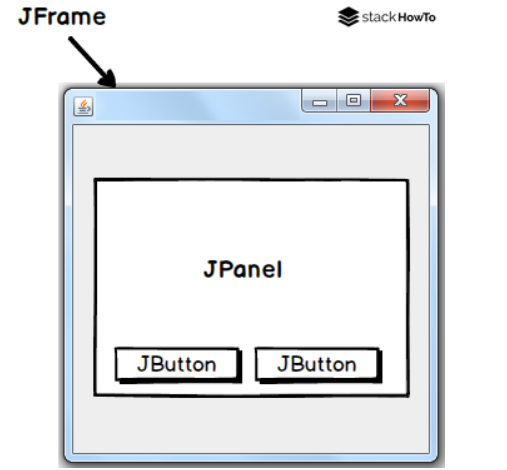
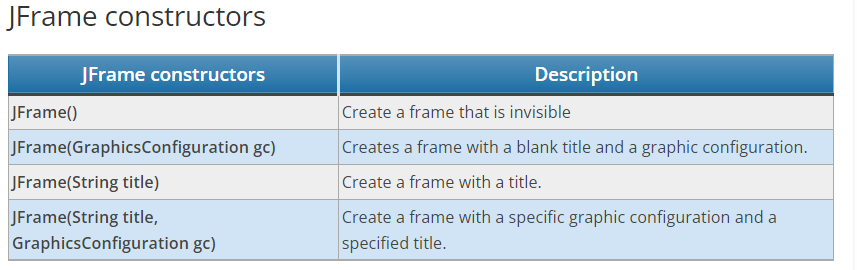
**JFrame**

JFrame is a class found in javax.swing package that inherits from java.awt.frame, it adds support for the SWING component architecture. It is a top-level window, with a border and a title bar. JFrame class has many methods that can be used to customize it.





**Create a JFrame**

*import javax.swing.JFrame;*

*public class Main*

*{*

*public static void main(String[] args)*

*{*

*JFrame frame = new JFrame();*

*frame.setVisible(true);*

*}*

*}*

**Output :**



**Change the size of a JFrame window**

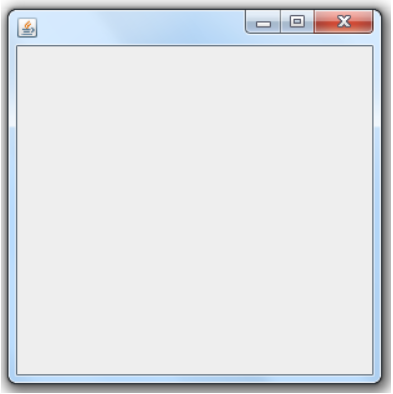
To resize a frame, JFrame provides a method JFrame.setSize(int width, int height), it needs two parameters width and height.

*JFrame frame = new JFrame();*

*frame.setSize(300, 300);*

*frame.setVisible(true);*

**Output:**

****

**Set the title of JFrame**

To set the title of a JFrame, you can use JFrame.setTitle(String title).

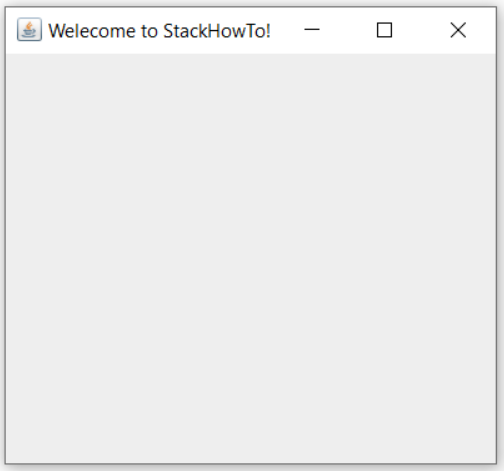
*JFrame frame = new JFrame();*

*frame.setSize(300, 300);*

*frame.setTitle("Welecome to StackHowTo!");*

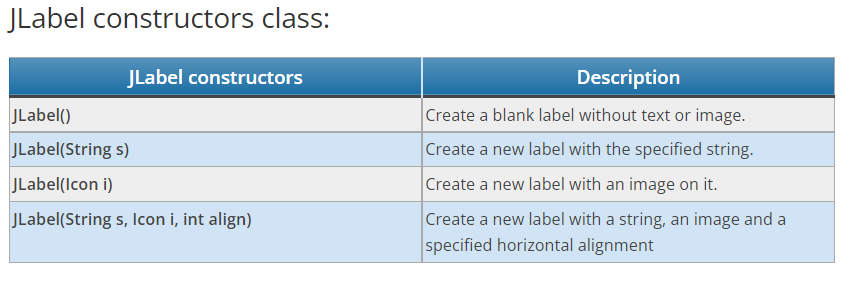
*frame.setVisible(true);*

**Output:**

****

**JLabel**

JLabel is a java Swing class. JLabel is a field to display a short string or an image or both. JLabel is only used to display text or images and it can’t get focus. JLabel is inactive to capture events such as mouse focus or keyboard focus. By default, labels are centered vertically but the user can change the alignment of JLabel.



**Commonly used methods of JLabel class:**

getIcon() : returns the image that the label displays

setIcon(Icon i) : sets the image that the label will display

getText() : returns the text displayed in the label

setText(String s) : sets the text of the label

**Example of JLabel in Java Swing:**

*import javax.swing.\*;*

*public class Main*

*{*

*public static void main(String[] args)*

*{*

*//Create a new frame*

*JFrame frame = new JFrame("JLabel Example");*

*//Create a label to display centered text*

*JLabel label = new JLabel("Welcome to StackHowTo!", JLabel.CENTER);*

*//Add label to frame*

*frame.add(label);*

*frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);*

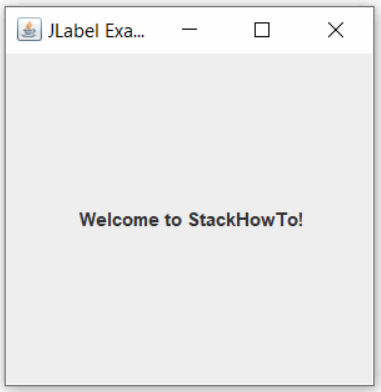
*frame.setSize(250, 250);*

*frame.setVisible(true);*

*}*

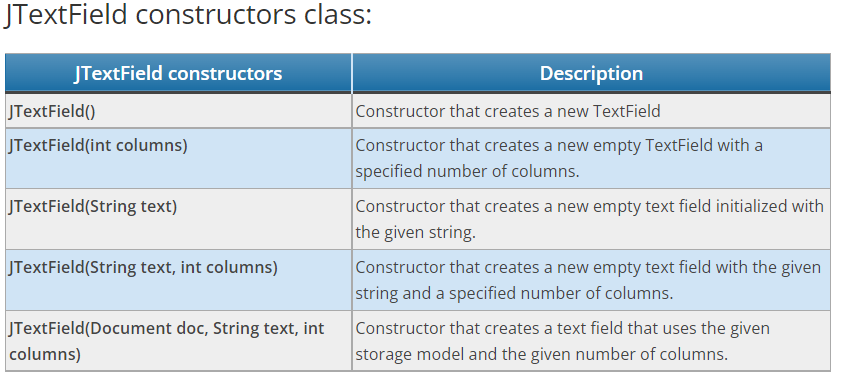
*}*

**Output:**

**

**JTextField**

JTextField is part of the javax.swing package. JTextField class is a component that allows modifying a single line of text. JTextField inherits from JTextComponent class and uses the SwingConstants interface.



Commonly used methods of JTextField class:

* setColumns(int n) : set the number of columns of JTextField.
* setFont(Font f) : set the font of the text displayed in JTextField.
* addActionListener(ActionListener l) : set an ActionListener on JTextField.
* int getColumns() : gets the number of columns in JTextField.

**Program**

*import javax.swing.\*;*

*class Main*

*{*

*public static void main(String args[])*

*{*

*JFrame frame = new JFrame("JTextField Example");*

*JTextField text1 = new JTextField();*

*text1.setBounds(20,40,200,28);*

*JTextField text2 = new JTextField("Welcome To StackHowTo!");*

*text2.setBounds(20,80,200,28);*

*frame.add(text1);*

*frame.add(text2);*

*frame.setSize(250,250);*

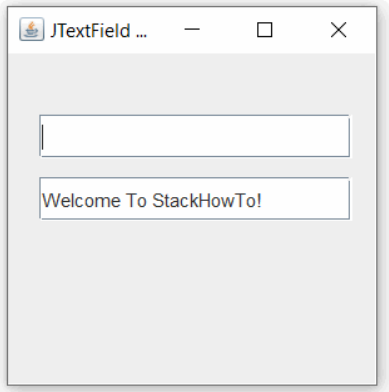
*frame.setLayout(null);*

*frame.setVisible(true);*

*}*

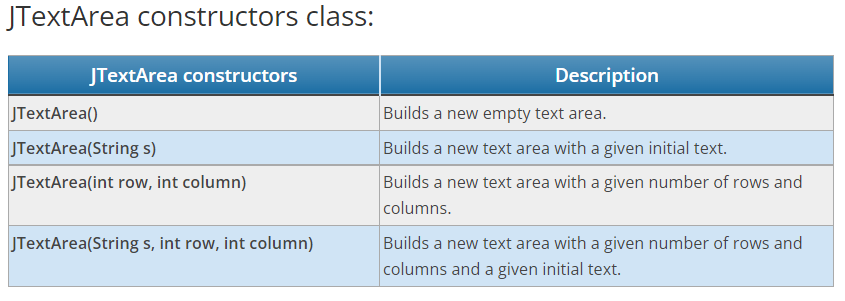
*}*

**Output**

****

**JTextArea**

JTextArea is part of the Java Swing package. It represents an area on several lines that displays text. It is used to edit text. JTextArea inherits from the JComponent class. The text in JTextArea can be set to different available fonts. The text area can be customized according to the user’s needs.



Commonly used methods:

* append(String s): adds the given string to the text in JTextArea.
* getLineCount(): gets the number of lines of text in the JTextArea.
* setFont(Font f): sets the font of JTextArea to the given font.
* setColumns(int c): sets the number of columns in JTextArea to a given integer.
* setRows(int r): sets the number of lines in JTextArea to a given integer.
* getColumns(): gets the number of columns in JTextArea.
* getRows(): gets the number of lines in JTextArea.

**Program :**

*import javax.swing.\*;*

*import java.awt.event.\*;*

*public class TextAreaTest implements ActionListener*

*{*

*JLabel l1, l2;*

*JTextArea text;*

*TextAreaTest()*

*{*

*JFrame f = new JFrame();*

*l1 = new JLabel();*

*l1.setBounds(45,175,100,30);*

*l2 = new JLabel();*

*l2.setBounds(150,175,100,30);*

*text = new JTextArea();*

*text.setBounds(15,20,250,150);*

*JButton btn = new JButton("Counting words");*

*btn.setBounds(50,210,180,30);*

*btn.addActionListener(this);*

*f.add(text);*

*f.add(l1);*

*f.add(l2);*

*f.add(btn);*

*f.setSize(300,300);*

*f.setLayout(null);*

*f.setVisible(true);*

*}*

*public void actionPerformed(ActionEvent e)*

*{*

*String str = text.getText();*

*String words[] = str.split("\\s");*

*l1.setText("Cords: "+ words.length);*

*l2.setText("Character: "+ str.length());*

*}*

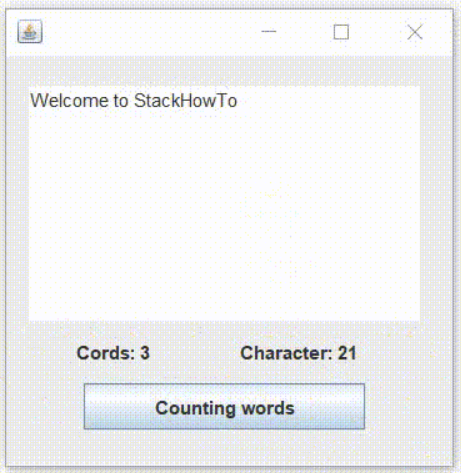
*public static void main(String[] args) {*

*new TextAreaTest();*

*}*

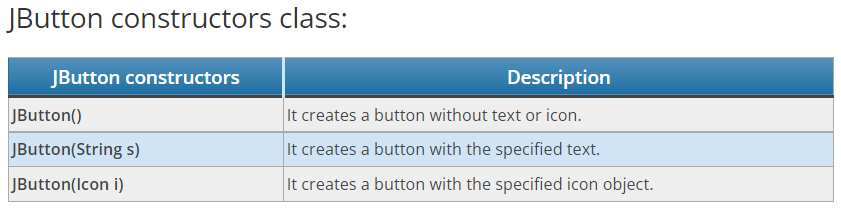
*}*

**Output**



**JButton**

JButton is a component of Java Swing. JButton class is used to create a labeled button with a platform-independent implementation. The application causes an action when the button is clicked. It can be configured to have different actions, using the Event Listener. JButton inherits from the AbstractButton class.



Commonly used methods of JButton class:

* void setText(String s) : It is used to set the text specified on the button.
* String getText() : It is used to return the text of the button.
* void setEnabled(boolean b) : It is used to enable or disable the button.
* void setIcon(Icon b) : It is used to set the icon on JButton.
* Icon getIcon() : It is used to get the button icon.
* void setMnemonic(int a) : It is used to set the mnemonic on the button.
* void addActionListener(ActionListener a) : It is used to add action listener to this object.

**Program**

*import javax.swing.\*;*

*public class Main*

*{*

*public static void main(String[] args)*

*{*

*//Create a new frame*

*JFrame frame = new JFrame("JButton Example");*

*//Create button*

*JButton btn = new JButton("Click here");*

*//Set button position*

*btn.setBounds(100,100,100,40);*

*//Add button to frame*

*frame.add(btn);*

*frame.setSize(300,300);*

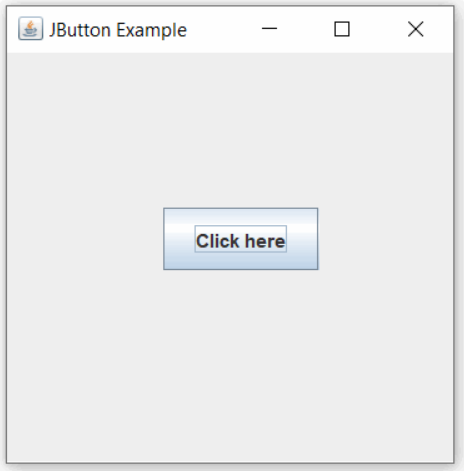
*frame.setLayout(null);*

*frame.setVisible(true);*

*}*

*}*

**Output**

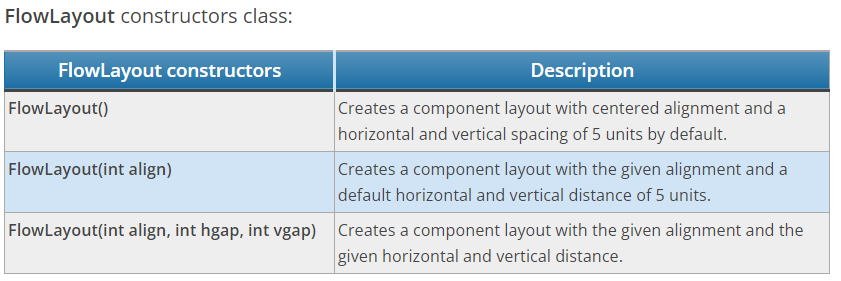
****

**FlowLayout**

FlowLayout is used to arrange components line by line, one after the other (in a flow). This is the default layout of the applet or panel.

Fields of the FlowLayout class:

* public static final int LEFT
* public static final int RIGHT
* public static final int CENTER
* public static final int LEADING
* public static final int TRAILING



**Example**

*import java.awt.\*;*

*import javax.swing.\*;*

*public class MyFlowLayout*

*{*

*MyFlowLayout()*

*{*

*JFrame frame = new JFrame();*

*JButton btn1 = new JButton("A");*

*JButton btn2 = new JButton("B");*

*JButton btn3 = new JButton("C");*

*frame.add(btn1);*

*frame.add(btn2);*

*frame.add(btn3);*

*//set the layout to the right*

*frame.setLayout(new FlowLayout(FlowLayout.RIGHT));*

*frame.setSize(300,300);*

*frame.setVisible(true);*

*}*

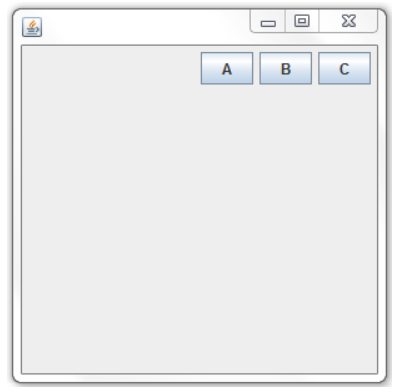
*public static void main(String[] args) {*

*new MyFlowLayout();*

*}*

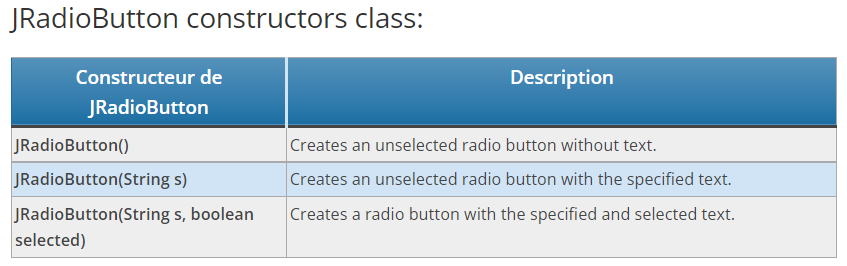
*}*

**Output:**

****

**JRadioButton**

The radio button is used to select one of several options. It is used to fill in forms, online documents, and MCQs. We add radio buttons to a group so that only one radio button can be selected at a time. We use the “ButtonGroup” class to create a button group and add a radio button to a group.



Commonly used methods of JRadioButton class

* void setText(String s) : It is used to set the text on the button.
* String getText() : It is used to return the text of the button.
* void setEnabled(boolean b) : It is used to enable or disable the button.
* void setIcon(Icon b) : It is used to set the icon on the button.
* Icon getIcon() : It is used to get the button icon.
* void setMnemonic(int a) : It is used to set the mnemonic on the button.
* void addActionListener(ActionListener a) : It is used to add action listener to this object.

**Program**

*import javax.swing.\*;*

*public class RadioButtonTest*

*{*

*JFrame frame;*

*RadioButtonTest()*

*{*

*frame = new JFrame();*

*// Create the label*

*JLabel label = new JLabel("1 - Give the abbreviation of AWT?", JLabel.CENTER);*

*label.setBounds(20,0,200,80);*

*// Create the radio buttons*

*JRadioButton btn1 = new JRadioButton("A) Applet Windowing Toolkit");*

*JRadioButton btn2 = new JRadioButton("B) Abstract Windowing Toolkit");*

*JRadioButton btn3 = new JRadioButton("C) Absolute Windowing Toolkit");*

*// Set the position of the radio buttons*

*btn1.setBounds(40,60,200,50);*

*btn2.setBounds(40,100,200,50);*

*btn3.setBounds(40,140,200,50);*

*// Add radio buttons to group*

*ButtonGroup bg = new ButtonGroup();*

*bg.add(btn1);*

*bg.add(btn2);*

*bg.add(btn3);*

*// Add buttons to frame*

*frame.add(label);*

*frame.add(btn1);*

*frame.add(btn2);*

*frame.add(btn3);*

*frame.setSize(300,300);*

*frame.setLayout(null);*

*frame.setVisible(true);*

*}*

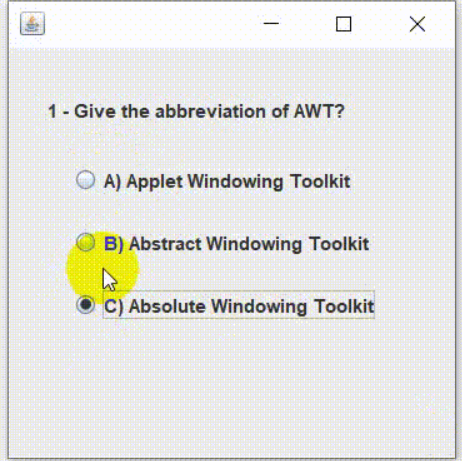
*public static void main(String[] args) {*

*new RadioButtonTest();*

*}*

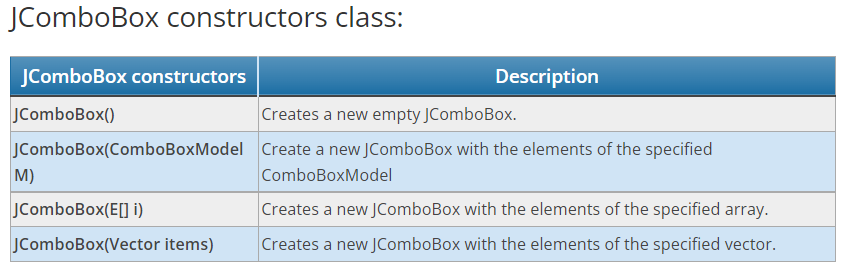
*}*

**Output**

****

**JComboBox**

JComboBox is part of the Java Swing package. JComboBox inherits from the JComponent class. JComboBox displays a contextual menu as a list that allows the user to select an option from the specified list. JComboBox can be editable or read-only according to the programmer’s choice.



Commonly used methods:

* addItem(E item) : Adds the element to JComboBox
* addItemListener( ItemListener l) : Adds an ItemListener to JComboBox.
* getItemAt(int i) : Returns the element at index i
* getItemCount() : Returns the number of elements in the list
* getSelectedItem() : Returns the element that is selected
* removeItemAt(int i) : Deletes the element at index i
* setEditable(boolean b) : The boolean b determines if the list is modifiable or not. If the value “true” is transmitted, the list is modifiable, if “false” the list is not modifiable.
* setSelectedIndex(int i) : Selects the JComboBox element at index i.
* showPopup() : Allows the drop-down list to display its pop-up window.
* setEnabled(boolean b) : Activates the drop-down list so that the items can be selected.
* removeItem(Object anObject) : Removes an element from the list of elements.
* removeAllItems() : Deletes all the elements of the list.
* removeActionListener(ActionListener l) : Deletes ActionListener.
* isPopupVisible() : Determines popup visibility.
* getItemCount() : Returns the number of elements in the list.

**Program**

*import javax.swing.\*;*

*import java.awt.\*;*

*import java.awt.event.\*;*

*class ComboBoxExample extends JFrame implements ItemListener {*

*// frame*

*static JFrame frame;*

*// combobox*

*static JComboBox combobox;*

*// label*

*static JLabel l1, l2;*

*public static void main(String[] args)*

*{*

*// create a new frame*

*frame = new JFrame("frame");*

*// create an object*

*ComboBoxExample obj = new ComboBoxExample();*

*// set the layout of the frame*

*frame.setLayout(new FlowLayout());*

*// array of strings containing languages*

*String s1[] = { "Java", "PHP", "Python", "C++", "Ruby" };*

*// create a checkbox*

*combobox = new JComboBox(s1);*

*// add ItemListener*

*combobox.addItemListener(obj);*

*// create labels*

*l1 = new JLabel("What is your favorite language? ");*

*l2 = new JLabel("[Java]");*

*// set the text color*

*l2.setForeground(Color.blue);*

*// create a new panel*

*JPanel p = new JPanel();*

*// add combobox and labels to the panel*

*p.add(l1);*

*p.add(combobox);*

*p.add(l2);*

*// add panel to frame*

*frame.add(p);*

*// set the frame size*

*frame.setSize(400, 200);*

*frame.show();*

*}*

*public void itemStateChanged(ItemEvent e)*

*{*

*// check if the state of the combobox is changed*

*if (e.getSource() == combobox) {*

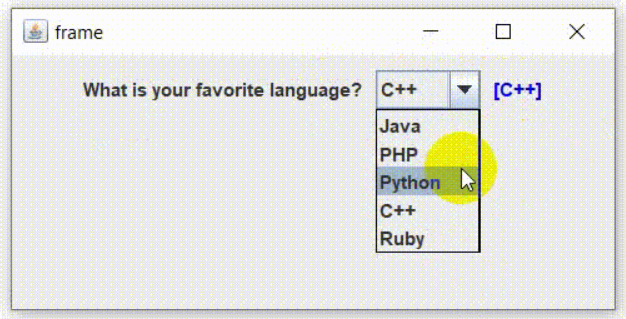
*l2.setText(" ["+combobox.getSelectedItem()+"]");*

*}*

*}*

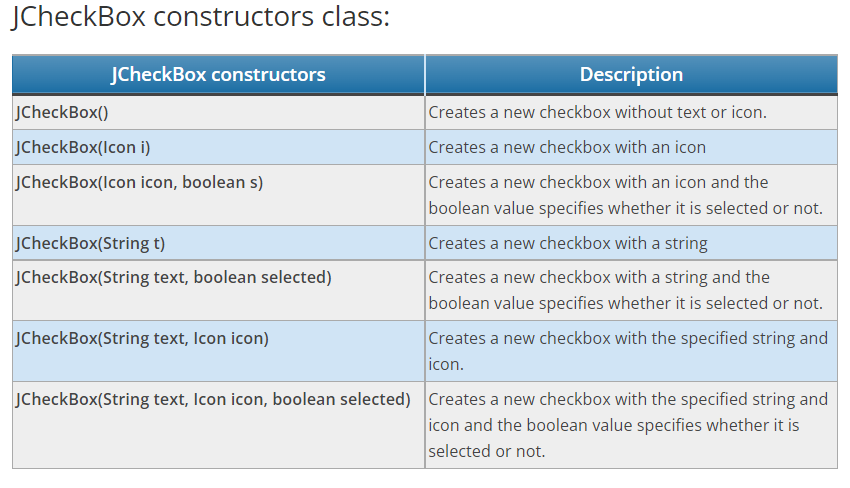
*}*

**Output**

****

**JCheckBox**

JCheckBox is a Swing component that represents an element that shows a selected or unselected state. The user can change this state by clicking on the checkbox. A standard JCheckBox component contains a checkbox and a label that describes the purpose of the checkbox. An icon and a mnemonic key can also be defined for this component.



Commonly used methods:

* setIcon(Icon i): sets the checkbox icon to the given icon
* setText(String s): sets the checkbox text to the given text
* setSelected(boolean b): sets the checkbox if the transmitted boolean value is true or vice versa
* getIcon(): returns the image of the checkbox
* getText(): returns the text of the checkbox
* updateUI(): resets the GUI property to the current Look & Feel value.
* getUI(): returns the Look & Feel object which makes this component.
* paramString(): returns a string representation of this JCheckBox.
* getUIClassID(): gets the AccessibleContext associated with this JCheckBox.
* getAccessibleContext(): gets the AccessibleContext associated with this JCheckBox.
* isBorderPaintedFlat(): gets the value of the borderPaintedFlat property.
* setBorderPaintedFlat(boolean b): sets the borderPaintedFlat property.

**Program**

*import java.awt.\*;*

*import javax.swing.\*;*

*class Main extends JFrame*

*{*

*static JFrame f;*

*public static void main(String[] args)*

*{*

*// create a new frame*

*f = new JFrame("Checkbox Example");*

*// set the frame layout*

*f.setLayout(new FlowLayout());*

*// create a checkbox*

*JCheckBox check1 = new JCheckBox("Male");*

*JCheckBox check2 = new JCheckBox("Female");*

*// create a new panel*

*JPanel p = new JPanel();*

*// add a checkbox to the panel*

*p.add(check1);*

*p.add(check2);*

*// add panel to frame*

*f.add(p);*

*// set the frame size*

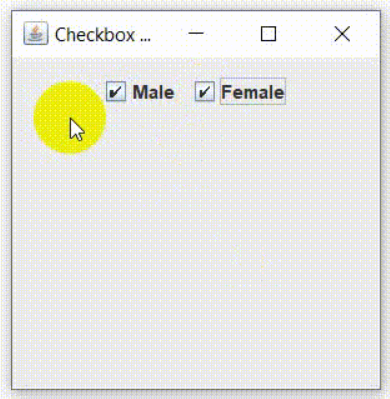
*f.setSize(250, 250);*

*f.show();*

*}*

*}*

**Output**

****

**How to Change Font Size and Font Style of a JLabel**

**Program**

*import java.awt.\*;*

*import javax.swing.\*;*

*public class StyleJLabel*

*{*

*StyleJLabel()*

*{*

*JFrame frame = new JFrame();*

*frame.setLayout(new GridLayout(4,1));*

*JLabel label = new JLabel("This is a label!", SwingConstants.CENTER);*

*label.setFont(new Font("Serif", Font.BOLD, 20));*

*label.setForeground(Color.RED);*

*label.setBackground(Color.ORANGE);*

*label.setOpaque(true);*

*frame.add(label);*

*frame.setSize(300,150);*

*frame.setVisible(true);*

*frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);*

*}*

*public static void main(String[] args)*

*{*

*new StyleJLabel();*

*}*

*}*

**Output**



**Java Program to Change Font Color and Font Size of a JTextField:**

**Program**

*import java.awt.\*;*

*import javax.swing.\*;*

*public class TextStyle*

*{*

*TextStyle()*

*{*

*JFrame frame = new JFrame();*

*frame.setLayout(new GridLayout(4,1));*

*JTextField text = new JTextField();*

*// Change text font size*

*text.setFont(new Font("Serif",Font.BOLD,30));*

*// Change text font color*

*text.setForeground(Color.RED);*

*frame.add(text);*

*frame.setSize(300,200);*

*frame.setVisible(true);*

*frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);*

*}*

*public static void main(String[] args)*

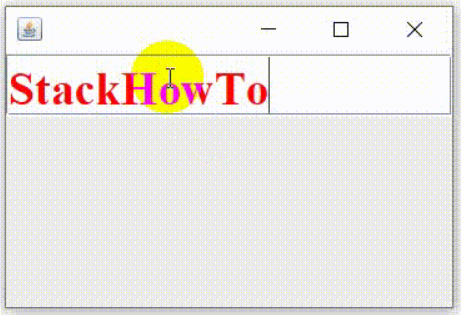
*{*

*new TextStyle();*

*}*

*}*

**Output:**

****