (int cont);* Returns the height of the client size.
j_getinsets	*int j_getinsets (int obj , int side);* Returns the width of the specified inset.
j_getinwidth	*int j_getinwidth (int cont);* Returns the width of the client size.
j_getlayoutid	*int j_getlayoutid (int obj);* Returns the event number of the layoutmanager for containers **obj**.
j_getlength	*int j_getlength (int obj);* Returns the length of dialog 's label or text.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of dialog **obj**.
j_gettext	*char* j_gettext (int obj , char* str);* returns the dialog 's text or label.
j_getwidth	*int j_getwidth (int obj);* Returns the width of dialog **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of dialog **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of dialog **obj** in its parent's coordinate space.
j_graphicbutton	*int j_graphicbutton (int obj , char* filename);* Creates a new graphicbutton component with the image loaded from **filename** and returns its event number.
j_graphiclabel	*int j_graphiclabel (int obj , char* str);* Creates a new graphiclabel component with the image loaded from **filename** and returns its event number.

j_hide	*void j_hide (int obj);* Hides the dialog **obj**.
j_hscrollbar	*int j_hscrollbar (int obj);* Creates a new horizontal scrollbar and returns its event number.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isresizable	*int j_isresizable (int obj);* returns true if dialog is resizable, false otherwise
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to dialog **obj**, and returns its event number.
j_label	*int j_label (int obj , char* label);* Creates a new label component with the specified **label** and returns its event number.
j_led	*int j_led (int obj , int style , int color);* Creates a new led component with the specified **style** and the specified color **color**.
j_line	*int j_line (int obj , int orient , int style , int length);* Creates a new line component with the specified **length** and returns its event number.
j_list	*int j_list (int obj , int rows);* Creates a new list component with the specified number of **rows** and returns its event number.
j_meter	*int j_meter (int obj , char* title);* Creates a new pointer–intrument with the specified label **titel**.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to dialog **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_pack	*void j_pack (int obj);* Resizes dialog to the minimal size of contained components.
j_panel	*int j_panel (int obj);* Creates a new panel component and returns its event number.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the dialog .

j_progressbar	*int j_progressbar (int obj , int orient);* Creates a new progressbar with the specified **orient**ation.
j_radiogroup	*int j_radiogroup (int obj);* Creates a new radiogroup and returns its event number.
j_releaseall	*void j_releaseall (int obj);* Releases all components from dialog **obj**.
j_release	*void j_release (int obj);* Releases dialog **obj** from its parent component (container).
j_scrollpane	*int j_scrollpane (int obj);* Creates a new scrollpane component and returns its event number.
j_setalign	*void j_setalign (int obj , int align);* Sets the alignment in dialog **obj** to **align**. Needs a flowlayout Manager.
j_setborderlayout	*void j_setborderlayout (int obj);* Adds a borderlayout manager to dialog **obj**.
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves dialog **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the dialog 's **obj** cursor to the specified **cursor**.
j_setfixlayout	*void j_setfixlayout (int obj);* Adds a fixlayout manager to dialog **obj** (default layout manager).
j_setflowfill	*void j_setflowfill (int obj , int bool);* Resizes all containing component to the height (width) of dialog **obj**. Needs a flowlayout manager.
j_setflowlayout	*void j_setflowlayout (int obj , int align);* Adds a flowlayout manager to dialog **obj** with the specified **align**ment.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to dialog **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.

j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setgridlayout	*void j_setgridlayout (int obj , int row , int col);* Adds a gridlayout manager to dialog **obj** with the specified **row**s and **col**umns.
j_sethgap	*void j_sethgap (int obj , int hgap);* Sets the horizontal gap between components to **hgap** Pixel.
j_setinsets	*void j_setinsets (int obj , int top , int bottom , int left , int right);* Set the insets to the specified values.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setnolayout	*void j_setnolayout (int obj);* Removes the current layout manager from dialog **obj** .
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the dialog **obj** to the specified Position (**xpos**,**ypos**).
j_setresizable	*void j_setresizable (int obj , int resizable);* The dialog cannot be resized, if **resizable** is J_FALSE .
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes dialog **obj** to specified **width** and **height**.
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the dialog **obj** to **str**.
j_setvgap	*void j_setvgap (int obj , int vgap);* Sets the vertical gap between components to **hgap** Pixel.
j_sevensegment	*int j_sevensegment (int obj , int color);* Creates a new sevensegment display with the specified color **color**.
j_show	*void j_show (int obj);* Shows the dialog **obj**.
j_textarea	*int j_textarea (int obj , int rows , int columns);* Creates a new textarea component with the specified number of **rows columns** and returns its event number.
j_textfield	*int j_textfield (int obj , int columns);* Creates a new textfield component with the specified number of **columns** and returns its event number.

j_vscrollbar *int j_vscrollbar (int obj);*
Creates a new vertical scrollbar and returns its event number.

j_windowlistener *int j_windowlistener (int window , int kind);*
Adds a new windowlistener to **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

Focuslistener

j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to focuslistener **obj**, and returns its event number.
j_dispose	*void j_dispose (int obj);* Releases the resources of the focuslistener **obj**.
j_hasfocus	*int j_hasfocus (int obj);* Returns J_TRUE if the focuslistener has the focus, J_FALSE otherwise.

Frame

j_frame	*int j_frame (char* label);* Creates a new frame component with the specified **label** and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds frame **obj** to container **cont**
j_alertbox	*void j_alertbox (int obj , char* title , char* text , char* button);* Shows a alertbox with the specified **title**, **text** and **button**.
j_borderpanel	*int j_borderpanel (int obj , int type);* Creates a new borderpanel component with the style **type** and returns its event number.
j_button	*int j_button (int obj , char* label);* Creates a new button component with the specified **label** and returns its event number.
j_canvas	*int j_canvas (int obj , int width , int height);* Creates a new canvas component with the given **width** and **height** and returns its event number. A canvas can be used for general drawing functions. A canvas generates an event, if its size changes. On error -1 will be returned.
j_checkbox	*int j_checkbox (int obj , char* label);* Creates a new checkbox component with the specified **label** and returns its event number.
j_choicebox2	*void j_choicebox2 (int obj , char* title , char* text , char* button1 , char* button2);* Shows a choicebox with the specified **title**, **text** and two buttons.
j_choicebox3	*void j_choicebox3 (int obj , char* title , char* text , char* button1 , char* button2 , char* button3);* Shows a choicebox with the specified **title**, **text** and three buttons.
j_choice	*int j_choice (int obj);* Creates a new choice component and returns its event number.
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to frame **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_dialog	*int j_dialog (int obj , char* label);* Creates a new dialog window with the specified **label** and returns its event number.
j_disable	*void j_disable (int obj);*

	Disables frame **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the frame **obj**.
j_enable	*void j_enable (int obj);* enables the frame **obj**.
j_filedialog	*char* j_filedialog (int frame , char* title , char* directory , char* filename);* Opens a filedialog box in the specified **directory** with the specified **title** and returns the selected **filename**.
j_fileselect	*char* j_fileselect (int frame , char* title , char* filter , char* filename);* Opens a fileslector box with the preselected **filename** and the specified **title** and returns the selected **filename**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to frame **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of frame **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of frame **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of frame **obj**.
j_getinheight	*int j_getinheight (int cont);* Returns the height of the client size.
j_getinsets	*int j_getinsets (int obj , int side);* Returns the width of the specified inset.
j_getinwidth	*int j_getinwidth (int cont);* Returns the width of the client size.
j_getlayoutid	*int j_getlayoutid (int obj);* Returns the event number of the layoutmanager for containers **obj**.
j_getlength	*int j_getlength (int obj);* Returns the length of frame 's label or text.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of frame **obj**.

j_gettext	*char* j_gettext (int obj , char* str);*	returns the frame 's text or label.
j_getwidth	*int j_getwidth (int obj);*	Returns the width of frame **obj**.
j_getxpos	*int j_getxpos (int obj);*	Returns the current horizontal position of frame **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);*	Returns the current vertical position of frame **obj** in its parent's coordinate space.
j_graphicbutton	*int j_graphicbutton (int obj , char* filename);*	Creates a new graphicbutton component with the image loaded from **filename** and returns its event number.
j_graphiclabel	*int j_graphiclabel (int obj , char* str);*	Creates a new graphiclabel component with the image loaded from **filename** and returns its event number.
j_hide	*void j_hide (int obj);*	Hides the frame **obj**.
j_hscrollbar	*int j_hscrollbar (int obj);*	Creates a new horizontal scrollbar and returns its event number.
j_isparent	*int j_isparent (int obj , int cont);*	Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isresizable	*int j_isresizable (int obj);*	returns true if frame is resizable, false otherwise
j_isvisible	*int j_isvisible (int obj);*	Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);*	Adds a new key listener to frame **obj**, and returns its event number.
j_label	*int j_label (int obj , char* label);*	Creates a new label component with the specified **label** and returns its event number.
j_led	*int j_led (int obj , int style , int color);*	Creates a new led component with the specified **style** and the specified color **color**.
j_line	*int j_line (int obj , int orient , int style , int length);*	Creates a new line component with the specified **length** and returns its event number.
j_list	*int j_list (int obj , int rows);*	

	Creates a new list component with the specified number of **rows** and returns its event number.
j_menubar	*int j_menubar (int obj);* Creates a new menubar and returns its event number.
j_messagebox	*void j_messagebox (int obj , char* title , char* text);* Shows a messagebox with the specified **title** and **text** and returns its event number.
j_meter	*int j_meter (int obj , char* title);* Creates a new pointer–intrument with the specified label **titel**.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to frame **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_pack	*void j_pack (int obj);* Resizes frame to the minimal size of contained components.
j_panel	*int j_panel (int obj);* Creates a new panel component and returns its event number.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_printer	*int j_printer (int frame);* Creates a new object, representing a paper of the printer.
j_print	*void j_print (int obj);* prints the frame .
j_progressbar	*int j_progressbar (int obj , int orient);* Creates a new progressbar with the specified **orient**ation.
j_radiogroup	*int j_radiogroup (int obj);* Creates a new radiogroup and returns its event number.
j_releaseall	*void j_releaseall (int obj);* Releases all components from frame **obj**.
j_release	*void j_release (int obj);* Releases frame **obj** from its parent component (container).
j_scrollpane	*int j_scrollpane (int obj);* Creates a new scrollpane component and returns its event number.
j_setalign	*void j_setalign (int obj , int align);* Sets the alignment in frame **obj** to **align**. Needs a flowlayout Manager.
j_setborderlayout	*void j_setborderlayout (int obj);* Adds a borderlayout manager to frame **obj**.

j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves frame **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the frame 's **obj** cursor to the specified **cursor**.
j_setfixlayout	*void j_setfixlayout (int obj);* Adds a fixlayout manager to frame **obj** (default layout manager).
j_setflowfill	*void j_setflowfill (int obj , int bool);* Resizes all containing component to the height (width) of frame **obj**. Needs a flowlayout manager.
j_setflowlayout	*void j_setflowlayout (int obj , int align);* Adds a flowlayout manager to frame **obj** with the specified **align**ment.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to frame **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setgridlayout	*void j_setgridlayout (int obj , int row , int col);* Adds a gridlayout manager to frame **obj** with the specified **row**s and **col**umns.
j_sethgap	*void j_sethgap (int obj , int hgap);* Sets the horizontal gap between components to **hgap** Pixel.
j_seticon	*void j_seticon (int frame , int icon);* Sets the image **icon** to display when the **frame** is iconized. Not all platforms support the concept of iconizing a window.
j_setinsets	*void j_setinsets (int obj , int top , int bottom , int left , int right);* Set the insets to the specified values.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);*

	Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setnolayout	*void j_setnolayout (int obj);* Removes the current layout manager from frame **obj** .
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the frame **obj** to the specified Position (**xpos**,**ypos**).
j_setresizable	*void j_setresizable (int obj , int resizable);* The frame cannot be resized, if **resizable** is J_FALSE .
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes frame **obj** to specified **width** and **height**.
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the frame **obj** to **str**.
j_setvgap	*void j_setvgap (int obj , int vgap);* Sets the vertical gap between components to **hgap** Pixel.
j_sevensegment	*int j_sevensegment (int obj , int color);* Creates a new sevensegment display with the specified color **color**.
j_show	*void j_show (int obj);* Shows the frame **obj**.
j_textarea	*int j_textarea (int obj , int rows , int columns);* Creates a new textarea component with the specified number of **rows columns** and returns its event number.
j_textfield	*int j_textfield (int obj , int columns);* Creates a new textfield component with the specified number of **columns** and returns its event number.
j_vscrollbar	*int j_vscrollbar (int obj);* Creates a new vertical scrollbar and returns its event number.
j_windowlistener	*int j_windowlistener (int window , int kind);* Adds a new windowlistener to **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_window	*int j_window (int obj);* Creates a new simple window and returns its event number.

Helpmenu

j_helpmenu	int j_helpmenu (int obj , char* label); Creates a new helpmenu component with the specified **label** and returns its event number.
j_checkmenuitem	int j_checkmenuitem (int obj , char* label); creates a new checkmenuitem with the specified **label** and returns its event number.
j_disable	void j_disable (int obj); Disables helpmenu **obj** so that it is unresponsive to user interactions
j_dispose	void j_dispose (int obj); Releases the resources of the helpmenu **obj**.
j_enable	void j_enable (int obj); enables the helpmenu **obj**.
j_getlength	int j_getlength (int obj); Returns the length of helpmenu 's label or text.
j_gettext	char* j_gettext (int obj , char* str); returns the helpmenu 's text or label.
j_menuitem	int j_menuitem (int obj , char* label); Creates a new menuitem with the specified **label** and returns its event number.
j_seperator	void j_seperator (int obj); Adds a separator bar to the helpmenu .
j_setfontname	void j_setfontname (int obj , int name); Changes the font to the given **name**.
j_setfont	void j_setfont (int obj , int name , int style , int size); Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	void j_setfontsize (int obj , int size); Changes the font to the given **size**.
j_setfontstyle	void j_setfontstyle (int obj , int style); Changes the font to the given **style**.
j_setshortcut	void j_setshortcut (int obj , char chr); Changes the shortcut **chr** of the helpmenu .
j_settext	void j_settext (int obj , char* str); Sets the content or the label of the helpmenu **obj** to **str**.

Hscrollbar

j_hscrollbar	*int j_hscrollbar (int obj);* Creates a new horizontal scrollbar and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds hscrollbar **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to hscrollbar **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables hscrollbar **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the hscrollbar **obj**.
j_enable	*void j_enable (int obj);* enables the hscrollbar **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to hscrollbar **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of hscrollbar **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of hscrollbar **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of hscrollbar **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of hscrollbar **obj**.
j_getvalue	*int j_getvalue (int obj);* Returns the current setting of the scrollbar.

j_getwidth	*int j_getwidth (int obj);* Returns the width of hscrollbar **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of hscrollbar **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of hscrollbar **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the hscrollbar **obj**.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to hscrollbar **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to hscrollbar **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the hscrollbar .
j_release	*void j_release (int obj);* Releases hscrollbar **obj** from its parent component (container).
j_setblockinc	*int j_setblockinc (int obj , int val);* Changes the block increment amount for the hscrollbar to **val**.
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves hscrollbar **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the hscrollbar 's **obj** cursor to the specified **cursor**.

j_setfocus	*int j_setfocus (int obj);* Directs the input focus to hscrollbar **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setmax	*int j_setmax (int obj , int val);* Changes the maximum value for the hscrollbar to **val**.
j_setmin	*int j_setmin (int obj , int val);* Changes the minimum value for the hscrollbar to **val**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the hscrollbar **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes hscrollbar **obj** to specified **width** and **height**.
j_setslidesize	*int j_setslidesize (int obj , int val);* Changes the slide size to **val**.
j_setunitinc	*int j_setunitinc (int obj , int val);* Changes the unit increment amount for the hscrollbar to **val**
j_setvalue	*void j_setvalue (int obj , int val);* Changes the current value of the hscrollbar to **val**.
j_show	*void j_show (int obj);* Shows the hscrollbar **obj**.

Graphicbutton

j_graphicbutton *int j_graphicbutton (int obj , char* filename);*
Creates a new graphicbutton component with the image loaded from **filename** and returns its event number.

j_add *void j_add (int obj , int cont);*
Adds graphicbutton **obj** to container **cont**

j_componentlistener *int j_componentlistener (int obj , int kind);*
Adds a new componentlistener to graphicbutton **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

j_disable *void j_disable (int obj);*
Disables graphicbutton **obj** so that it is unresponsive to user interactions

j_dispose *void j_dispose (int obj);*
Releases the resources of the graphicbutton **obj**.

j_enable *void j_enable (int obj);*
enables the graphicbutton **obj**.

j_focuslistener *int j_focuslistener (int obj);*
Adds a new focus listener to graphicbutton **obj**, and returns its event number.

j_getfontascent *int j_getfontascent (int obj);*
Returns the ascent (space above the baseline) of the actual font of graphicbutton **obj**.

j_getfontheight *int j_getfontheight (int obj);*
Returns the total pixel height of the actual font of graphicbutton **obj**.

j_getheight *int j_getheight (int obj);*
Returns the height of graphicbutton **obj**.

j_getparentid *int j_getparentid (int obj);*
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getparent *int j_getparent (int obj);*
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getstringwidth *int j_getstringwidth (int obj , char* str);*
Returns the length of **str** of the actual font of graphicbutton **obj**.

j_getwidth *int j_getwidth (int obj);*
Returns the width of graphicbutton **obj**.

j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of graphicbutton **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of graphicbutton **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the graphicbutton **obj**.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to graphicbutton **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to graphicbutton **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the graphicbutton .
j_release	*void j_release (int obj);* Releases graphicbutton **obj** from its parent component (container).
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves graphicbutton **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the graphicbutton 's **obj** cursor to the specified **cursor**.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to graphicbutton **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.

j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setimage	*void j_setimage (int obj , int image);* Sets the **image** to be displayed in **obj**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the graphicbutton **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes graphicbutton **obj** to specified **width** and **height**.
j_show	*void j_show (int obj);* Shows the graphicbutton **obj**.

Graphiclabel

j_graphiclabel	*int j_graphiclabel (int obj , char* str);* Creates a new graphiclabel component with the image loaded from **filename** and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds graphiclabel **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to graphiclabel **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables graphiclabel **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the graphiclabel **obj**.
j_enable	*void j_enable (int obj);* enables the graphiclabel **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to graphiclabel **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of graphiclabel **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of graphiclabel **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of graphiclabel **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of graphiclabel **obj**.
j_getwidth	*int j_getwidth (int obj);* Returns the width of graphiclabel **obj**.

j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of graphiclabel **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of graphiclabel **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the graphiclabel **obj**.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to graphiclabel **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to graphiclabel **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the graphiclabel .
j_release	*void j_release (int obj);* Releases graphiclabel **obj** from its parent component (container).
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves graphiclabel **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the graphiclabel 's **obj** cursor to the specified **cursor**.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to graphiclabel **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.

j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setimage	*void j_setimage (int obj , int image);* Sets the **image** to be displayed in **obj**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the graphiclabel **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes graphiclabel **obj** to specified **width** and **height**.
j_show	*void j_show (int obj);* Shows the graphiclabel **obj**.

Image

j_image	*int j_image (int width , int height);* Creates a new (memory) image component with the given **width** and **height** and returns its event number.
j_cliprect	*void j_cliprect (int obj , int x , int y , int width , int height);* Changes current clipping region to the specified rectangle (**x, y, width, height**).
j_dispose	*void j_dispose (int obj);* Releases the resources of the image **obj**.
j_drawarc	*void j_drawarc (int obj , int x , int y , int rx , int ry , int arc1 , int arc2);* Draws an unfilled arc from angle **arc1** to angle **arc2** with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.
j_drawcircle	*void j_drawcircle (int obj , int x , int y , int r);* Draws an unfilled circle with center (**x, y**) and radius **x**.
j_drawimage	*void j_drawimage (int obj , int image , int x , int y);* Copies the image, given by its eventnumber **image**, to position (**x, y**).
j_drawimagesource	*void j_drawimagesource (int obj , int x , int y , int w , int h , int* r , int* g , int* b);* Paints an image at Position (**x, y,**) with **w**idth and **h**eight. The red, green and blue values of each pixel are given by the arrays **r, g, b**.
j_drawline	*void j_drawline (int obj , int x1 , int y1 , int x2 , int y2);* Draws a line connecting **(x1,y1)** and **(x2,y2)**.
j_drawoval	*void j_drawoval (int obj , int x , int y , int rx , int ry);* Draws an unfilled oval with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.
j_drawpixel	*void j_drawpixel (int obj , int x , int y);* Draws a pixel at **(x,y)**.
j_drawpolygon	*void j_drawpolygon (int obj , int len , int* x , int* y);* Draws an unfilled polygon based on first **len** elements in **x** and **y**.
j_drawpolyline	*void j_drawpolyline (int obj , int len , int* x , int* y);* Draws a series of line segments based on first **len** elements in **x** and **y**.
j_drawrect	*void j_drawrect (int obj , int x , int y , int width , int height);* Draws an unfilled rectangle from **(x,y)** of size **width** x **height**.
j_drawroundrect	*void j_drawroundrect (int obj , int x , int y , int width , int height , int arcx , int arcy);*

	Draws an unfilled rectangle from **(x,y)** of size **width** x **height** with rounded corners. **arcx** and **arcy** specify the radius of rectangle corners.
j_drawscaleddimage	*void j_drawscaleddimage (int obj , int image , int sx , int sy , int sw , int sh , int tx , int ty , int tw , int th);* Copy the contents of the rectangular area defined by **x, y,**) width **sw**, and height **sh** of the **image** to position (**tx, ty**. The area will be scaled to target width **th** and target height **th**.
j_drawstring	*void j_drawstring (int obj , int x , int y , char* str);* Draws text on screen at position (**x,y**).
j_fillarc	*void j_fillarc (int obj , int x , int y , int rx , int ry , int arc1 , int arc2);* Draws an filled arc from angle **arc1** to angle **arc2** with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.
j_fillcircle	*void j_fillcircle (int obj , int x , int y , int r);* Draws an filled circle with center (**x, y**) and radius **x**.
j_filloval	*void j_filloval (int obj , int x , int y , int rx , int ry);* Draws an filled oval with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.
j_fillpolygon	*void j_fillpolygon (int obj , int len , int* x , int* y);* Draws an filled polygon based on first **len** elements in **x** and **y**.
j_fillrect	*void j_fillrect (int obj , int x , int y , int width , int height);* Draws an filled rectangle from **(x,y)** of size **width** x **height**.
j_fillroundrect	*void j_fillroundrect (int obj , int x , int y , int width , int height , int arcx , int arcy);* Draws an filled rectangle from **(x,y)** of size **width** x **height** with rounded corners. **arcx** and **arcy** specify the radius of rectangle corners.
j_getheight	*int j_getheight (int obj);* Returns the height of image **obj**.
j_getimage	*int j_getimage (int obj);* Copy the contents of image **obj** into an image and return its eventnumber.
j_getimagesource	*int j_getimagesource (int obj , int x , int y , int w , int h , int* r , int* g , int* b);* Returns an image of the specified size (**x, y, w**idth, **h**eight) of image . The red, green and blue values of each pixel will be stored in **r, g, b**
j_getscaledimage	*int j_getscaledimage (int obj , int x , int y , int sw , int sh , int tw , int th);* Copy the contents of the rectangular area defined by **x, y,** width **sw**, and height **sh** into an image and return its eventnumber. The image will be scaled to target width **th** and target height **th**.
j_getwidth	*int j_getwidth (int obj);* Returns the width of image **obj**.

j_print *void j_print (int obj);*
prints the image .

j_setxor *void j_setxor (int obj , int bool);*
Changes painting mode to XOR mode, if bool = J_TRUE . In this mode, drawing the same object in the same color at the same location twice has no net effect.

j_translate *void j_translate (int obj , int x , int y);*
Moves the origin of drawing operations to (**x, y**).

Keylistener

j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to keylistener **obj**, and returns its event number.
j_dispose	*void j_dispose (int obj);* Releases the resources of the keylistener **obj**.
j_getkeychar	*int j_getkeychar (int obj);* Returns the ascii value of the last pressed key.
j_getkeycode	*int j_getkeycode (int obj);* Returns the integer key code of the last pressed key.

Label

j_label	*int j_label (int obj , char* label);* Creates a new label component with the specified **label** and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds label **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to label **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables label **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the label **obj**.
j_enable	*void j_enable (int obj);* enables the label **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to label **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of label **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of label **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of label **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of label **obj**.
j_gettext	*char* j_gettext (int obj , char* str);* returns the label 's text or label.
j_getwidth	*int j_getwidth (int obj);*

	Returns the width of label **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of label **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of label **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the label **obj**.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to label **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to label **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the label .
j_release	*void j_release (int obj);* Releases label **obj** from its parent component (container).
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves label **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the label 's **obj** cursor to the specified **cursor**.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to label **obj**.
j_setfontname	*void j_setfontname (int obj , int name);*

	Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the label **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes label **obj** to specified **width** and **height**.
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the label **obj** to **str**.
j_show	*void j_show (int obj);* Shows the label **obj**.

Led

j_led	*int j_led (int obj , int style , int color);* Creates a new led component with the specified **style** and the specified color **color**.
j_add	*void j_add (int obj , int cont);* Adds led **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to led **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables led **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the led **obj**.
j_enable	*void j_enable (int obj);* enables the led **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to led **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of led **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of led **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of led **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstate	*int j_getstate (int obj);* Returns J_TRUE , if led is selected, J_FALSE otherwise.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of led **obj**.
j_getwidth	*int j_getwidth (int obj);*

	Returns the width of led **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of led **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of led **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the led **obj**.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to led **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to led **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the led .
j_release	*void j_release (int obj);* Releases led **obj** from its parent component (container).
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves led **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the led 's **obj** cursor to the specified **cursor**.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to led **obj**.
j_setfontname	*void j_setfontname (int obj , int name);*

	Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the led **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes led **obj** to specified **width** and **height**.
j_setstate	*void j_setstate (int obj , int bool);* The led becomes selected, if **bool** is J_TRUE .
j_show	*void j_show (int obj);* Shows the led **obj**.

List

j_list	*int j_list (int obj , int rows);* Creates a new list component with the specified number of **rows** and returns its event number.
j_additem	*void j_additem (int obj , char* str);* adds a new item containing **str** to list **obj**.
j_add	*void j_add (int obj , int cont);* Adds list **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to list **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_deselect	*int j_deselect (int obj , int item);* Deselects the item at the designated position **item**, if selected.
j_disable	*void j_disable (int obj);* Disables list **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the list **obj**.
j_enable	*void j_enable (int obj);* enables the list **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to list **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of list **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of list **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of list **obj**.
j_getitemcount	*int j_getitemcount (int obj);* Returns the number of items of list **obj**.
j_getitem	*char* j_getitem (int obj , int item , char* str);* returns the label of the given **item**.
j_getparentid	*int j_getparentid (int obj);*

Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getparent *int j_getparent (int obj);*
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getselect *int j_getselect (int obj);*
Returns the position of currently selected item.

j_getstringwidth *int j_getstringwidth (int obj , char* str);*
Returns the length of **str** of the actual font of list **obj**.

j_getwidth *int j_getwidth (int obj);*
Returns the width of list **obj**.

j_getxpos *int j_getxpos (int obj);*
Returns the current horizontal position of list **obj** in its parent's coordinate space.

j_getypos *int j_getypos (int obj);*
Returns the current vertical position of list **obj** in its parent's coordinate space.

j_hide *void j_hide (int obj);*
Hides the list **obj**.

j_insert *int j_insert (int obj , int pos , char* label);*
inserts a new item to list **obj** at position **pos** with the specified **label**.

j_isparent *int j_isparent (int obj , int cont);*
Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.

j_isselect *int j_isselect (int obj , int item);*
Returns J_TRUE if the particular **item** is currently selected, J_FALSE otherwise.

j_isvisible *int j_isvisible (int obj);*
Returns J_TRUE if **obj** is visible, J_FALSE otherwise.

j_keylistener *int j_keylistener (int obj);*
Adds a new key listener to list **obj**, and returns its event number.

j_mouselistener *int j_mouselistener (int obj , int kind);*
Adds a new mouse listener to list **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

j_multiplemode *int j_multiplemode (int obj , int bool);*
if **bool** is J_TRUE , selection mode is turned to multiplemode.

j_popupmenu *int j_popupmenu (int obj , char* label);*
Creates a new popupmenu with the specified **label** and returns its event number.

j_print	*void j_print (int obj);*	prints the list .
j_release	*void j_release (int obj);*	Releases list **obj** from its parent component (container).
j_removeall	*int j_removeall (int obj);*	Removes all items from the list .
j_removeitem	*int j_removeitem (int obj , char* item);*	remove the first occurrence of **item** from the list .
j_remove	*int j_remove (int obj , int item);*	removes the Item with the Index **item** from the list .
j_select	*int j_select (int obj , int item);*	Makes the given **item** the selected one for the list .
j_setborderpos	*void j_setborderpos (int obj , int pos);*	Moves list **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);*	Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);*	Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);*	Changes the list 's **obj** cursor to the specified **cursor**.
j_setfocus	*int j_setfocus (int obj);*	Directs the input focus to list **obj**.
j_setfontname	*void j_setfontname (int obj , int name);*	Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);*	Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);*	Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);*	Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);*	Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);*	Sets the foreground color to a predefined **color**.

j_setpos *void j_setpos (int obj , int xpos , int ypos);*
Relocates the list **obj** to the specified Position (**xpos**,**ypos**).

j_setsize *void j_setsize (int obj , int width , int height);*
Resizes list **obj** to specified **width** and **height**.

j_show *void j_show (int obj);*
Shows the list **obj**.

Menu

j_menu	*int j_menu (int obj , char* str);* Creates a new menu component with the specified **label** and returns its event number.
j_checkmenuitem	*int j_checkmenuitem (int obj , char* label);* creates a new checkmenuitem with the specified **label** and returns its event number.
j_disable	*void j_disable (int obj);* Disables menu **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the menu **obj**.
j_enable	*void j_enable (int obj);* enables the menu **obj**.
j_getlength	*int j_getlength (int obj);* Returns the length of menu 's label or text.
j_gettext	*char* j_gettext (int obj , char* str);* returns the menu 's text or label.
j_helpmenu	*int j_helpmenu (int obj , char* label);* Creates a new helpmenu component with the specified **label** and returns its event number.
j_menuitem	*int j_menuitem (int obj , char* label);* Creates a new menuitem with the specified **label** and returns its event number.
j_menu	*int j_menu (int obj , char* str);* Creates a new menu component with the specified **label** and returns its event number.
j_seperator	*void j_seperator (int obj);* Adds a separator bar to the menu .
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);*

Changes the font to the given **style**.

j_setshortcut *void j_setshortcut (int obj , char chr);*
Changes the shortcut **chr** of the menu .

j_settext *void j_settext (int obj , char* str);*
Sets the content or the label of the menu **obj** to **str**.

Menuitem

j_menuitem	*int j_menuitem (int obj , char* label);* Creates a new menuitem with the specified **label** and returns its event number.
j_disable	*void j_disable (int obj);* Disables menuitem **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the menuitem **obj**.
j_enable	*void j_enable (int obj);* enables the menuitem **obj**.
j_getlength	*int j_getlength (int obj);* Returns the length of menuitem 's label or text.
j_gettext	*char* j_gettext (int obj , char* str);* returns the menuitem 's text or label.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setshortcut	*void j_setshortcut (int obj , char chr);* Changes the shortcut **chr** of the menuitem .
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the menuitem **obj** to **str**.

Meter

j_meter	*int j_meter (int obj , char* title);* Creates a new pointer–intrument with the specified label **titel**.
j_add	*void j_add (int obj , int cont);* Adds meter **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to meter **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables meter **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the meter **obj**.
j_enable	*void j_enable (int obj);* enables the meter **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to meter **obj**, and returns its event number.
j_getdanger	*void j_getdanger (int obj);* Returns the danger value of meter **obj**.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of meter **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of meter **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of meter **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of meter **obj**.
j_getwidth	*int j_getwidth (int obj);*

Returns the width of meter **obj**.

j_getxpos *int j_getxpos (int obj);*
Returns the current horizontal position of meter **obj** in its parent's coordinate space.

j_getypos *int j_getypos (int obj);*
Returns the current vertical position of meter **obj** in its parent's coordinate space.

j_hide *void j_hide (int obj);*
Hides the meter **obj**.

j_isparent *int j_isparent (int obj , int cont);*
Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.

j_isvisible *int j_isvisible (int obj);*
Returns J_TRUE if **obj** is visible, J_FALSE otherwise.

j_keylistener *int j_keylistener (int obj);*
Adds a new key listener to meter **obj**, and returns its event number.

j_mouselistener *int j_mouselistener (int obj , int kind);*
Adds a new mouse listener to meter **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

j_popupmenu *int j_popupmenu (int obj , char* label);*
Creates a new popupmenu with the specified **label** and returns its event number.

j_print *void j_print (int obj);*
prints the meter .

j_release *void j_release (int obj);*
Releases meter **obj** from its parent component (container).

j_setborderpos *void j_setborderpos (int obj , int pos);*
Moves meter **obj** at a certain position. The outer container needs a border layout manager.

j_setcolorbg *void j_setcolorbg (int obj , int r , int g, , int b);*
Sets the background color to the (**r, g, b**) values.

j_setcolor *void j_setcolor (int obj , int r , int g, , int b);*
Sets the foreground color to the (**r, g, b**) values.

j_setcursor *int j_setcursor (int obj , int cursor);*
Changes the meter 's **obj** cursor to the specified **cursor**.

j_setdanger *void j_setdanger (int obj , int val);*
Changes the danger value of meter **obj** to **val**.

j_setfocus *int j_setfocus (int obj);*

Directs the input focus to meter **obj**.

j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setmax	*int j_setmax (int obj , int val);* Changes the maximum value for the meter to **val**.
j_setmin	*int j_setmin (int obj , int val);* Changes the minimum value for the meter to **val**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the meter **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes meter **obj** to specified **width** and **height**.
j_setvalue	*void j_setvalue (int obj , int val);* Changes the current value of the meter to **val**.
j_show	*void j_show (int obj);* Shows the meter **obj**.

Mouselistener

j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to mouselistener **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_dispose	*void j_dispose (int obj);* Releases the resources of the mouselistener **obj**.
j_getmousebutton	*int j_getmousebutton (int mouselistener);* Returns the latest used mousebutton.
j_getmousex	*int j_getmousex (int mouselistener);* Returns the current horizontal position of the mouse in its parent's coordinate space.
j_getmousey	*int j_getmousey (int mouselistener);* Returns the current vertical position of the mouse in its parent's coordinate space.

Panel

j_panel	*int j_panel (int obj);* Creates a new panel component and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds panel **obj** to container **cont**
j_borderpanel	*int j_borderpanel (int obj , int type);* Creates a new borderpanel component with the style **type** and returns its event number.
j_button	*int j_button (int obj , char* label);* Creates a new button component with the specified **label** and returns its event number.
j_canvas	*int j_canvas (int obj , int width , int height);* Creates a new canvas component with the given **width** and **height** and returns its event number. A canvas can be used for general drawing functions. A canvas generates an event, if its size changes. On error -1 will be returned.
j_checkbox	*int j_checkbox (int obj , char* label);* Creates a new checkbox component with the specified **label** and returns its event number.
j_choice	*int j_choice (int obj);* Creates a new choice component and returns its event number.
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to panel **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables panel **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the panel **obj**.
j_enable	*void j_enable (int obj);* enables the panel **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to panel **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of panel **obj**.
j_getfontheight	*int j_getfontheight (int obj);*

Returns the total pixel height of the actual font of panel **obj**.

j_getheight
int j_getheight (int obj);
Returns the height of panel **obj**.

j_getinheight
int j_getinheight (int cont);
Returns the height of the client size.

j_getinsets
int j_getinsets (int obj , int side);
Returns the width of the specified inset.

j_getinwidth
int j_getinwidth (int cont);
Returns the width of the client size.

j_getlayoutid
int j_getlayoutid (int obj);
Returns the event number of the layoutmanager for containers **obj**.

j_getparentid
int j_getparentid (int obj);
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getparent
int j_getparent (int obj);
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getstringwidth
int j_getstringwidth (int obj , char str);*
Returns the length of **str** of the actual font of panel **obj**.

j_getwidth
int j_getwidth (int obj);
Returns the width of panel **obj**.

j_getxpos
int j_getxpos (int obj);
Returns the current horizontal position of panel **obj** in its parent's coordinate space.

j_getypos
int j_getypos (int obj);
Returns the current vertical position of panel **obj** in its parent's coordinate space.

j_graphicbutton
int j_graphicbutton (int obj , char filename);*
Creates a new graphicbutton component with the image loaded from **filename** and returns its event number.

j_graphiclabel
int j_graphiclabel (int obj , char str);*
Creates a new graphiclabel component with the image loaded from **filename** and returns its event number.

j_hide
void j_hide (int obj);
Hides the panel **obj**.

j_hscrollbar
int j_hscrollbar (int obj);
Creates a new horizontal scrollbar and returns its event number.

j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to panel **obj**, and returns its event number.
j_label	*int j_label (int obj , char* label);* Creates a new label component with the specified **label** and returns its event number.
j_led	*int j_led (int obj , int style , int color);* Creates a new led component with the specified **style** and the specified color **color**.
j_line	*int j_line (int obj , int orient , int style , int length);* Creates a new line component with the specified **length** and returns its event number.
j_list	*int j_list (int obj , int rows);* Creates a new list component with the specified number of **rows** and returns its event number.
j_meter	*int j_meter (int obj , char* title);* Creates a new pointer–intrument with the specified label **titel**.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to panel **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_pack	*void j_pack (int obj);* Resizes panel to the minimal size of contained components.
j_panel	*int j_panel (int obj);* Creates a new panel component and returns its event number.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the panel .
j_progressbar	*int j_progressbar (int obj , int orient);* Creates a new progressbar with the specified **orient**ation.
j_radiogroup	*int j_radiogroup (int obj);* Creates a new radiogroup and returns its event number.
j_releaseall	*void j_releaseall (int obj);* Releases all components from panel **obj**.

j_release	*void j_release (int obj);* Releases panel **obj** from its parent component (container).
j_scrollpane	*int j_scrollpane (int obj);* Creates a new scrollpane component and returns its event number.
j_setalign	*void j_setalign (int obj , int align);* Sets the alignment in panel **obj** to **align**. Needs a flowlayout Manager.
j_setborderlayout	*void j_setborderlayout (int obj);* Adds a borderlayout manager to panel **obj**.
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves panel **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the panel 's **obj** cursor to the specified **cursor**.
j_setfixlayout	*void j_setfixlayout (int obj);* Adds a fixlayout manager to panel **obj** (default layout manager).
j_setflowfill	*void j_setflowfill (int obj , int bool);* Resizes all containing component to the height (width) of panel **obj**. Needs a flowlayout manager.
j_setflowlayout	*void j_setflowlayout (int obj , int align);* Adds a flowlayout manager to panel **obj** with the specified **align**ment.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to panel **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setgridlayout	*void j_setgridlayout (int obj , int row , int col);* Adds a gridlayout manager to panel **obj** with the specified **row**s and **col**umns.

j_sethgap	*void j_sethgap (int obj , int hgap);* Sets the horizontal gap between components to **hgap** Pixel.
j_setinsets	*void j_setinsets (int obj , int top , int bottom , int left , int right);* Set the insets to the specified values.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setnolayout	*void j_setnolayout (int obj);* Removes the current layout manager from panel **obj** .
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the panel **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes panel **obj** to specified **width** and **height**.
j_setvgap	*void j_setvgap (int obj , int vgap);* Sets the vertical gap between components to **hgap** Pixel.
j_sevensegment	*int j_sevensegment (int obj , int color);* Creates a new sevensegment display with the specified color **color**.
j_show	*void j_show (int obj);* Shows the panel **obj**.
j_textarea	*int j_textarea (int obj , int rows , int columns);* Creates a new textarea component with the specified number of **rows columns** and returns its event number.
j_textfield	*int j_textfield (int obj , int columns);* Creates a new textfield component with the specified number of **columns** and returns its event number.
j_vscrollbar	*int j_vscrollbar (int obj);* Creates a new vertical scrollbar and returns its event number.

Popupmenu

j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_checkmenuitem	*int j_checkmenuitem (int obj , char* label);* creates a new checkmenuitem with the specified **label** and returns its event number.
j_disable	*void j_disable (int obj);* Disables popupmenu **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the popupmenu **obj**.
j_enable	*void j_enable (int obj);* enables the popupmenu **obj**.
j_getlength	*int j_getlength (int obj);* Returns the length of popupmenu 's label or text.
j_gettext	*char* j_gettext (int obj , char* str);* returns the popupmenu 's text or label.
j_menuitem	*int j_menuitem (int obj , char* label);* Creates a new menuitem with the specified **label** and returns its event number.
j_seperator	*void j_seperator (int obj);* Adds a separator bar to the popupmenu .
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setshortcut	*void j_setshortcut (int obj , char chr);* Changes the shortcut **chr** of the popupmenu .
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the popupmenu **obj** to **str**.

j_showpopup *void j_showpopup (int obj , int xpos , int ypos);*
Shows the popupmenu at specified Position (**xpos**,**ypos**).

Printer

j_printer	*int j_printer (int frame);* Creates a new object, representing a paper of the printer.
j_cliprect	*void j_cliprect (int obj , int x , int y , int width , int height);* Changes current clipping region to the specified rectangle (**x, y, width, height**).
j_dispose	*void j_dispose (int obj);* Releases the resources of the printer **obj**.
j_drawarc	*void j_drawarc (int obj , int x , int y , int rx , int ry , int arc1 , int arc2);* Draws an unfilled arc from angle **arc1** to angle **arc2** with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.
j_drawcircle	*void j_drawcircle (int obj , int x , int y , int r);* Draws an unfilled circle with center (**x, y**) and radius **x**.
j_drawimage	*void j_drawimage (int obj , int image , int x , int y);* Copies the image, given by its eventnumber **image**, to position (**x, y**).
j_drawimagesource	*void j_drawimagesource (int obj , int x , int y , int w , int h , int* r , int* g , int* b);* Paints an image at Position (**x, y,**) with **w**idth and **h**eight. The red, green and blue values of each pixel are given by the arrays **r, g, b**.
j_drawline	*void j_drawline (int obj , int x1 , int y1 , int x2 , int y2);* Draws a line connecting **(x1,y1)** and **(x2,y2)**.
j_drawoval	*void j_drawoval (int obj , int x , int y , int rx , int ry);* Draws an unfilled oval with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.
j_drawpixel	*void j_drawpixel (int obj , int x , int y);* Draws a pixel at **(x,y)**.
j_drawpolygon	*void j_drawpolygon (int obj , int len , int* x , int* y);* Draws an unfilled polygon based on first **len** elements in **x** and **y**.
j_drawpolyline	*void j_drawpolyline (int obj , int len , int* x , int* y);* Draws a series of line segments based on first **len** elements in **x** and **y**.
j_drawrect	*void j_drawrect (int obj , int x , int y , int width , int height);* Draws an unfilled rectangle from **(x,y)** of size **width** x **height**.
j_drawroundrect	*void j_drawroundrect (int obj , int x , int y , int width , int height , int arcx , int arcy);*

Draws an unfilled rectangle from **(x,y)** of size **width** x **height** with rounded corners. **arcx** and **arcy** specify the radius of rectangle corners.

j_drawscaleddimage *void j_drawscaleddimage (int obj , int image , int sx , int sy , int sw , int sh , int tx , int ty , int tw , int th);*
Copy the contents of the rectangular area defined by **x, y,**) width **sw**, and height **sh** of the **image** to position (**tx, ty**. The area will be scaled to target width **th** and target height **th**.

j_drawstring *void j_drawstring (int obj , int x , int y , char* str);*
Draws text on screen at position (**x,y**).

j_fillarc *void j_fillarc (int obj , int x , int y , int rx , int ry , int arc1 , int arc2);*
Draws an filled arc from angle **arc1** to angle **arc2** with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.

j_fillcircle *void j_fillcircle (int obj , int x , int y , int r);*
Draws an filled circle with center (**x, y**) and radius **x**.

j_filloval *void j_filloval (int obj , int x , int y , int rx , int ry);*
Draws an filled oval with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.

j_fillpolygon *void j_fillpolygon (int obj , int len , int* x , int* y);*
Draws an filled polygon based on first **len** elements in **x** and **y**.

j_fillrect *void j_fillrect (int obj , int x , int y , int width , int height);*
Draws an filled rectangle from **(x,y)** of size **width** x **height**.

j_fillroundrect *void j_fillroundrect (int obj , int x , int y , int width , int height , int arcx , int arcy);*
Draws an filled rectangle from **(x,y)** of size **width** x **height** with rounded corners. **arcx** and **arcy** specify the radius of rectangle corners.

j_print *void j_print (int obj);*
prints the printer .

j_setxor *void j_setxor (int obj , int bool);*
Changes painting mode to XOR mode, if bool = J_TRUE . In this mode, drawing the same object in the same color at the same location twice has no net effect.

j_translate *void j_translate (int obj , int x , int y);*
Moves the origin of drawing operations to (**x, y**).

Progressbar

j_progressbar	*int j_progressbar (int obj , int orient);* Creates a new progressbar with the specified **orient**ation.
j_add	*void j_add (int obj , int cont);* Adds progressbar **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to progressbar **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables progressbar **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the progressbar **obj**.
j_enable	*void j_enable (int obj);* enables the progressbar **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to progressbar **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of progressbar **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of progressbar **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of progressbar **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of progressbar **obj**.
j_getwidth	*int j_getwidth (int obj);* Returns the width of progressbar **obj**.

j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of progressbar **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of progressbar **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the progressbar **obj**.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to progressbar **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to progressbar **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the progressbar .
j_release	*void j_release (int obj);* Releases progressbar **obj** from its parent component (container).
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves progressbar **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the progressbar 's **obj** cursor to the specified **cursor**.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to progressbar **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.

j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the progressbar **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes progressbar **obj** to specified **width** and **height**.
j_show	*void j_show (int obj);* Shows the progressbar **obj**.

Radiobutton

j_radiobutton	*int j_radiobutton (int obj , char* label);* Creates a new radiobutton with the specified **label** and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds radiobutton **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to radiobutton **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables radiobutton **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the radiobutton **obj**.
j_enable	*void j_enable (int obj);* enables the radiobutton **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to radiobutton **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of radiobutton **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of radiobutton **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of radiobutton **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstate	*int j_getstate (int obj);* Returns J_TRUE , if radiobutton is selected, J_FALSE otherwise.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of radiobutton **obj**.

j_gettext	*char* j_gettext (int obj , char* str);* returns the radiobutton 's text or label.
j_getwidth	*int j_getwidth (int obj);* Returns the width of radiobutton **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of radiobutton **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of radiobutton **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the radiobutton **obj**.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to radiobutton **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to radiobutton **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the radiobutton .
j_release	*void j_release (int obj);* Releases radiobutton **obj** from its parent component (container).
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves radiobutton **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the radiobutton 's **obj** cursor to the specified **cursor**.

j_setfocus	*int j_setfocus (int obj);* Directs the input focus to radiobutton **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the radiobutton **obj** to the specified Position (**xpos**,**ypos**).
j_setradiogroup	*int j_setradiogroup (int rbutton, , int rgroup);* Sets radiobuttons **rbutton** group to be the specified radiogroup **rgroup**. If the radiobuttons is already in a different radiogroup, it is first taken out of that group.
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes radiobutton **obj** to specified **width** and **height**.
j_setstate	*void j_setstate (int obj , int bool);* The radiobutton becomes selected, if **bool** is J_TRUE .
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the radiobutton **obj** to **str**.
j_show	*void j_show (int obj);* Shows the radiobutton **obj**.

Sevensegment

j_sevensegment	*int j_sevensegment (int obj , int color);* Creates a new sevensegment display with the specified color **color**.
j_add	*void j_add (int obj , int cont);* Adds sevensegment–component **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to sevensegment–component **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables sevensegment–component **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the sevensegment–component **obj**.
j_enable	*void j_enable (int obj);* enables the sevensegment–component **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to sevensegment–component **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of sevensegment–component **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of sevensegment–component **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of sevensegment–component **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of sevensegment–component **obj**.
j_getwidth	*int j_getwidth (int obj);*

Returns the width of sevensegment–component **obj**.

j_getxpos *int j_getxpos (int obj);*
Returns the current horizontal position of sevensegment–component **obj** in its parent's coordinate space.

j_getypos *int j_getypos (int obj);*
Returns the current vertical position of sevensegment–component **obj** in its parent's coordinate space.

j_hide *void j_hide (int obj);*
Hides the sevensegment–component **obj**.

j_isparent *int j_isparent (int obj , int cont);*
Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.

j_isvisible *int j_isvisible (int obj);*
Returns J_TRUE if **obj** is visible, J_FALSE otherwise.

j_keylistener *int j_keylistener (int obj);*
Adds a new key listener to sevensegment–component **obj**, and returns its event number.

j_mouselistener *int j_mouselistener (int obj , int kind);*
Adds a new mouse listener to sevensegment–component **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

j_popupmenu *int j_popupmenu (int obj , char* label);*
Creates a new popupmenu with the specified **label** and returns its event number.

j_print *void j_print (int obj);*
prints the sevensegment–component .

j_release *void j_release (int obj);*
Releases sevensegment–component **obj** from its parent component (container).

j_setborderpos *void j_setborderpos (int obj , int pos);*
Moves sevensegment–component **obj** at a certain position. The outer container needs a border layout manager.

j_setcolorbg *void j_setcolorbg (int obj , int r , int g, , int b);*
Sets the background color to the (**r, g, b**) values.

j_setcolor *void j_setcolor (int obj , int r , int g, , int b);*
Sets the foreground color to the (**r, g, b**) values.

j_setcursor *int j_setcursor (int obj , int cursor);*
Changes the sevensegment–component 's **obj** cursor to the specified **cursor**.

j_setfocus *int j_setfocus (int obj);*
Directs the input focus to sevensegment–component **obj**.

j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the sevensegment–component **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes sevensegment–component **obj** to specified **width** and **height**.
j_setvalue	*void j_setvalue (int obj , int val);* Changes the current value of the sevensegment–component to **val**.
j_show	*void j_show (int obj);* Shows the sevensegment–component **obj**.

Scrollpane

j_scrollpane	*int j_scrollpane (int obj);* Creates a new scrollpane component and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds scrollpane **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to scrollpane **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables scrollpane **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the scrollpane **obj**.
j_enable	*void j_enable (int obj);* enables the scrollpane **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to scrollpane **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of scrollpane **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of scrollpane **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of scrollpane **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of scrollpane **obj**.
j_getviewportheight	*int j_getviewportheight (int obj);* Returns the height of the scrollpane 's **obj** port (the area that is shown)

j_getviewportwidth	*int j_getviewportwidth (int obj);* Returns the width of the scrollpane 's **obj** port (the area that is shown)
j_getwidth	*int j_getwidth (int obj);* Returns the width of scrollpane **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of scrollpane **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of scrollpane **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the scrollpane **obj**.
j_hscrollbar	*int j_hscrollbar (int obj);* Creates a new horizontal scrollbar and returns its event number.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to scrollpane **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to scrollpane **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the scrollpane .
j_release	*void j_release (int obj);* Releases scrollpane **obj** from its parent component (container).
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves scrollpane **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.

j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the scrollpane 's **obj** cursor to the specified **cursor**.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to scrollpane **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the scrollpane **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes scrollpane **obj** to specified **width** and **height**.
j_show	*void j_show (int obj);* Shows the scrollpane **obj**.
j_vscrollbar	*int j_vscrollbar (int obj);* Creates a new vertical scrollbar and returns its event number.

Textarea

j_textarea	*int j_textarea (int obj , int rows , int columns);* Creates a new textarea component with the specified number of **rows columns** and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds textarea **obj** to container **cont**
j_appendtext	*void j_appendtext (int obj , char* text);* Appends the given **text** to the **obj** current text.
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to textarea **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_delete	*void j_delete (int obj , int start , int end);* Delets text from starting position **start** to ending position **end**.
j_disable	*void j_disable (int obj);* Disables textarea **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the textarea **obj**.
j_enable	*void j_enable (int obj);* enables the textarea **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to textarea **obj**, and returns its event number.
j_getcolumns	*void j_getcolumns (int obj);* Gets the number of columns in **obj**.
j_getcurpos	*int j_getcurpos (int obj);* Returns the position, in characters, of the text cursor.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of textarea **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of textarea **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of textarea **obj**.
j_getlength	*int j_getlength (int obj);*

Returns the length of textarea 's label or text.

j_getparentid *int j_getparentid (int obj);*
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getparent *int j_getparent (int obj);*
Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

j_getrows *void j_getrows (int obj);*
Gets the number of rows in **obj**.

j_getselend *int j_getselend (int obj);*
Returns the ending position of any selected text.

j_getselstart *int j_getselstart (int obj);*
Returns the initial position of any selected text.

j_getseltext *char* j_getseltext (int obj , char* text);*
Returns the currently selected text of textarea **obj**.

j_getstringwidth *int j_getstringwidth (int obj , char* str);*
Returns the length of **str** of the actual font of textarea **obj**.

j_gettext *char* j_gettext (int obj , char* str);*
returns the textarea 's text or label.

j_getwidth *int j_getwidth (int obj);*
Returns the width of textarea **obj**.

j_getxpos *int j_getxpos (int obj);*
Returns the current horizontal position of textarea **obj** in its parent's coordinate space.

j_getypos *int j_getypos (int obj);*
Returns the current vertical position of textarea **obj** in its parent's coordinate space.

j_hide *void j_hide (int obj);*
Hides the textarea **obj**.

j_inserttext *void j_inserttext (int obj , char* text , int pos);*
Places additional text within the textarea at the given position **pos**.

j_isparent *int j_isparent (int obj , int cont);*
Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.

j_isvisible *int j_isvisible (int obj);*
Returns J_TRUE if **obj** is visible, J_FALSE otherwise.

j_keylistener *int j_keylistener (int obj);*
Adds a new key listener to textarea **obj**, and returns its event number.

j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to textarea **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the textarea .
j_release	*void j_release (int obj);* Releases textarea **obj** from its parent component (container).
j_replacetext	*void j_replacetext (int obj , char* text , int start , int end);* Replaces the text from starting position **start** to ending position **end** with the given **text**.
j_selectall	*void j_selectall (int obj);* Selects all the text in the textarea .
j_selecttext	*void j_selecttext (int obj , int start , int end);* Selects text from starting position **start** to ending position **end**.
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves textarea **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcolumns	*void j_setcolumns (int obj , int columns);* Sets the number of columns for **obj** to **columns**.
j_setcurpos	*void j_setcurpos (int obj , int pos);* Change the location of the text cursor to the specified position **pos**.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the textarea 's **obj** cursor to the specified **cursor**.
j_seteditable	*void j_seteditable (int obj , int bool);* Allows to make the textarea editable (**bool**=J_TRUE) or read-only (**bool**=J_FALSE).
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to textarea **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.

j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the textarea **obj** to the specified Position (**xpos**,**ypos**).
j_setrows	*void j_setrows (int obj , int rows);* Sets the number of rows for **obj** to **rows**.
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes textarea **obj** to specified **width** and **height**.
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the textarea **obj** to **str**.
j_show	*void j_show (int obj);* Shows the textarea **obj**.

Textfield

j_textfield	*int j_textfield (int obj , int columns);* Creates a new textfield component with the specified number of **columns** and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds textfield **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to textfield **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables textfield **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the textfield **obj**.
j_enable	*void j_enable (int obj);* enables the textfield **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to textfield **obj**, and returns its event number.
j_getcolumns	*void j_getcolumns (int obj);* Gets the number of columns in **obj**.
j_getcurpos	*int j_getcurpos (int obj);* Returns the position, in characters, of the text cursor.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of textfield **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of textfield **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of textfield **obj**.
j_getlength	*int j_getlength (int obj);* Returns the length of textfield 's label or text.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.

j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getselend	*int j_getselend (int obj);* Returns the ending position of any selected text.
j_getselstart	*int j_getselstart (int obj);* Returns the initial position of any selected text.
j_getseltext	*char* j_getseltext (int obj , char* text);* Returns the currently selected text of textfield **obj**.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of textfield **obj**.
j_gettext	*char* j_gettext (int obj , char* str);* returns the textfield 's text or label.
j_getwidth	*int j_getwidth (int obj);* Returns the width of textfield **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of textfield **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of textfield **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the textfield **obj**.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to textfield **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to textfield **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the textfield .

j_release	*void j_release (int obj);*	Releases textfield **obj** from its parent component (container).
j_selectall	*void j_selectall (int obj);*	Selects all the text in the textfield .
j_selecttext	*void j_selecttext (int obj , int start , int end);*	Selects text from starting position **start** to ending position **end**.
j_setborderpos	*void j_setborderpos (int obj , int pos);*	Moves textfield **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);*	Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);*	Sets the foreground color to the (**r, g, b**) values.
j_setcolumns	*void j_setcolumns (int obj , int columns);*	Sets the number of columns for **obj** to **columns**.
j_setcurpos	*void j_setcurpos (int obj , int pos);*	Change the location of the text cursor to the specified position **pos**.
j_setcursor	*int j_setcursor (int obj , int cursor);*	Changes the textfield 's **obj** cursor to the specified **cursor**.
j_setechochar	*void j_setechochar (int obj , char chr);*	Changes the character **chr** that is used to echo all user input in the textfield .
j_seteditable	*void j_seteditable (int obj , int bool);*	Allows to make the textfield editable (**bool**=J_TRUE) or read-only (**bool**=J_FALSE).
j_setfocus	*int j_setfocus (int obj);*	Directs the input focus to textfield **obj**.
j_setfontname	*void j_setfontname (int obj , int name);*	Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);*	Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);*	Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);*	Changes the font to the given **style**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);*	Sets the background color to a predefined **color**.

j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the textfield **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes textfield **obj** to specified **width** and **height**.
j_settext	*void j_settext (int obj , char* str);* Sets the content or the label of the textfield **obj** to **str**.
j_show	*void j_show (int obj);* Shows the textfield **obj**.

Vscrollbar

j_vscrollbar	*int j_vscrollbar (int obj);* Creates a new vertical scrollbar and returns its event number.
j_add	*void j_add (int obj , int cont);* Adds vscrollbar **obj** to container **cont**
j_componentlistener	*int j_componentlistener (int obj , int kind);* Adds a new componentlistener to vscrollbar **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_disable	*void j_disable (int obj);* Disables vscrollbar **obj** so that it is unresponsive to user interactions
j_dispose	*void j_dispose (int obj);* Releases the resources of the vscrollbar **obj**.
j_enable	*void j_enable (int obj);* enables the vscrollbar **obj**.
j_focuslistener	*int j_focuslistener (int obj);* Adds a new focus listener to vscrollbar **obj**, and returns its event number.
j_getfontascent	*int j_getfontascent (int obj);* Returns the ascent (space above the baseline) of the actual font of vscrollbar **obj**.
j_getfontheight	*int j_getfontheight (int obj);* Returns the total pixel height of the actual font of vscrollbar **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of vscrollbar **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of vscrollbar **obj**.
j_getvalue	*int j_getvalue (int obj);* Returns the current setting of the scrollbar.

j_getwidth	*int j_getwidth (int obj);* Returns the width of vscrollbar **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of vscrollbar **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of vscrollbar **obj** in its parent's coordinate space.
j_hide	*void j_hide (int obj);* Hides the vscrollbar **obj**.
j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to vscrollbar **obj**, and returns its event number.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to vscrollbar **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the vscrollbar .
j_release	*void j_release (int obj);* Releases vscrollbar **obj** from its parent component (container).
j_setblockinc	*int j_setblockinc (int obj , int val);* Changes the block increment amount for the vscrollbar to **val**.
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves vscrollbar **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the vscrollbar 's **obj** cursor to the specified **cursor**.

j_setfocus	*int j_setfocus (int obj);* Directs the input focus to vscrollbar **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setmax	*int j_setmax (int obj , int val);* Changes the maximum value for the vscrollbar to **val**.
j_setmin	*int j_setmin (int obj , int val);* Changes the minimum value for the vscrollbar to **val**.
j_setnamedcolorbg	*void j_setnamedcolorbg (int obj , int color);* Sets the background color to a predefined **color**.
j_setnamedcolor	*void j_setnamedcolor (int obj , int color);* Sets the foreground color to a predefined **color**.
j_setpos	*void j_setpos (int obj , int xpos , int ypos);* Relocates the vscrollbar **obj** to the specified Position (**xpos**,**ypos**).
j_setsize	*void j_setsize (int obj , int width , int height);* Resizes vscrollbar **obj** to specified **width** and **height**.
j_setslidesize	*int j_setslidesize (int obj , int val);* Changes the slide size to **val**.
j_setunitinc	*int j_setunitinc (int obj , int val);* Changes the unit increment amount for the vscrollbar to **val**
j_setvalue	*void j_setvalue (int obj , int val);* Changes the current value of the vscrollbar to **val**.
j_show	*void j_show (int obj);* Shows the vscrollbar **obj**.

Window

j_window *int j_window (int obj);*
Creates a new simple window and returns its event number.

j_add *void j_add (int obj , int cont);*
Adds window **obj** to container **cont**

j_borderpanel *int j_borderpanel (int obj , int type);*
Creates a new borderpanel component with the style **type** and returns its event number.

j_button *int j_button (int obj , char* label);*
Creates a new button component with the specified **label** and returns its event number.

j_canvas *int j_canvas (int obj , int width , int height);*
Creates a new canvas component with the given **width** and **height** and returns its event number. A canvas can be used for general drawing functions. A canvas generates an event, if its size changes. On error -1 will be returned.

j_checkbox *int j_checkbox (int obj , char* label);*
Creates a new checkbox component with the specified **label** and returns its event number.

j_choice *int j_choice (int obj);*
Creates a new choice component and returns its event number.

j_componentlistener *int j_componentlistener (int obj , int kind);*
Adds a new componentlistener to window **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

j_disable *void j_disable (int obj);*
Disables window **obj** so that it is unresponsive to user interactions

j_dispose *void j_dispose (int obj);*
Releases the resources of the window **obj**.

j_enable *void j_enable (int obj);*
enables the window **obj**.

j_focuslistener *int j_focuslistener (int obj);*
Adds a new focus listener to window **obj**, and returns its event number.

j_getfontascent *int j_getfontascent (int obj);*
Returns the ascent (space above the baseline) of the actual font of window **obj**.

j_getfontheight *int j_getfontheight (int obj);*

	Returns the total pixel height of the actual font of window **obj**.
j_getheight	*int j_getheight (int obj);* Returns the height of window **obj**.
j_getinheight	*int j_getinheight (int cont);* Returns the height of the client size.
j_getinsets	*int j_getinsets (int obj , int side);* Returns the width of the specified inset.
j_getinwidth	*int j_getinwidth (int cont);* Returns the width of the client size.
j_getlayoutid	*int j_getlayoutid (int obj);* Returns the event number of the layoutmanager for containers **obj**.
j_getparentid	*int j_getparentid (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getparent	*int j_getparent (int obj);* Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.
j_getstringwidth	*int j_getstringwidth (int obj , char* str);* Returns the length of **str** of the actual font of window **obj**.
j_getwidth	*int j_getwidth (int obj);* Returns the width of window **obj**.
j_getxpos	*int j_getxpos (int obj);* Returns the current horizontal position of window **obj** in its parent's coordinate space.
j_getypos	*int j_getypos (int obj);* Returns the current vertical position of window **obj** in its parent's coordinate space.
j_graphicbutton	*int j_graphicbutton (int obj , char* filename);* Creates a new graphicbutton component with the image loaded from **filename** and returns its event number.
j_graphiclabel	*int j_graphiclabel (int obj , char* str);* Creates a new graphiclabel component with the image loaded from **filename** and returns its event number.
j_hide	*void j_hide (int obj);* Hides the window **obj**.
j_hscrollbar	*int j_hscrollbar (int obj);* Creates a new horizontal scrollbar and returns its event number.

j_isparent	*int j_isparent (int obj , int cont);* Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
j_isvisible	*int j_isvisible (int obj);* Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
j_keylistener	*int j_keylistener (int obj);* Adds a new key listener to window **obj**, and returns its event number.
j_label	*int j_label (int obj , char* label);* Creates a new label component with the specified **label** and returns its event number.
j_led	*int j_led (int obj , int style , int color);* Creates a new led component with the specified **style** and the specified color **color**.
j_line	*int j_line (int obj , int orient , int style , int length);* Creates a new line component with the specified **length** and returns its event number.
j_list	*int j_list (int obj , int rows);* Creates a new list component with the specified number of **rows** and returns its event number.
j_meter	*int j_meter (int obj , char* title);* Creates a new pointer–intrument with the specified label **titel**.
j_mouselistener	*int j_mouselistener (int obj , int kind);* Adds a new mouse listener to window **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.
j_pack	*void j_pack (int obj);* Resizes window to the minimal size of contained components.
j_panel	*int j_panel (int obj);* Creates a new panel component and returns its event number.
j_popupmenu	*int j_popupmenu (int obj , char* label);* Creates a new popupmenu with the specified **label** and returns its event number.
j_print	*void j_print (int obj);* prints the window .
j_progressbar	*int j_progressbar (int obj , int orient);* Creates a new progressbar with the specified **orient**ation.
j_radiogroup	*int j_radiogroup (int obj);* Creates a new radiogroup and returns its event number.
j_releaseall	*void j_releaseall (int obj);* Releases all components from window **obj**.

j_release	*void j_release (int obj);* Releases window **obj** from its parent component (container).
j_scrollpane	*int j_scrollpane (int obj);* Creates a new scrollpane component and returns its event number.
j_setalign	*void j_setalign (int obj , int align);* Sets the alignment in window **obj** to **align**. Needs a flowlayout Manager.
j_setborderlayout	*void j_setborderlayout (int obj);* Adds a borderlayout manager to window **obj**.
j_setborderpos	*void j_setborderpos (int obj , int pos);* Moves window **obj** at a certain position. The outer container needs a border layout manager.
j_setcolorbg	*void j_setcolorbg (int obj , int r , int g, , int b);* Sets the background color to the (**r, g, b**) values.
j_setcolor	*void j_setcolor (int obj , int r , int g, , int b);* Sets the foreground color to the (**r, g, b**) values.
j_setcursor	*int j_setcursor (int obj , int cursor);* Changes the window 's **obj** cursor to the specified **cursor**.
j_setfixlayout	*void j_setfixlayout (int obj);* Adds a fixlayout manager to window **obj** (default layout manager).
j_setflowfill	*void j_setflowfill (int obj , int bool);* Resizes all containing component to the height (width) of window **obj**. Needs a flowlayout manager.
j_setflowlayout	*void j_setflowlayout (int obj , int align);* Adds a flowlayout manager to window **obj** with the specified **align**ment.
j_setfocus	*int j_setfocus (int obj);* Directs the input focus to window **obj**.
j_setfontname	*void j_setfontname (int obj , int name);* Changes the font to the given **name**.
j_setfont	*void j_setfont (int obj , int name , int style , int size);* Changes the font to the given characteristics **name**, **style** and **size**.
j_setfontsize	*void j_setfontsize (int obj , int size);* Changes the font to the given **size**.
j_setfontstyle	*void j_setfontstyle (int obj , int style);* Changes the font to the given **style**.
j_setgridlayout	*void j_setgridlayout (int obj , int row , int col);*

Adds a gridlayout manager to window **obj** with the specified **row**s and **col**umns.

j_sethgap *void j_sethgap (int obj , int hgap);*
Sets the horizontal gap between components to **hgap** Pixel.

j_setinsets *void j_setinsets (int obj , int top , int bottom , int left , int right);*
Set the insets to the specified values.

j_setnamedcolorbg *void j_setnamedcolorbg (int obj , int color);*
Sets the background color to a predefined **color**.

j_setnamedcolor *void j_setnamedcolor (int obj , int color);*
Sets the foreground color to a predefined **color**.

j_setnolayout *void j_setnolayout (int obj);*
Removes the current layout manager from window **obj** .

j_setpos *void j_setpos (int obj , int xpos , int ypos);*
Relocates the window **obj** to the specified Position (**xpos**,**ypos**).

j_setsize *void j_setsize (int obj , int width , int height);*
Resizes window **obj** to specified **width** and **height**.

j_setvgap *void j_setvgap (int obj , int vgap);*
Sets the vertical gap between components to **hgap** Pixel.

j_sevensegment *int j_sevensegment (int obj , int color);*
Creates a new sevensegment display with the specified color **color**.

j_show *void j_show (int obj);*
Shows the window **obj**.

j_textarea *int j_textarea (int obj , int rows , int columns);*
Creates a new textarea component with the specified number of **rows columns** and returns its event number.

j_textfield *int j_textfield (int obj , int columns);*
Creates a new textfield component with the specified number of **columns** and returns its event number.

j_vscrollbar *int j_vscrollbar (int obj);*
Creates a new vertical scrollbar and returns its event number.

j_windowlistener *int j_windowlistener (int window , int kind);*
Adds a new windowlistener to **obj**, and returns its event number. An event occures, if the user action is of kind **kind**.

Kapitel 2

Functions

additem

Synopsis void **j_additem** (int obj , char* str);

Arguments

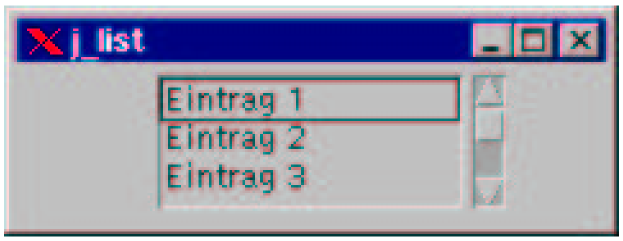
	obj	int
	str	char*

Description adds a new item containing **str** to component **obj**.

Targets List, Choice

Example

```
:
list = j_list(frame,3);
j_additem(list,"Eintrag 1");
j_additem(list,"Eintrag 2");
:
```



add

Synopsis	void **j_add** (int obj , int cont);
Arguments	obj int cont int
Description	Adds component **obj** to container **cont**
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

alertbox

Synopsis

void **j_alertbox** (int obj , char* title , char* text , char* button);

Arguments

obj	int
title	char*
text	char*
button	char*

Description

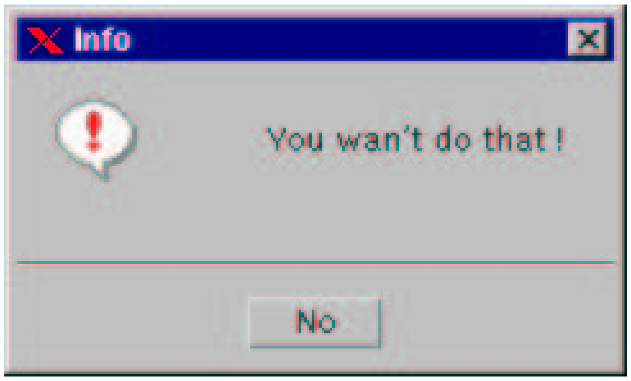
Shows a alertbox with the specified **title**, **text** and **button**. Alertboxes are modal dialogs, the application is blocked until the button or the closeicon is clicked. The return value is 0 if the closeicon is clicked and 1 if the buttons is used.

Targets

Frame

Example

```
:
retval = j_alertbox(frame,"Info","You wan’t do that !","   No   ");
:
```



appendtext

Synopsis	void **j_appendtext** (int obj , char* text);
Arguments	obj int text char*
Description	Appends the given **text** to the **obj** current text.
Targets	Textarea

beep

Synopsis void **j_beep** ( );

Description Emits an audio beep.

borderpanel

Synopsis int **j_borderpanel** (int obj , int type);

Arguments

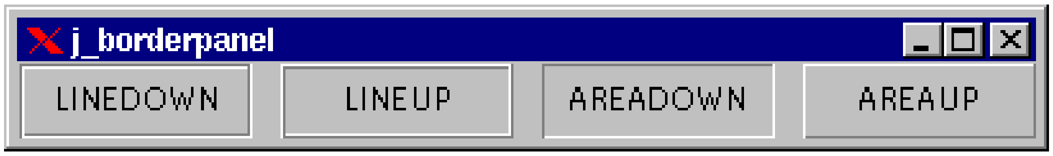
obj	int
type	int

Description Creates a new borderpanel component with the style **type** and returns its event number.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
j_setgridlayout(frame,1,4);
p1 = j_borderpanel(frame,J_LINEDOWN);
p2 = j_borderpanel(frame,J_LINEUP);
p3 = j_borderpanel(frame,J_AREADOWN);
p4 = j_borderpanel(frame,J_AREAUP);
:
```



button

Synopsis	int **j_button** (int obj , char* label);

Arguments

obj	int
label	char*

Description

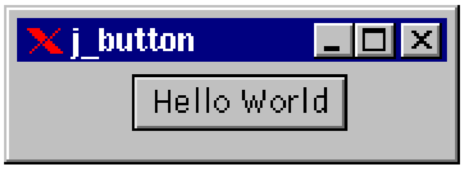
Creates a new button component with the specified **label** and returns its event number.

Targets

Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame  = j_frame("j_button");
button = j_button(frame,"Hello World");
:
```



canvas

Synopsis int **j_canvas** (int obj , int width , int height);

Arguments

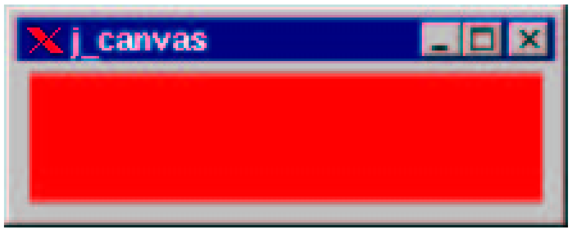
obj	int
width	int
height	int

Description Creates a new canvas component with the given **width** and **height** and returns its event number. A canvas can be used for general drawing functions. A canvas generates an event, if its size changes. On error −1 will be returned.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

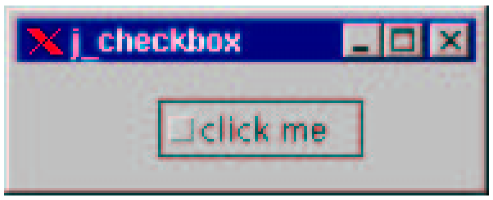
```
:
canvas = j_canvas(frame,200,50);
j_setnamedcolorbg(canvas,J_RED);
:
```



checkbox

Synopsis	int **j_checkbox** (int obj , char* label);
Arguments	obj int label char*
Description	Creates a new checkbox component with the specified **label** and returns its event number.
Targets	Panel, Borderpanel, Window, Dialog, Frame
Example	

```
:
frame    = j_frame("j_checkbox");
checkbox = j_checkbox(frame,"click me");
:
```



checkmenuitem

Synopsis int **j_checkmenuitem** (int obj , char* label);

Arguments

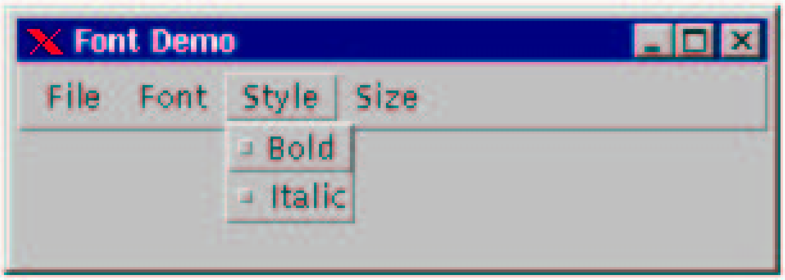
obj	int
label	char*

Description creates a new checkmenuitem with the specified **label** and returns its event number.

Targets Menu, Popupmenu, Helpmenu

Example

```
:
menubar = j_menubar(frame)
:
style = j_menu(menubar,"Style");
bold  = j_checkmenuitem(style,"Bold");
italic= j_checkmenuitem(style,"Italic");
:
```



choicebox2

Synopsis

void **j_choicebox2** (int obj , char* title , char* text , char* button1 , char* button2);

Arguments

obj	int
title	char*
text	char*
button1	char*
button2	char*

Description

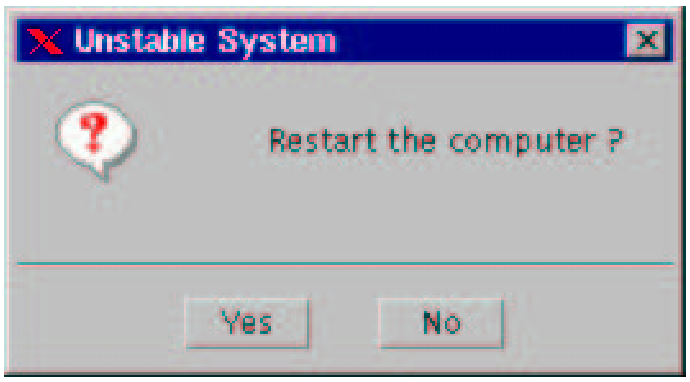
Shows a choicebox with the specified **title**, **text** and two buttons. Choiceboxes are modal dialogs, the application is blocked until a button or the closeicon is clicked. The focus is set to the first button. The return value is 0 if the closeicon is clicked, 1 for the first button and 2 for the second one.

Targets

Frame

Example

```
:
retval = j_choicebox2(frame,"Unstable System","Restart the computer ?",
                             "  Yes  ","No");
:
```



choicebox3

Synopsis

void **j_choicebox3** (int obj , char* title , char* text , char* button1 , char* button2 , char* button3);

Arguments

obj	int
title	char*
text	char*
button1	char*
button2	char*
button3	char*

Description

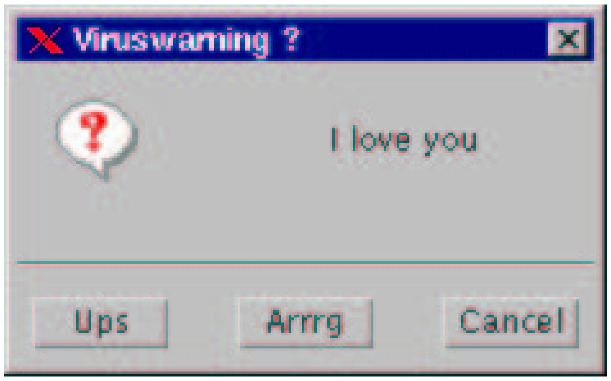
Shows a choicebox with the specified **title**, **text** and three buttons. Choiceboxes are modal dialogs, the application is blocked until a button or the closeicon is clicked. The focus is set to the first button. The return value is 0 if the closeicon is clicked, 1 for the first button, 2 for the second and 3 for the third one.

Targets

Frame

Example

```
:
retval = j_choicebox2(frame,"Viruswarning ?","I love you",
                            "Ups","Arrrg","Cancel");
:
```



choice

Synopsis int **j_choice** (int obj);

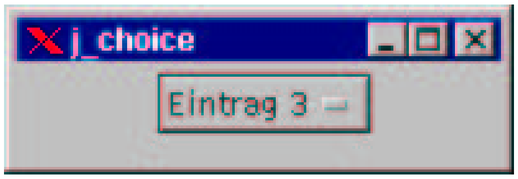
Arguments obj int

Description Creates a new choice component and returns its event number.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
choice = j_choice(frame);
j_additem(choice,"Eintrag 1");
j_additem(choice,"Eintrag 2");
:
```



cliprect

Synopsis	void **j_cliprect** (int obj , int x , int y , int width , int height);

Arguments

obj	int
x	int
y	int
width	int
height	int

Description: Changes current clipping region to the specified rectangle (**x, y, width, height**).

Targets: Canvas, Image, Printer

componentlistener

Synopsis	int **j_componentlistener** (int obj , int kind);

Arguments	obj	int
	kind	int

Description

Adds a new componentlistener to component **obj**, and returns its event number. An event occures, if the user action is of kind **kind**. Posible values for **kind**:

- J_RESIZED : An event occures when the component has been resized.
- J_HIDDEN : An event occures when the component has been hidden.
- J_SHOWN : An event occures when the component has been shown.

Targets

Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

connect

Synopsis int **j_connect** (char* hostname);

Arguments hostname char*

Description Connects a running japi kernel on host **hostname**.

Example

```
:
if( ! j_connect("atan.japi.de"))

or

if( ! j_connect("127.0.0.1"))
:
```

delete

Synopsis	void **j_delete** (int obj , int start , int end);
Arguments	obj int start int end int
Description	Delets text from starting position **start** to ending position **end**.
Targets	Textarea

deselect

Synopsis	int **j_deselect** (int obj , int item);
Arguments	obj int item int
Description	Deselects the item at the designated position **item**, if selected.
Targets	List

dialog

Synopsis

int **j_dialog** (int obj , char* label);

Arguments

obj	int
label	char*

Description

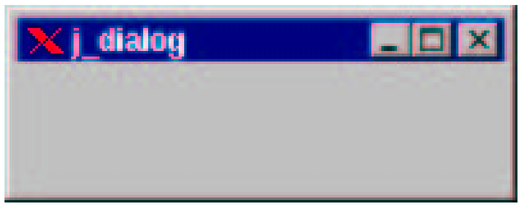
Creates a new dialog window with the specified **label** and returns its event number.

Targets

Frame

Example

```
:
dialog = j_dialog(frame,"j_dialog");
j_setsize(dialog,200,80);
j_show(dialog);
:
```



disable

Synopsis	void **j_disable** (int obj);
Arguments	obj int
Description	Disables component **obj** so that it is unresponsive to user interactions
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu

dispose

Synopsis	void **j_dispose** (int obj);
Arguments	obj int
Description	Releases the resources of the component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Canvas, Image, Printer, Keylistener, Focus-listener, Mouselistener

drawarc

Synopsis

void **j_drawarc** (int obj , int x , int y , int rx , int ry , int arc1 , int arc2);

Arguments

obj	int
x	int
y	int
rx	int
ry	int
arc1	int
arc2	int

Description

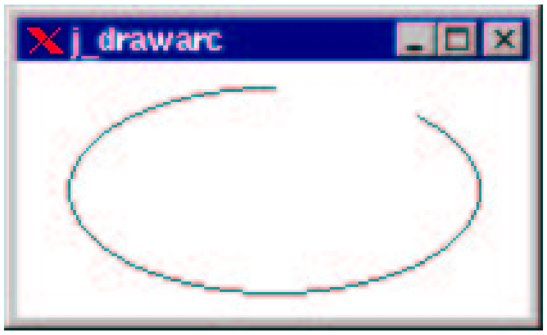
Draws an unfilled arc from angle **arc1** to angle **arc2** with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.

Targets

Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,200,100);
j_drawarc(canvas,100,50,80,40,45,-270);
:
```



drawcircle

Synopsis

void **j_drawcircle** (int obj , int x , int y , int r);

Arguments

obj	int
x	int
y	int
r	int

Description

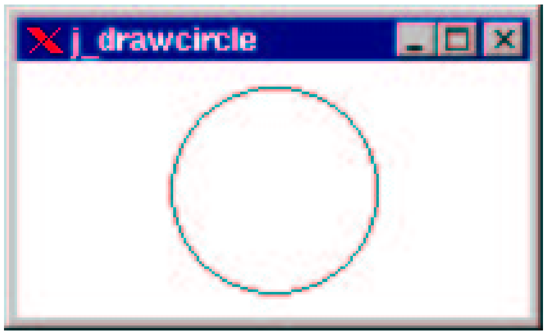
Draws an unfilled circle with center (**x, y**) and radius **x**.

Targets

Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,200,100);
j_drawcircle(canvas,100,50,40);
:
```



drawimagesource

Synopsis	void **j_drawimagesource** (int obj , int x , int y , int w , int h , int* r , int* g , int* b);
Arguments	obj int x int y int w int h int r int* g int* b int*
Description	Paints an image at Position (**x, y,**) with **w**idth and **h**eight. The red, green and blue values of each pixel are given by the arrays **r, g, b**.
Targets	Canvas, Image, Printer

drawimage

Synopsis	void **j_drawimage** (int obj , int image , int x , int y);
Arguments	obj int image int x int y int
Description	Copies the image, given by its eventnumber **image**, to position (**x, y**).
Targets	Canvas, Image, Printer

drawline

Synopsis void **j_drawline** (int obj , int x1 , int y1 , int x2 , int y2);

Arguments

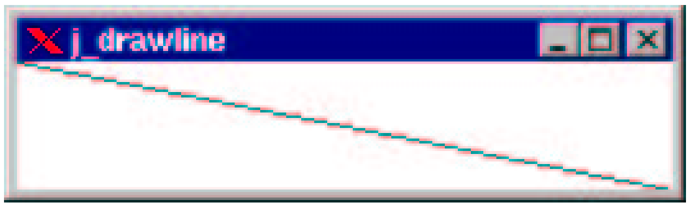
obj	int
x1	int
y1	int
x2	int
y2	int

Description Draws a line connecting **(x1,y1)** and **(x2,y2)**.

Targets Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,256,50);
j_drawline(canvas,0,0,256,50);
:
```



drawoval

Synopsis void **j_drawoval** (int obj , int x , int y , int rx , int ry);

Arguments

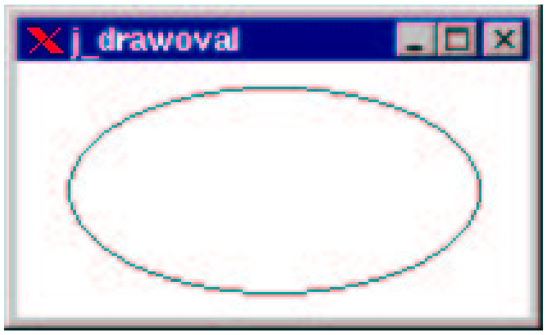
obj	int
x	int
y	int
rx	int
ry	int

Description Draws an unfilled oval with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.

Targets Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,200,100);
j_drawoval(canvas,100,50,80,40);
:
```



drawpixel

Synopsis void **j_drawpixel** (int obj , int x , int y);

Arguments

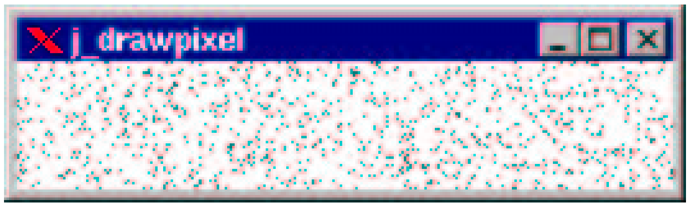
obj	int
x	int
y	int

Description Draws a pixel at **(x,y)**.

Targets Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,256,50);
for(i=0;i<1000;i++)
    j_drawpixel(canvas,j_random()%256,,j_random()%256);
:
```



drawpolygon

Synopsis	void **j_drawpolygon** (int obj , int len , int* x , int* y);

Arguments

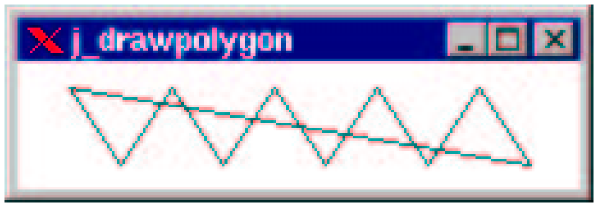
obj	int
len	int
x	int*
y	int*

Description Draws an unfilled polygon based on first **len** elements in **x** and **y**.

Targets Canvas, Image, Printer

Example

```
:
int x[10]={20,40,60,80,100,120,140,160,180,200};
int y[10]={10,40,10,40,10,40,10,40,10,40};
canvas = j_canvas(frame,256,50);
j_drawpolygon(canvas,10,x,y);
:
```



drawpolyline

Synopsis void **j_drawpolyline** (int obj , int len , int* x , int* y);

Arguments

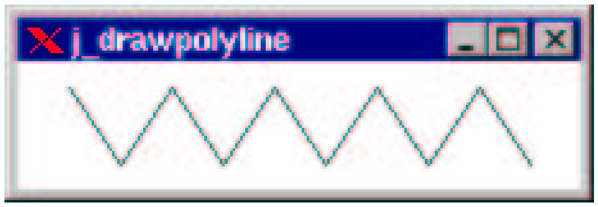
obj	int
len	int
x	int*
y	int*

Description Draws a series of line segments based on first **len** elements in **x** and **y**.

Targets Canvas, Image, Printer

Example

```
:
int x[10]={20,40,60,80,100,120,140,160,180,200};
int y[10]={10,40,10,40,10,40,10,40,10,40};
canvas = j_canvas(frame,256,50);
j_drawpolyline(canvas,10,x,y);
:
```



drawrect

Synopsis

void **j_drawrect** (int obj , int x , int y , int width , int height);

Arguments

obj	int
x	int
y	int
width	int
height	int

Description

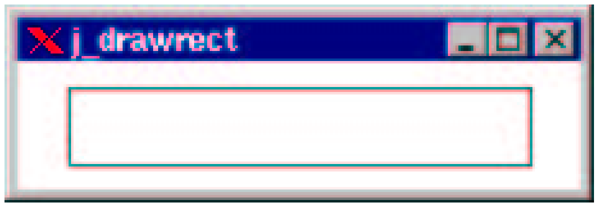
Draws an unfilled rectangle from **(x,y)** of size **width** x **height**.

Targets

Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,220,50);
j_drawrect(canvas,20,10,180,30);
:
```



drawroundrect

Synopsis

void **j_drawroundrect** (int obj , int x , int y , int width , int height , int arcx , int arcy);

Arguments

obj	int
x	int
y	int
width	int
height	int
arcx	int
arcy	int

Description

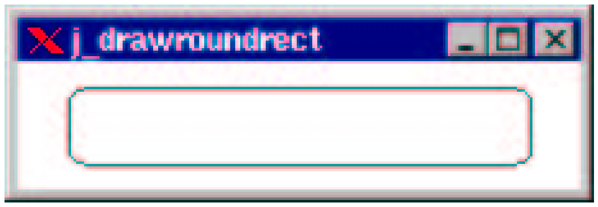
Draws an unfilled rectangle from **(x,y)** of size **width** x **height** with rounded corners. **arcx** and **arcy** specify the radius of rectangle corners.

Targets

Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,220,50);
j_drawroundrect(canvas,20,10,180,30,10,5);
:
```



drawscaleddimage

Synopsis

void **j_drawscaleddimage** (int obj , int image , int sx , int sy , int sw , int sh , int tx , int ty , int tw , int th);

Arguments

obj	int
image	int
sx	int
sy	int
sw	int
sh	int
tx	int
ty	int
tw	int
th	int

Description

Copy the contents of the rectangular area defined by **x, y,**) width **sw**, and height **sh** of the **image** to position (**tx, ty**. The area will be scaled to target width **th** and target height **th**.

Targets

Canvas, Image, Printer

drawstring

Synopsis void **j_drawstring** (int obj , int x , int y , char* str);

Arguments

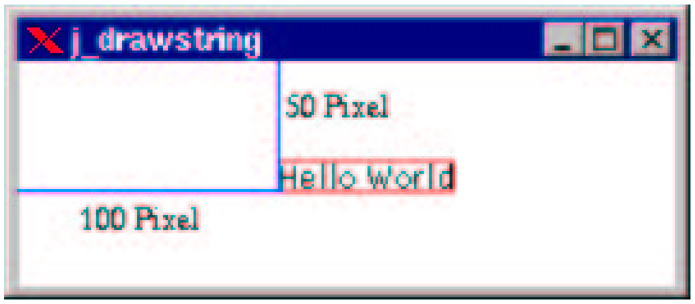
obj	int
x	int
y	int
str	char*

Description Draws text on screen at position (**x,y**).

Targets Canvas, Image, Printer

Example

```
:
j_drawstring(canvas,100,50,"Hello World");
:
```



enable

Synopsis	void **j_enable** (int obj);
Arguments	obj int
Description	enables the component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu

filedialog

Synopsis char* **j_filedialog** (int frame , char* title , char* directory , char* filename);

Arguments

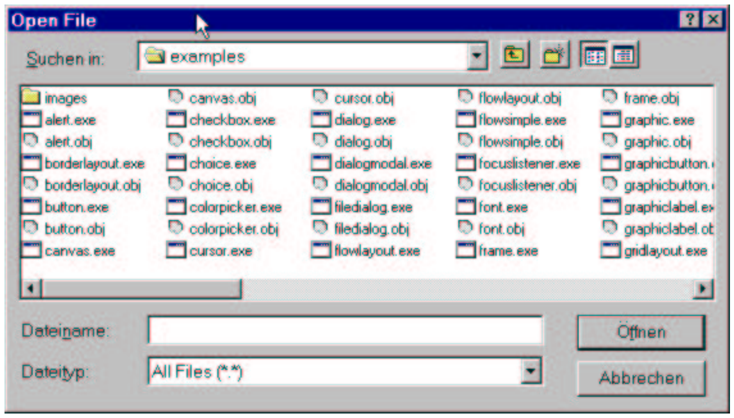
frame	int
title	char*
directory	char*
filename	char*

Description Opens a filedialog box in the specified **directory** with the specified **title** and returns the selected **filename**. If **title** contains "/S" the SAVE–filedialog will be called. The substring "/S" will be removed.

Targets Frame

Example

```
:
filename = j_filedialog(frame,"Save/S File","..",filename);
:
```



Suchen in: examples
Dateityp: All Files (*.*)

fileselect

Synopsis: char* **j_fileselect** (int frame , char* title , char* filter , char* filename);

Arguments:

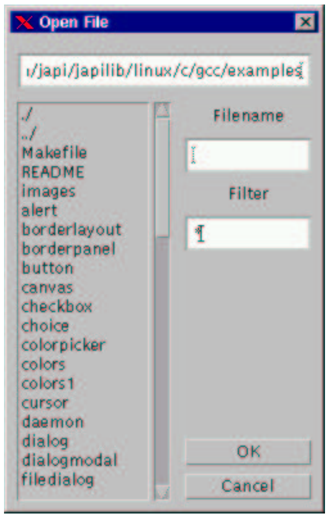
frame	int
title	char*
filter	char*
filename	char*

Description: Opens a fileslector box with the preselected **filename** and the specified **title** and returns the selected **filename**. **filter** specifies the Filename Filter. A Fileselector can be used with output redirections via j_connect();

Targets: Frame

Example

```
:
filename = j_fileselect(frame,"Open File","*",filename);
:
```



fillarc

Synopsis

void **j_fillarc** (int obj , int x , int y , int rx , int ry , int arc1 , int arc2);

Arguments

obj	int
x	int
y	int
rx	int
ry	int
arc1	int
arc2	int

Description

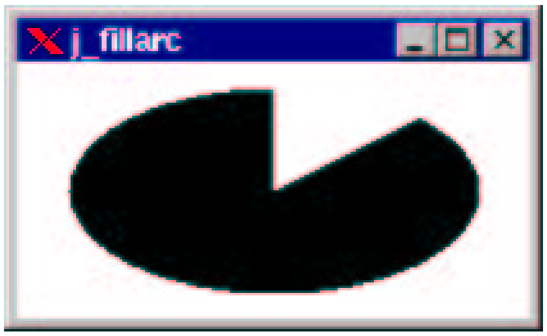
Draws an filled arc from angle **arc1** to angle **arc2** with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.

Targets

Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,200,100);
j_fillarc(canvas,100,50,80,40,45,-270);
:
```



fillcircle

Synopsis void **j_fillcircle** (int obj , int x , int y , int r);

Arguments

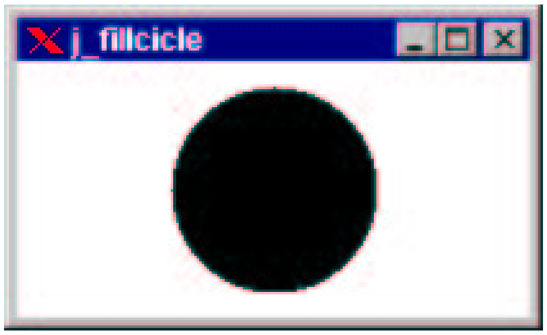
obj	int
x	int
y	int
r	int

Description Draws an filled circle with center (**x, y**) and radius **x**.

Targets Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,200,100);
j_fillcircle(canvas,100,50,40);
:
```



filloval

Synopsis void **j_filloval** (int obj , int x , int y , int rx , int ry);

Arguments

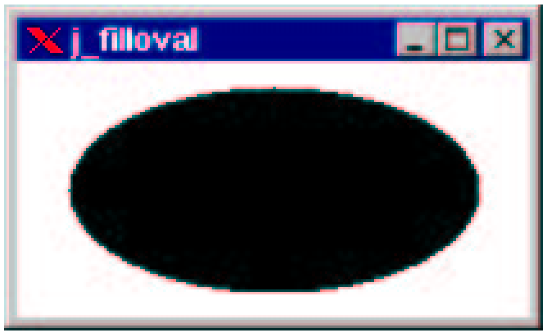
obj	int
x	int
y	int
rx	int
ry	int

Description Draws an filled oval with the center (**x, y**) and the horizontal radius **rx** and the vertical radius **ry**.

Targets Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,200,100);
j_filloval(canvas,100,50,80,40);
:
```



fillpolygon

Synopsis void **j_fillpolygon** (int obj , int len , int* x , int* y);

Arguments

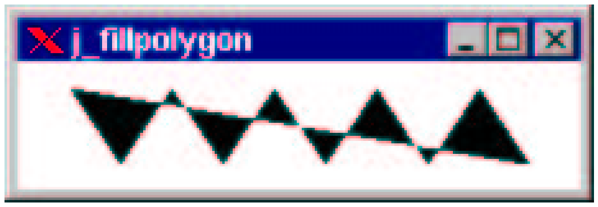
obj	int
len	int
x	int*
y	int*

Description Draws an filled polygon based on first **len** elements in **x** and **y**.

Targets Canvas, Image, Printer

Example

```
:
int x[10]={20,40,60,80,100,120,140,160,180,200};
int y[10]={10,40,10,40, 10,40,10,40,10,40};
canvas = j_canvas(frame,256,50);
j_fillpolygon(canvas,10,x,y);
:
```



fillrect

Synopsis

void **j_fillrect** (int obj , int x , int y , int width , int height);

Arguments

obj	int
x	int
y	int
width	int
height	int

Description

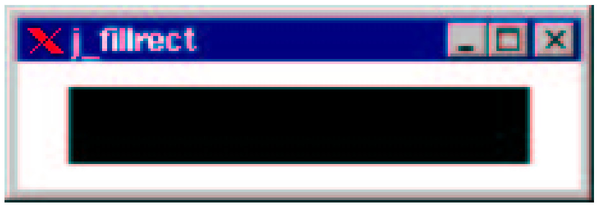
Draws an filled rectangle from **(x,y)** of size **width** x **height**.

Targets

Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,220,50);
j_fillrect(canvas,20,10,180,30);
:
```



fillroundrect

Synopsis

void **j_fillroundrect** (int obj , int x , int y , int width , int height , int arcx , int arcy);

Arguments

obj	int
x	int
y	int
width	int
height	int
arcx	int
arcy	int

Description

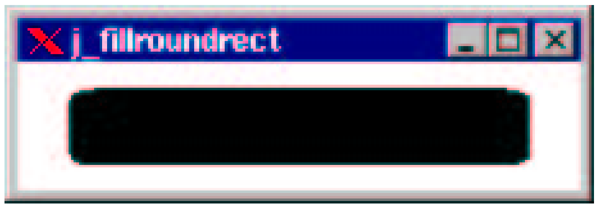
Draws an filled rectangle from **(x,y)** of size **width** x **height** with rounded corners. **arcx** and **arcy** specify the radius of rectangle corners.

Targets

Canvas, Image, Printer

Example

```
:
canvas = j_canvas(frame,220,50);
j_fillroundrect(canvas,20,10,180,30,10,5);
:
```



focuslistener

Synopsis	int **j_focuslistener** (int obj);
Arguments	obj int
Description	Adds a new focus listener to component **obj**, and returns its event number.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

frame

Synopsis int **j_frame** (char* label);

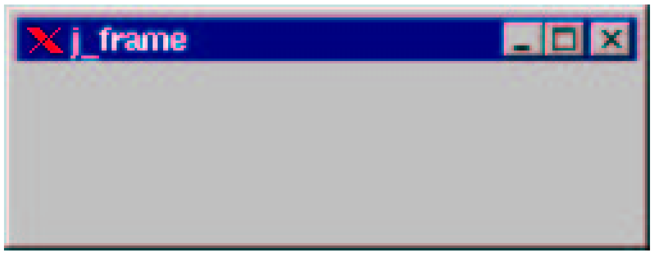
Arguments

label	char*

Description Creates a new frame component with the specified **label** and returns its event number.

Example

```
:
frame = j_frame("j_frame");
j_show(frame);
:
```



getaction

Synopsis int **j_getaction** ( );

Description returns the next event, or 0 if no event available

getcolumns

Synopsis void **j_getcolumns** (int obj);

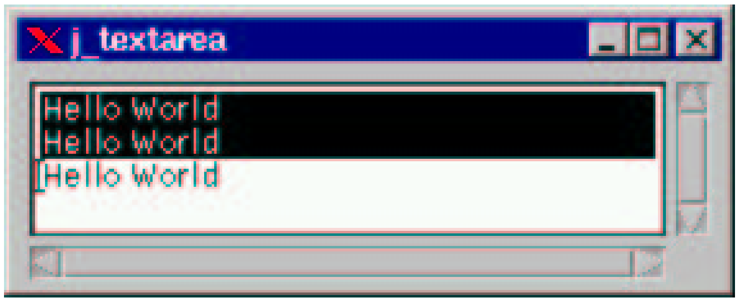
Arguments obj int

Description Gets the number of columns in **obj**.

Targets Textarea, Textfield, Gridlayout

Example

```
:
text = j_text(frame,30,4);
j_getcolumns(text);
:
> 30
```



Hello World
Hello World
Hello World

getcurpos

Synopsis	int **j_getcurpos** (int obj);
Arguments	obj int
Description	Returns the position, in characters, of the text cursor.
Targets	Textarea, Textfield

getdanger

Synopsis	void **j_getdanger** (int obj);	
Arguments	obj	int
Description	Returns the danger value of component **obj**.	
Targets	Meter	

getfontascent

Synopsis	int **j_getfontascent** (int obj);
Arguments	obj int
Description	Returns the ascent (space above the baseline) of the actual font of component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

getfontheight

Synopsis	int **j_getfontheight** (int obj);
Arguments	obj int
Description	Returns the total pixel height of the actual font of component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

getheight

Synopsis	int **j_getheight** (int obj);
Arguments	obj int
Description	Returns the height of component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Image

Example

```
:
label = j_getlabel(frame,"Hello World");
printf("%d\n",j_getheight(label));
:

> 22
```

getimagesource

Synopsis

int **j_getimagesource** (int obj , int x , int y , int w , int h , int* r , int* g , int* b);

Arguments

obj	int
x	int
y	int
w	int
h	int
r	int*
g	int*
b	int*

Description

Returns an image of the specified size (**x, y, w**idth, **h**eight) of component . The red, green and blue values of each pixel will be stored in **r, g, b**

Targets

Canvas, Image

getimage

Synopsis	int **j_getimage** (int obj);	
Arguments	obj	int
Description	Copy the contents of component **obj** into an image and return its eventnumber.	
Targets	Canvas, Image	

getinheight

Synopsis int **j_getinheight** (int cont);

Arguments	cont	int

Description Returns the height of the client size.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame = j_frame("Hello World")
j_setsize(frame,300,400)
printf("%d\n",j_getinheight(label));
:
> 370
```

getinsets

Synopsis int **j_getinsets** (int obj , int side);

Arguments

obj	int
side	int

Description Returns the width of the specified inset. **side** can take the following values:

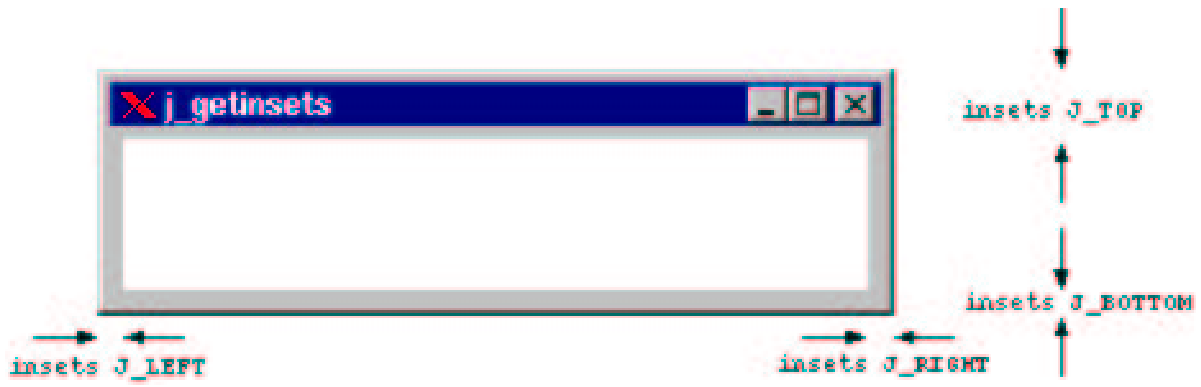
- J_TOP: returns the height of the top inset.
- J_BOTTOM: returns the height of the bottom inset.
- J_LEFT: returns the width of the left inset.
- J_RIGHT: returns the width of the right inset.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame  = j_frame("j_getinsets");
printf("%d %d %d %d\n",j_getinsets(frame,J_TOP),j_getinsets(frame,J_BOTTOM),
                       j_getinsets(frame,J_LEFT),j_getinsets(frame,J_RIGHT));
:

> 25 5 5 6
```



getinwidth

Synopsis	int **j_getinwidth** (int cont);
Arguments	cont int
Description	Returns the width of the client size.
Targets	Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame = j_frame("Hello World")
j_setsize(frame,300,400)
printf("%d\n",j_getinwidth(label));
:
> 289
```

getitemcount

Synopsis	int **j_getitemcount** (int obj);
Arguments	obj int
Description	Returns the number of items of component **obj**.
Targets	List, Choice

getitem

Synopsis	char* **j_getitem** (int obj , int item , char* str);
Arguments	obj int item int str char*
Description	returns the label of the given **item**.
Targets	List, Choice

getkeychar

Synopsis	int **j_getkeychar** (int obj);
Arguments	obj int
Description	Returns the ascii value of the last pressed key.
Targets	Keylistener

getkeycode

Synopsis	int **j_getkeycode** (int obj);
Arguments	obj int
Description	Returns the integer key code of the last pressed key.
Targets	Keylistener

getlayoutid

Synopsis int **j_getlayoutid** (int obj);

Arguments obj int

Description Returns the event number of the layoutmanager for containers **obj**.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
j_setgridlayout(frame,2,2);
grid = j_getlayoutid(frame);
:
```

getlength

Synopsis	int **j_getlength** (int obj);	
Arguments	obj	int
Description	Returns the length of component 's label or text.	
Targets	Textarea, Textfield, Dialog, Frame, Button, Menuitem, CheckBox-Menuitem,Menu, HelpMenu, Popupmenu	

getmousebutton

Synopsis int **j_getmousebutton** (int mouselistener);

Arguments mouselistener int

Description Returns the latest used mousebutton. The return value is:

- J_LEFT left mousebutton
- J_CENTER middle mousebutton
- J_RIGHT right mousebutton

Targets Mouselistener

getmousex

Synopsis int **j_getmousex** (int mouselistener);

Arguments mouselistener int

Description Returns the current horizontal position of the mouse in its parent's coordinate space.

Targets Mouselistener

getmousey

Synopsis	int **j_getmousey** (int mouselistener);
Arguments	mouselistener int
Description	Returns the current vertical position of the mouse in its parent's coordinate space.
Targets	Mouselistener

getparentid

Synopsis int **j_getparentid** (int obj);

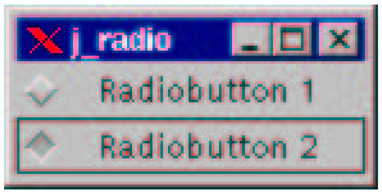
Arguments obj int

Description Returns the parent event number of component **obj**. If **obj** is a frame −1 will be returned.

Targets Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Menubar, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu, Radiogroup

Example

```
:
radio1     = j_radiobutton(j_radiogroup(frame),"Radiobutton 1");
radio2     = j_radiobutton(j_getparentid(radio1),"Radiobutton 2");
:
```



getparent

Synopsis int **j_getparent** (int obj);

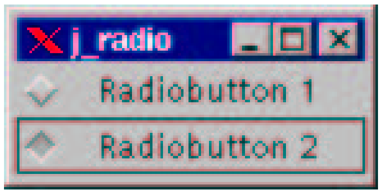
Arguments obj int

Description Returns the parent event number of component **obj**. If **obj** is a frame -1 will be returned.

Targets Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Menubar, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu, Radiogroup

Example

```
:
radio1     = j_radiobutton(j_radiogroup(frame),"Radiobutton 1");
radio2     = j_radiobutton(j_getparent(radio1),"Radiobutton 2");
:
```

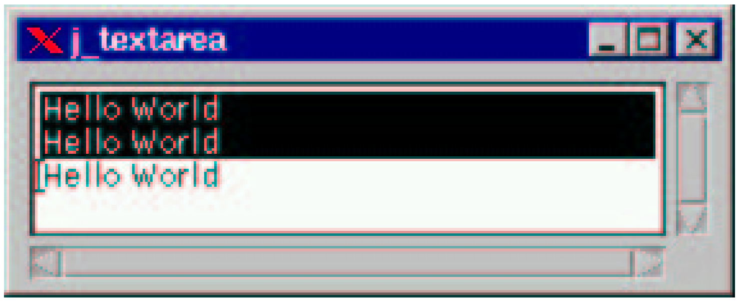


getrows

Synopsis	void **j_getrows** (int obj);
Arguments	obj int
Description	Gets the number of rows in **obj**.
Targets	Textarea, Gridlayout

Example

```
:
text = j_text(frame,30,4);
j_getrows(text);
:
> 4
```



getscaledimage

Synopsis	int **j_getscaledimage** (int obj , int x , int y , int sw , int sh , int tw , int th);

Arguments

obj	int
x	int
y	int
sw	int
sh	int
tw	int
th	int

Description

Copy the contents of the rectangular area defined by **x, y,** width **sw**, and height **sh** into an image and return its eventnumber. The image will be scaled to target width **th** and target height **th**.

Targets

Canvas, Image

getscreenheight

Synopsis

int **j_getscreenheight** ( );

Description

Returns the screens height in pixel. If a virtual screen is installed, the virtual height will be returned.

Example

```
:
printf("%d %d\n", j_getscreenwidth(), j_getscreenheight());
:

> 1280 1024
```

getscreenwidth

Synopsis

int **j_getscreenwidth** ( );

Description

Returns the screens width in pixel. If a virtual screen is installed, the virtual width will be returned.

Example

```
:
printf("%d %d\n", j_getscreenwidth(), j_getscreenheight());
:

> 1280 1024
```

getselect

Synopsis	int **j_getselect** (int obj);	
Arguments	obj	int
Description	Returns the position of currently selected item.	
Targets	List, Choice	

getselend

Synopsis	int **j_getselend** (int obj);
Arguments	obj int
Description	Returns the ending position of any selected text.
Targets	Textarea, Textfield

getselstart

Synopsis	int **j_getselstart** (int obj);
Arguments	obj int
Description	Returns the initial position of any selected text.
Targets	Textarea, Textfield

getseltext

Synopsis	char* **j_getseltext** (int obj , char* text);
Arguments	obj int text char*
Description	Returns the currently selected text of component **obj**.
Targets	Textarea, Textfield

getstate

Synopsis	int **j_getstate** (int obj);
Arguments	obj int
Description	Returns J_TRUE , if component is selected, J_FALSE otherwise.
Targets	Checkbox, Radiobutton, Checkmenuitem, Led

getstringwidth

Synopsis	int **j_getstringwidth** (int obj , char* str);
Arguments	obj int str char*
Description	Returns the length of **str** of the actual font of component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

gettext

Synopsis	char* **j_gettext** (int obj , char* str);
Arguments	obj int str char*
Description	returns the component 's text or label.
Targets	Button, Label, Checkbox, Radiobutton, Dialog, Frame, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu, Textarea, Textfield
Example	

```
char str[256];
:
label = j_label(frame,"Hello World");
printf("%s",j_gettext(label,str));
:

> Hello World
```

getvalue

Synopsis	int **j_getvalue** (int obj);
Arguments	obj int
Description	Returns the current setting of the scrollbar.
Targets	Scrollbar

getviewportheight

Synopsis	int **j_getviewportheight** (int obj);
Arguments	obj int
Description	Returns the height of the component 's **obj** port (the area that is shown)
Targets	Scrollpane

getviewportwidth

Synopsis	int **j_getviewportwidth** (int obj);
Arguments	obj int
Description	Returns the width of the component 's **obj** port (the area that is shown)
Targets	Scrollpane

getwidth

Synopsis	int **j_getwidth** (int obj);
Arguments	obj int
Description	Returns the width of component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Image

Example

```
:
label = j_getlabel(frame,"Hello World");
printf("%d\n",j_getwidth(label));
:

> 84
```

getxpos

Synopsis	int **j_getxpos** (int obj);
Arguments	obj int
Description	Returns the current horizontal position of component **obj** in its parent's coordinate space.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

getypos

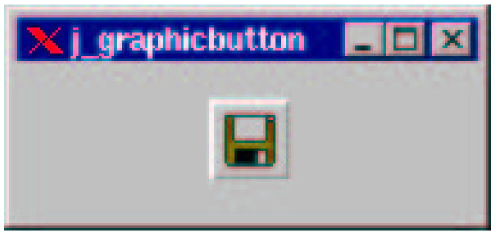
Synopsis	int **j_getypos** (int obj);
Arguments	obj int
Description	Returns the current vertical position of component **obj** in its parent's coordinate space.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

graphicbutton

Synopsis	int **j_graphicbutton** (int obj , char* filename);
Arguments	obj int filename char*
Description	Creates a new graphicbutton component with the image loaded from **filename** and returns its event number.
Targets	Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame  = j_frame("j_graphicbutton");
button = j_graphicbutton(frame,"save.gif");
:
```



graphiclabel

Synopsis	int **j_graphiclabel** (int obj , char* str);

Arguments

obj	int
str	char*

Description

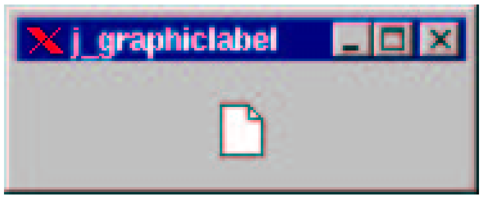
Creates a new graphiclabel component with the image loaded from **filename** and returns its event number.

Targets

Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame  = j_frame("j_graphiclabel");
label = j_graphiclabel(frame,"new.gif");
:
```



hasfocus

Synopsis	int **j_hasfocus** (int obj);
Arguments	obj int
Description	Returns J_TRUE if the component has the focus, J_FALSE otherwise.
Targets	Focuslistener

helpmenu

Synopsis	int **j_helpmenu** (int obj , char* label);

Arguments

obj	int
label	char*

Description

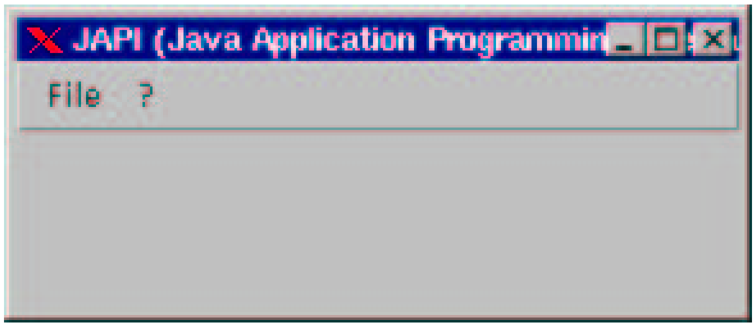
Creates a new helpmenu component with the specified **label** and returns its event number.

Targets

Menubar

Example

```
:
frame   = j_frame("Menu Komponenten");
menubar = j_menubar(frame);
file= j_menu(menubar,"File");
help= j_helpmenu(menubar,"?");
:
```



hide

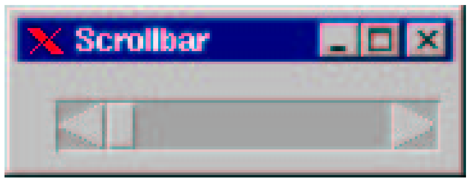
Synopsis	void **j_hide** (int obj);
Arguments	obj int
Description	Hides the component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

hscrollbar

Synopsis	int **j_hscrollbar** (int obj);
Arguments	obj int
Description	Creates a new horizontal scrollbar and returns its event number.
Targets	Panel, Borderpanel, Window, Dialog, Frame, Scrollpane

Example

```
:
scroll=j_hscrollbar(frame);
j_setpos(scroll,20,40);
j_setsize(scroll,150,20);
:
```



image

Synopsis	int **j_image** (int width , int height);

Arguments

width	int
height	int

Description

Creates a new (memory) image component with the given **width** and **height** and returns its event number. The return value is the eventnumber of the image. On error -1 will be returned.

Example

```
:
image = j_image(200,200);
:
```

insert

Synopsis	int **j_insert** (int obj , int pos , char* label);

Arguments

obj	int
pos	int
label	char*

Description: inserts a new item to component **obj** at position **pos** with the specified **label**.

Targets: List, Choice

inserttext

Synopsis	void **j_inserttext** (int obj , char* text , int pos);
Arguments	obj int text char* pos int
Description	Places additional text within the component at the given position **pos**.
Targets	Textarea

isparent

Synopsis	int **j_isparent** (int obj , int cont);
Arguments	obj int cont int
Description	Returns J_TRUE if **cont** is parent of **obj**, J_FALSE otherwise.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Menubar, Menuitem, CheckBox-Menuitem,Menu, HelpMenu, Popupmenu, Radiogroup

isresizable

Synopsis	int **j_isresizable** (int obj);
Arguments	obj int
Description	returns true if component is resizable, false otherwise
Targets	Dialog, Frame

isselect

Synopsis	int **j_isselect** (int obj , int item);
Arguments	obj int item int
Description	Returns J_TRUE if the particular **item** is currently selected, J_FALSE otherwise.
Targets	List

isvisible

Synopsis	int **j_isvisible** (int obj);
Arguments	obj int
Description	Returns J_TRUE if **obj** is visible, J_FALSE otherwise.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

keylistener

Synopsis	int **j_keylistener** (int obj);
Arguments	obj int
Description	Adds a new key listener to component **obj**, and returns its event number.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

label

Synopsis

int **j_label** (int obj , char* label);

Arguments

obj	int
label	char*

Description

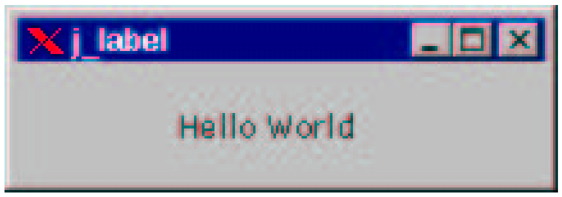
Creates a new label component with the specified **label** and returns its event number.

Targets

Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame = j_frame("j_label");
label = j_label(frame,"Hello World");
:
```



led

Synopsis

int **j_led** (int obj , int style , int color);

Arguments

obj	int
style	int
color	int

Description

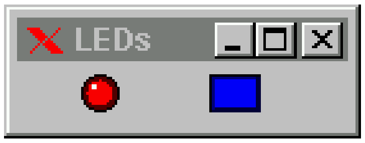
Creates a new led component and returns its event number. The LEDs shape could be round if **style**=J_ROUND or a rectangle if **style**=J_RECT. The color could be one of the predefined colors (eg. J_RED, J_GREEN).

Targets

Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
led1 = j_led(frame,J_ROUND,J_RED);
led2 = j_led(frame,J_RECT,J_BLUE);
:
```



line

Synopsis

int **j_line** (int obj , int orient , int style , int length);

Arguments

obj	int
orient	int
style	int
length	int

Description

Creates a new line component with the specified **length** and returns its event number. A line may be used to separate groups of components. On Error -1 will returned. The parameter **orient** specifies the orientation of the line:

- J_HORIZONTAL : horizontal line
- J_VERTICAL : vertical line

The Parameter **style** specifies the linestyle:

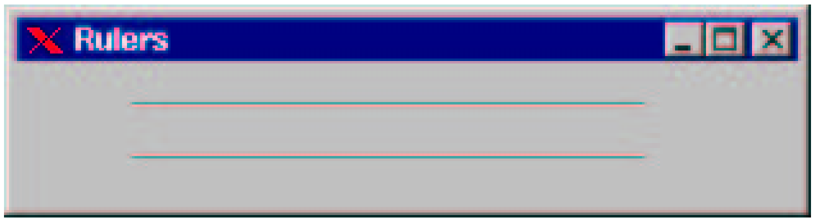
- J_LINEDOWN : etched-in linestyle.
- J_LINEUP : etchet-out linestyle.

Targets

Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
line1 = j_line(frame,J_HORIZONTAL,J_LINEDOWN,200);
line2 = j_line(frame,J_HORIZONTAL,J_LINEUP,200);
:
```



list

Synopsis int **j_list** (int obj , int rows);

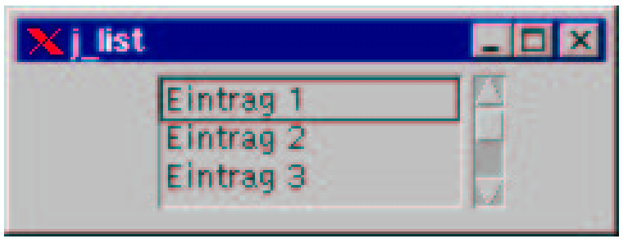
Arguments		
	obj	int
	rows	int

Description Creates a new list component with the specified number of **rows** and returns its event number.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
list = j_list(frame,3);
j_additem(list,"Eintrag 1");
j_additem(list,"Eintrag 2");
:
```



loadimage

Synopsis int **j_loadimage** (char* filename);

Arguments filename char*

Description Loads the Image from file **filename** and returns its eventnumber.
The file could be of the following format:

- GIF
- JPEG
- BMP
- PPM

Example

```
:
image = j_loadimage("mandel.jpg");
:
```

menubar

Synopsis int **j_menubar** (int obj);

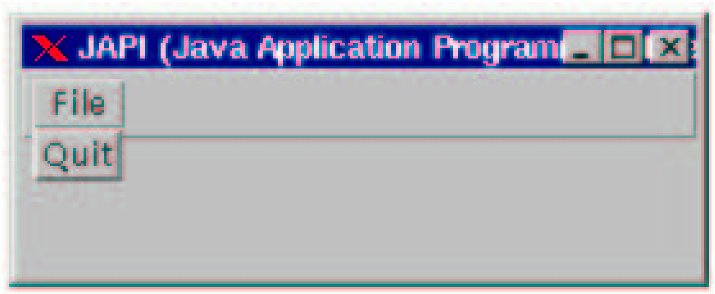
Arguments	obj int

Description Creates a new menubar and returns its event number.

Targets Frame

Example

```
:
frame   = j_frame("Menu Komponenten");
menubar = j_menubar(frame);
file    = j_menu(menubar,"File");
quit    = j_menuitem(file,"Quit");
:
```



menuitem

Synopsis	int **j_menuitem** (int obj , char* label);

Arguments

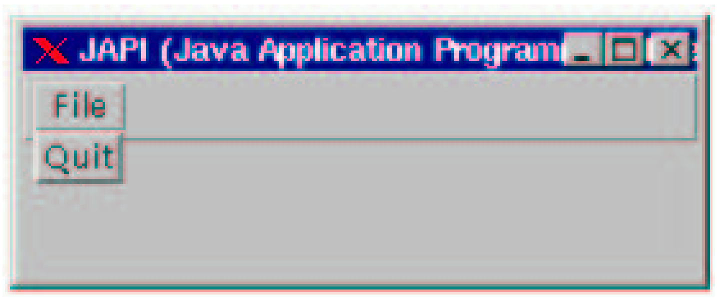
obj	int
label	char*

Description: Creates a new menuitem with the specified **label** and returns its event number.

Targets: Menu, Popupmenu, Helpmenu

Example

```
:
frame   = j_frame("Menu Komponenten");
menubar = j_menubar(frame);
file    = j_menu(menubar,"File");
quit    = j_menuitem(file,"Quit");
:
```



menu

Synopsis int **j_menu** (int obj , char* str);

Arguments

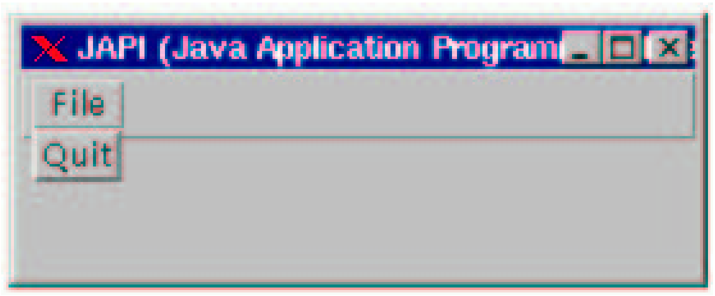
obj	int
str	char*

Description Creates a new menu component with the specified **label** and returns its event number.

Targets Menubar, Menu

Example

```
:
frame   = j_frame("Menu Komponenten");
menubar = j_menubar(frame);
file    = j_menu(menubar,"File");
quit    = j_menuitem(file,"Quit");
:
```

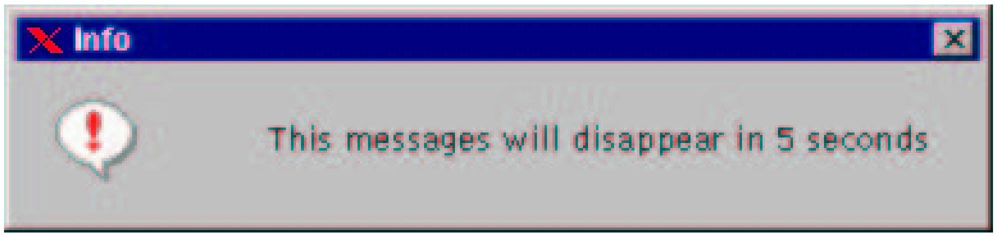


messagebox

Synopsis	void **j_messagebox** (int obj , char* title , char* text);
Arguments	obj int title char* text char*
Description	Shows a messagebox with the specified **title** and **text** and returns its event number. In the case of error −1 will be returend. A Messagebox generates an event, if the close icon is clicked.
Targets	Frame

Example

```
:
mbox = j_messagebox(frame,"Info","This messages will disappear in 5 seconds");
j_sleep(5000);
j_dispose(mbox);
:
```



This messages will disappear in 5 seconds

meter

Synopsis int **j_meter** (int obj , char* title);

Arguments

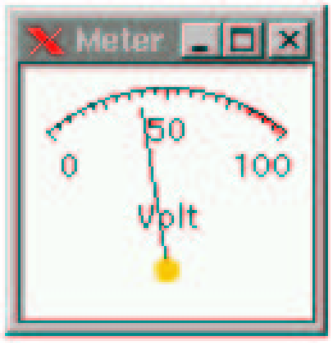
obj	int
title	char*

Description Creates a new pointer–intrument with the specified label **titel** and returns its event number. The meter has predifiend values from 0 to 100. This can be canged via j_setmin() and j_setmax(). A danger value is set to 80 and can be justified with j_setdanger().

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
meter = j_meter(frame,"Volt");
j_setvalue(meter,40);
:
```



mouselistener

Synopsis

int **j_mouselistener** (int obj , int kind);

Arguments

obj	int
kind	int

Description

Adds a new mouse listener to component **obj**, and returns its event number. An event occures, if the user action is of kind **kind**. Posible values for **kind**:

- J_ENTERED : An event occures if the mouse cursor has been moved into the component **obj**.
- J_MOVED : An event occures if the mouse cursor has been moved inside the component **obj**.
- J_EXITED : An event occures if the mouse cursor has been moved out of the component **obj**.
- J_PRESSED : An event occures if a mouse button was pressed.
- J_DRAGGED : An event occures if the mouse cursor has been dragged (moved with pressed button) inside the component **obj**.
- J_RELEASED : An event occures if a mouse button was released.
- J_DOUBLECLICK : An event occures if a mouse button was doubleclicked.

Targets

Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

multiplemode

Synopsis	int **j_multiplemode** (int obj , int bool);
Arguments	obj int bool int
Description	if **bool** is J_TRUE , selection mode is turned to multiplemode.
Targets	List

nextaction

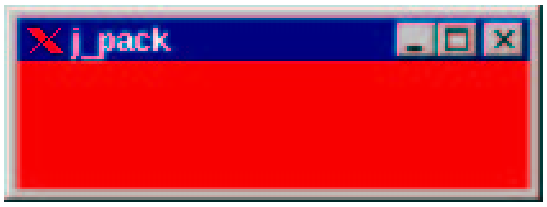
Synopsis	int **j_nextaction** ( );
Description	Waits for the next event.

pack

Synopsis	void **j_pack** (int obj);
Arguments	obj int
Description	Resizes component to the minimal size of contained components.
Targets	Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
j_setflowlayout(jframe,J_HORIZOMTAL);
canvas = j_canvas(frame,200,50);
j_setnamedcolorbg(canvas,J_RED);
j_pack(frame);
:
```



panel

Synopsis int **j_panel** (int obj);

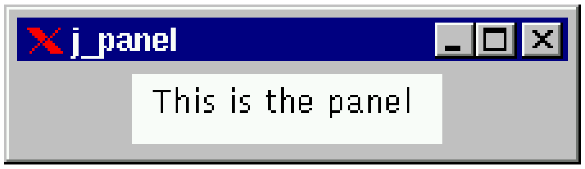
Arguments	obj	int

Description Creates a new panel component and returns its event number.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
panel = j_panel(frame);
j_setnamedcolorbg(panel,J_WHITE);
j_setpos(panel,50,30);
label = j_label(panel,"This is the panel");
j_setpos(label,0,0);
:
```



popupmenu

Synopsis int **j_popupmenu** (int obj , char* label);

Arguments

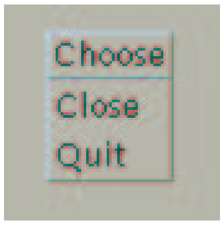
obj	int
label	char*

Description Creates a new popupmenu with the specified **label** and returns its event number.

Targets Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

Example

```
:
choose = j_popupmenu(frame,"Choose");
close  = j_menuitem(choose,"Close");
quit   = j_menuitem(choose,"Quit");
j_showpopup(choose,100,100);
:
```



printer

Synopsis int **j_printer** (int frame);

Arguments

frame	int

Description Creates a new object, representing a paper of the printer and returns its event number. On error -1 will be returned. A printer object can be used like a canvas, where all drawing funktions will be passed to the printer, instead of a window. A printer generates no event.

Targets Frame

Example

```
:
printer = j_printer(frame);
j_drawimage(printer,image,100,100);
:
```

print

Synopsis

void **j_print** (int obj);

Arguments

obj	int

Description

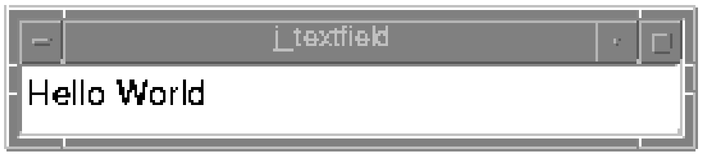
prints the component . With X-Windows all components have Motif-look.
If component is a printer, the actual page will be closed, and a new page will be opened. The pages are not jet printed. To print all pages call j_dispose(printer);

Targets

Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Canvas, Image, Printer

Example

```
:
frame = j_frame("j_textfield");
text  = j_textfield(frame,30)
:
j_print(frame);
:
```



progressbar

Synopsis int **j_progressbar** (int obj , int orient);

Arguments

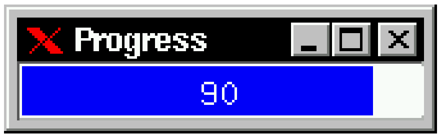
obj	int
orient	int

Description Creates a new progressbar with the specified **orient**ation and returns its event number. Orientation could be J_HORIZONTAL or J_VERTICAL. The progressbar has predifiend values from 0 to 100. This can be canged via *j_setmin*() and *j_setmax*().

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
progress = j_progressbar(frame,J_HORIZONTAL);
j_setvalue(progress,90);
:
```



quit

Synopsis	void **j_quit** ( );
Description	Cancels the connection to the JAPI Kernel.

radiobutton

Synopsis int **j_radiobutton** (int obj , char* label);

Arguments

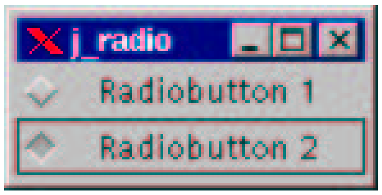
obj	int
label	char*

Description Creates a new radiobutton with the specified **label** and returns its event number.

Targets Radiogroup

Example

```
:
radiogroup = j_radiogroup(frame);
radio1     = j_radiobutton(radiogroup,"Radiobutton 1");
radio2     = j_radiobutton(radiogroup,"Radiobutton 2");
:
```



radiogroup

Synopsis int **j_radiogroup** (int obj);

Arguments

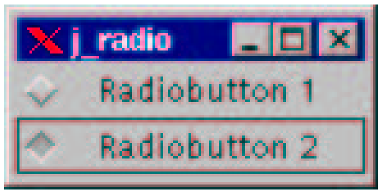
obj	int

Description Creates a new radiogroup and returns its event number.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
radiogroup = j_radiogroup(frame);
radio1     = j_radiobutton(radiogroup,"Radiobutton 1");
radio2     = j_radiobutton(radiogroup,"Radiobutton 2");
:
```



random

Synopsis	int **j_random** ( );
Description	Generates a pseudo random number. The returned value will be in the range of 0 to 2147483647 ($2^{31} - 1$).

releaseall

Synopsis	void **j_releaseall** (int obj);
Arguments	obj int
Description	Releases all components from component **obj**.
Targets	Panel, Borderpanel, Window, Dialog, Frame

release

Synopsis	void **j_release** (int obj);
Arguments	obj int
Description	Releases component **obj** from its parent component (container).
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

removeall

Synopsis	int **j_removeall** (int obj);
Arguments	obj int
Description	Removes all items from the component .
Targets	List, Choice

removeitem

Synopsis	int **j_removeitem** (int obj , char* item);
Arguments	obj int item char*
Description	remove the first occurrence of **item** from the component .
Targets	List, Choice

remove

Synopsis	int **j_remove** (int obj , int item);
Arguments	obj int item int
Description	removes the Item with the Index **item** from the component .
Targets	List, Choice

replacetext

Synopsis	void **j_replacetext** (int obj , char* text , int start , int end);
Arguments	obj int text char* start int end int
Description	Replaces the text from starting position **start** to ending position **end** with the given **text**.
Targets	Textarea

saveimage

Synopsis int **j_saveimage** (int obj , char* filename , int filetyp);

Arguments

obj	int
filename	char*
filetyp	int

Description Saves the components image to file **filename**. The specified file format can be:

- J_BMP Win32 Bitmap Format
- J_PPM Portable pixmap

Example

```
:
if(! j_saveimage(canvas,"mandel.bmp",J_BMP))
    printf("Error saving Bitmap file\n");
:
```

scrollpane

Synopsis int **j_scrollpane** (int obj);

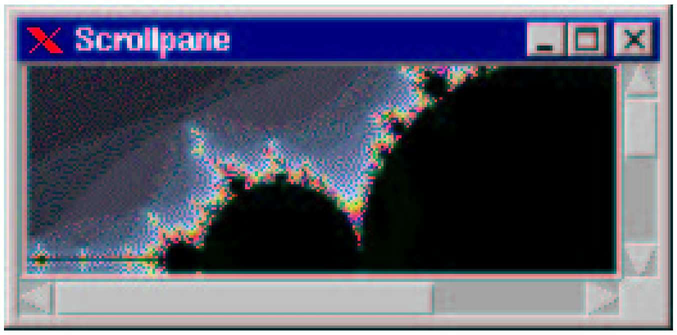
Arguments obj int

Description Creates a new scrollpane component and returns its event number.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
scrollpane  = j_scrollpane(frame);
image = j_graphiclabel(scrollpane,"mandel.gif");
j_setsize(scrollpane,240,100);
:
```



selectall

Synopsis	void **j_selectall** (int obj);
Arguments	obj int
Description	Selects all the text in the component .
Targets	Textarea, Textfield

select

Synopsis	int **j_select** (int obj , int item);
Arguments	obj int item int
Description	Makes the given **item** the selected one for the component .
Targets	List, Choice

selecttext

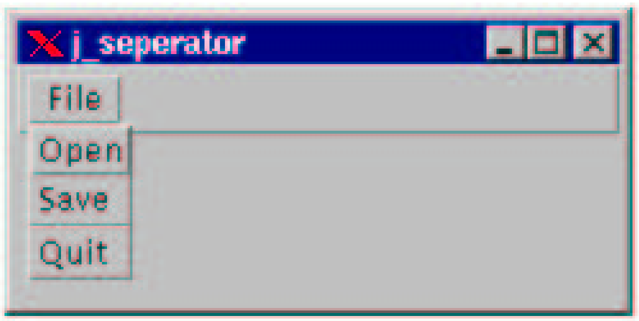
Synopsis	void **j_selecttext** (int obj , int start , int end);
Arguments	obj int start int end int
Description	Selects text from starting position **start** to ending position **end**.
Targets	Textarea, Textfield

seperator

Synopsis	void **j_seperator** (int obj);
Arguments	obj int
Description	Adds a separator bar to the component .
Targets	Menu, HelpMenu, Popupmenu

Example

```
:
file  = j_menu(menubar,"File");
open  = j_menuitem(file,"Open");
save  = j_menuitem(file,"Save");
j_seperator(file);
quit  = j_menuitem(file,"Quit");
:
```



setalign

Synopsis	void **j_setalign** (int obj , int align);
Arguments	obj int align int
Description	Sets the alignment in component **obj** to **align**. Needs a flowlayout Manager.
Targets	Panel, Borderpanel, Window, Dialog, Frame

setblockinc

Synopsis	int **j_setblockinc** (int obj , int val);
Arguments	obj int val int
Description	Changes the block increment amount for the component to **val**.
Targets	Scrollbar

setborderlayout

Synopsis	void **j_setborderlayout** (int obj);
Arguments	obj int
Description	Adds a borderlayout manager to component **obj**.
Targets	Panel, Borderpanel, Window, Dialog, Frame

setborderpos

Synopsis	void **j_setborderpos** (int obj , int pos);
Arguments	obj int pos int
Description	Moves component **obj** at a certain position. The outer container needs a border layout manager.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

setcolorbg

Synopsis

void **j_setcolorbg** (int obj , int r , int g, , int b);

Arguments

obj	int
r	int
g,	int
b	int

Description

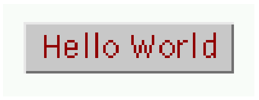
Sets the background color to the (**r, g, b**) values.

Targets

Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

Example

```
:
button = j_button(frame,"Hello World");
j_setcolorbg(button,150,0,0);
j_settext(button,"Hello World");
:
```



setcolor

Synopsis void **j_setcolor** (int obj , int r , int g, , int b);

Arguments

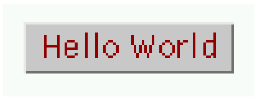
obj	int
r	int
g,	int
b	int

Description Sets the foreground color to the (**r, g, b**) values.

Targets Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

Example

```
:
button = j_button(frame,"Hello World");
j_setcolor(button,150,0,0);
j_settext(button,"Hello World");
:
```



setcolumns

Synopsis void **j_setcolumns** (int obj , int columns);

Arguments

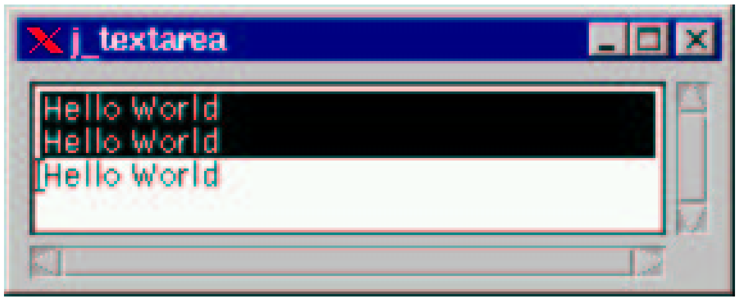
obj	int
columns	int

Description Sets the number of columns for **obj** to **columns**.

Targets Textarea, Textfield, Gridlayout

Example

```
:
text = j_text(frame,10,4);
j_setcolumns(text,30);
:
```



setcurpos

Synopsis	void **j_setcurpos** (int obj , int pos);
Arguments	obj int pos int
Description	Change the location of the text cursor to the specified position **pos**.
Targets	Textarea, Textfield

setcursor

Synopsis	int **j_setcursor** (int obj , int cursor);
Arguments	obj int cursor int
Description	Changes the component 's **obj** cursor to the specified **cursor**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

setdanger

Synopsis	void **j_setdanger** (int obj , int val);
Arguments	obj int val int
Description	Changes the danger value of component **obj** to **val**.
Targets	Meter

setdebug

Synopsis	void **j_setdebug** (int level);
Arguments	level int
Description	Sets the debuglevel to **level**.

setechochar

Synopsis	void **j_setechochar** (int obj , char chr);
Arguments	obj int chr char
Description	Changes the character **chr** that is used to echo all user input in the component .
Targets	Textfield

seteditable

Synopsis	void **j_seteditable** (int obj , int bool);
Arguments	obj int bool int
Description	Allows to make the component editable (**bool**=J_TRUE) or read-only (**bool**=J_FALSE).
Targets	Textarea, Textfield

setfixlayout

Synopsis	void **j_setfixlayout** (int obj);
Arguments	obj int
Description	Adds a fixlayout manager to component **obj** (default layout manager).
Targets	Panel, Borderpanel, Window, Dialog, Frame

setflowfill

Synopsis	void **j_setflowfill** (int obj , int bool);
Arguments	obj int bool int
Description	Resizes all containing component to the height (width) of component **obj**. Needs a flowlayout manager.
Targets	Panel, Borderpanel, Window, Dialog, Frame

setflowlayout

Synopsis	void **j_setflowlayout** (int obj , int align);	
Arguments	obj	int
	align	int
Description	Adds a flowlayout manager to component **obj** with the specified **align**ment.	
Targets	Panel, Borderpanel, Window, Dialog, Frame	

setfocus

Synopsis	int **j_setfocus** (int obj);
Arguments	obj int
Description	Directs the input focus to component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

setfontname

Synopsis void **j_setfontname** (int obj , int name);

Arguments

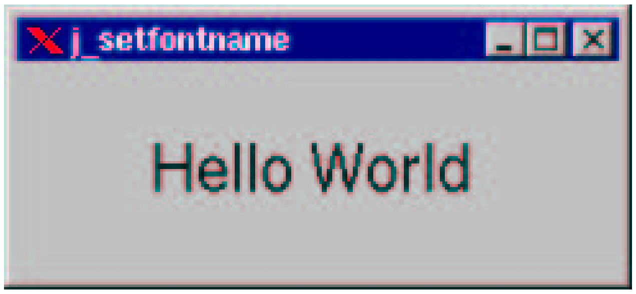
obj	int
name	int

Description Changes the font to the given **name**.

Targets Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu

Example

```
:
label = j_label(jframe,"Hello World");
j_setfontname(label,J_HELVETIA);
:
```



setfontsize

Synopsis void **j_setfontsize** (int obj , int size);

Arguments

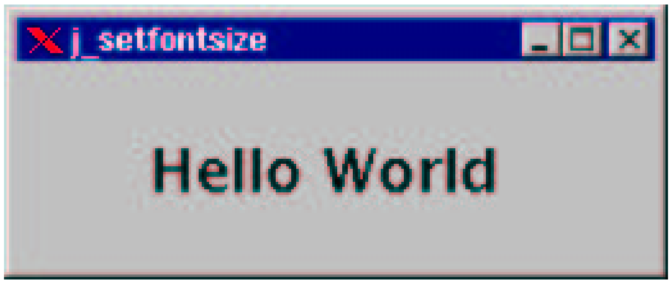
obj	int
size	int

Description Changes the font to the given **size**.

Targets Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu

Example

```
:
label = j_label(jframe,"Hello World");
j_setfontsize(label,24);
:
```



setfontstyle

Synopsis	void **j_setfontstyle** (int obj , int style);

Arguments

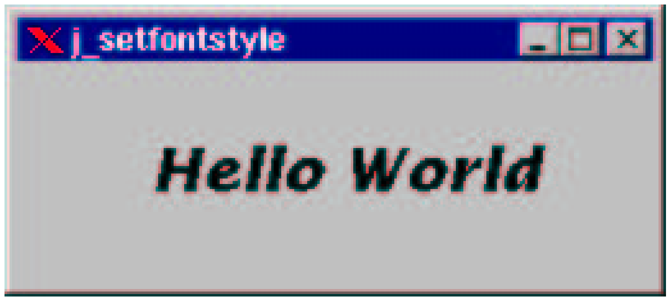
obj	int
style	int

Description: Changes the font to the given **style**.

Targets: Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu

Example

```
:
label = j_label(jframe,"Hello World");
j_setfontstyle(label,J_BOLD+J_ITALIC);
:
```



setfont

Synopsis void **j_setfont** (int obj , int name , int style , int size);

Arguments

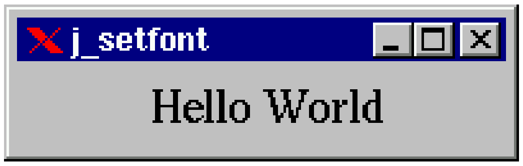
obj	int
name	int
style	int
size	int

Description Changes the font to the given characteristics **name**, **style** and **size**.

Targets Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu

Example

```
:
label = j_label(jframe,"Hello World");
j_setfont(label,J_TIMES,J_PLAIN,18);
:
```



setgridlayout

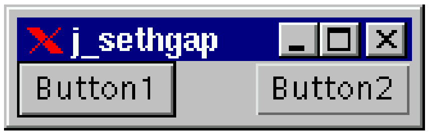
Synopsis	void **j_setgridlayout** (int obj , int row , int col);
Arguments	obj int row int col int
Description	Adds a gridlayout manager to component **obj** with the specified **row**s and **col**umns.
Targets	Panel, Borderpanel, Window, Dialog, Frame

sethgap

Synopsis	void **j_sethgap** (int obj , int hgap);
Arguments	obj int hgap int
Description	Sets the horizontal gap between components to **hgap** Pixel.
Targets	Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
j_flowlayout(frame,J_HORIZONTAL);
button1 = j_button(frame,"Button1");
button2 = j_button(frame,"Button2");
j_sethgap(frame,30);
:
```



seticon

Synopsis void **j_seticon** (int frame , int icon);

Arguments

frame	int
icon	int

Description Sets the image **icon** to display when the **frame** is iconized. Not all platforms support the concept of iconizing a window.

Targets Frame

Example

```
:
frame = j_frame("Hello World");
j_seticon(frame,j_loadimage("icon.gif"));
:
```

setimage

Synopsis void **j_setimage** (int obj , int image);

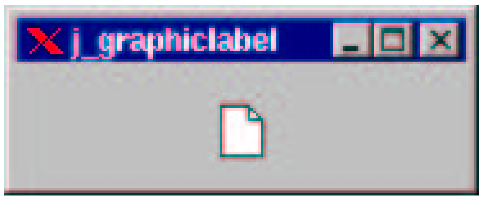
Arguments	obj	int
	image	int

Description Sets the **image** to be displayed in **obj**.

Targets Graphicbutton, Graphiclabel

Example

```
:
label = j_graphiclabel(frame,"mandel.gif");
image = j_image("new.gif");
j_setimage(label,image);
:
```



setinsets

Synopsis

void **j_setinsets** (int obj , int top , int bottom , int left , int right);

Arguments

obj	int
top	int
bottom	int
left	int
right	int

Description

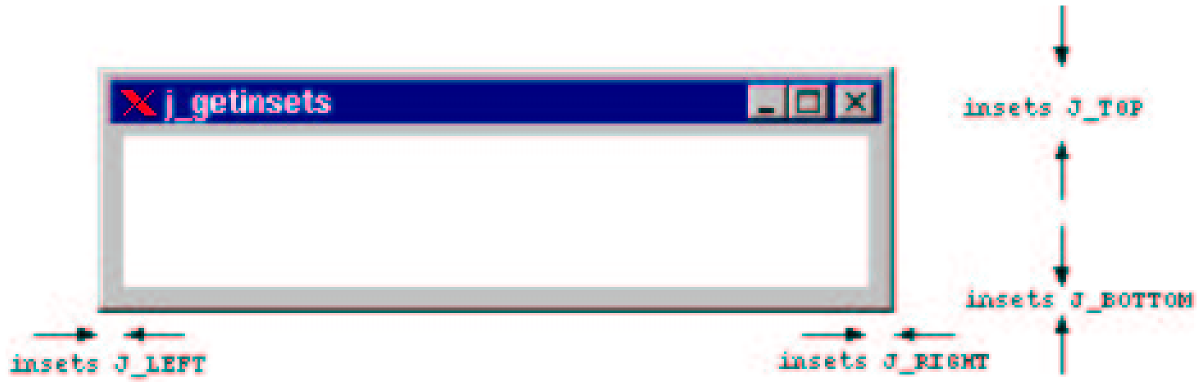
Set the insets to the specified values.

Targets

Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame  = j_frame("j_getinsets");
j_setinsets(frame,30,10,10,10);
:
```



setmax

Synopsis	int **j_setmax** (int obj , int val);
Arguments	obj int val int
Description	Changes the maximum value for the component to **val**.
Targets	Scrollbar, Meter, Progress

setmin

Synopsis	int **j_setmin** (int obj , int val);
Arguments	obj int val int
Description	Changes the minimum value for the component to **val**.
Targets	Scrollbar, Meter, Progress

setnamedcolorbg

Synopsis	void **j_setnamedcolorbg** (int obj , int color);
Arguments	obj int color int
Description	Sets the background color to a predefined **color**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

setnamedcolor

Synopsis	void **j_setnamedcolor** (int obj , int color);

Arguments

obj	int
color	int

Description — Sets the foreground color to a predefined **color**.

Targets — Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

setnolayout

Synopsis	void **j_setnolayout** (int obj);
Arguments	obj int
Description	Removes the current layout manager from component **obj** .
Targets	Panel, Borderpanel, Window, Dialog, Frame

start

Synopsis	void **j_start** (int port);
Arguments	port int
Description	Replace the default Port by **port**. This can be usefull if the default port is used by an other application, or if you want to start several independent kernels on one machine. This functions must be called before calling j_start();

Example

```
:
j_setport(12345);
if(j_start() != J_TRUE)
:
```

setpos

Synopsis	void **j_setpos** (int obj , int xpos , int ypos);
Arguments	obj int xpos int ypos int
Description	Relocates the component **obj** to the specified Position (**xpos**,**ypos**).
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

setradiogroup

Synopsis	int **j_setradiogroup** (int rbutton, , int rgroup);
Arguments	rbutton, int rgroup int
Description	Sets radiobuttons **rbutton** group to be the specified radiogroup **rgroup**. If the radiobuttons is already in a different radiogroup, it is first taken out of that group.
Targets	Radiobutton

setresizable

Synopsis void **j_setresizable** (int obj , int resizable);

Arguments

obj	int
resizable	int

Description The component cannot be resized, if **resizable** is J_FALSE .

Targets Dialog, Frame

Example

```
:
frame = j_frame("fixsized Frame");
j_setrezisable(frame,J_FALSE);
:
```

setrows

Synopsis void **j_setrows** (int obj , int rows);

Arguments

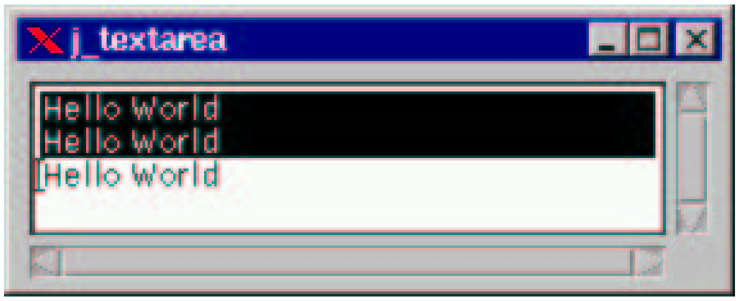
obj	int
rows	int

Description Sets the number of rows for **obj** to **rows**.

Targets Textarea, Gridlayout

Example

```
:
text = j_text(frame,30,10);
j_setcolumns(text,4);
:
```



setshortcut

Synopsis	void **j_setshortcut** (int obj , char chr);
Arguments	obj int chr char
Description	Changes the shortcut **chr** of the component .
Targets	Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu

setsize

Synopsis void **j_setsize** (int obj , int width , int height);

Arguments

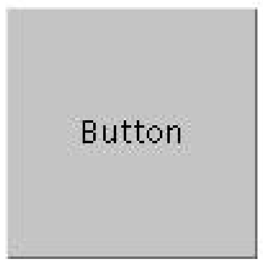
obj	int
width	int
height	int

Description Resizes component **obj** to specified **width** and **height**.

Targets Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

Example

```
:
button = j_button(frame,"Button");
j_setsize(button,100,100);
:
```



setslidesize

Synopsis	int **j_setslidesize** (int obj , int val);
Arguments	obj int val int
Description	Changes the slide size to **val**.
Targets	Scrollbar

setstate

Synopsis	void **j_setstate** (int obj , int bool);
Arguments	obj int bool int
Description	The component becomes selected, if **bool** is J_TRUE .
Targets	Checkbox, Radiobutton, Checkmenuitem, Led

settext

Synopsis

void **j_settext** (int obj , char* str);

Arguments

obj	int
str	char*

Description

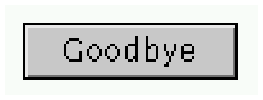
Sets the content or the label of the component **obj** to **str**.

Targets

Button, Label, Checkbox, Radiobutton, Dialog, Frame, Menuitem, CheckBoxMenuitem,Menu, HelpMenu, Popupmenu, Textarea, Textfield

Example

```
:
button = j_button(frame,"Hello World");
j_settext(button,"Goodbye");
:
```



setunitinc

Synopsis	int **j_setunitinc** (int obj , int val);
Arguments	obj int val int
Description	Changes the unit increment amount for the component to **val**
Targets	Scrollbar

setvalue

Synopsis	void **j_setvalue** (int obj , int val);
Arguments	obj int val int
Description	Changes the current value of the component to **val**.
Targets	Scrollbar, Progress, Meter, Sevensegment

setvgap

Synopsis void **j_setvgap** (int obj , int vgap);

Arguments

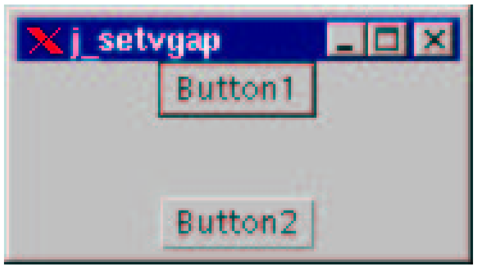
obj	int
vgap	int

Description Sets the vertical gap between components to **hgap** Pixel.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
j_setflowlayout(frame,J_VERTICAL);
button1 = j_button(frame,"Button1");
button2 = j_button(frame,"Button2");
j_setvgap(frame,30);
:
```



setxor

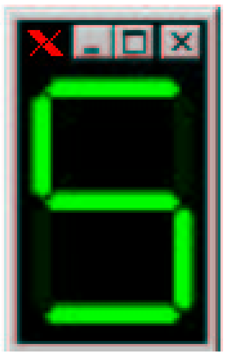
Synopsis	void **j_setxor** (int obj , int bool);
Arguments	obj int bool int
Description	Changes painting mode to XOR mode, if bool = J_TRUE . In this mode, drawing the same object in the same color at the same location twice has no net effect.
Targets	Canvas, Image, Printer

sevensegment

Synopsis	int **j_sevensegment** (int obj , int color);
Arguments	obj int color int
Description	Creates a new sevensegment display and returns its event number. The color could be one of the predefined colors (eg. J_RED, J_GREEN).
Targets	Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
seven = j_sevensegment(frame,J_GREEN);
j_setvalue(seven,5);
:
```



showpopup

Synopsis	void **j_showpopup** (int obj , int xpos , int ypos);

Arguments		
	obj	int
	xpos	int
	ypos	int

Description — Shows the component at specified Position (**xpos**,**ypos**).

Targets — Popupmenu

show

Synopsis	void **j_show** (int obj);
Arguments	obj int
Description	Shows the component **obj**.
Targets	Button, Graphicbutton, Canvas, Checkbox, Radiobutton, Choice, Label, Graphiclabel, List, Scrollbar, Panel, Borderpanel, Window, Dialog, Frame, Scrollpane, Textarea, Textfield, Led, Progressbar, Meter, Sevensegment

sleep

Synopsis	int **j_sleep** (int msec);	
Arguments	msec	int
Description	Suspends the execution for **msec** milliseconds.	

start

Synopsis int **j_start** ( );

Description Get in touch with a running japi kernel or start a neu one.

Example

```
:
if(j_start() != J_TRUE)
{
   printf("can't connect to JAPI Kernel\n");
   exit(0);
}
:
```

sync

Synopsis	void **j_sync** ( );
Description	Synchronizes the application with the JAPI kernel.

textarea

Synopsis int **j_textarea** (int obj , int rows , int columns);

Arguments

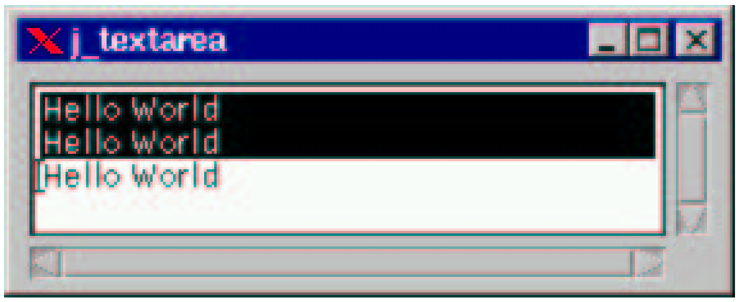
obj	int
rows	int
columns	int

Description Creates a new textarea component with the specified number of **rows columns** and returns its event number.

Targets Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame = j_frame("j_textarea");
text  = j_textarea(frame,30,4)
:
```



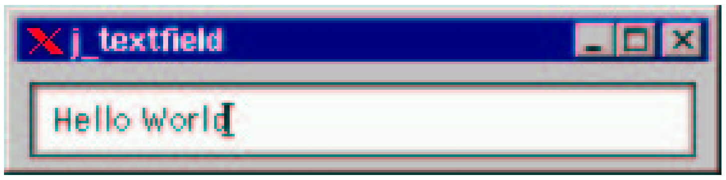
```
Hello World
Hello World
Hello World
```

textfield

Synopsis	int **j_textfield** (int obj , int columns);
Arguments	obj int columns int
Description	Creates a new textfield component with the specified number of **columns** and returns its event number.
Targets	Panel, Borderpanel, Window, Dialog, Frame

Example

```
:
frame = j_frame("j_textfield");
text  = j_textfield(frame,30)
:
```



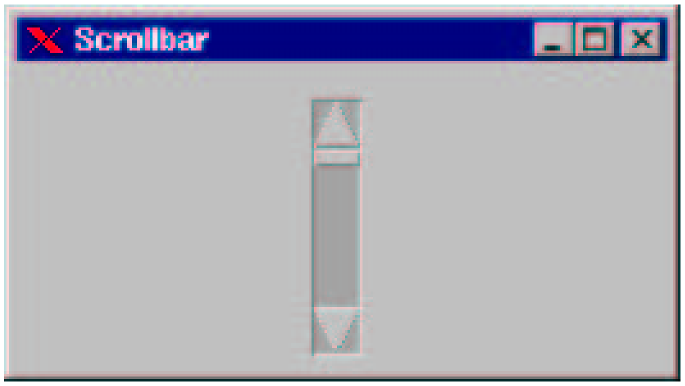
translate

Synopsis	void **j_translate** (int obj , int x , int y);	
Arguments	obj	int
	x	int
	y	int
Description	Moves the origin of drawing operations to (**x, y**).	
Targets	Canvas, Image, Printer	

vscrollbar

Synopsis	int **j_vscrollbar** (int obj);
Arguments	obj int
Description	Creates a new vertical scrollbar and returns its event number.
Targets	Panel, Borderpanel, Window, Dialog, Frame, Scrollpane
Example	

```
:
scroll=j_vscrollbar(frame);
j_setpos(scroll,120,40);
j_setsize(scroll,20,100);
:
```



windowlistener

Synopsis	int **j_windowlistener** (int window , int kind);

Arguments	window	int
	kind	int

Description

Adds a new windowlistener to component **obj**, and returns its event number. An event occures, if the user action is of kind **kind**. Posible values for **kind**:

- J_ACTIVATED : An event occures when the component is activated.
- J_DEACTIVATED : An event occures when the component is deactivated.
- J_OPENED : An event occures when the component has been opened.
- J_CLOSED : An event occures when the component has been closed.
- J_ICONFIED : An event occures when the component is iconfied.
- J_DEICONFIED : An event occures when the component is deiconfied.
- J_CLOSING : An event occures when the close icon has been clicked .

Targets Window, Dialog, Frame

window

Synopsis int **j_window** (int obj);

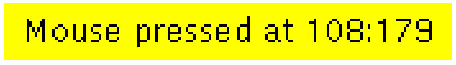
Arguments	obj	int

Description Creates a new simple window and returns its event number.

Targets Frame

Example

```
:
window = j_window(frame);
label  = j_label(window,"Mouse pressed at ... ");
j_setnamedcolorbg(label,J_YELLOW);
:
```


Mouse pressed at 108:179