**8.11 Text Components :-**

**Java AWT TextField**

The [object](https://www.javatpoint.com/object-and-class-in-java) of a TextField class is a text component that allows the editing of a single line text. It inherits TextComponent class.

AWT TextField Class Declaration

**public** **class** TextField **extends** TextComponent

**import** java.awt.\*;

**class** TextFieldExample{

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

    Frame f= **new** Frame("TextField Example");

    TextField t1,t2;

    t1=**new** TextField("Welcome to Javatpoint.");

    t1.setBounds(50,100, 200,30);

    t2=**new** TextField("AWT Tutorial");

    t2.setBounds(50,150, 200,30);

    f.add(t1); f.add(t2);

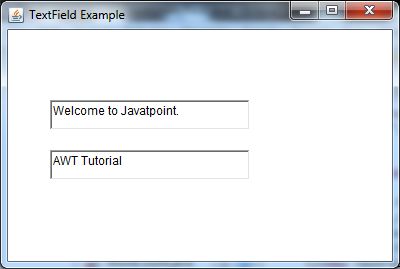
    f.setSize(400,400);

    f.setLayout(**null**);

    f.setVisible(**true**);

}

}



**Java AWT TextArea**

The [object](https://www.javatpoint.com/object-and-class-in-java) of a TextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits TextComponent class.

AWT TextArea Class Declaration

**public** **class** TextArea **extends** TextComponent

**import** java.awt.\*;

**public** **class** TextAreaExample

{

     TextAreaExample(){

        Frame f= **new** Frame();

            TextArea area=**new** TextArea("Welcome to javatpoint");

        area.setBounds(10,30, 300,300);

        f.add(area);

        f.setSize(400,400);

        f.setLayout(**null**);

        f.setVisible(**true**);

     }

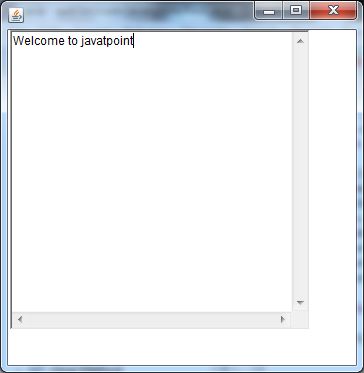
**public** **static** **void** main(String args[])

{

**new** TextAreaExample();

}

}



This section provides background information you might need when using Swing text components. If you intend to use an unstyled text component — a text field, password field, formatted text field, or text area — go to its how-to page and return here only if necessary. If you intend to use a styled text component, see How to Use Editor Panes and Text Panes, and read this section as well. If you do not know which component you need, read on.

Swing text components display text and optionally allow the user to edit the text. Programs need text components for tasks ranging from the straightforward (enter a word and press Enter) to the complex (display and edit styled text with embedded images in an Asian language).

Swing provides six text components, along with supporting classes and interfaces that meet even the most complex text requirements. In spite of their different uses and capabilities, all Swing text components inherit from the same superclass, JTextComponent, which provides a highly-configurable and powerful foundation for text manipulation.

The following figure shows the JTextComponent hierarchy.

Swing's hierarchy of text components

The following picture shows an application called TextSamplerDemo that uses each Swing text component.

An application that provides a sample of each Swing text component

Try this:

Click the Launch button to run TextSamplerDemo using Java™ Web Start (download JDK 6 or later). Alternatively, to compile and run the example yourself, consult the example index.Launches the TextSamplerDemo Application

Type some text in the text field and press Enter. Do the same in the password field. The label beneath the fields is updated when you press Enter.

Try entering valid and invalid dates into the formatted text field. Note that when you press Enter the label beneath the fields is updated only if the date is valid.

Select and edit text in the text area and the text pane. Use keyboard bindings, Ctrl-X, Ctrl-C, and Ctrl-V, to cut, copy, and paste text, respectively.

Try to edit the text in the editor pane, which has been made uneditable with a call to setEditable.

Look in the text pane to find an example of an embedded component and an embedded icon.

The TextSamplerDemo example uses the text components in very basic ways. The following table tells you more about what you can do with each kind of text component.

Group Description Swing Classes

**Text Controls**

Also known simply as text fields, text controls can display only one line of editable text. Like buttons, they generate action events. Use them to get a small amount of textual information from the user and perform an action after the text entry is complete. JTextField and its subclasses JPasswordField and JFormattedTextField

**Plain Text Areas**

JTextArea can display multiple lines of editable text. Although a text area can display text in any font, all of the text is in the same font. Use a text area to allow the user to enter unformatted text of any length or to display unformatted help information. JTextArea

**Styled Text Areas**

A styled text component can display editable text using more than one font. Some styled text components allow embedded images and even embedded components. Styled text components are powerful and multi-faceted components suitable for high-end needs, and offer more avenues for customization than the other text components.

Because they are so powerful and flexible, styled text components typically require more initial programming to set up and use. One exception is that editor panes can be easily loaded with formatted text from a URL, which makes them useful for displaying uneditable help information.

JEditorPane

and its subclass

JTextPane

This Tutorial provides information about the foundation laid by the JTextComponent class and tells you how to accomplish some common text-related tasks. Because the JTextComponent class and its subclasses have too many features to be completely described in this Tutorial, please visit the Swing & AWT forum at java.net for help and information.

Learn more about text components in JavaFX, see the Using Text and Text Effects in JavaFX and Using JavaFX UI Controls: Text Field tutorials.

