

Graphical User Interfaces

Advanced GUI Components

COMP2603

Object Oriented Programming 1

Week 7

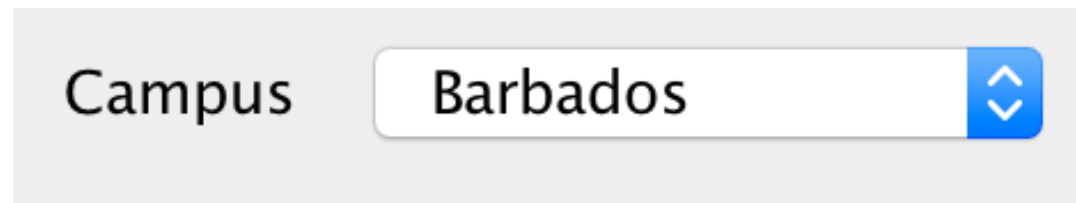
Outline

- Graphical User Interfaces
 - Advanced GUI Components
 - ComboBox
 - Radio Button
 - Check Box

Advanced GUI Components

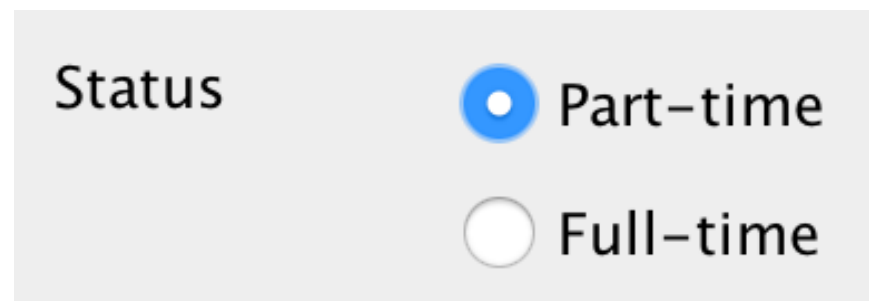
Three advanced GUI components are:

- Combo Box



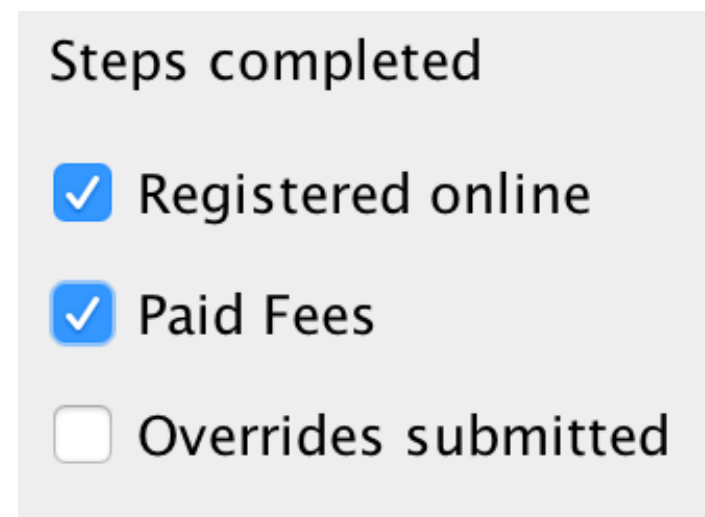
Campus Barbados

- Radio Button



Status ☒ Part-time
☐ Full-time

- Check Box



Steps completed

☒ Registered online
☒ Paid Fees
☐ Overrides submitted

Combo Box

A combo box combines a button with a drop-down list.

It is available in Swing as `JComboBox`

If a user clicks on the button, a drop-down list is displayed.

The user can then scroll down the drop-down list and select a value which is then displayed.

Combo Box Examples

Campus

Barbados



Automatic option displayed

Campus

✓ Barbados

Jamaica

Trinidad

Open Campus



Selecting from the list

Campus

Jamaica



Selected option displayed

Creating a Combo Box

```
/* short hand for creating an array and filling it with
data */
String[] countries = new String{"Barbados", "Jamaica",
"Trinidad", "Open Campus"};

// #1 Creating a combo box and passing in the array
JComboBox countriesCB = new JComboBox(countries);

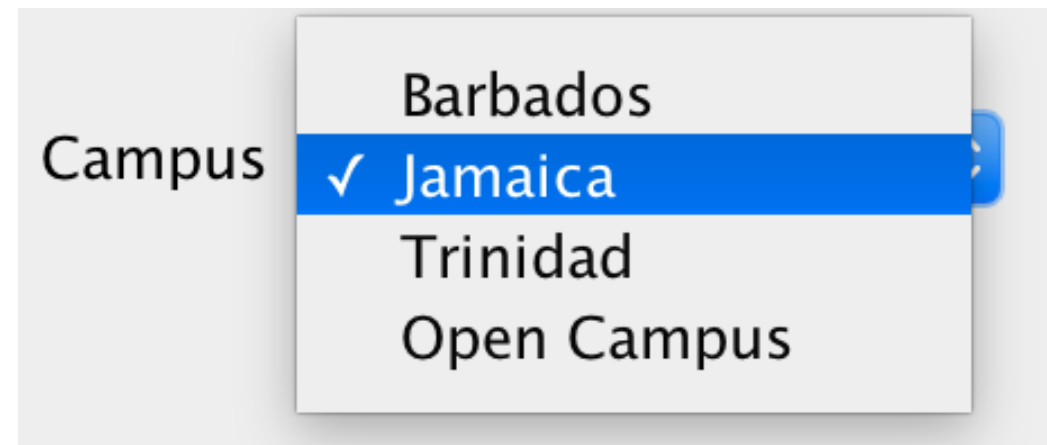
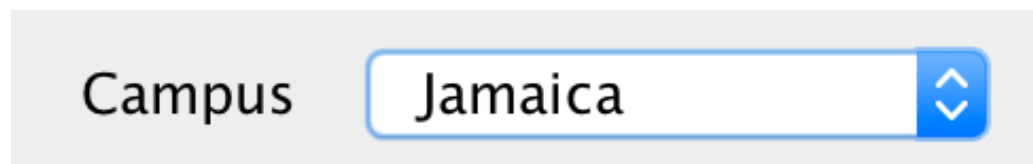
// #2 Creating a combo box and setting options
JComboBox countriesCB = new JComboBox();
countriesCB.addItem("Barbados");
countriesCB.addItem("Jamaica");
countriesCB.addItem("Trinidad");
countriesCB.addItem("Open Campus");
```

Combo Box Methods

Automatic Position of Selected Item

//#1 Automatically position combo box at particular option
`countriesCB.setSelectedIndex(1); // Jamaica`

//#2 Automatically position combo box at particular option
`countriesCB.setSelectedItem("Jamaica"); // Jamaica`



Combo Box Methods

Getting Value of Selected Item

```
/* Retrieve the Object selected from the combo box  
   and get its String representation */  
String country = countriesCB.getSelectedItem().toString();
```


Combo Box Methods

Setting the Value to a Selected Item

```
// Suppose we have a Student object with a campus location
String country = student.getCampusLocation();

/* If we want to set the combo box automatically to this
location, we have to write a method in our GUI class to parse
the data model of the combo box and extract the index of the
student's country. Return -1 if not found */
int countryIndex = getCountryIndex(country);

/* If valid, set the combo box to display country at that index
if(countryIndex >= 0)
    countriesCB.setSelectedIndex(countryIndex);
```

Radio Button

A radio button can be selected or de-selected by the user. It is available in Swing as `JRadioButton`.

A `ButtonGroup` object can be used to group together a set of `JRadioButton` objects so that only one `JRadioButton` can be selected at a time.

Radio Button

Status

☐ Part-time

☐ Full-time

No selection

Status

☒ Part-time

☐ Full-time

Selecting one option

Status

☒ Part-time

☒ Full-time

**Selecting both options.
If not correct, then a
ButtonGroup should be used**

Creating Radio Button Objects

```
//declare Radio Buttons
```

```
JRadioButton status1, status2;
```

```
//declare Button Group for grouping Radio Buttons together
```

```
ButtonGroup statusGroup;
```

```
//initialise Radio Buttons
```

```
status1 = new JRadioButton("Part-time");
```

```
status2 = new JRadioButton("Full-time");
```

```
//initialise Button Group
```

```
statusGroup = new ButtonGroup();
```

```
/* add Radio Buttons to Button Group -> only 1 can be  
selected now on the GUI */
```

```
statusGroup.add(status1);
```

```
statusGroup.add(status2);
```

Radio Button Method Automatic Selected Item

```
// Automatically select particular option  
status1.setSelected(true);
```

Status

☒ Part-time

☐ Full-time

Radio Button Methods

Getting Value of Selected Item

```
String status;    //for saving selected value
if(status1.isSelected())
    status = status1.getText(); //get displayed value
else
    status = status2.getText();
```

Status will be Full-time

Status

☐ Part-time

☒ Full-time

Check Box

A check box is similar to a radio button and can be selected or de-selected by the user.

A check mark is usually placed inside the check box to indicate it has been selected.

If a group of check boxes is used, the user can select as many as required.

It is available in Swing as `JCheckBox`.

Check Box

Steps completed

- ☐ Registered online
- ☐ Paid Fees
- ☐ Overrides submitted

No selection

Steps completed

- ☒ Registered online
- ☒ Paid Fees
- ☐ Overrides submitted

A few selections

Creating Check Boxes

```
//create a check box array
```

```
JCheckBox[] steps = new JCheckBox[3];
```

```
//enter options
```

```
steps[0] = new JCheckBox("Registered Online");
```

```
steps[1] = new JCheckBox("Paid Fees");
```

```
steps[2] = new JCheckBox("Overrides Submitted");
```

Getting Values of Check Boxes

//see slide on ArrayLists

```
ArrayList<String> stepsCompleted;
```

```
stepsCompleted = new ArrayList<String>();
```

```
for(int i = 0; i< steps.length; i++){ //for all boxes
    if(steps[i].isSelected()){ //if box is selected
        String label = steps[i].getText(); //get box value
        stepsCompleted.add(label); //add value to list
    }
}
```

Netbeans Editor