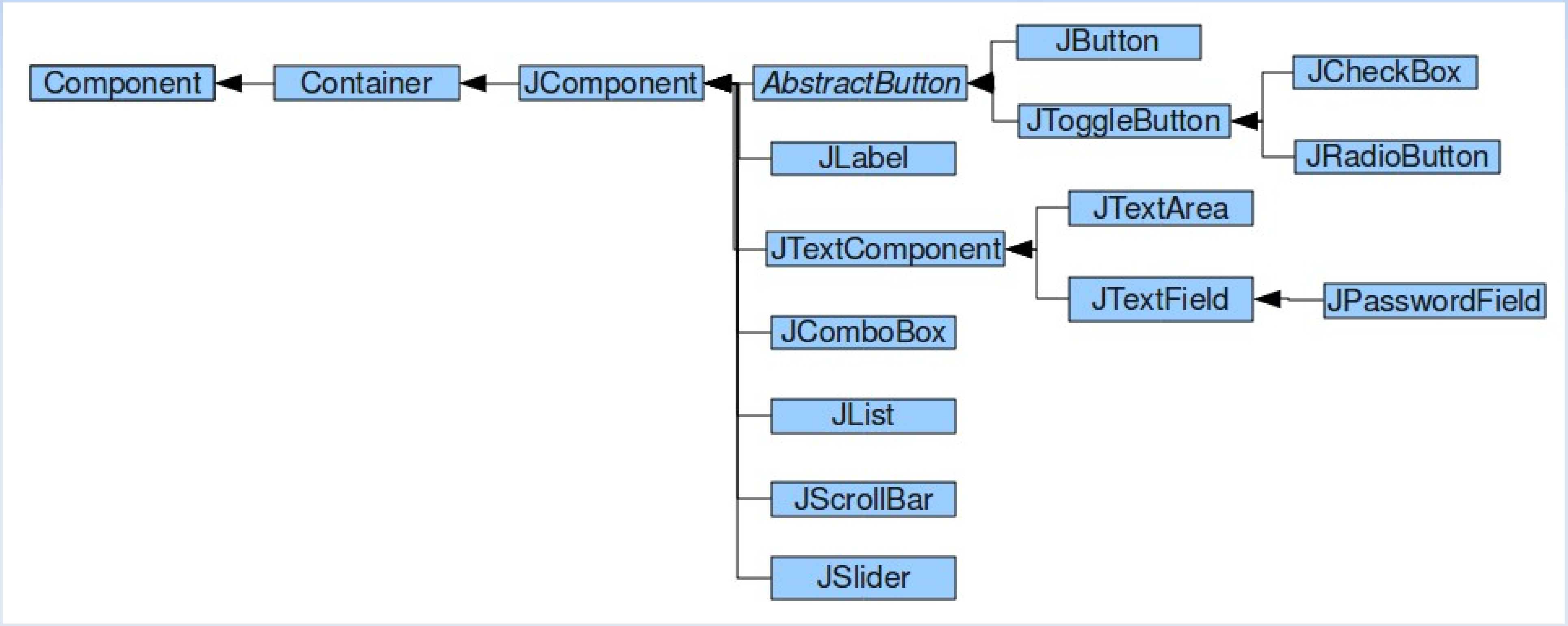
Chapter 16

Creating User Interfaces

Introduction

- So far we have covered how to make basic GUI interfaces, how to draw custom graphics onto a screen, and how to link those graphics and GUI's to listeners to perform useful tasks.
- Today we will be diving deeper into the swing library to expand upon our knowledge of basic components as well as lean about new components that will assist in making our GUI's more interactive and intuitive.

Commonly used swing components



AbstractButton

- We will begin our dive into the swing library by taking a closer look at AbstractButton

javax.swing.AbstractButton	
-actionCommand: String	The action command of the button (defualt == text)
-text: String	The text on the button
-icon: javax.swing.Icon	The buttons default icon
-pressedIcon: javax.swing.Icon	The icon displayed when the button is being pressed
-rolloverIcon: javax.swing.Icon	The icon displayed when the mouse is over the button
-mnemonic: int	The key code for the ALT+<KEY> shortcut for the button
-horizontalAlignment: int	horizontal alignment for icon and text (CENTER)
-horizontalTextPosition: int	horizontal text position relative to the icon(RIGHT)
-verticalAlignment: int	vertical alignment of icon and text(CENTER)
-verticalTextPosition: int	vertical text relative to the icon(CENTER)
-borderPainted: boolean	Indicates if the boarder for the button should be painted (TRUE)
-iconTextGap: int	The amount of space between the text and the icon
-selected(): boolean	State of the button, TRUE if check box, or radio button is selected

AbstractButton

- AbstractButton is the parent class of JButton, JRadioButton, and JCheckBox
- It defines that common methods that a button should use
- If you want to make your own button from scratch, you can extend the AbstractButton class.
- Lets look at JButton and see how to use the functionally provided by AbstractButton

JButton

- We will first go over setting the different icons on a button and see how they work
- We will then examine the different alignments for the text and icons.

JButton

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

public class ShowButtonMethods extends JFrame{
   private ImageIcon windowsIcon = new ImageIcon("images/win.jpg");
   private ImageIcon linuxIcon = new ImageIcon("images/linux.jpg");
   private ImageIcon macIcon = new ImageIcon("images/mac.jpg");
   private ImageIcon enabledIcon = new ImageIcon("images/user.jpg");
   private ImageIcon disabledIcon = new ImageIcon("images/user-
                                                    disabled.jpg");

   private JButton jbtButton1 = new JButton(windowsIcon);
   private JButton jbtButton2 = new JButton();


   public ShowButtonMethods(){
      JPanel panel = new JPanel();
      panel.setLayout(new GridLayout(1,2,5,5));

      panel.add(jbtButton1);
      panel.add(jbtButton2);
```

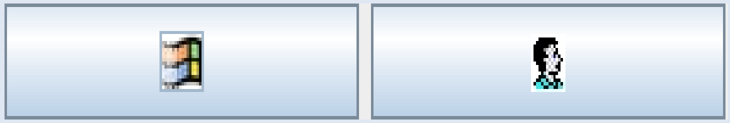
JButton

```
        jbtButton1.setPressedIcon(macIcon);
        jbtButton1.setRolloverIcon(linuxIcon);

        jbtButton2.setIcon(enabledIcon);
        jbtButton2.setDisabledIcon(disabledIcon);

        jbtButton1.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                if (jbtButton2.isEnabled()){
                    jbtButton2.setEnabled(false);
                }
                else{
                    jbtButton2.setEnabled(true);
                }
            }
        });

        add(panel);

    }
```

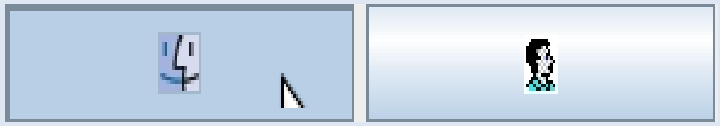
JButton

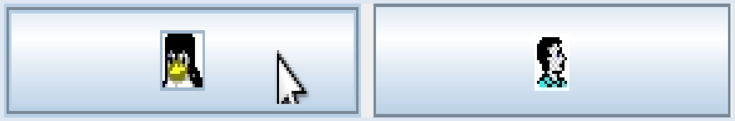
```
    public static void main(String[] args){
        JFrame frame = new ShowButtonMethods();
        frame.setTitle("Buttons with icons");
        frame.setSize(300,60);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Normal icons

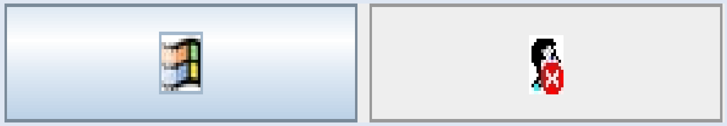


Click first button





Rollover first button



Button 2 disabled,
changed to disabled icon

JButton Alignments

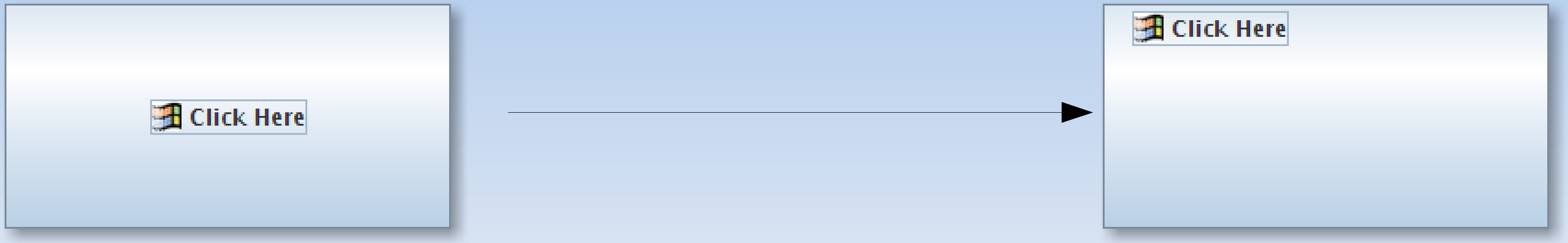
- You can move the icons and text around the button in different combinations to make the button more presentable
- Vertical alignment
- Horizontal alignment
- Vertical text alignment
- Horizontal text alignment

JButton Alignment

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

public class ShowButtonAlignments extends JFrame{
    private ImageIcon windowsIcon = new ImageIcon("images/win.jpg");
    private JButton jbtButton1 = new JButton("Click Here",
windowsIcon);

    public ShowButtonAlignments(){
        jbtButton1.setHorizontalAlignment(JButton.LEFT);
        jbtButton1.setVerticalAlignment(JButton.TOP);
        add(jbtButton1);

    }

    public static void main(String[] args){
        JFrame frame = new ShowButtonAlignments();
        frame.setTitle("Buttons with icons");
        frame.setSize(250,150);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

JButton Alignment

```
jbtButton1.setHorizontalAlignment(JButton.LEFT);
jbtButton1.setVerticalAlignment(JButton.TOP);
```



Normal

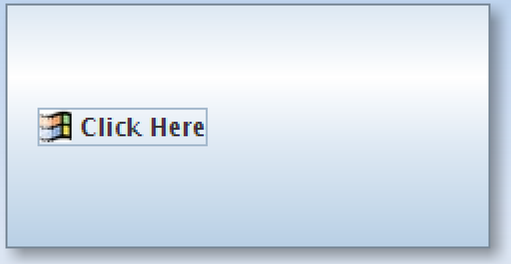
Left and top

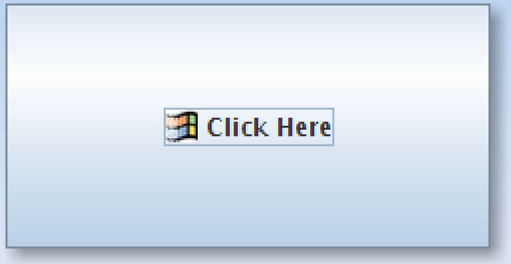
- Horizontal alignment can be adjusted by using the following constants
  - LEADING
  - LEFT
  - CENTER
  - RIGHT
  - TRAILING

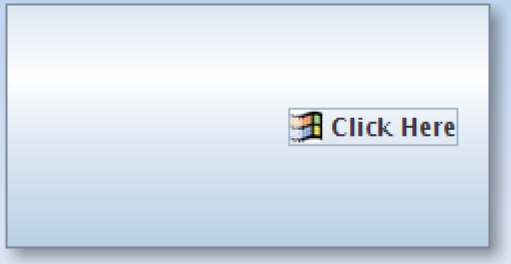
$\left.\begin{array}{l}\text{LEADING}\\ \text{LEFT}\end{array}\right\}$ Same behavior

$\left.\begin{array}{l}\text{RIGHT}\\ \text{TRAILING}\end{array}\right\}$ Same behavior

JButton Alignment







JButton.LEFT

JButton.CENTER

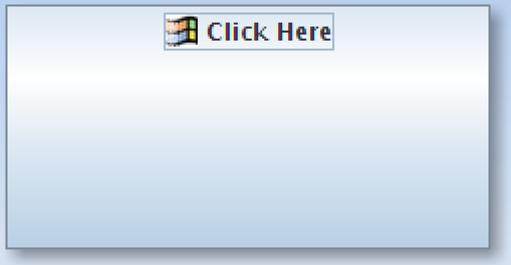
JButton.RIGHT

$\uparrow$

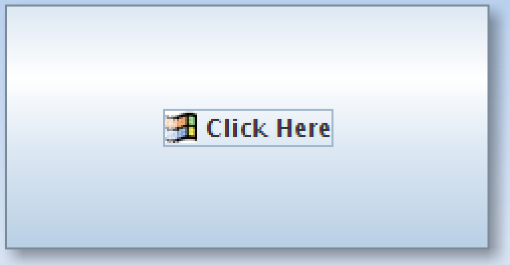
default

```
jbtButton1.setHorizontalAlignment(JButton.LEFT);
```

JButton Alignment



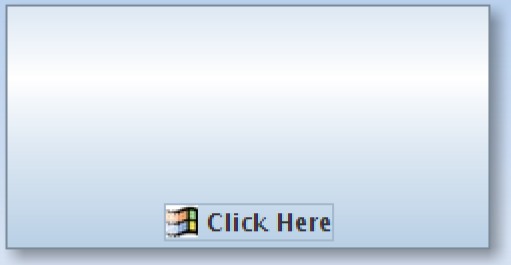
JButton.TOP



JButton.CENTER

↑

default



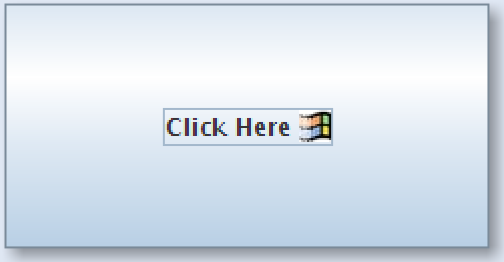
JButton.BOTTOM

```
jbtButton1.setVerticalAlignment(JButton.BOTTOM);
```

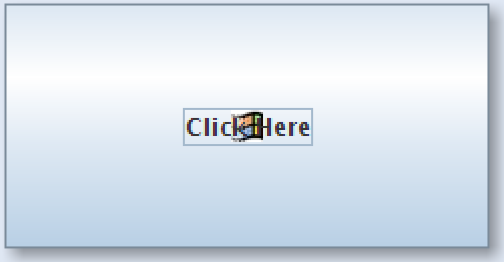
JButton Text Alignment

- You can also adjust where the text is relative to the icon

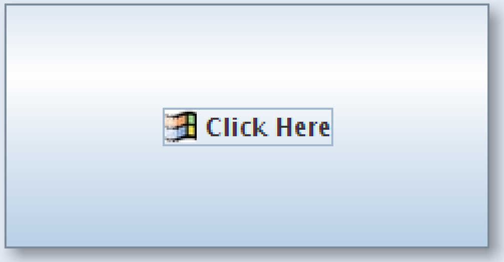
```
public ShowButtonAlignments(){
      jbtButton1.setHorizontalTextPosition(JButton.RIGHT);
      add(jbtButton1);

   }
```



JButton.LEFT



JButton.CENTER



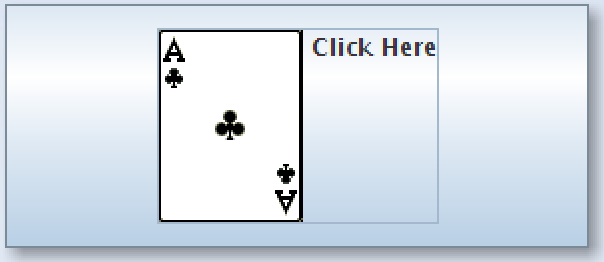
JButton.RIGHT

$\uparrow$

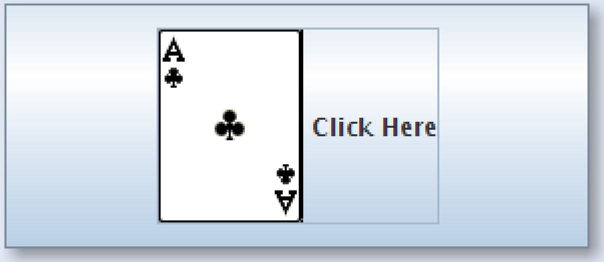
default

JButton Text Alignment

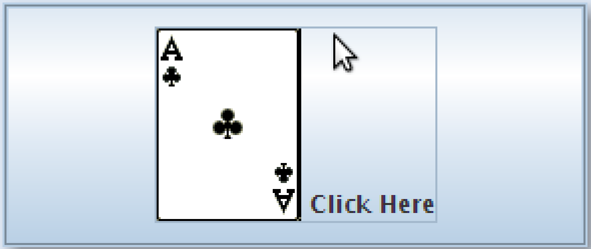
```
public ShowButtonAlignments(){
     jbtButton1.setVerticalTextPosition(JButton.TOP);
     add(jbtButton1);

  }
```



JButton.TOP



JButton.CENTER



JButton.BOTTOM

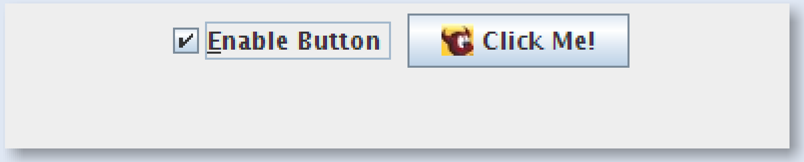
$\uparrow$

default

- The text is relative to the icon, so if the icon is too small, the text won't move!

Check Boxes

- Check boxes inherit from JToggleButton and represent a 2 state button (like a light switch)
- JCheckBox can be used to allow the user the ability to enable or disable a feature
- You can consider it a graphical representation of a boolean



JCheckBox

javax.swing.JCheckBox

JCheckBox()
JCheckBox(text: String)
JCheckBox(text: String, selected: boolean)
JCheckBox(icon: Icon);
JCheckBox(text: String, icon: Icon);
JCheckBox(text: String, icon: Icon, selected: boolean)
 	
IsSelected(): boolean
setMnemonic(c: char)

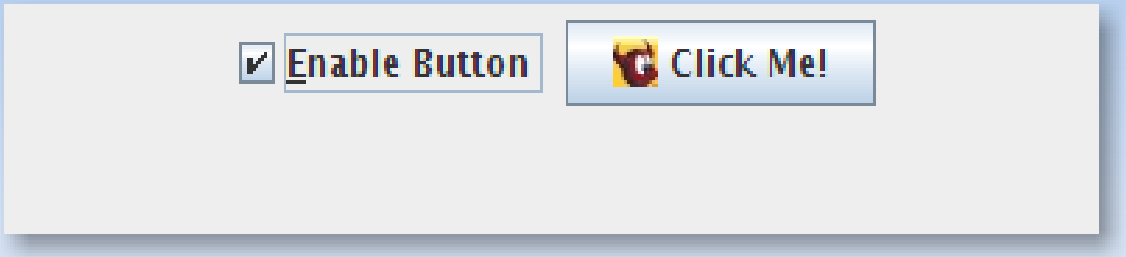
JCheckBox

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

public class CheckBoxDemo extends JFrame{
    private ImageIcon freebsdIcon = new ImageIcon("images/freebsd.jpg");
    private JButton jbtButton = new JButton("Click Me!", freebsdIcon);

    private JCheckBox jcbEnableButton = new JCheckBox("Enable Button ",
                                                                    true);

    public CheckBoxDemo(){
        JPanel panel = new JPanel(new FlowLayout());
        panel.add(jcbEnableButton);
        panel.add(jbtButton);

        jcbEnableButton.setMnemonic('E');
```

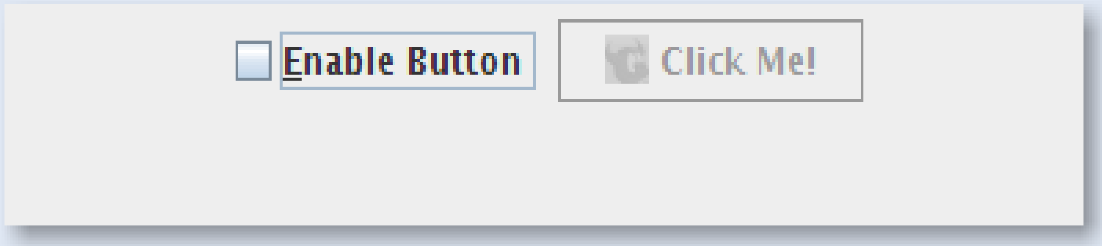
JCheckBox

```
        jcbEnableButton.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                if (jcbEnableButton.isSelected()){
                    jbtButton.setEnabled(true);
                }
                else{
                    jbtButton.setEnabled(false);
                }
            }
        });

        add(panel);
    }

    public static void main(String[] args){
        JFrame frame = new CheckBoxDemo();
        frame.setTitle("Check Box Demo");
        frame.setSize(400,100);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

JCheckBox

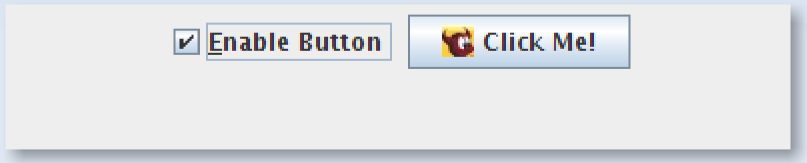


Uncheck the box, or press Alt+E to disable the button, press it again to re-enable it.

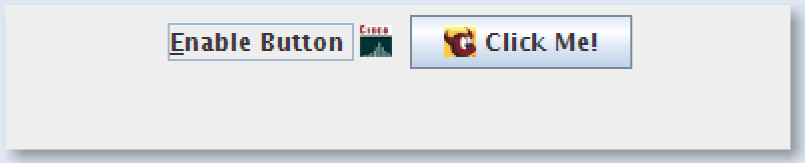


JCheckBox

- All of the formatting that you can do with the JButton apply to the JCheckBox



```
jcbEnableButton.setHorizontalTextPosition(JCheckBox.LEFT);
jcbEnableButton.setIcon(new ImageIcon("images/cisco.jpg"));
```



JRadioButton

- JRadioButtons are used to toggle between a set of mutually exclusive options.
- JradioButtons are added to ButtonGroups in order to make them mutually exclusive.
- ButtonGroups are only used to connect buttons together, a ButtonGroup does not extends component and therefor cannot be added to a Frame

JRadioButton

javax.swing.JRadioButton

JRadioButton()
JRadioButton(text: String)
JRadioButton(text: String, selected: boolean)
JRadioButton(icon: Icon);
JRadioButton(text: String, icon: Icon);
JRadioButton(text: String, icon: Icon, selected: boolean)
 	
IsSelected(): boolean
setMnemonic(c: char)

JRadioButton

```
import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.border.TitledBorder;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ChangeText extends JFrame implements ActionListener{
   TextBuilder textBuilder = new TextBuilder();
   private JCheckBox jchbBlinking = new JCheckBox("Blinking");
   private JRadioButton jrbBlack = new JRadioButton("Black");
   private JRadioButton jrbBlue = new JRadioButton("Blue");
   private JRadioButton jrbRed = new JRadioButton("Red");
   private JRadioButton jrbCyan = new JRadioButton("Cyan");
   private JRadioButton jrbGreen = new JRadioButton("Green");
   private JRadioButton jrbYellow = new JRadioButton("Yellow");
```

JRadioButton

```
public ChangeText() {
    JPanel jButtons = new JPanel();

    jButtons.setLayout(new GridLayout(1,7));
    jButtons.add(jrbBlack);
    jButtons.add(jrbBlue);
    jButtons.add(jrbRed);
    jButtons.add(jrbCyan);
    jButtons.add(jrbGreen);
    jButtons.add(jrbYellow);
    jButtons.add(jchbBlinking);
    jButtons.setBorder(new TitledBorder("Text Graphic Options"));

    ButtonGroup bg = new ButtonGroup();
    bg.add(jrbBlack);
    bg.add(jrbBlue);
    bg.add(jrbRed);
    bg.add(jrbCyan);
    bg.add(jrbGreen);
    bg.add(jrbYellow);
```

JRadioButton

```
        jrbBlack.addActionListener(this);
        jrbBlue.addActionListener(this);
        jrbRed.addActionListener(this);
        jrbCyan.addActionListener(this);
        jrbGreen.addActionListener(this);
        jrbYellow.addActionListener(this);

        jchbBlinking.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                textBuilder.isBlinking(jchbBlinking.isSelected());
            }
        });

        jrbBlack.setSelected(true);
        add(textBuilder, BorderLayout.CENTER);
        add(jButtons, BorderLayout.SOUTH);

    }
```

JRadioButton

```
public void actionPerformed(ActionEvent e){
    if (jrbBlack.isSelected()){
        textBuilder.setColor(Color.BLACK);
    }
    else if (jrbBlue.isSelected()){
        textBuilder.setColor(Color.BLUE);
    }
    else if (jrbRed.isSelected()){
        textBuilder.setColor(Color.RED);
    }
    else if (jrbCyan.isSelected()){
        textBuilder.setColor(Color.CYAN);
    }
    else if (jrbGreen.isSelected()){
        textBuilder.setColor(Color.GREEN);
    }
    else if (jrbYellow.isSelected()){
        textBuilder.setColor(Color.YELLOW);
    }

}
```

JRadioButton

```
class TextBuilder extends JPanel {
    private Timer blinkTimer = new Timer(100, new TimerListener());
    private Color colorChoice;
    private boolean blinking;
    private boolean blk = false;
    private String textToChange;

    public TextBuilder(){
        this(Color.BLACK,false,"Default Message");
    }

    public TextBuilder(String newString){
        this(Color.BLACK,false,newString);
    }

    public TextBuilder(Color colorChoice, boolean blinking, String
                                                                   message){
        this.colorChoice = colorChoice;
        this.blinking = blinking;
        this.textToChange = message;
    }
```

JRadioButton

```
public void setColor(Color newColor){
    this.colorChoice = newColor;
    repaint();
}

public Color getColor(){
    return colorChoice;
}

public void isBlinking(boolean blink){
    if (blink){
        startBlinking();
    }
    else {
        stopBlinking();
    }
}
public Dimension getPreferredSize() {
    return new Dimension(600,300);
}
```

JRadioButton

```
private void startBlinking(){
    blinking = true;
    blinkTimer.start();
}

private void stopBlinking(){
    blinking = false;
    blinkTimer.stop();
}

public void setMessage(String newMessage){
    textToChange = newMessage;
    repaint();
}
```

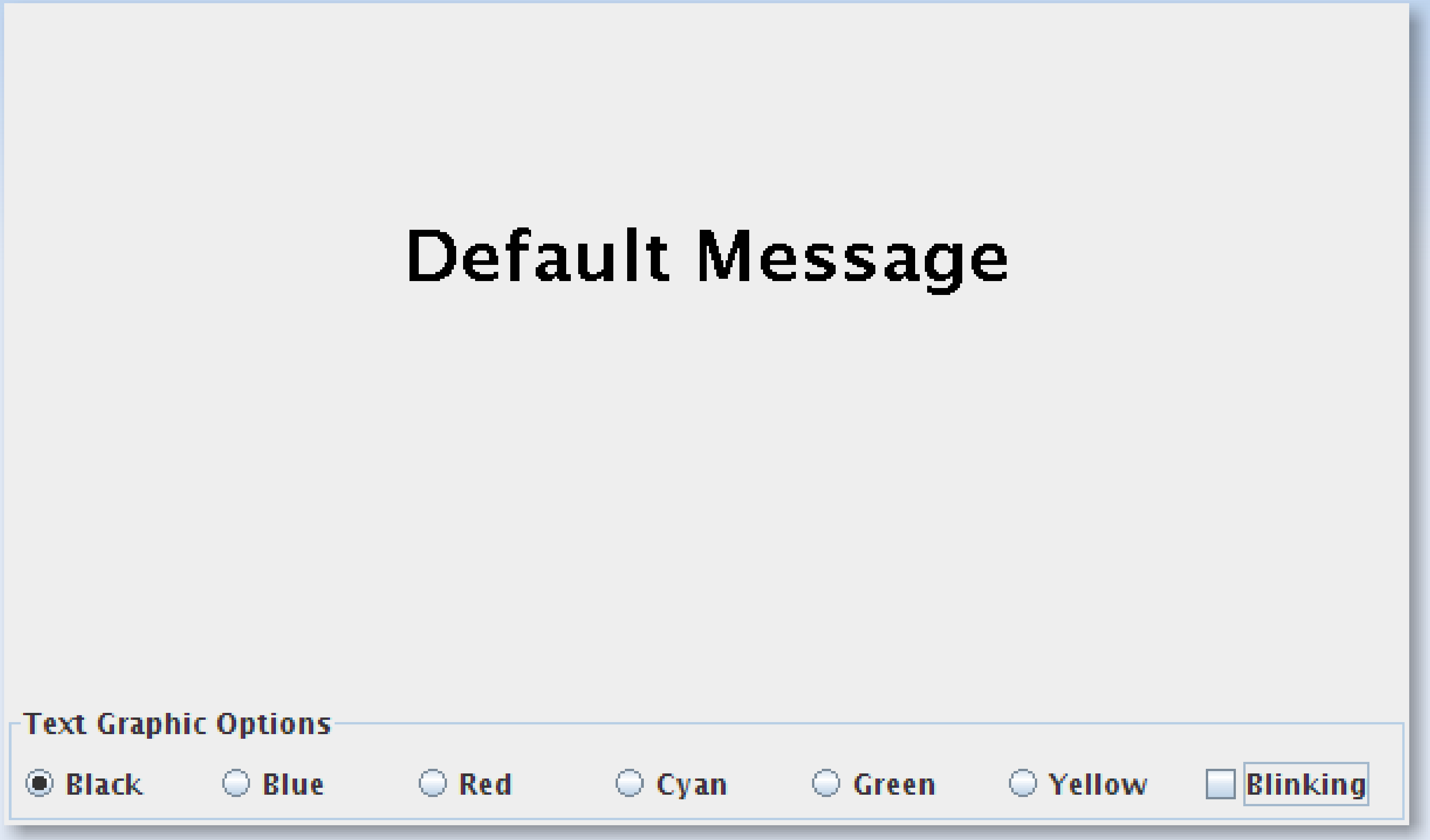
JRadioButton

```
protected void paintComponent(Graphics g){
    super.paintComponent(g);
    Font font = new Font(Font.SANS_SERIF, Font.BOLD, 30);
    g.setFont(font);
    FontMetrics fm = g.getFontMetrics();

    g.setColor(colorChoice);

    if (blinking){
        if (blk){
            g.drawString(textToChange, (getWidth() / 2) -
 fm.stringWidth(textToChange) / 2, (getHeight() / 2) - fm.getAscent());
        }
        else {
            g.setColor(Color.WHITE);
            g.drawString(textToChange, (getWidth() / 2) -
 fm.stringWidth(textToChange) / 2, (getHeight() / 2) - fm.getAscent());
        }
    }
    else {
        g.drawString(textToChange, (getWidth() / 2) -
 fm.stringWidth(textToChange) / 2, (getHeight() / 2) - fm.getAscent());
    }

}
```

JRadioButton

```
        class TimerListener implements ActionListener {
            public void actionPerformed(ActionEvent e) {
                if (blk){
                    blk = false;
                }
                else {
                    blk = true;
                }
                repaint();
            }

        }

    }

}
```

JRadioButton

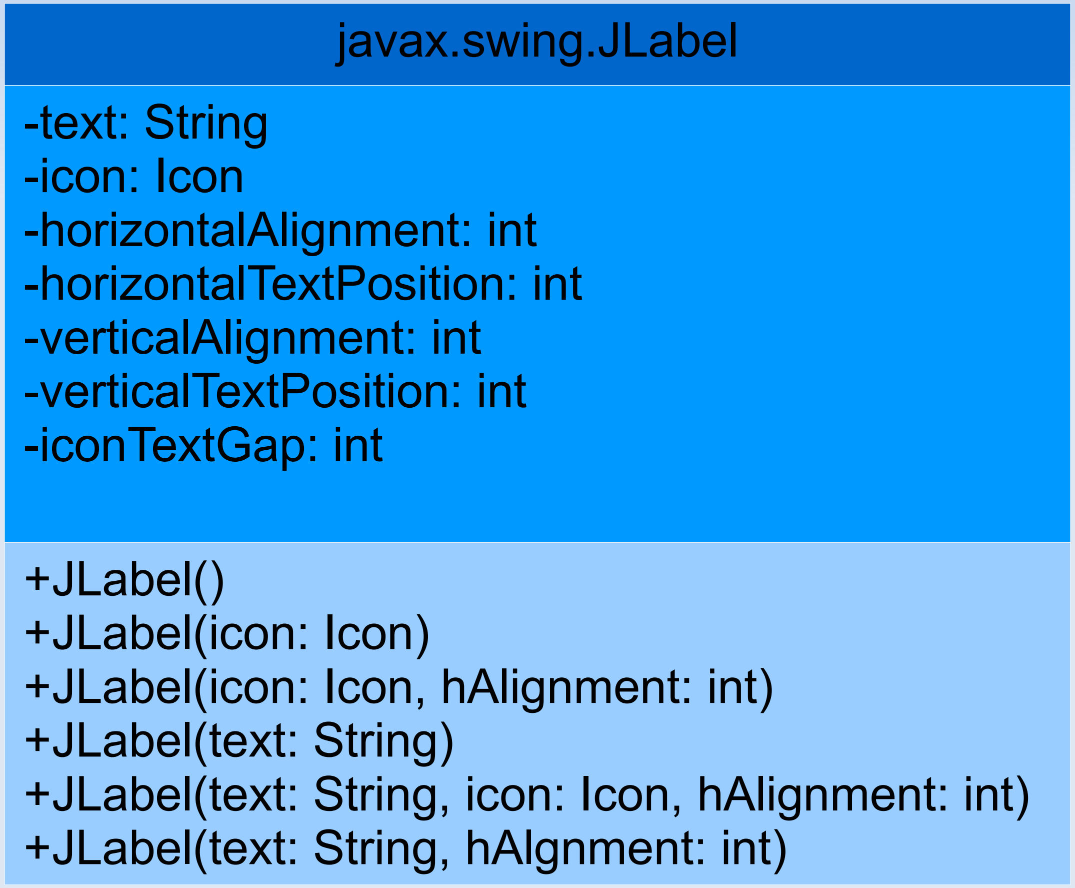

Default Message

Text Graphic Options

(● Black) ○ Blue ○ Red ○ Cyan ○ Green ○ Yellow ☐ Blinking

JLabel

- A JLabel can display text, images or both



javax.swing.JLabel

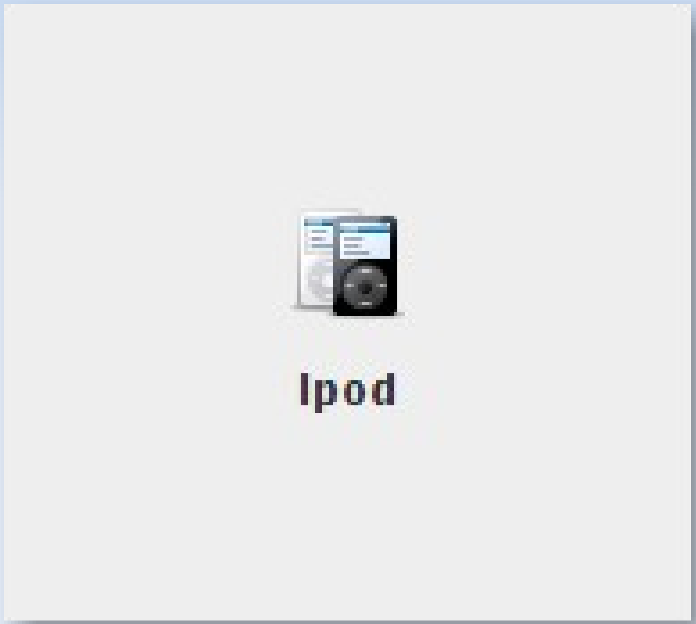
-text: String
-icon: Icon
-horizontalAlignment: int
-horizontalTextPosition: int
-verticalAlignment: int
-verticalTextPosition: int
-iconTextGap: int

+JLabel()
+JLabel(icon: Icon)
+JLabel(icon: Icon, hAlignment: int)
+JLabel(text: String)
+JLabel(text: String, icon: Icon, hAlignment: int)
+JLabel(text: String, hAlgnment: int)

JLabel

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

public class LabelDemo extends JFrame{
    private ImageIcon ipodsIcon = new ImageIcon("images/ipods.png");
    private JLabel jlblIpods = new JLabel("Ipod", ipodsIcon,
                                                    JLabel.CENTER);

    public LabelDemo(){
        jlblIpods.setHorizontalTextPosition(JLabel.CENTER);
        jlblIpods.setVerticalTextPosition(JLabel.BOTTOM);
        jlblIpods.setIconTextGap(10);
        add(jlblIpods);
    }

    public static void main(String[] args){
        JFrame frame = new LabelDemo();
        frame.setTitle("Label Demo");
        frame.setSize(200,200);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }

}
```

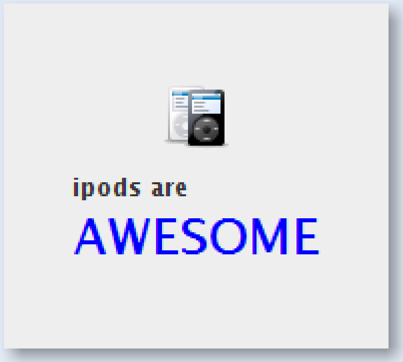
JLabel



Jlabel (not in book)

- Did you know you can style a JLabel using HTML?
- A lot (but not all) of swing components can interpret HTML

```
private ImageIcon ipodsIcon = new ImageIcon("images/ipods.png");
String textString = "<HTML>ipods are<P><FONT color=blue" +
                                    "size=+2>AWESOME</font></html>";
private JLabel jlblIpods = new JLabel(textString, ipodsIcon, JLabel.CENTER);
```


ipods are
AWESOME

HTML Support

- The swing components that can render HTML are
  - JLabel
  - JButton
  - JToggleButton
  - JMenu
  - JMenuItem
  - JComboBox
  - JList
  - Tabs of the JTabbedPane
  - JTable
  - JTree

TextFields

- A text field can be used to enter or display a string
- When you press enter on a JTextField, it fires and ActionEvent
- We can use these action events to pass the data typed into the JTextField into a String and use it in our program

JTextFields

javax.swing.JTextField
-columns: int -horizontalAlignment: int -text: String -editable: boolean

+JTextField()
+JTextField(column: int)
+JTextField(text: String)
+JTextField(text: String, columns: int)

JTextFields

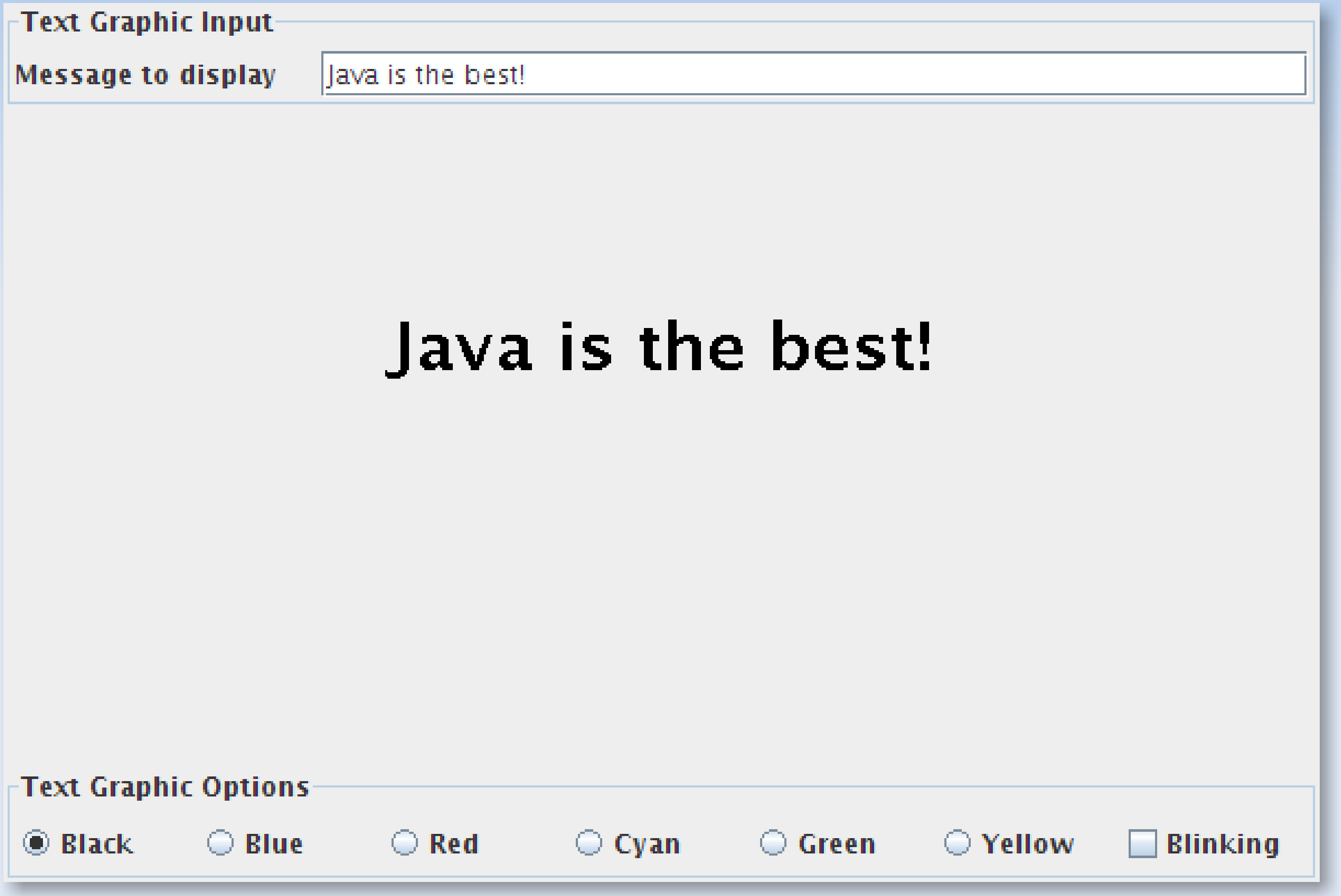
- Lets edit our JRadioButton example to include the ability to change the default message

```
private JRadioButton jrbGreen = new JRadioButton("Green");
private JRadioButton jrbYellow = new JRadioButton("Yellow");
private JTextField jtfMessageToDisplay = new JtextField(15);

public ChangeText() {
        JPanel jButtons = new JPanel();
        JPanel jText = new JPanel();
        jText.setLayout(new BorderLayout());
        jText.add(new JLabel("Message to display     "), BorderLayout.WEST);
        jText.add(jtfMessageToDisplay, BorderLayout.CENTER);
        jText.setBorder(new TitledBorder("Text Graphic Input"));
```

JTextField

```
jtfMessageToDisplay.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e){
            textBuilder.setMessage(jtfMessageToDisplay.getText());
            jtfMessageToDisplay.requestFocusInWindow();
        }
    });


    jrbBlack.setSelected(true);
    add(jText, BorderLayout.NORTH);
    add(textBuilder, BorderLayout.CENTER);
    add(jButtons, BorderLayout.SOUTH);

}
```

JTextField



Text Graphic Input

Message to display: Java is the best!

Java is the best!

Text Graphic Options

(● Black) ○ Blue ○ Red ○ Cyan ○ Green ○ Yellow ☐ Blinking

JTextArea

- JTextAreas are used if you want to enter, or display multiple lines of text
- JTextAreas cannot scroll, but they can be added to JScrollPanes which can handle the scrolling for them

javax.swing.JTextArea
-columns: int -rows: int -tabSize: int -lineWrap: boolean -wordStyleWrap: boolean

JTextArea()
JTextArea(rows: int, columns: int
JTextArea(text: String)
JTextArea(text: String, rows: int, columns: int)
 	
append(s: String)
insert(s: String, pos: int)
replaceRange(s: String, start: int, end: int)
getLineCount(): int

JTextArea

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


public class TextAreaDemo extends JFrame{
    private JTextArea jtaDisplay = new JTextArea();
    private JRadioButton jrbWrapWords = new JRadioButton("Wrap Words", true);
    private JRadioButton jrbWrapChar = new JRadioButton("Wrap Characters");
    private JCheckBox jcbWrap = new JCheckBox("Wrap", true);

    public TextAreaDemo(){
        jtaDisplay.setLineWrap(true);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(1,3,5,5));

        ButtonGroup wrapGroup = new ButtonGroup();
        wrapGroup.add(jrbWrapWords);
        wrapGroup.add(jrbWrapChar);
```

JTextArea

```
buttonPanel.add(jcbWrap);
buttonPanel.add(jrbWrapWords);
buttonPanel.add(jrbWrapChar);

buttonPanel.setBorder(new TitledBorder("Wrap Options"));

jcbWrap.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        if (jcbWrap.isSelected()){
            jrbWrapWords.setEnabled(true);
            jrbWrapChar.setEnabled(true);
            if(jrbWrapWords.isSelected()){
                jtaDisplay.setLineWrap(true);
                jtaDisplay.setWrapStyleWord(true);
            }
            else{
                jtaDisplay.setLineWrap(true);
                jtaDisplay.setWrapStyleWord(false);
            }
        }
        else{
            jrbWrapWords.setEnabled(false);
            jrbWrapChar.setEnabled(false);
            jtaDisplay.setLineWrap(false);
            jtaDisplay.setWrapStyleWord(false);
        }
    }
});
```

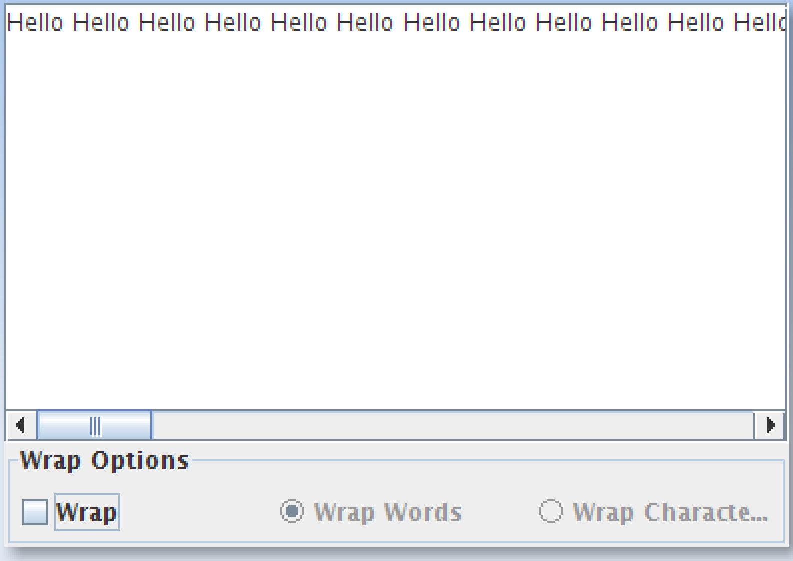
JTextArea

```
    jrbWrapWords.addActionListener(new ActionListener(){
       public void actionPerformed(ActionEvent e){
          jtaDisplay.setWrapStyleWord(true);
       }
    });

    jrbWrapChar.addActionListener(new ActionListener(){
       public void actionPerformed(ActionEvent e){
          jtaDisplay.setWrapStyleWord(false);
       }
    });

    setLayout(new BorderLayout());
    add(new JScrollPane(jtaDisplay), BorderLayout.CENTER);
    add(buttonPanel, BorderLayout.SOUTH);

}
```

JTextArea

```
    public static void main(String[] args){
        JFrame frame = new TextAreaDemo();
        frame.setTitle("Text Area Demo");
        frame.setSize(400,300);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

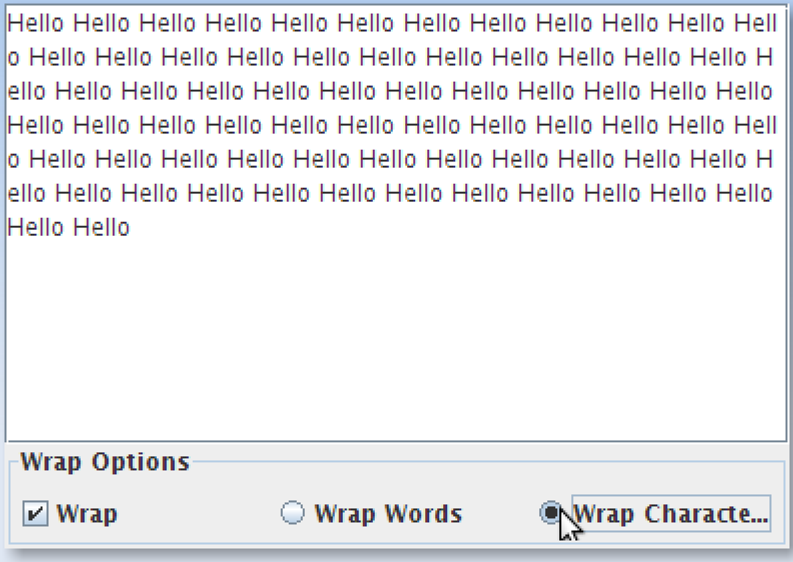
JTextArea



```
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
```

Wrap Options

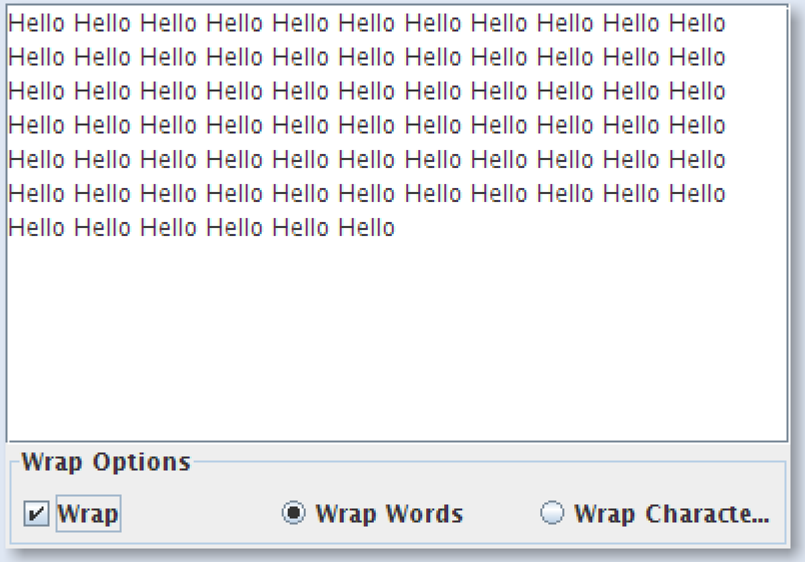
Wrap [✓ Wrap Words] Wrap Characte...



```
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hell
o Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello H
ello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hell
o Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello H
ello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello
```

Wrap Options

[✓ Wrap] Wrap Words [✓ Wrap Characte...]



```
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello
```

Wrap Options

[✓ Wrap] [✓ Wrap Words] Wrap Characte...

JComboBox

- A JComboBox represents a drop-down list of choices that a user can make.
- It is useful to limit the users range of choices and eliminate the need for data validation

javax.swing.JComboBox

JcomboBox()
JcomboBox(items: Object[])
addItem(item: Object)
getItemAt(index: int): Object
getItemCount(): int
getSelectedIndex(): int
setSelectedIndex(index: int)
getSelectedItem(): Object
setSelectedItem(item: Object)
removeItem(obj: Object)
removeItemAt(index: int)
removeAllIItems()

JComboBox

```
public class DisplayFont extends JFrame {
    FontBuilder fb = new FontBuilder();
    private JComboBox jcbFontSelection = new JComboBox(fb.getAvilableFonts());
    private JComboBox jcbFontSizes = new JComboBox(fb.getAvilableFontSizes());

    public DisplayFont() {
        JPanel boxPanel = new JPanel();
        boxPanel.setLayout(new GridLayout(2,2,5,5));
        boxPanel.add(new JLabel("     Font name "));
        boxPanel.add(jcbFontSelection);
        boxPanel.add(new JLabel("     Font size "));
        boxPanel.add(jcbFontSizes);

        jcbFontSelection.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                fb.setFontName(jcbFontSelection.getSelectedItem().toString());
            }
        });
        jcbFontSizes.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                fb.setSize(jcbFontSizes.getSelectedIndex() + 1);
            }
        });
        boxPanel.setBorder(new TitledBorder("Set Font and Size"));
        add(boxPanel, BorderLayout.NORTH);
        fb.setBackground(Color.WHITE);
        add(fb);
    }
```

JComboBox

```
class FontBuilder extends JPanel {
        private static final long serialVersionUID = 5549814319828008883L;
        private String selectedFontName;
        private Font selectedFont;
        private int fontSize;
        private String[] avilableFontSizes = new String[30];
        private String[] avilableFonts;

        public FontBuilder() {
            fontSize = 14;
            selectedFontName = Font.SANS_SERIF;
            getSystemFonts();
            setSystemFontSizes();
        }

        private void getSystemFonts() {
            GraphicsEnvironment e =
                    GraphicsEnvironment.getLocalGraphicsEnvironment();
            avilableFonts = e.getAvailableFontFamilyNames();
        }

        private void setSystemFontSizes() {
            for (int i = 1; i < 31; i++) {
                avilableFontSizes[i - 1] = new Integer(i).toString();
            }
        }
```

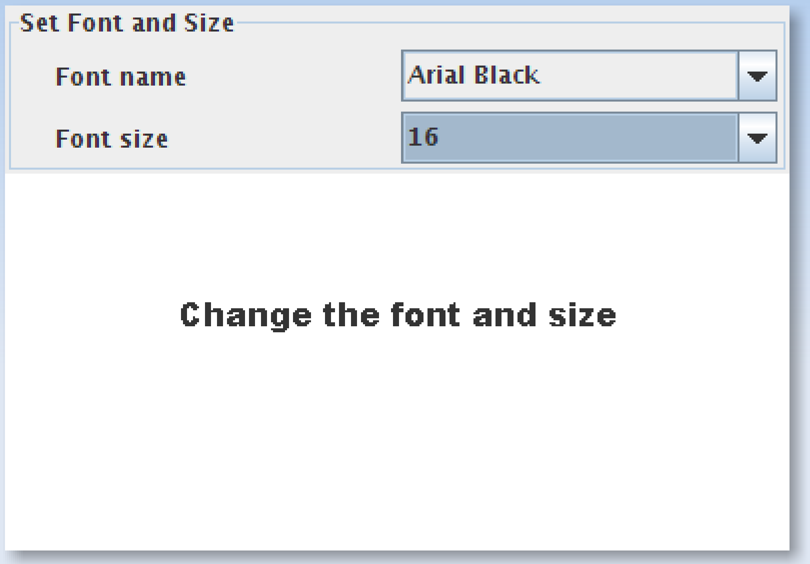
JComboBox

```
public String[] getAvilableFontSizes() {
        return avilableFontSizes;
}

public String[] getAvilableFonts() {
        return avilableFonts;
}

public void setSize(int newSize) {
        fontSize = newSize;
        repaint();
}

public void setFontName(String newFontToUse) {
        selectedFontName = newFontToUse;
        repaint();
}
```

JComboBox

```
        protected void paintComponent(Graphics g){
            super.paintComponent(g);
            selectedFont = new Font(selectedFontName,
                                            Font.PLAIN, fontSize);
            g.setFont(selectedFont);
            g.drawString("Change the font and size",
                    (getWidth() / 2 - (g.getFontMetrics().stringWidth("Change"+
                    "the font and size") / 2)), getHeight() / 2 -
                    (g.getFontMetrics().getAscent()));

        }

    }

    public static void main(String[] args) {
        JFrame frame = new DisplayFont();
        frame.setTitle("Avilable Fonts");
        frame.setSize(400,300);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);


    }
}
```

JComboBox

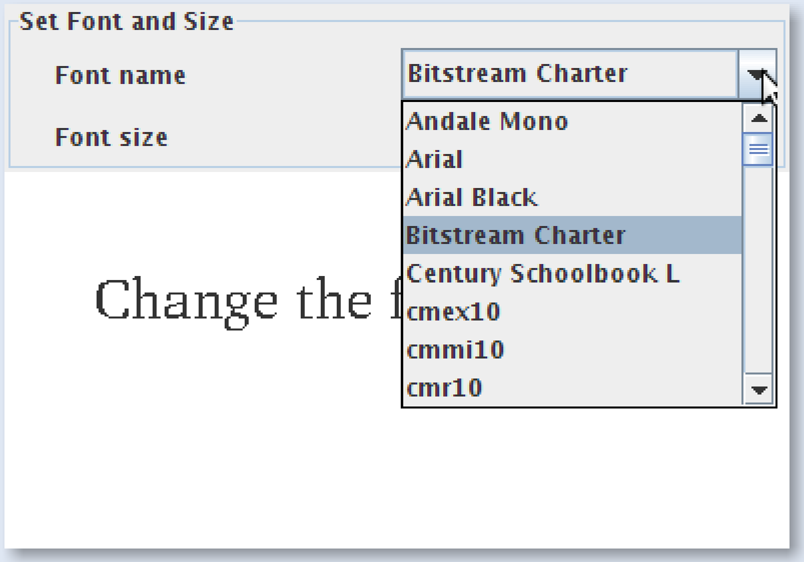


Set Font and Size

Font name: Arial Black

Font size: 16

Change the font and size



Set Font and Size

Font name: Bitstream Charter

Font size

Change the f

Lists

- JLists are just like JComboBoxes except that they allow you to make multiple selections.
- They can be set to allow single, group, or multi-interval selections

javax.swing.JList
-selectedIndex: int -selectedIndices: int[] -selectedValue: Object -visibleRowCount: int -selectionBackground: Color -selectionForeground: Color -selectionMode: int

JList()
JList(items: Object[])

JList

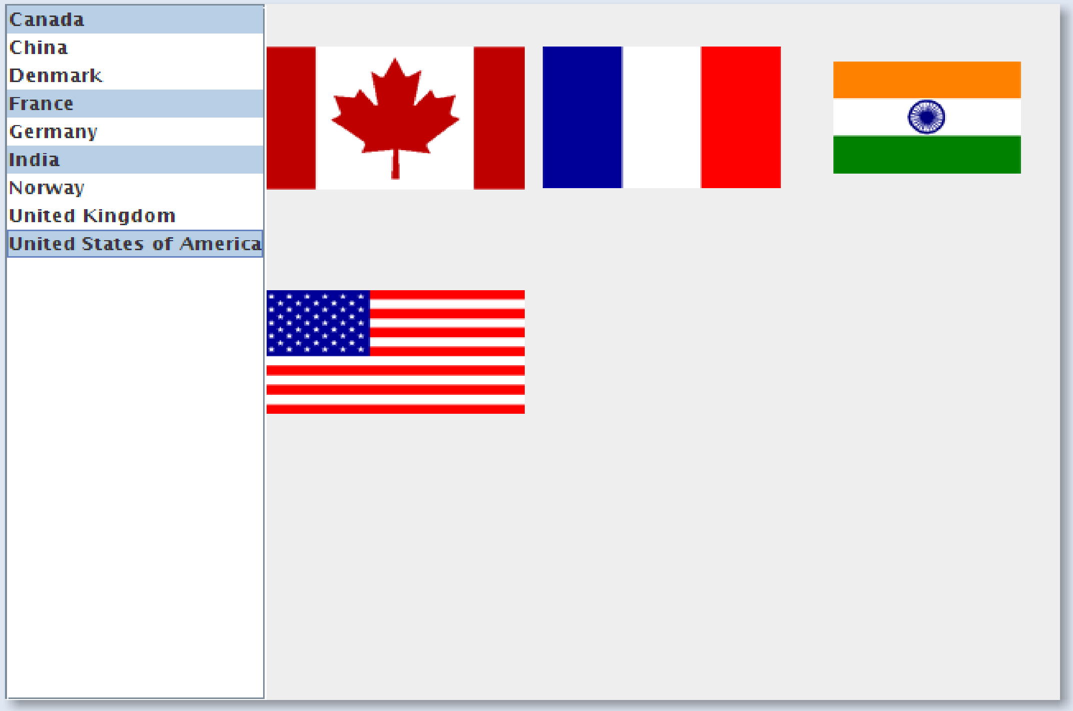
```
import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;

public class ListDemo extends JFrame {
   final int NUMBER_OF_FLAGS = 9;

   private String[] flagTitles = {
      "Canada", "China", "Denmark",
      "France", "Germany", "India", "Norway", "United Kingdom",
      "United States of America"};

   private JList jlst = new JList(flagTitles);
   private JLabel[] jlblImageViewer = new JLabel[NUMBER_OF_FLAGS];
   private ImageIcon[] imageIcons = {
         new ImageIcon("image/ca.gif"),
         new ImageIcon("image/china.gif"),
         new ImageIcon("image/denmark.gif"),
         new ImageIcon("image/fr.gif"),
         new ImageIcon("image/germany.gif"),
         new ImageIcon("image/india.gif"),
         new ImageIcon("image/norway.gif"),
         new ImageIcon("image/uk.gif"),
         new ImageIcon("image/us.gif")
   };
```

JList

```
public ListDemo() {
    JPanel p = new JPanel(new GridLayout(3, 3, 5, 5));

    for (int i = 0; i < NUMBER_OF_FLAGS; i++) {
       p.add(jlblImageViewer[i] = new JLabel());
       jlblImageViewer[i].setHorizontalAlignment
       (SwingConstants.CENTER);
    }

    add(p, BorderLayout.CENTER);
    add(new JScrollPane(jlst), BorderLayout.WEST);

    jlst.addListSelectionListener(new ListSelectionListener() {
       public void valueChanged(ListSelectionEvent e) {
          int[] indices = jlst.getSelectedIndices();

          int i;
          for (i = 0; i < indices.length; i++) {
             jlblImageViewer[i].setIcon(imageIcons[indices[i]]);
          }

          for (; i < NUMBER_OF_FLAGS; i++) {
             jlblImageViewer[i].setIcon(null);
          }
       }
    });
}
```

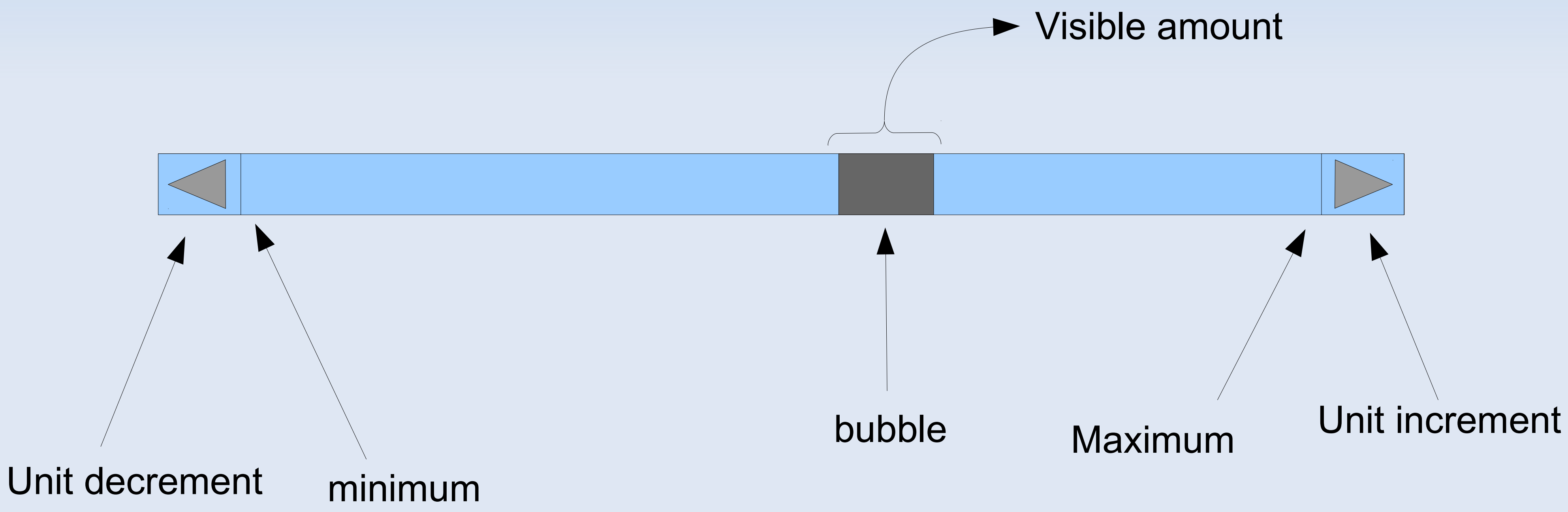
JList

```
    public static void main(String[] args) {
        ListDemo frame = new ListDemo();
        frame.setSize(650, 500);
        frame.setTitle("ListDemo");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



Scroll Bars

- JScrollBar is a component that enables the user to select from a range of values



JScrollBar

javax.swing.JScrollBar
-orientation: int -Maximum: int -minimum: int -visibleAmount: int -value: int -blockImcrement: int -unitIncrement: int

JscrollBar()
JscrollBar(orentation: int)
JscrollBar(orientation: int, value: int, extent: int, min: int, max: int)

The "extent" is the size of the viewable area. It is also known as the "visible amount".

JScrollBar

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

public class ScrollBarDemo extends JFrame {
    private JScrollBar jscbHort =
        new JScrollBar(JScrollBar.HORIZONTAL);
    private JScrollBar jscbVert =
        new JScrollBar(JScrollBar.VERTICAL);

    private MessagePanel messagePanel =
        new MessagePanel();

    public static void main(String[] args) {
        ScrollBarDemo frame = new ScrollBarDemo();
        frame.setTitle("ScrollBarDemo");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
```

JScrollBar

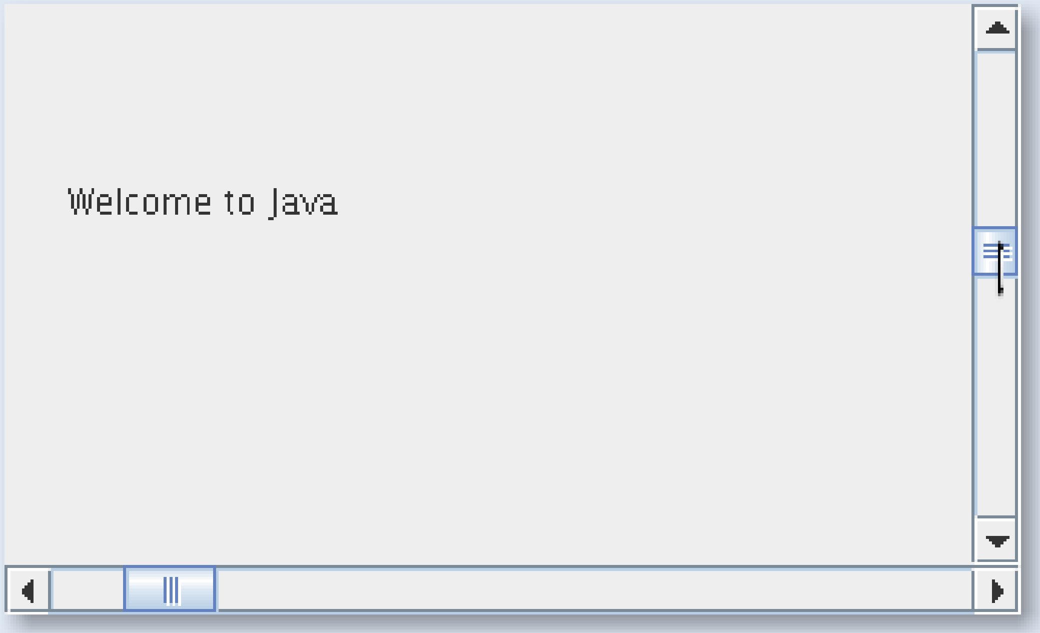
```
public ScrollBarDemo() {
    setLayout(new BorderLayout());
    add(messagePanel, BorderLayout.CENTER);
    add(jscbVert, BorderLayout.EAST);
    add(jscbHort, BorderLayout.SOUTH);

    jscbHort.addAdjustmentListener(new AdjustmentListener() {
        public void adjustmentValueChanged(AdjustmentEvent e) {
            double value = jscbHort.getValue();
            double maximumValue = jscbHort.getMaximum();
            double newX = (value * messagePanel.getWidth() /
                    maximumValue);
            messagePanel.setXCoordinate((int)newX);
        }
    });
    jscbVert.addAdjustmentListener(new AdjustmentListener() {
        public void adjustmentValueChanged(AdjustmentEvent e) {
            double value = jscbVert.getValue();
            double maximumValue = jscbVert.getMaximum();
            double newY = (value * messagePanel.getHeight() /
                    maximumValue);
            messagePanel.setYCoordinate((int)newY);
        }
    });
}
```

JScrollBar

```
class MessagePanel extends JPanel {
    private String message = "Welcome to Java";
    private int xCoordinate = 20;
    private int yCoordinate = 20;

    public MessagePanel() {

    }

    public int getXCoordinate() {
        return xCoordinate;
    }

    public void setXCoordinate(int x) {
        this.xCoordinate = x;
        repaint();
    }

    public int getYCoordinate() {
        return yCoordinate;
    }

    public void setYCoordinate(int y) {
        this.yCoordinate = y;
        repaint();
    }
```

JScrollBar

```
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.drawString(message, xCoordinate, yCoordinate);
        }

        public Dimension getPreferredSize() {
            return new Dimension(200, 30);
        }
    }

}
```



Welcome to Java

Using Multiple Windows

- Frames can not contain frames, but a program can use more then one frame to get job done.
- You need a main frame that will handle all of the other frames.
- All of the other frames must have there DefaultCloseOperation set to Jframe.DISPOSE_ON_CLOSE or the program will exit each time you close a frame

Using Multiple Windows

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

public class DemoSelector extends JFrame{
    private JButton jbtCheckBox = new JButton("Check Box Demo");
    private JButton jbtChangeText = new JButton("Change Text Demo");
    private JButton jbtLabelDemo = new JButton("Label Demo");
    private JButton jbtTextAreaDemo = new JButton("Text Area Demo");

    public DemoSelector(){
        setLayout(new GridLayout(2,2,5,5));
        add(jbtCheckBox);
        add(jbtChangeText);
        add(jbtLabelDemo);
        add(jbtTextAreaDemo);

        jbtCheckBox.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                JFrame frame = new CheckBoxDemo();
                frame.setTitle("Check Box Demo");
                frame.setSize(400,100);
                frame.setLocationRelativeTo(null);
                frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                frame.setVisible(true);
            }
        });
```

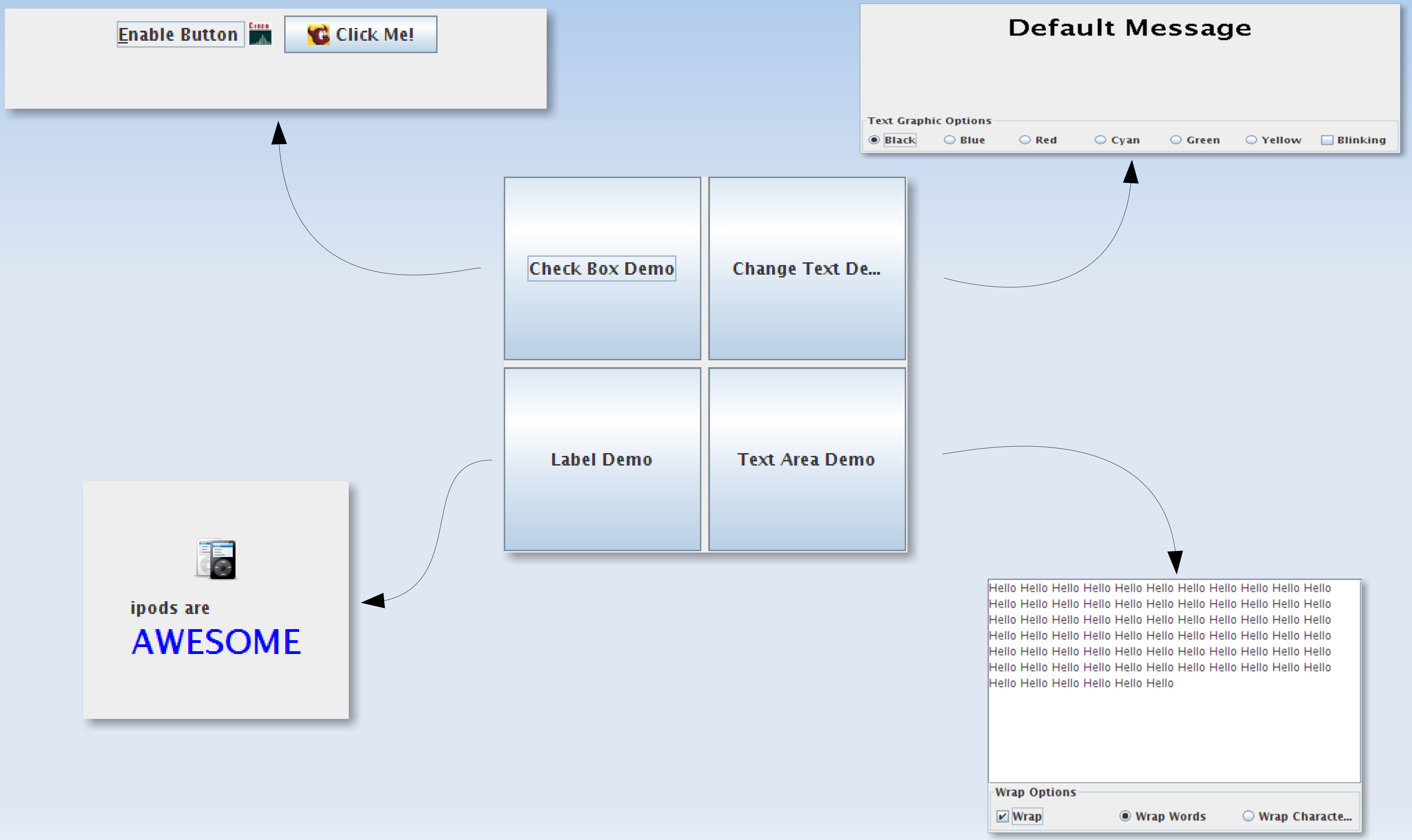
Using Multiple Windows

```
jJbtChangeText.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent e){
      JFrame frame = new ChangeText();
      frame.setTitle("Choose a style");
      frame.pack();
      frame.setLocationRelativeTo(null);
      frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      frame.setVisible(true);
   }
});

jbtLabelDemo.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent e){
      JFrame frame = new LabelDemo();
      frame.setTitle("Label Demo");
      frame.setSize(200,200);
      frame.setLocationRelativeTo(null);
      frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      frame.setVisible(true);
   }
});
```

Using Multiple Windows

```
        jbtTextAreaDemo.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                JFrame frame = new TextAreaDemo();
                frame.setTitle("Text Area Demo");
                frame.setSize(400,300);
                frame.setLocationRelativeTo(null);
                frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                frame.setVisible(true);
            }
        });

    }

    public static void main(String[] args){
        JFrame frame = new DemoSelector();
        frame.setTitle("Demo Selector");
        frame.setSize(300,300);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Using Multiple Windows



Text Graphic Options: [✓ Black] Blue Red Cyan Green Yellow ☐ Blinking

```
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello Hello
```

Wrap Options: ☑ Wrap [✓ Wrap Words] Wrap Characte...

Lab Assignment

- 16.6* Investment Calculator

Homework

- 16.9* Jlabel Properities
- 16.3** Traffic Light

Advanced Homework

- 16.17*** Calendar

Acknowledgments

Introduction to Java Programming by Y. Daniel Liang