# Class 23 Notes
# Check Boxes

A component which allows a user to toggle a device on or off

Here is a JFrame with two check Boxes


Checking a box generates an ActionEvent (and also an ItemEvent but we will not need that)
Generates ActionEvent
Listener: ActionListener -- for ActionEvents
Method to implement: void actionPerformed(ActionEvent e) -- for ActionEvents

Constructors

- JCheckBox(String s)
  **ex. JCheckBox cb = new JCheckBox("Bold");**
- JCheckBox(String s, boolean selected) // true is box is to start as checked
  **ex. JCheckBox cb = new JCheckBox("Bold", true);**
- JCheckBox(Icon i) // you can "label" a check box with a picture
  ex. JCheckBox cb = new JCheckBox("new ImageIcon("Homer.jpg"));
- JcheckBox(Icon i, boolean selected) //

  JCheckBox(String s , Icon i) // label a check box with both a string and a picture
- 
- JcheckBox(String s, Icon I, boolean selected)

**Some Methods:**

- boolean isSelected() // returns true if the box is checked
- void setSelected(boolean x) // you program can check the box

**Sometimes , you do not need to use a listener—you can use the isSelected() method to determine whether or not a box is checked.**

Example:
The following example uses check boxes so that the user may decide how he/she wishes to display a message – bold, italic, bold and Italic, neither (plain)

No box selected – plain text

Bold selected

Bold and italic selected

Here is a class that creates the Abba dabba frame

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class CheckBoxes extends JFrame
{

 private JCheckBox  bold, italic;
 private JLabel  label;    //  for what is displayed  ---  Abba  dabba etc

 public CheckBoxes()  // constructor
 {

    String monkeyTalk = "Abba dabba dabba dabba dabba dabba dabba";
    setBounds(100,250,420,100);

   label = new JLabel(monkeyTalk);                    // put the string on a label
   label.setFont(new Font("Ariel",Font.PLAIN,18));  // set the font
   label.setForeground(Color.BLUE);                   // printing in blue

   JPanel labelPanel = new JPanel();                  // for the label
   labelPanel.setBackground(Color.YELLOW);
   labelPanel.add(label);                             // add the label to the panel
   add(labelPanel);                                  // add the panel to the CENTER (default)
```

```
   // Make a panel for the check boxes
   JPanel p = new JPanel();
   p.setBackground(Color.YELLOW);        // background for panel

   // create two check boxes
   bold = new JCheckBox("Bold");
   italic = new JCheckBox("Italic");

   // make the background of the check boxes yellow so they blend in with the panel
   bold.setBackground(Color.YELLOW);
   italic.setBackground(Color.YELLOW);

   //add the check boxes to a panel
   p.add(bold);  // add to panel
   p.add(italic );

   // add the panel to the frame
   add(p, BorderLayout.SOUTH);

   // Register listeners
   bold.addActionListener(new CheckBoxHandler());
   italic.addActionListener(new CheckBoxHandler());

   setVisible(true);
  }
  private class CheckBoxHandler implements ActionListener
  {
  public void actionPerformed(ActionEvent e)
  {
   if (e.getSource() == bold || e.getSource()== italic)
   {
    // examine EACH checkbox
     int selection =Font.PLAIN;        // or  you can say selection = 0;

     if (bold.isSelected())
          selection =  Font.BOLD;       // otherwise selection stays PLAIN

     if (italic.isSelected())
       selection = selection+Font.ITALIC;  // add BOLD to whatever was already selected

     // adjust the label using the above selection
     label.setFont(new Font("Arial", selection, 18));
   }
  }
}
public static void main(String[] args)
  {
   CheckBoxes frame = new CheckBoxes();
  }
}
```

I initially put the label directly on the frame but could not get the background to be yellow. When I made a panel, with background yellow, put the label in the panel, then put the panel in the frame the colors were fine. Tricky.

**Radio Buttons**

The frame below has three radio buttons (Black, Blue, and Green). Unlike the check boxes you can only select one radio button at any time.


**The class isJRadioButton.**

Generates ActionEvent,
Listener: ActionListener
Methods to implement: void actionPerformed(ActionEvent e) //ActionListener

**So how do you specify that only one radio button can be selected at any time?.**

- You create any number ofJRadionButton objects
- Then create a ButtonGroup.
- Then you add the buttons to the button group.

Once you do this, selection of one button automatically causes deselection of the others in the group.

A change in the selection causes .an event
The constructors and methods are similar to check boxes

**Constructors: si**
JRadioButton()
JRadioButton( String s)
JRadioButton(String s, boolean selected)
JRadioButton(Icon i)
JRadioButton(Icon i, boolean selected)
JRadioButton(String s, Icon i)
JRadioButton(String s, Icon i, boolean selected)

**Other Methods:**
boolean isSelected()
addActionListener(ActionListener a)
addItemListener(ItemListener a)

To make the ButtonGroup and add radio buttons:

**Constructor of the ButtonGroup Class**
ButtonGroup()
Ex : ButtonGroup. Bg = new ButtonGroup()

**Method of ButtonGroup class**
void add(JRadioButton b)

Ex.
JRadioButton  red = new JRadioButton(“Red”);
JRadioButton  blue= new JRadioButton(“Blue”);
bg.add(red);
bg.add(blue);

**Example:**

The following example adds a button group to the program of the previous problem.  The buttons can be used to change the color of the font:


```
// New code is highlighted
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class CheckBoxesAndRadioButtons extends JFrame
{
 private JCheckBox  bold, italic;
 private JRadioButton black,blue, green;
 private ButtonGroup bg;
 private JLabel  label;


 public CheckBoxesAndRadioButtons()   // constructor
 {
  super("Monkey Talk");
  setBounds(100,250,430,100);
  setBackground(Color.YELLOW);

  String monkeyTalk = "Abba dabba dabba dabba dabba dabba dabba";
  label = new JLabel(monkeyTalk);
  label.setFont(new Font("Ariel",Font.PLAIN,18));
  label.setForeground(Color.blue);

  add(label, BorderLayout.CENTER);  // add the label to the frame

  // Get a panel for the check boxes and radio buttons
  JPanel p = new JPanel();
  p.setBackground(Color.yellow);
```

```
// make the check boxes
bold = new JCheckBox("Bold");
italic = new JCheckBox("Italic");
bold.setBackground(Color.yellow);
italic.setBackground(Color.yellow);

// add the check boxes to the panel
p.add(bold);
p.add(italic );

//create radio buttons
black = new JRadioButton("Black",false);
blue = new JRadioButton("Blue",true);   // selected when program starts
green = new JRadioButton("Green",false);
black.setBackground(Color.yellow);
blue.setBackground(Color.yellow);
green.setBackground(Color.yellow);

//create the button group and place the buttons in the group
bg = new ButtonGroup();
bg.add(black);
bg.add(blue);
bg.add(green);

//add radio buttons to the panel
p.add(black);
p.add(blue);
p.add(green);

// add the panel to the frame
add(p, BorderLayout.SOUTH);
setVisible(true);

// Register listeners
bold.addActionListener(new BoxesAndButtons());
italic.addActionListener(new BoxesAndButtons());
blue.addActionListener(new BoxesAndButtons());
black.addActionListener(new BoxesAndButtons());
green.addActionListener(new BoxesAndButtons());
}
```

```
// inner listener class – for check boxes and radio buttons
  private class BoxesAndButtons implements ActionListener
  {
   public void actionPerformed (ActionEvent e)
   {
    if (e.getSource()  instanceof  JCheckBox )// notice use of instanceof
    {
     int selection = 0;    //0 is Font.PLAIN
     if (bold.isSelected())
       selection =  Font.BOLD;   // Font.BOLD == 1
     if (italic.isSelected())
       selection = selection+Font.ITALIC;
     label.setFont(new Font("Ariel", selection, 18));
    }
    else if (e.getSource() instanceof JRadioButton)
    {
      // find which is selected
     if (blue.isSelected())
       label.setForeground(Color.blue);  // print in blue
     else if (black.isSelected())
       label.setForeground(Color.black);
     else
       label.setForeground(Color.green);
    }
   }
  }
  public static void main(String[] args)
  {
   CheckBoxesAndRadioButtons frame = new CheckBoxesAndRadioButtons();
  }
}
```

And if you want to see the Abba Dabba song here it is in a VERY  old movie…with Debbie Reyonlds (Princess Leah's mom!)

https://www.youtube.com/watch?v=VJHJAkhacGU

Some notes

- When I created the blue radio button the second parameter is *true*
  **blue = new JRadioButton("Blue",true);**
  This makes the blue button selected when the program starts

- setForeground(Color c) is the color that is used for printing

- Notice I used the method
  boolean isSelected()

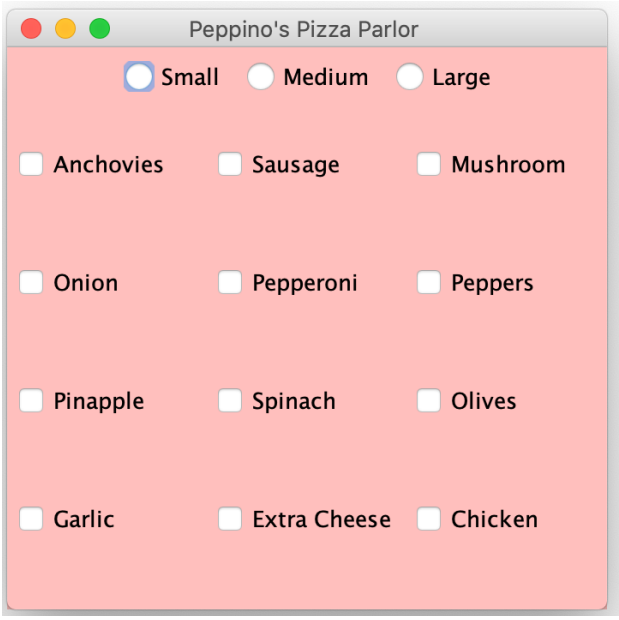
- You add the radio buttons to the panel—not the button group

- In the listener I used instanceof
  I could have also said
  if( e.getSource()== bold || e.getSource() == italic)
  but saying
  if (e.getSource() instanceof JCheckBox)
  covers both cases.

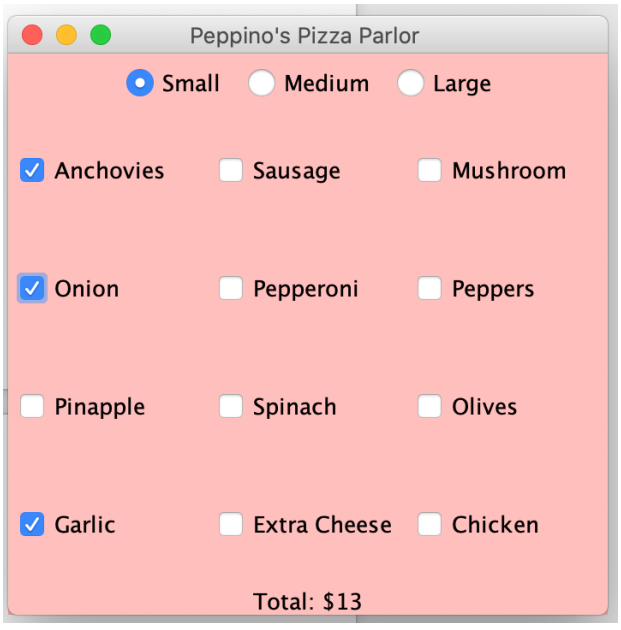
**Problem/Example**

At Peppino's Pizza Parlor a small pizza costs $10, a medium costs $12 ,and a large $15. Each additional topping is $1.00.

Design a GUI with check boxes and that will calculate the cost of a pizza. It should look like this.




When the buttons are selected the price is displayed on a label in the SOUTH




If you change any buttons the price is recalculated:


Notes:

- The radio buttons are on a panel in the NORTH. They will be in a ButtonGroup
- The check boxes are on a panel in the CENTER. The panel might use a GridLayout manager -- 4 x 3
- The price is a label in the SOUTH. The price should be in the center of the label (use SwingConstants.CENTER when creating the label)
- We will use an array of JCheckBox .
- The listener does not need to check the source. When a button or checkbox is selected, a event is fired and each is examined to see it it is selected or not. This will become clearer when you see the code

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class PizzaOrder1 extends JFrame
{
  // The names for the check boxes, one name for each check box
  String [] names =   {"anchovies", "sausage", "mushroom", "onion", "pepperoni", "peppers",
                       "pineapple", "spinach","olives", "garlic", "excheese", "chicken"};

  JCheckBox [] checkBoxes;  // notice an array of check boxes
   JRadioButton small,medium, large;
  JLabel price;
  int cost = 0;

  public PizzaOrder1()  //default constructor, sets up GUI
  {
    super("Peppino's Pizza Parlor ");
    setBounds(50,50, 350,350);
    setBackground(Color.PINK);

    // create the radio buttons, none are initially selected
    small = new JRadioButton("Small", false);
    medium = new JRadioButton("Medium", false);
    large = new JRadioButton("Large", false);

    // add radio buttons to a button group
    ButtonGroup bg = new ButtonGroup();
    bg.add(small);
    bg.add(medium);
    bg.add(large);

    // place the radio buttons in panel and place in the NORTH
    JPanel radioPanel = new JPanel();
    radioPanel.add(small);
    radioPanel.add(medium);
    radioPanel.add(large);
    add(radioPanel, BorderLayout.NORTH);

    //make  a label with the total cost and place  in the SOUTH
    price = new JLabel("  ", SwingConstants.CENTER);  // initially empty
    price.setFont(new Font("Arial", Font.BOLD, 28));
    add(price, BorderLayout.SOUTH);

    // a panel for all the check boxes
    JPanel menuPanel = new JPanel(new GridLayout(4,3));

    // create the array of check boxes and put each in the menuPanel
    checkBoxes = new JCheckBox[names.length];
    for (int i = 0; i < checkBoxes.length; i++)
    {
      checkBoxes[i] = new JCheckBox(names[i]);  // names[i] is the name of the check box
      menuPanel.add(checkBoxes[i]);
     }
```

```
    // add the menuPanel to the CENTER (default)
    add(menuPanel);

    //register the check boxes
    for (int i = 0; i < checkBoxes.length; i++)
      checkBoxes[i]. addActionListener(new Handler());

    // register the radio buttons
    small.addActionListener(new Handler());
    medium.addActionListener(new Handler());
    large.addActionListener(new Handler());

    setVisible(true);
  }

  // an event will occur if any check box or radio button
  // is selected or deselected
  private class Handler implements ActionListener
  {
    public void actionPerformed(ActionEvent e)
    {

      cost = 0;
      int count = 0;  // keep track of number of toppings  $1/topping

      {
        // check the radio buttons
        if (small.isSelected())
          cost = cost+ 10;
        else if (medium.isSelected())
          cost = cost+ 12;
         else if (large.isSelected())
          cost = cost+ 15;


       // how many check boxes have been selected?
       for (int i = 0; i<checkBoxes.length; i++)
          if (checkBoxes[i].isSelected())
             count++;

        cost = cost+count;  // "count check boxes" means "count dollars" added
        price.setText("Total: $" + cost);
    }
    }
  }

  public static void main(String[] args)
  {
    JFrame f = new PizzaOrder1();
  }
}
```

**Here is another version that does not use an array of check boxes. It is not a compact and (I think) not as nice. But, it does work.**
**Statements in blue are places that differ from the previous version**

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;


public class PizzaOrder extends JFrame
{
   // Declares each check box, does not use an array of check boxes
   JCheckBox  anchovies, sausage, mushroom, onion, pepperoni, peppers, pineapple,
                spinach,olives, garlic, excheese, chicken;

  JRadioButton small,medium, large;
  JLabel price;
  int cost = 0;

   public PizzaOrder()  // sets up GUI
  {
    super("Peppino's Pizza Parlor ");
    setBounds(50,50, 350,350);
    setBackground(Color.PINK);

    // the radio button stuff is  identical to the previous version
    small = new JRadioButton("Small", false);
    medium = new JRadioButton("Medium", false);
    large = new JRadioButton("Large", false);

    // add radio buttons to a button group
    ButtonGroup bg = new ButtonGroup();
    bg.add(small);
    bg.add(medium);
    bg.add(large);

    // place the radio buttons on a panel
    JPanel radioPanel = new JPanel();
    radioPanel.add(small);
    radioPanel.add(medium);
    radioPanel.add(large);
    add(radioPanel, BorderLayout.NORTH);


     // create the labe for the total cost
     price = new JLabel("  ", SwingConstants.CENTER);
    price.setFont(new Font("Arial", Font.BOLD, 28));
    add(price, BorderLayout.SOUTH);
```

```
// create a panel for the check boxes
JPanel menuPanel = new JPanel(new GridLayout(4,3));

// create wach individual check box
// the previous version used an array of JCheckBox
anchovies = new JcheckBox(“Anchovies”);
sausage  = new JcheckBox(“Sausage”);
mushroom  = new JcheckBox(“Mushroom”);
onion  = new JcheckBox(“Onion”);
pepperoni = new JcheckBox(“Pepperoni”);
peppers = new JcheckBox(“Peppers”);
pineapple = new JcheckBox(“pineapple”);
spinach = new JcheckBox(“Spinach”);
olives = new JcheckBox(“Olives”);
garlic = new JcheckBox(“Garlic”);
excheese = new JcheckBox(“Extra Cheese”) ;
chicken = new JcheckBox(“Chicken”);

 // add the check boxes to the panel, again this is done one by one
 menuPanel.add(anchovies);
 menuPanel.add(sausage);
 menuPanel.add(mushroom);
 menuPanel.add( onion);
 menuPanel.add(pepperoni);
 menuPanel.add(peppers);
 menuPanel.add(pineapple);
 menuPanel.add(spinach);
 menuPanel.add(olives);
 menuPanel.add(garlic);
 menuPanel.add(excheese);
 menuPanel.add(chicken);

 add(menuPanel);

 //register the radio buttons and check boxes
 // the previous version used a loop th register
 anchovies.addActionListener(new Handler());
 mushroom.addActionListener(new Handler());
 sausage.addActionListener(new Handler());
 onion.addActionListener(new Handler());
 pepperoni.addActionListener(new Handler());
 peppers.addActionListener(new Handler());
 pineapple.addActionListener(new Handler());
 spinach.addActionListener(new Handler());
 olives.addActionListener(new Handler());
 garlic.addActionListener(new Handler());
 excheese.addActionListener(new Handler());
 chicken.addActionListener(new Handler());
 small.addActionListener(new Handler());
 medium.addActionListener(new Handler());
 large.addActionListener(new Handler());
 setVisible(true);
}

private class Handler implements ActionListener
```

```
  {
    public void actionPerformed(ActionEvent e)
    {

      cost = 0;
      int count = 0;

      {
          if (small.isSelected())
            cost = cost+ 10;
          if (medium.isSelected())
             cost = cost+ 12;
          if (large.isSelected())
             cost = cost+ 15;


        // the previous version looped theough the array of check boxes

          if (anchovies.isSelected()) count++;
          if (sausage.isSelected()) count++;
          if (mushroom.isSelected()) count++;
          if (onion.isSelected()) count++;
          if (pepperoni.isSelected()) count++;
          if (peppers.isSelected()) count++;
          if (pineapple.isSelected()) count++;
          if (spinach.isSelected()) count++;
          if (olives.isSelected()) count++;
          if (garlic.isSelected()) count++;
          if (excheese.isSelected()) count++;
          if (chicken.isSelected()) count++;


        cost = cost+count;
        price.setText("Total: $" + cost);
    }
    }
  }

  public static void main(String[] args)
  {
       JFrame f = new PizzaOrder();
  }
}
```