**JCheckbox**

Checkboxes are used for choices where a user can pick more than one option. This is good for "Pick any or all" type questions. They allow a user to select any or all of the checkboxes.

|  |
| --- |
| class Check extends JPanel implements itemListener{  boxes = **new** JCheckBox[size]; //an array of checkboxes    **for** (**int** i=0;i<size;i++){//loop through each of the checkboxes  boxes[i] = **new** JCheckBox();//initialize the boxes  boxes[i].setSelected(**false**);//make the boxes unselected  boxes[i].addItemListener(**this**);  /\*allows us to check if the box is checked, we add this because this is an object that implements itemListener. This checks for an ItemEvent which is an event that says if an object is selected or deselected. Using an itemListener requires the use of a method public void itemStateChanged(ItemEvent e) as this is what gets called when the itemListener picks up an itemEvent\*/  add(boxes[i]);  }  **public** **void** itemStateChanged(ItemEvent e) {/\*part of the itemListener interface, checks boxes, when an ItemEvent is fired, the ItemListener catches it and calls this method, passing the ItemEvent that was fired as e, this would be the selecting or deselecting of a checkbox\*/  **int** numberChecked=0;  **for** (**int** i=0;i<size;i++){//loop through each of the checkboxes  **if** (boxes[i].isSelected()) {//if it is selected  ++numberChecked;//here we are counting how many boxes are checked  }  }  **if** (numberChecked > toCheck) {//if too many boxes are checked  upperText.setText((numberChecked - toCheck) + " fewer!" );  }  **else** {//if not enough boxes are checked  upperText.setText((toCheck - numberChecked) + " more.");  }  }  } |

**JRadioButton**

A radio button is a button that allows only one option from a list to be chosen. This is useful for questions that can only have one answer per person. Yes and No questions are an example, you can be yes or no, not both.

|  |
| --- |
| class Radio extends JPanel implements ActionListener{  JRadioButton[] buttons = new JRadioButton[5];//array of buttons  JPanel panel = new JPanel();//panel to put all the buttons on  ButtonGroup group = new ButtonGroup();/\*putting the buttons in a group is how we know if one button is already selected\*/  **for** (**int** i = 0 ; i < buttons.length ; i++){//loop to initialize all the buttons  buttons[i] = **new** JRadioButton("Button "+i);//creates the button, and names it  buttons[i].addActionListener(**this**);  /\*Adds an ActionListener to tell us if the button is clicked, this is an object that implements ActionListener, this ActionListener checks for ActionEvents that are events specific to the thing that fired it, in this case, a JRadioButton. Using an actionListener requires the presence of a method called public void actionPerformed(ActionEvent e) because this is called every time an action event is fired and caught by the actionListener\*/  buttons[i].setMnemonic(KeyEvent.VK\_1 + i-1);/\*allows you to press a key instead of click \*\*HAVE TO PRESS ALT FOR IT TO WORK\*\* \*/  buttons[i].setAlignmentX(CENTER\_ALIGNMENT);  panel.add(buttons[i]);//adds the button to the panel  group.add(buttons[i]);//adds the button to the group so we can't select more than one  }  **public** **void** actionPerformed(ActionEvent e){/\*This method is called when an ActionEvent is caught by the ActionListener. In this case, all ActionEvents are JRadioButtons being clicked.\*/  **if**(buttons[choice].isSelected()){//checks if you selected the right button  label.setText("You hit the right button");//congratulates you  }  **else**{//if it isn't the right button  label.setText("That was the wrong button, please press button "+choice);/\*tells you to fix your mistake\*/  }  }  } |

**JButton**

A JButton is a button that can be clicked to do something instantly. Unlike the first two objects listed above, it doesn't track if something has been checked or not, you click it to fire an actionEvent, and that is all.

|  |
| --- |
| We really only have 3 pieces of code to look at for this type of button  JButton button = new JButton(String label);  The string passed in the constructor becomes a label on the button.  setEnabled(boolean bool)  this method sets whether or not the button can be clicked  public void actionPerformed(ActionEvent e){  /\*  Put stuff to do when the button is clicked here  \*/  } |