# Java JButton

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

## JButton class declaration

Let's see the declaration for javax.swing.JButton class.

1. **public** **class** JButton **extends** AbstractButton **implements** Accessible

### Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JButton() | It creates a button with no text and icon. |
| JButton(String s) | It creates a button with the specified text. |
| JButton(Icon i) | It creates a button with the specified icon object. |

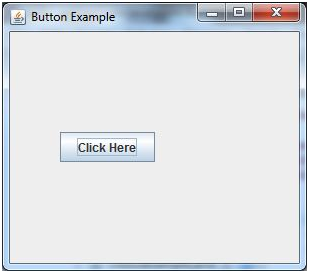
### Commonly used Methods of AbstractButton class:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| void setText(String s) | It is used to set specified text on 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 specified Icon on the button. |
| Icon getIcon() | It is used to get the Icon of the button. |
| void setMnemonic(int a) | It is used to set the mnemonic on the button. |
| void addActionListener(ActionListener a) | It is used to add the [action listener](https://www.javatpoint.com/java-actionlistener) to this object. |

## Java JButton Example

1. **import** javax.swing.\*;
2. **public** **class** ButtonExample {
3. **public** **static** **void** main(String[] args) {
4. JFrame f=**new** JFrame("Button Example");
5. JButton b=**new** JButton("Click Here");
6. b.setBounds(50,100,95,30);
7. f.add(b);
8. f.setSize(400,400);
9. f.setLayout(**null**);
10. f.setVisible(**true**);
11. }
12. }

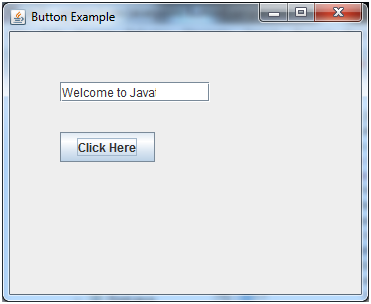
Output:



## Java JButton Example with ActionListener

1. **import** java.awt.event.\*;
2. **import** javax.swing.\*;
3. **public** **class** ButtonExample {
4. **public** **static** **void** main(String[] args)
5. {
6. JFrame f=**new** JFrame("Button Example");
7. **final** JTextField tf=**new** JTextField();
8. tf.setBounds(50,50, 150,20);
9. JButton b=**new** JButton("Click Here");
10. b.setBounds(50,100,95,30);
11. b.addActionListener(**new** ActionListener(){
12. **public** **void** actionPerformed(ActionEvent e){
13. tf.setText("Welcome to Java.");
14. }
15. });
16. f.add(b);f.add(tf);
17. f.setSize(400,400);
18. f.setLayout(**null**);
19. f.setVisible(**true**);
20. }
21. }

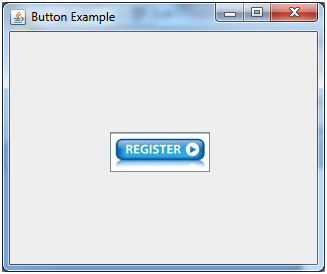
Output:



## Example of displaying image on the button:

1. **import** javax.swing.\*;
2. **public** **class** ButtonExample{
3. ButtonExample(){
4. JFrame f=**new** JFrame("Button Example");
5. JButton b=**new** JButton(**new** ImageIcon("D:\\icon.png"));
6. b.setBounds(100,100,100, 40);
7. f.add(b);
8. f.setSize(300,400);
9. f.setLayout(**null**);
10. f.setVisible(**true**);
11. f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
12. }
13. **public** **static** **void** main(String[] args) {
14. **new** ButtonExample();
15. }
16. }

Output:



# Java JLabel

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

## JLabel class declaration

Let's see the declaration for javax.swing.JLabel class.

1. **public** **class** JLabel **extends** JComponent **implements** SwingConstants, Accessible

### Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JLabel() | Creates a JLabel instance with no image and with an empty string for the title. |
| JLabel(String s) | Creates a JLabel instance with the specified text. |
| JLabel(Icon i) | Creates a JLabel instance with the specified image. |
| JLabel(String s, Icon i, int horizontalAlignment) | Creates a JLabel instance with the specified text, image, and horizontal alignment. |

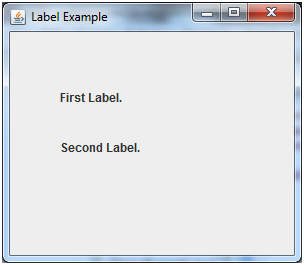
### Commonly used Methods:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| String getText() | t returns the text string that a label displays. |
| void setText(String text) | It defines the single line of text this component will display. |
| void setHorizontalAlignment(int alignment) | It sets the alignment of the label's contents along the X axis. |
| Icon getIcon() | It returns the graphic image that the label displays. |
| int getHorizontalAlignment() | It returns the alignment of the label's contents along the X axis. |

## JLabel Example

1. **import** javax.swing.\*;
2. **class** LabelExample
3. {
4. **public** **static** **void** main(String args[])
5. {
6. JFrame f= **new** JFrame("Label Example");
7. JLabel l1,l2;
8. l1=**new** JLabel("First Label.");
9. l1.setBounds(50,50, 100,30);
10. l2=**new** JLabel("Second Label.");
11. l2.setBounds(50,100, 100,30);
12. f.add(l1); f.add(l2);
13. f.setSize(300,300);
14. f.setLayout(**null**);
15. f.setVisible(**true**);
16. }
17. }

Output:

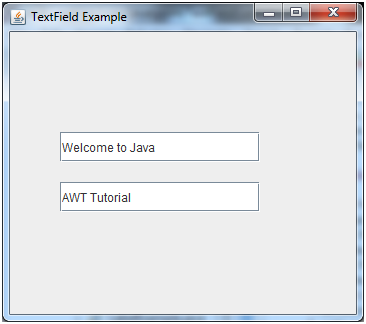


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| JTextField The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class. JTextField class declaration Let's see the declaration for javax.swing.JTextField class.   1. **public** **class** JTextField **extends** JTextComponent **implements** SwingConstants  Commonly used Constructors:  |  |  | | --- | --- | | **Constructor** | **Description** | | JTextField() | Creates a new TextField | | JTextField(String text) | Creates a new TextField initialized with the specified text. | | JTextField(String text, int columns) | Creates a new TextField initialized with the specified text and columns. | | JTextField(int columns) | Creates a new empty TextField with the specified number of columns. |  Commonly used Methods:  |  |  | | --- | --- | | **Methods** | **Description** | | void addActionListener(ActionListener l) | It is used to add the specified action listener to receive action events from this textfield. | | Action getAction() | It returns the currently set Action for this ActionEvent source, or null if no Action is set. | | void setFont(Font f) | It is used to set the current font. | | void removeActionListener(ActionListener l) | It is used to remove the specified action listener so that it no longer receives action events from this textfield. | |

## Java JTextField Example

1. **import** javax.swing.\*;
2. **class** TextFieldExample
3. {
4. **public** **static** **void** main(String args[])
5. {
6. JFrame f= **new** JFrame("TextField Example");
7. JTextField t1,t2;
8. t1=**new** JTextField("Welcome to Java.");
9. t1.setBounds(50,100, 200,30);
10. t2=**new** JTextField("AWT Tutorial");
11. t2.setBounds(50,150, 200,30);
12. f.add(t1); f.add(t2);
13. f.setSize(400,400);
14. f.setLayout(**null**);
15. f.setVisible(**true**);
16. }
17. }

Output:



# JTextArea

The object of a JTextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits JTextComponent class

## JTextArea class declaration

Let's see the declaration for javax.swing.JTextArea class.

1. **public** **class** JTextArea **extends** JTextComponent

### Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JTextArea() | Creates a text area that displays no text initially. |
| JTextArea(String s) | Creates a text area that displays specified text initially. |
| JTextArea(int row, int column) | Creates a text area with the specified number of rows and columns that displays no text initially. |
| JTextArea(String s, int row, int column) | Creates a text area with the specified number of rows and columns that displays specified text. |

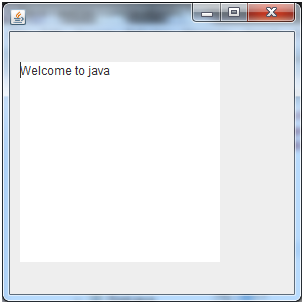
### Commonly used Methods:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| void setRows(int rows) | It is used to set specified number of rows. |
| void setColumns(int cols) | It is used to set specified number of columns. |
| void setFont(Font f) | It is used to set the specified font. |
| void insert(String s, int position) | It is used to insert the specified text on the specified position. |
| void append(String s) | It is used to append the given text to the end of the document. |

## Java JTextArea Example

1. **import** javax.swing.\*;
2. **public** **class** TextAreaExample
3. {
4. TextAreaExample(){
5. JFrame f= **new** JFrame();
6. JTextArea area=**new** JTextArea("Welcome to java");
7. area.setBounds(10,30, 200,200);
8. f.add(area);
9. f.setSize(300,300);
10. f.setLayout(**null**);
11. f.setVisible(**true**);
12. }
13. **public** **static** **void** main(String args[])
14. {
15. **new** TextAreaExample();
16. }}

Output:



# JPasswordField

The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text. It inherits JTextField class.

## JPasswordField class declaration

Let's see the declaration for javax.swing.JPasswordField class.

1. **public** **class** JPasswordField **extends** JTextField

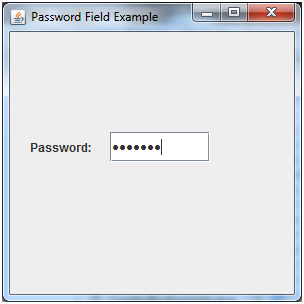
### Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JPasswordField() | Constructs a new JPasswordField, with a default document, null starting text string, and 0 column width. |
| JPasswordField(int columns) | Constructs a new empty JPasswordField with the specified number of columns. |
| JPasswordField(String text) | Constructs a new JPasswordField initialized with the specified text. |
| JPasswordField(String text, int columns) | Construct a new JPasswordField initialized with the specified text and columns. |

## Java JPasswordField Example

1. **import** javax.swing.\*;
2. **public** **class** PasswordFieldExample {
3. **public** **static** **void** main(String[] args) {
4. JFrame f=**new** JFrame("Password Field Example");
5. JPasswordField value = **new** JPasswordField();
6. JLabel l1=**new** JLabel("Password:");
7. l1.setBounds(20,100, 80,30);
8. value.setBounds(100,100,100,30);
9. f.add(value); f.add(l1);
10. f.setSize(300,300);
11. f.setLayout(**null**);
12. f.setVisible(**true**);
13. }
14. }

Output:



## JPasswordField Example with ActionListener

1. **import** javax.swing.\*;
2. **import** java.awt.event.\*;
3. **public** **class** PasswordFieldExample {
4. **public** **static** **void** main(String[] args) {
5. JFrame f=**new** JFrame("Password Field Example");
6. **final** JLabel label = **new** JLabel();
7. label.setBounds(20,150, 200,50);
8. **final** JPasswordField value = **new** JPasswordField();
9. value.setBounds(100,75,100,30);
10. JLabel l1=**new** JLabel("Username:");
11. l1.setBounds(20,20, 80,30);
12. JLabel l2=**new** JLabel("Password:");
13. l2.setBounds(20,75, 80,30);
14. JButton b = **new** JButton("Login");
15. b.setBounds(100,120, 80,30);
16. **final** JTextField text = **new** JTextField();
17. text.setBounds(100,20, 100,30);
18. f.add(value); f.add(l1); f.add(label); f.add(l2); f.add(b); f.add(text);
19. f.setSize(300,300);
20. f.setLayout(**null**);
21. f.setVisible(**true**);
22. b.addActionListener(**new** ActionListener() {
23. **public** **void** actionPerformed(ActionEvent e) {
24. String data = "Username " + text.getText();
25. data += ", Password: " + **new** String(value.getPassword());
26. label.setText(data);
27. }
28. });
29. }
30. }

Output:

