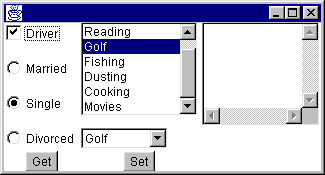
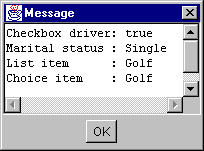
**Checkboxes, Radio Buttons, Scrolling Lists, and Choice Lists**

To our growing list of window controls we now add checkboxes, radio button groups, scrolling lists, and popup choice lists. Before diving into the details, we present a simple demonstration program. An initial screenshot for the demo program is below and contains:

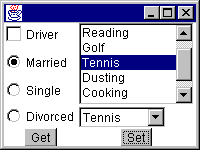
|  | A ***checkbox*** labeled **Driver** that is currently checked. |
| --- | --- |
|  | A ***radio button group*** containing buttons labeled **Married**, **Single**, and **Divorced** with the option **Single** selected. |
|  | A ***scrolling list box***. The list contains the words *Swimming*, *Reading*, *Golf*, *Fishing*, *Dusting*, *Cooking*, and *Movies*. Only some of these are visible, and the word *Golf* has been selected |
|  | A ***popup choice list***. The list contains the words *Swimming*, *Reading*, *Golf*, *Fishing*, *Dusting*, *Cooking*, and *Movies*, but because the word *Golf* has been selected, it is the only word visible. |
|  | A ***text area***. This area displays a message when the user selects an item in the scrolling list. |



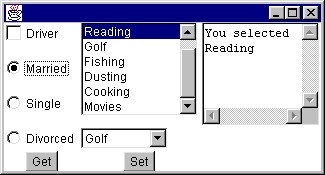
The user and the program can both manipulate the settings of the various window controls. The program made the settings shown above at startup. When the user selects the **Get** command, the program reads the current settings and displays them in a message box (shown next).



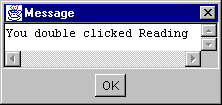
When the user selects the **Set** command, the program modifies the contents of the scrolling and popup lists and makes several selections, as shown next.



Whenever the user selects, with a single click, an item in a scrolling list, the program can respond to this selection. In the next illustration, a message is displayed in the text area.



Any time the user double clicks an item in a scrolling list, the program is informed of both a single click (see previous example) and a double click. Our demo program displays a message box indicating the item double-clicked, and also updates the text area with the results of a single click (the message box is shown next).



Now that we have seen a simple demonstration, we turn to some of the details.

***Checkboxes***

Checkboxes are declared in a now familiar manner, for instance,

Checkbox cbDriver = addCheckbox ("Driver", 1,1,1,1);

Here are four frequently used CheckBox methods:

| Name of Method | What the Method Does |
| --- | --- |
| setState (aBoolean) | If a Boolean is true, then draw a check mark else clear check mark. |
| getState()  returns boolean | Return true if checked else return false. |
| setLabel (aString) | Set the label to a string. |
| getLabel()  returns String | Return the current label. |

***Radio Buttons***

When checkboxes are placed in a group, they behave like radio buttons, meaning that selecting one automatically deselects the others. To use radio buttons, one must declare a CheckboxGroup, several checkboxes, and then add each checkbox to the group. Later, when the program needs to know which checkbox has been selected, it sends the getSelectedCheckbox() message to the checkbox group control. Here are relevant lines of code from the demo program:

CheckboxGroup cbgMaritalStatus = new CheckboxGroup(); // Radio button

Checkbox cbMarried = addCheckbox ("Married", 2,1,1,1); // group

Checkbox cbSingle = addCheckbox ("Single", 3,1,1,1); //

Checkbox cbDivorced = addCheckbox ("Divorced", 4,1,1,1); //

// Place the married, single, and divorced checkboxes in a

// radio button group, and select the "Single" option

cbMarried.setCheckboxGroup (cbgMaritalStatus);

cbSingle.setCheckboxGroup (cbgMaritalStatus);

cbDivorced.setCheckboxGroup (cbgMaritalStatus);

cbSingle.setState (true);

... cbgMaritalStatus.getSelectedCheckbox().getLabel()

The following table contains a summary of the methods just discussed:

| **Type of Control** | **Name of Method** | **What the Method Does** |
| --- | --- | --- |
| Checkbox | setCheckboxGroup  (aCheckboxGroup) | Add the checkbox to aCheckboxGroup. |
| Checkbox  Group | getSelectedCheckbox()  returns aCheckbox | Return the selected checkbox. |

***Scrolling Lists***

The strings in a scrolling list can be manipulated by the user and by the program. When the user clicks on a string, it is automatically highlighted. Later the program can query the list to determine which string the user selected. When the user double clicks on a string, it is highlighted, and the program is notified immediately. The program is also able to select, and thus highlight, a string. Finally, the program can add strings to and remove strings from the list. Here are some fragments of code that show how the demo program declares and manipulates a scrolling list.

List ltHobbies = addList (1,2,1,3); // Scrolling list

// Load up the list control

ltHobbies.add ("Swimming", 999); // 999 is larger than list size

ltHobbies.add ("Reading", 999); // therefore string added to

ltHobbies.add ("Golf", 999); // end of list

ltHobbies.add ("Fishing", 999);

ltHobbies.add ("Dusting", 999);

ltHobbies.add ("Cooking", 999);

ltHobbies.add ("Movies", 999);

ltHobbies.select (2); // select the 3rd string

// Determine which string has been selected and retrieve it

i = ltHobbies.getSelectedIndex();

selection = ltHobbies.getItem(i);

ltHobbies.remove (3); // Remove 4th item, "Fishing"

ltHobbies.add ("Tennis", 3); // Add "Tennis" in 4th position

ltHobbies.select (3); // Select 4th item

The next table summarizes the commonly used Java List methods. Note that positions within a list are zero based.

| **Name of Method** | **What the Method Does** |
| --- | --- |
| add(aString,anInteger)  returns void | To the list, add a string at the indicated position. If anInteger is –1 or is larger than the list’s size, add the string to the list’s end. |
| getItem(anInteger)  returns String | Return the string at the indicated position. An exception is thrown if anInteger is out of bounds. |
| getItemCount()  returns anInteger | Return the list’s length. |
| getItems() returns  arrayOfString | Return an array containing all the strings in the list. |
| getSelectedIndex()  returns int | Return the index of the selected item and –1 if no item is selected. |
| remove(anInteger)  returns void | Remove the indicated item. An exception is thrown if anInteger is out of bounds. |
| remove(aSring)  returns void | Remove the first instance of aString. An exception is thrown if aString is absent. |
| removeAll()  returns void | Remove all the items from the list. |
| replaceItem  (aString, anInteger) | Replace the item at the indicated position with aString. |
| select(anInteger)  returns void | Select the item at the indicated position. An exception is thrown if anInteger is out of bounds. |

The method addList is provided for your convenience by BreezyGUI. This package also provides two methods for responding to events that occur in scrolling lists, as shown in the next table.

| **Name of the Method** | **What the Method Does** |
| --- | --- |
| void listItemSelected  (List listObj) | The framework invokes this method when the user selects a list item with a single click or a double click. The application may or may not implement this method to take the appropriate action. The parameter is the list in which the item was selected. The programmer can use the List methods getSelectedIndex() and getItem(int) to determine the selected item. |
| void listDoubleClicked  (List listObj,  String itemClicked) | The framework invokes this method when a list item is double clicked. The application should implement this method to take the appropriate action. The parameters are the list and the list item where the event occurred. *Note*: This method is invoked *after* the method listItemSelected. |

Applications are free to ignore single clicks in a scrolling list and omit the implementation of listItemSelected. However, if the application is to ignore double clicks, the programmer must implement a listDoubleClicked method that does nothing. Otherwise, a message box displays a reminder to implement this method.

***Popup Choice Lists***

A popup choice list and a scrolling list look very different, but in other respects they have much in common. Their main difference, from the perspective of this demonstration, is that double clicking on a popup list has no effect. The next table summarizes some useful Choice methods:

| **Name of Method** | **What the Method Does** |
| --- | --- |
| insert  (aString,anInteger)  returns void | In the list, insert a string at the indicated position. If anInteger is larger than the list’s size, add the string to the list’s end. An exception is thrown if anInteger is negative. |
| getItem(anInteger)  returns aString | Return the string at the indicated position. An exception is thrown if anInteger is out of bounds. |
| getItemCount()  returns anInteger | Return the list’s length. |
| getSelectedIndex()  returns anInteger | Return the index of the selected item and –1 if no item is selected. |
| remove(anInteger)  returns void | Remove the indicated item. An exception is thrown if anInteger is out of bounds. |
| remove(aSring)  returns void | Remove the first instance of aString. An exception is thrown if aString is absent. |
| removeAll()  returns void | Remove all the items from the list. |
| select(anInteger)  returns void | Select the item at the indicated position. An exception is thrown if anInteger is out of bounds. |

***Code for the Demo***

Here is the complete code for the demonstration program.

import java.awt.\*;

import BreezyGUI.\*;

public class Test extends GBFrame{

// Declare the window controls. The variables have been given prefixes

// that indicate their type. Doing this is not necessary, but hopefully

// it makes the program more readable.

//

// Prefix Type of Control

// cb Checkbox

// ch Choice

// lt List

// bt Button

Checkbox cbDriver = addCheckbox ("Driver", 1,1,1,1); // Checkbox

CheckboxGroup cbgMaritalStatus = new CheckboxGroup(); // Radio button

Checkbox cbMarried = addCheckbox ("Married", 2,1,1,1); // group

Checkbox cbSingle = addCheckbox ("Single", 3,1,1,1); //

Checkbox cbDivorced = addCheckbox ("Divorced", 4,1,1,1); //

Choice chHobbies = addChoice (4,2,1,1); // Popup choice list

List ltHobbies = addList (1,2,1,3); // Scrolling list

TextArea ltHobbyOut = addTextArea ("", 1,3,1,3); // Output of selected

// list item.

Button btGet = addButton ("Get", 5,1,1,1); // Command buttons

Button btSet = addButton ("Set", 5,2,1,1);

// Constructor

public Test(){

// Load up the list control

ltHobbies.add ("Swimming", 999); // 999 larger than list size

ltHobbies.add ("Reading", 999); // therefore string added to

ltHobbies.add ("Golf", 999); // end of list

ltHobbies.add ("Fishing", 999);

ltHobbies.add ("Dusting", 999);

ltHobbies.add ("Cooking", 999);

ltHobbies.add ("Movies", 999);

ltHobbies.select (2); // select the 3rd string

// Load up the choice control

chHobbies.insert ("Swimming", 999); // 999 larger than list size

chHobbies.insert ("Reading", 999); // therefore inserts at end

chHobbies.insert ("Golf", 999); // of list

chHobbies.insert ("Fishing", 999);

chHobbies.insert ("Dusting", 999);

chHobbies.insert ("Cooking", 999);

chHobbies.insert ("Movies", 999);

chHobbies.select (2); // select the 3rd string

// Mark the driver checkbox

cbDriver.setState (true);

// Place the married, single, and divorced checkboxes in a

// radio button group, and select the "Single" option

cbMarried.setCheckboxGroup (cbgMaritalStatus);

cbSingle.setCheckboxGroup (cbgMaritalStatus);

cbDivorced.setCheckboxGroup (cbgMaritalStatus);

cbSingle.setState (true);

}

public void buttonClicked (Button buttonObj){

String str;

int i;

if (buttonObj == btGet){

// Read the data from the screen and display it in a message box.

// Examine the driver checkbox

str = "Checkbox driver: " + cbDriver.getState() + "\n";

// Examine the radio button group

str +=

"Marital status : " +

cbgMaritalStatus.getSelectedCheckbox().getLabel() + "\n";

// Examine the scrolling list

i = ltHobbies.getSelectedIndex();

str += "List item : " + ltHobbies.getItem(i) + "\n";

// Examine the popup choice list

i = chHobbies.getSelectedIndex();

str += "Choice item : " + chHobbies.getItem(i) + "\n";

// Display the data in a message box

messageBox (str);

}else{

// Modify some of the data on the screen.

cbDriver.setState (false); // Uncheck driver

cbMarried.setState (true); // Choose radio button "Married"

ltHobbies.remove (3); // Remove 4th item, "Fishing"

ltHobbies.add ("Tennis", 3); // Add "Tennis" in 4th position

ltHobbies.select (3); // Select 4th item

chHobbies.remove (3); // Remove 4th item, "Fishing"

chHobbies.insert ("Tennis", 3); // Add "Tennis" in 4th position

chHobbies.select (3); // Select 4th item

}

}

public void listDoubleClicked (List listObj, String itemClicked){

// Handle double clicks on a list item.

messageBox ("You double clicked " + itemClicked);

}

public void listItemSelected (List listObj){

// Handle selection (single click) of a list item.

int index = listObj.getSelectedIndex();

String item = listObj.getItem(index);

ltHobbyOut.setText("You selected \n" + item);

}

public static void main (String[] args){

Frame frm = new Test();

frm.setSize (325, 175);

frm.setVisible(true);

}

}