

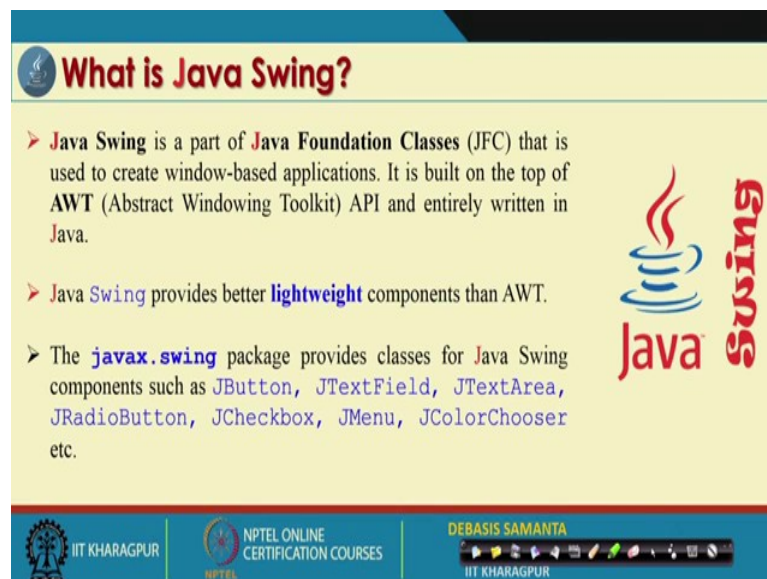
**Programming in Java**  
**Prof. Debasis Samanta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 43**  
**Swing Programming - I**

We have learned about AWT and AWT is sufficient enough to develop any graphical user interface based program. However, Java developer was not so, happy with the AWT, they dream to give something more than AWT which should be more elegant and then user friendly I should say rather programmer friendly; that means, programmer will find it so, easy to include there those components into their program and that is basically the process of it is called that in a light weight way to develop this thing without much of the code details they can develop their window program.

So, with this m Java developer introduced from version 6 onwards another utility package API it is called the Java swing. So, in today's lecture we will discussed first part of the Java Swing because it is so, vast that we should take at least 2 modules to cover the entire package one by one. So, first we will discussed about exactly what is the Swing.

(Refer Slide Time: 01:25)



**What is Java Swing?**

- **Java Swing** is a part of **Java Foundation Classes (JFC)** that is used to create window-based applications. It is built on the top of **AWT (Abstract Windowing Toolkit)** API and entirely written in **Java**.
- **Java Swing** provides better **lightweight** components than AWT.
- The **javax.swing** package provides classes for Java Swing components such as **JButton**, **TextField**, **TextArea**, **JRadioButton**, **JCheckbox**, **JMenu**, **JColorChooser** etc.

**Java Swing**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

As I told you Java developer introduced it in version 6 onwards, but version 5 and lower version Java Swing is was not available. Now Java Swing is basically is a part of Java

foundation classes which is the one foundation programs it is developed usually the oracle to make the Java Swing is more platform independent.

So, Java Swing in fact, provides a better light weight components than AWT in the sense light weight means we understand that user a programmer can do many things without knowing exactly how all those things been done. That means, we do not have to bother about the implementation details only just idea is that, if this thing is available just use it and then solve your problem. So, it is just like a very good library we can say very sensible library rather ok, so, that we can get many different utilities easily comfortably.


Now, so, Java that is a Swing the swing is included in one package you see it is not exactly same as the java dot package name like, but it is the new thing javax dot swing. So, javax dot swing package is there it is again whenever you install jdk right with your latest version then you will be able to automatically install it is bundle with jdk itself.

Now, the these javax swing package contains many components as we have already familiar to some components like button text field list menu item like this it is there, but whenever the to make it difference from the swing component to AWT component, then they use basically one more capital characters prepended to each component like say JButton is for button in swing, JTextField is a text field in swing and like this one.

So, a 'J' characters are prepended to each component to. So, indicate that it is from the swing actually. So, a now all those buttons are there; obviously, question that are is then what is the difference from this button to JButton or like this and why JButton is more preferable than the button we will able to understand these things when we will start our discussion about the swing component, definitely it has more advantages and first of all the thing is that all the component that will appear in a interface looks may much elegant than the component that you can develop with the AWT.

(Refer Slide Time: 04:13)

AWT versus Swing	
Java AWT	Java Swing
AWT components are platform-dependent.	Java swing components are platform-independent.
AWT components are heavyweight.	Swing components are lightweight.
AWT doesn't support pluggable look and feel.	Swing supports pluggable look and feel.
AWT provides less components than Swing.	Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedpane, etc.
AWT doesn't follow MVC(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing follows MVC.



The slide features a comparison table between Java AWT and Java Swing. To the right of the table is a large graphic with the Java logo, the word 'Swing' in a stylized font, and 'AWT vs' below it. The footer contains logos for IIT Kharagpur, NPTEL, and the name DEBASIS SAMANTA.

Anyways, so, there are some salient difference between the two things that is AWT and Swing. The first thing is that AWT components are platform dependent this means that is java environment is required in order to run all those things no other environment you can use it.

However, Java Swing is platform independent whatever be the environment absolutely you so, you do not have to bother you can use it. And AWT a components are heavy weights; heavy weights means many thing that you have to write the codes of your own and they are not they are much abstract rather whereas, Swing components is much more abstract than the AWT components this means that you do not have to bother about how all those things coming and then how all those things will be implemented you just simply plug and play.

And AWT does not support pluggable look and feel; that means, whatever the idea that it looks like it is really coming it is not like that. So, sometimes it show something it is appearing in a different form like, but it is basically pluggable look and feel. So, what exactly it looks like you can find it like that. AWT provides less component compared to the Swing because it has only few; however, Swing provides much more components then the AWT components such as tables, lists, then scrollpanes, tabbedpane and then colorchooser, etc all those things were not there in AWT, but those are there here in Java Swing.

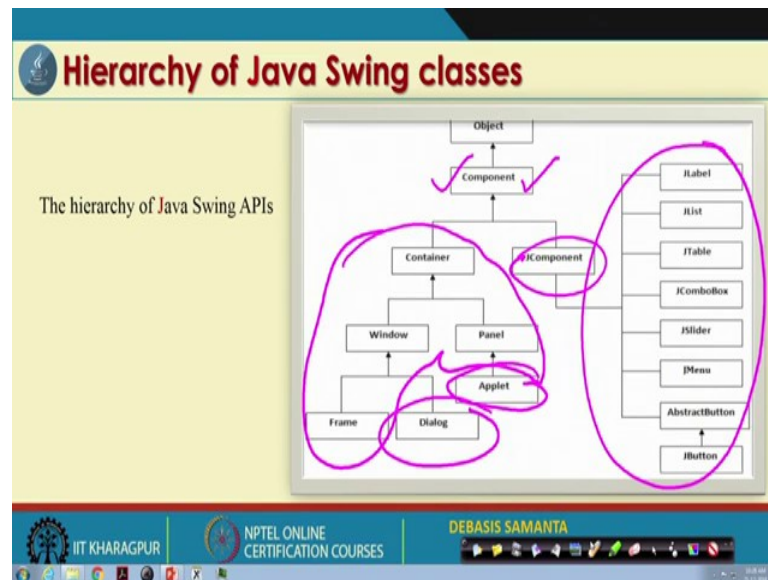
And another advantage of having the Java Swing is basically they follow one concept in graphical leisure intersplit programming is called the Model View Controller. So, it is basically called MVC actually AWT does not follow MVC then model view controller whereas, Swing follows MVC I do not want to discuss details about the MVC which is not so much relevant to this discussion. Anyway this is the one way how the graphical user interface can be developed.

(Refer Slide Time: 06:22)

The slide is titled "What is JFC?" in a red font. Below the title, it states: "The **Java Foundation Classes (JFC)** are a set of GUI components which simplify the development of desktop applications." To the right of this text is the "Java FOUNDATIONS" logo, which consists of a stylized flame icon and the text "Java™ FOUNDATIONS". In the bottom right corner, there is a video inset of a man, Debasis Samanta, wearing a suit and glasses. The slide has a blue header and footer. The footer contains the IIT Kharagpur logo, the text "NPTEL ONLINE CERTIFICATION COURSES", and the name "DEBASIS SAMANTA" with "IIT KHARAGPUR" below it. A navigation bar with arrows is also present in the footer.

Now, java is basically from the JFC; JFC developed by the oracle Java Foundation Classes for developing the graphical interface related to oracle systems. So, it is from the JFC only.

(Refer Slide Time: 06:37)



And then here is basically whatever the different classes are there so, for the JF Swing package is concerned we have shown here as we see here as it is right. So, JComponent is basically related to the component it is very same as the AWT components and in addition to these things there is a container and then container has say panel and then frame and all these things and other that thing dialog is there and container may be also an Applet. So, it is the basic idea and that is basically called the Swing components ok. So, Swing components basically as you see the JComponent is basically subclass of the components; that means, whatever the facilities that is available in AWT, you can have it here in swing; that means, both swing and AWT can be used together.

You can use anyone. So, that is why the idea it is there. So, they are all basically subclass of the AWT component class that is there in AWT package. And that is why whenever you have to use it we should import Java dot AWT dot star as well as Javax.Swing.\*; that means, to avail all the facility that is there and particularly the event handling facilities are defined there in AWT itself.

So, that anyway you have to use AWT package while when the whether you use AWT or in addition to AWT you use swing. So, this is the idea about the basic about the classes that is there.

Now, we will discuss about the different components that is there in swing as it is there is a large number of components as you have show told.

(Refer Slide Time: 08:28)

The slide is titled "Commonly used methods in Component class". Below the title, it states: "The methods of **Component** class are widely used in java swing that are given below." A table lists four methods with their descriptions. In the bottom right corner, there is a video inset of a man, Debasis Samanta, from IIT Kharagpur.

Method	Description
<code>public void add(Component c)</code>	Add a component on another component.
<code>public void setSize(int width,int height)</code>	Sets size of the component.
<code>public void setLayout(LayoutManager m)</code>	Sets the layout manager for the component.
<code>public void setVisible(boolean b)</code>	Sets the visibility of the component. It is by default false.

DEBASIS SAMANTA  
IIT KHARAGPUR

So, definitely it will take much time to discuss all the components. So, I will be little bit fast about I am just showing what exactly the component it looks and then what are the methods and then what are the other constructors that is there this is for your information.

So, my suggestion would be that if you practice it with the example which is here given in the slides as well as in the link that I have given and also you can practice any program collecting from any other sources from net like an if you practice it if you run then you can understand. Actually it is very difficult to learn from the very scares level another in the very in depth. So, better idea would be to learn it by your own practice. So, you have to practice it vigorously so, that you can learn it.

Now, first of all the component class that is there as you have already familiar to it has few methods like say add component. So, if you use the add any component can be added into some other either it is in a frame or it can be added into the applet like this one. And it also has the set size method we have already familiar to that and set layout and set visible those methods are already there and those methods should be also used in when whenever you include some Java Swing component in our program.

(Refer Slide Time: 09:50)

**Java Swing examples**

There are two ways to create a frame:

1. By creating the object of `Frame` class (**Association**)
2. By extending `Frame` class (**Inheritance**)

We can write the code of Swing inside the `main()`, constructor or any other method.

DEBASIS SAMANTA  
IIT KHARAGPUR

(Refer Slide Time: 09:54)

**Swing by association inside constructor**

We can also write all the codes of creating `JFrame`, `JButton` and method call inside the Java constructor.

```
import javax.swing.*;
public class Simple {
    JFrame f;
    Simple() {
        f = new JFrame();
        JButton b = new JButton("click");
        b.setBounds(130, 100, 100, 40);
        f.add(b);
        f.setSize(400, 500);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new Simple();
    }
}
```

DEBASIS SAMANTA  
IIT KHARAGPUR

And the Java Swing is basically first is that we will discussed about how the components can be added from a more precisely, the Java Swing component can be added into our program. There are mainly 3 ways I am giving you using a Frame we can add it. So, in this example as we see we create a Frame and in this Frame we can add the thing these are the most simplest way.

On the other hand we can create another way also by using the constructor and here is one thing you can see in this example, we create a frame and then this frame is basically



using a constructor of this class which is basically the main program, there we can include it and then that constructor can be called for that.

So, this is the one way of doing these things, it is basically same impact. So, this is the one way within the main program, without any creating subclass we can do it, but here creating a subclass we can do it and then you can use it. So, this is more useful because if we developed one and then same class can be used in some other program also this is an example like.

(Refer Slide Time: 11:05)

**Swing by inheritance**

We can also inherit the `JFrame` class, so there is no need to create the instance of `JFrame` class explicitly.

```
import javax.swing.*;
//inheriting JFrame
public class Simple2 extends JFrame{
    Simple2(){
        JButton b = new JButton("click");
        b.setBounds(130,100,100, 40);
        add(b);
        setSize(400,500);
        setLayout(null);
        setVisible(true);
    }

    public static void main(String[] args){
        new Simple2();
    }
}
```

DEBASIS SAMANTA

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

And another way the third way it is basically it is extending the Frame. It is almost the same as the previous one what is that this is basically class extend the frame and then the same way it is like this one. So, basically whether you can inherits a frame or not, in that case this basically program is limited to only Frame, but in the previous example you can create a Frame you can add it you can get other maybe panel, you can add some things into that panel or something else like.

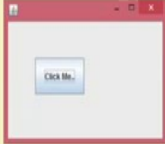
So, whatever the way you want to have it you can do it. So, these are the different way that we will follow we usually follow to include the swing components in our program anyway. So, any one method you can use it, but for the demonstration or for the illustration, we will consider the second method that we have discussed ok. So, these are the different way the frame the Java Swing component can be added into the frame and we will discuss the different classes one by one.



(Refer Slide Time: 12:10)

## Class 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.



Below is the declaration for `javax.swing.JButton` class.

```
public class JButton extends AbstractButton implements Accessible
```


IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

The first is JButton. JButton is basically as we have already familiar to this it is similar to that.

(Refer Slide Time: 12:16)

## Class JButton : Constructors

Constructor	Description
<code>JButton()</code>	It creates a button with no text and icon.
<code>JButton(String s)</code>	It creates a button with the specified text.
<code>JButton(Icon i)</code>	It creates a button with the specified icon object.



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

But in addition to the concept that is there it has something different than the previous one as we can see it is here. So, JButton icon as you can see this is basically default without any level and this is basically a button which will appear with certain level on it and this is basically one important Icon i; where icon is a class that is defined in the java dot Graphics dot Image and in that icon is basically shows an image; that means, a button

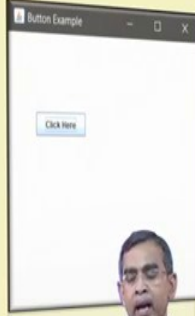
can includes an image. So, this is the idea about this one so; that means, a button can include nothing a button can include simple a text as a level, a button can include as an image also.

(Refer Slide Time: 12:57)

### Creating a JButton : An example

Let's see a simple swing example where we are creating one button and adding it on the JFrame object inside the main() method.

```
import javax.swing.*;
public class ButtonExample {
    public static void main(String[] args) {
        JFrame f=new JFrame("Button Example");
        JButton b=new JButton("Click Here");
        b.setBounds(50,100,95,30);
        f.add(b);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```




DEBASIS SAMANTA  
IIT KHARAGPUR

So, here is an example as a simple Button as we see this button is created and floated on a frame and then level of the Button is clicked here. And then another Button which is here as we see here text field and then the Button is also can be just here the basically the idea it is that if we click here and then this text field will appear.

(Refer Slide Time: 13:08)

### Java JButton : An example with ActionListener

```
import java.awt.event.*;
import javax.swing.*;
public class ButtonExample {
    public static void main(String[] args) {
        JFrame f=new JFrame("Button Example");
        final JTextField tf=new JTextField();
        tf.setBounds(50,50,150,20);
        JButton b=new JButton("Click Here");
        b.setBounds(50,100,95,30);
        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                tf.setText("Welcome to IIT Kharagpur.");
            }
        });
        f.add(b); f.add(tf);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```



DEBASIS SAMANTA  
IIT KHARAGPUR

So, this is basically action. So, what I want to see here is that all the event handling routine that we have already learnt also equally applicable to it also; that means, add ActionListener and then you can define the listener methods all these things.

As we see here in this program we do the same thing, we add action listener all the button and then we define the method for that action perform and then basically here is that whenever a button is clicked it will appear this kind of text. So, all.

(Refer Slide Time: 13:58)

**Java JButton : Displaying image on the button**

```
import javax.swing.*;
public class ButtonExample{
    ButtonExample() {
        JFrame f=new JFrame("Button Example");
        JButton b=new JButton(new ImageIcon("D:\\icon.png"));
        b.setBounds(100,100,100, 40);
        f.add(b);
        f.setSize(300,400);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        new ButtonExample();
    }
}
```

Button Example

DEBASIS SAMANTA  
IIT KHARAGPUR


IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, whatever the action handling mechanisms we have already learnt, that they can be applicable to this also. And this example indicates that with a Button can include an image as you see an image is also there on the Button itself. So, there are many small image can be embedded into the Button to go some more good feeling and good looking.

(Refer Slide Time: 14:19)

## Class 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.



Below is the declaration for **javax.swing.JLabel** class.

```
public class JLabel extends JComponent implements SwingConstants, Accessible
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

Now, JLabel same as the AWT Lable is basically same.

(Refer Slide Time: 14:19)

## Class JLabel : Constructors

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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

There is such different it has 3 4 different constructors.

(Refer Slide Time: 14:27)

Java JLabel methods	
Methods	Description
<code>String getText()</code>	It returns the text string that a label displays.
<code>void setText(String text)</code>	It defines the single line of text this component will display.
<code>void setHorizontalAlignment(int alignment)</code>	It sets the alignment of the label's contents along the X axis.
<code>Icon getIcon()</code>	It returns the graphic image that the label displays.
<code>int getHorizontalAlignment()</code>	It returns the alignment of the label's contents along the X axis.

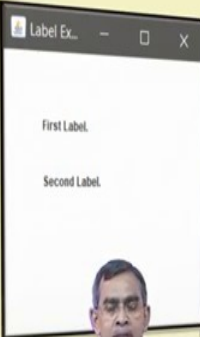
And also it has 5 different methods for accessing its or related to the JLabel we can have the different information by virtue of all these methods it is there.

(Refer Slide Time: 14:32)

### Creating a JLabel : An example

```
import javax.swing.*;

class LabelExample
{
    public static void main(String args[]) {
        JFrame f = new JFrame("Label Example");
        JLabel l1, l2;
        l1 = new JLabel("First Label.");
        l1.setBounds(50, 50, 100, 30);
        l2 = new JLabel("Second Label.");
        l2.setBounds(50, 100, 100, 30);
        f.add(l1); f.add(l2);
        f.setSize(300, 300);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```



The image shows a slide from a presentation. On the left, there is a code editor window displaying Java code for a class named 'LabelExample'. The code creates a 'JFrame' titled 'Label Example', then creates two 'JLabel' objects, 'l1' and 'l2'. 'l1' is initialized with the text 'First Label.' and 'l2' with 'Second Label.'. Both labels have their bounds set to (50, 50, 100, 30). They are added to the frame, the frame size is set to 300x300, the layout is set to null, and the frame is made visible. On the right, there is a screenshot of the resulting Java Swing window. The window has a title bar 'Label Ex...' and contains two labels stacked vertically: 'First Label' and 'Second Label'. At the bottom of the slide, there is a video feed of a man in a suit, presumably the presenter, and a footer with logos for IIT Kharagpur and NPTEL, along with the name 'DEBASIS SAMANTA'.

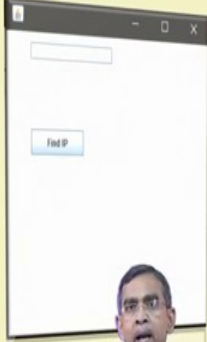
And this is an example as we see it basically display 2 Lables First Lable and Second Lable, the methods is almost same AWT which we have already discussed. So, I do not want to elaborate it.

(Refer Slide Time: 14:52)

### Java JLabel : An example with ActionListener

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

public class LabelExample extends Frame implements ActionListener{
    JTextField tf; JLabel l; JButton b;
    LabelExample(){
        tf=new JTextField();
        tf.setBounds(50,50, 150,20);
        l=new JLabel();
        l.setBounds(50,100, 250,20);
        b=new JButton("Find IP");
        b.setBounds(50,150,95,30);
        b.addActionListener(this);
        add(tf);add(tf);add(l);
        setSize(400,400);
        setLayout(null);
        setVisible(true);
    }
}
```




DEBASIS SAMANTA  
IIT KHARAGPUR

Same thing is there and then again any event can be added into this Lable as we have seen here. So, we can type it something and then if we type it, then the button you can use it and then to have many other things are there. So, it is action listing all these performance is also added can be added to a component as we have discussed.

(Refer Slide Time: 15:16)

### Java JTextField

The object of a **JTextField** class is a text component that allows the editing of a single line text. It inherits **JTextComponent** class.



Below is the declaration for **javax.swing.JTextField** class.

```
public class JTextField extends JTextComponent implements SwingConstants
```

DEBASIS SAMANTA  
IIT KHARAGPUR

JTextField is not similar to the same text field it is there.



(Refer Slide Time: 15:24)

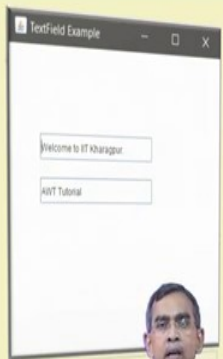
Class JTextField : Constructors	
Constructor	Description
JTextField()	Creates a new TextField
JTextField(String text)	Creates a new TextField initialized with the specified text.
JTextField(String text, int columns)	Creates a new TextField initialized with the specified text and columns.
JTextField(int columns)	Creates a new empty TextField with the specified number of columns.

And it is an example of the text field area it is there and it has so, many constructors 4 constructors and then 4 methods.

(Refer Slide Time: 15:28)

### Java JTextField : An example

```
import javax.swing.*;
class TextFieldExample
{
    public static void main(String args[]) {
        JFrame f= new JFrame("TextField Example");
        JTextField t1,t2;
        t1=new JTextField("Welcome to IIT Kharagpur.");
        t1.setBounds(50,100, 200,30);
        t2=new JTextField("AWT Tutorial");
        t2.setBounds(50,150, 200,30);
        f.add(t1); f.add(t2);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```



And here is an example as we see is very similar to the same example as AWT, this program includes 2 Text Field with default state welcome to IIT Kharagpur and AWT tutorial like.



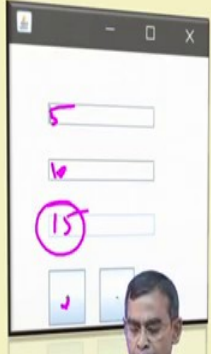
(Refer Slide Time: 15:40)

### Java JTextField : An example with ActionListener

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

public class TextFieldExample implements ActionListener{
    JTextField tf1,tf2,tf3;
    JButton b1,b2;

    TextFieldExample(){
        JFrame f= new JFrame();
        tf1=new JTextField();
        tf1.setBounds(50,50,150,20);
        tf2=new JTextField();
        tf2.setBounds(50,100,150,20);
        tf3=new JTextField();
        tf3.setBounds(50,150,150,20);
        tf3.setEditable(false);
        b1=new JButton("+");
        b1.setBounds(100,200,50,50);
        b2=new JButton("-");
        b2.setBounds(120,200,50,50);
        b1.addActionListener(this);
        b2.addActionListener(this);
        f.add(tf1);f.add(tf2);f.add(tf3);f.add(b1);f.add(b2);
    }
}
```



DEBASIS SAMANTA

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

And then action also can be added into any program like. So, here also another example JTestField with certain action added into there. Here is basically if you type something in this example if we type something say 5 and 10 and then plus you can add. So, 15 will appear. So, here is basically 3 Text Fields which is initially empty, but we have added event so, that if we type something here and there and then add it then result will be automatically come into this Text Field. So, the idea it is like this. So, action can be added in this. So, you can understand that how these kind of things can be utilized in your actual application.

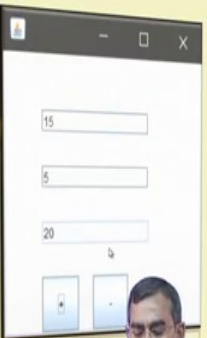
(Refer Slide Time: 16:26)

### Java JTextField : An example with ActionListener

```
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);

public void actionPerformed(ActionEvent e) {
    String s1=tf1.getText();
    String s2=tf2.getText();
    int a=Integer.parseInt(s1);
    int b=Integer.parseInt(s2);
    int c=0;
    if(e.getSource()==b1){
        c=a+b;
    }else if(e.getSource()==b2){
        c=a-b;
    }
    String result=String.valueOf(c);
    tf3.setText(result);
}

public static void main(String[] args) {
    new TextFieldExample();
}
```



DEBASIS SAMANTA

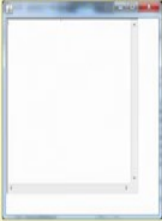
IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, this is the `TextFieldExample` that we have discussed this is an example.

(Refer Slide Time: 16:33)

**Class 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



Below is the declaration for `javax.swing.JTextField` class.

```
public class JTextArea extends JTextComponent
```

The slide footer includes the IIT Kharagpur logo, NPTEL Online Certification Courses logo, and the name DEBASIS SAMANTA.

And then text area also similar to the text area that we have discussed in the context of AWT. It looks like that.

(Refer Slide Time: 16:38)

**Class JTextArea : Constructors**

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

The slide footer includes the IIT Kharagpur logo, NPTEL Online Certification Courses logo, and the name DEBASIS SAMANTA.

It has 4 different methods and then 4 different construction and 5 different methods.


(Refer Slide Time: 16:41)


Class JTextArea : Methods	
Methods	Description
<code>void setRows(int rows)</code>	It is used to set specified number of rows.
<code>void setColumns(int cols)</code>	It is used to set specified number of columns.
<code>void setFont(Font f)</code>	It is used to set the specified font.
<code>void insert(String s, int position)</code>	It is used to insert the specified text on the specified position.
<code>void append(String s)</code>	It is used to append the given text to the end of the document.

(Refer Slide Time: 16:45)

### Java JTextArea : An example

```
import javax.swing.*;
public class TextAreaExample{
    TextAreaExample(){
        JFrame f= new JFrame();
        JTextArea area=new JTextArea("Welcome to IIT
Kharagpur");
        area.setBounds(10,30, 200,200);
        f.add(area);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        new TextAreaExample();
    }
}
```





And these are very simple example that text field area that will come here like this.

(Refer Slide Time: 16:54)

**Java JTextArea : An example with ActionListener**

```
f.setSize(450,450);  
f.setLayout(null);  
f.setVisible(true);  
}  
  
public void actionPerformed(ActionEvent e){  
    String text=area.getText();  
    String words[]=text.split(" ");  
    1. getText("Words: "+words.length);  
    2. setText("Characters: "+text.length());  
}  
  
public static void main(String[] args) {  
    new TextAreaExample();  
}  
}
```

The GUI shows a text area with the text "I am counting using the text to write multiple lines of text". Below the text area are two labels: "Words: 11" and "Characters: 59". A button labeled "Count Words" is at the bottom.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA

And action listening can be also added with action listener how the text will work it is shown there. Here is basically in this action listener one very important thing that you can note it is very interesting. So, this is basically your text area, in this text area you can add some type cut and paste also you can do no issue and then there is a button we have used it which basically count words. Now if we click it then automatically all these word count and then character count will come.

Now, how it is there you do not have to write any explicit code for that, you see what is the code it is here? So, setText the l1 and l2 are the 2 label 1 one and 1 two and then l1 dot setText ("words " " plus words dot length ) that mean here this words dot length already defined in the java dot at till how the lengths how many words are there in a text can be calculated similarly text dot length also this is a method which will return how many characters are there in this text.

So, it will display there. So, it is so, easy we do not have to do much programming actually, you do not have to account how many words is with a very critical program as you know probably in c or whatever it is other programming language. So, counting character also not that you have to stand one by one for the entire text. It is just simply the simple method that will help to do it very efficiently and effectively. So, this is a different application that which is coming in my mind to explain then I have just mentioned it.

(Refer Slide Time: 18:27)

**Java 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.

Enter the password:

Below is the declaration for `javax.swing.JPasswordField` class.

```
public class JPasswordField extends JTextField
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And then PasswordField it is a new one item which was not there in AWT components and you know the password is basically usually look like is it appear as a some wild character like they will star star star. So, it hide the actual character that you have typed it. So, this is this look like a text field area only, but it will appear with that kind of things are there.

(Refer Slide Time: 18:46)

**Class JPasswordField : Constructors**

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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

Now, so, first we the PasswordField constructor can be defined using these are the 3 4 constructors.




(Refer Slide Time: 18:54)

### Java JPasswordField : An example

```
import javax.swing.*;

public class PasswordFieldExample {
    public static void main(String[] args) {
        JFrame f=new JFrame("Password Field Example");
        JPasswordField value = new JPasswordField();
        JLabel l1=new JLabel("Password:");
        l1.setBounds(20,100, 80,30);
        value.setBounds(100,100,100,30);
        f.add(value); f.add(l1);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```



DEBASIS SAMANTA  
IIT KHARAGPUR

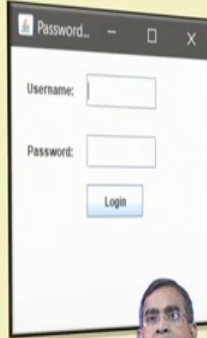
And it does not have any method and then this is a very simple example by which we can see you can practice it that how a PasswordField can be included in your frame and this is an example that you can find if you run this program.

(Refer Slide Time: 19:09)

### Java JPasswordField : An example with ActionListener

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

public class PasswordFieldExample {
    public static void main(String[] args) {
        JFrame f=new JFrame("Password Field Example");
        final JLabel label = new JLabel();
        label.setBounds(20,150, 200,30);
        final JPasswordField value = new JPasswordField();
        value.setBounds(100,75,100,30);
        JLabel l1=new JLabel("Username:");
        l1.setBounds(20,20, 80,30);
        JLabel l2=new JLabel("Password:");
        l2.setBounds(20,75, 80,30);
        JButton b = new JButton("Login");
        b.setBounds(100,120, 80,30);
        final JTextField text = new JTextField();
        text.setBounds(100,20, 100,30);
        f.add(value); f.add(l1); f.add(l2); f.add(b);
        f.add(text);
    }
}
```



DEBASIS SAMANTA  
IIT KHARAGPUR

And action listening also can be added there; that means, after getting the PasswordField you can verify that whether this is a correct password or not and then through verification you have to store I have to compare these things some store value somewhere else like. So, all these things you can do it here user name password user

name may be the text field and password may be say `PasswordField` and then login if you click it, then whatever the things that you have typed it login will be there.

(Refer Slide Time: 19:41)

The slide is titled "Java JPasswordField : An example with ActionListener". It features a code editor on the left with the following Java code:

```
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
b.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String data = "Username " + text.getText();
        data += ", Password: " + new String(value.getPassword());
        label.setText(data);
    }
})
```

On the right, there is a screenshot of a Java Swing window titled "Password ...". It contains two text fields: "Username:" with the text "Debasis" and "Password:" with masked characters "\*\*\*\*\*". Below these fields is a "Login" button. At the bottom of the window, a status bar displays "Username Debasis, Password: s...".

The slide footer includes the IIT Kharagpur logo, the text "NPTEL ONLINE CERTIFICATION COURSES", and the name "DEBASIS SAMANTA" with the IIT Kharagpur logo.

So, here you have to write an event handling methods for login whenever a user click the button login after giving text field login and then `PasswordField` password. So, this is the example that you can see. Here is basically user name Debasis password is something as it is there corrected. In this case we have just simply that value you have correctly read from the user interaction and then you can print it.

(Refer Slide Time: 20:04)

The slide is titled "Java JCheckBox". It contains the following text:

The **JCheckBox** class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a **JCheckBox** changes its state from "on" to "off" or from "off" to "on". It inherits **JToggleButton** class.

Below this text is a screenshot of a Java Swing window. It has an "E-mail" text field with "pubnol@gmail.com". Below the text field are three checkboxes labeled "Option 1", "Option 2", and "Option 3". "Option 1" and "Option 3" are checked, while "Option 2" is unchecked. At the bottom of the window is a "Submit Form" button.

Below the screenshot, the text says: "Below is the declaration for `javax.swing.JCheckBox` class."

The declaration is shown as:

```
public class JCheckBox extends JToggleButton implements Accessible
```

The slide footer includes the IIT Kharagpur logo, the text "NPTEL ONLINE CERTIFICATION COURSES", and the name "DEBASIS SAMANTA" with the IIT Kharagpur logo.



Now, checkbox is JCheckBox is similar to the AWT checkbox and as we see the checkbox is basically which lay option one. So, in this example as you see this is the basically checkbox, it has the 3 items and then we can click here multiple click is also possible or single click is also possible that you have to define it by means of its appropriate constructor, whether you have to do the single click or you have to do the multiple click.

(Refer Slide Time: 20:34)

Class JCheckBox : Constructors	
Constructor	Description
JCheckBox()	Creates an initially unselected check box button with no text, no icon.
JCheckBox(String s)	Creates an initially unselected check box with text.
JCheckBox(String text, boolean selected)	Creates a check box with text and specifies whether or not it is initially selected.
JCheckBox(Action a)	Creates a check box where properties are taken from the Action supplied.

And it has got 4 constructors and two methods only.


(Refer Slide Time: 20:37)

Java JCheckBox methods	
Methods	Description
AccessibleContext getAccessibleContext()	It is used to get the AccessibleContext associated with this JCheckBox.
protected String paramString()	It returns a <a href="#">string</a> representation of this JCheckBox.

(Refer Slide Time: 20:40)

### Java JCheckBox : An example

```
import javax.swing.*;
public class CheckBoxExample
{
    CheckBoxExample() {
        JFrame f= new JFrame("CheckBox Example");
        JCheckBox checkBox1 = new JCheckBox("C++");
        checkBox1.setBounds(100,100, 50,50);
        JCheckBox checkBox2 = new JCheckBox("Java", true);
        checkBox2.setBounds(100,150, 50,50);
        f.add(checkBox1);
        f.add(checkBox2);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        new CheckBoxExample();
    }
}
```



DEBASIS SAMANTA  
IIT KHARAGPUR

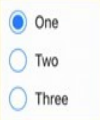
And this is a simple example how the JCheckBox can be done using a single multiple click C++ and Java are the 2 checkbox choices in the checkbox is included and it looks like that.

(Refer Slide Time: 20:54)

### Class JRadioButton

The **JRadioButton** class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.

It should be added in **ButtonGroup** to select one radio button only.



Below is the declaration for `javax.swing.JRadioButton` class.

```
public class JRadioButton extends JToggleButton implements Accessible
```

DEBASIS SAMANTA  
IIT KHARAGPUR

And JRadioButton radio button we have already familiar with the AWT it is also there, but here the loops of the JButton is much more elegant than the AWT and this is an example of a RadioButton.

(Refer Slide Time: 21:06)

### Class JRadioButton : Constructors

Constructor	Description
<code>JRadioButton()</code>	Creates an unselected radio button with no text.
<code>JRadioButton(String s)</code>	Creates an unselected radio button with specified text.
<code>JRadioButton(String s, boolean selected)</code>	Creates a radio button with the specified text and selected status.

DEBASIS SAMANTA  
IIT KHARAGPUR

Say button group actually the object that you have to create it and it has the 3 constructors.

(Refer Slide Time: 21:09)

### Class JRadioButton : Methods

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

DEBASIS SAMANTA  
IIT KHARAGPUR

And these are the so, many methods as you see.

(Refer Slide Time: 21:14)

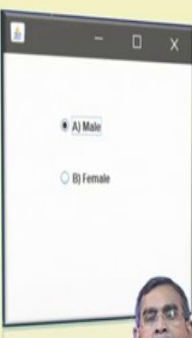
### Java JRadioButton : An example

```
import javax.swing.*;

public class RadioButtonExample {
    JFrame f;

    RadioButtonExample() {
        f = new JFrame();
        JRadioButton r1 = new JRadioButton("A) Male");
        JRadioButton r2 = new JRadioButton("B) Female");
        r1.setBounds(75, 50, 100, 30);
        r2.setBounds(75, 100, 100, 30);
        ButtonGroup bg = new ButtonGroup();
        bg.add(r1); bg.add(r2);
        f.add(r1); f.add(r2);
        f.setSize(300, 300);
        f.setLayout(null);
        f.setVisible(true);
    }

    public static void main(String[] args) {
        new RadioButtonExample();
    }
}
```



DEBASIS SAMANTA  
IIT KHARAGPUR

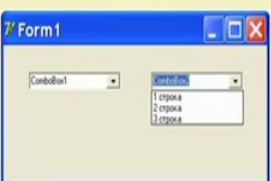
And then this is an simple example and if you implement this example you can find this kind of output that you can find it and this is a self explanatory. So, it has basically 2 RadioButtons as we can see one is A Male and then B Female and then you can select anyone whichever you select it will be highlighted by a Button form.

And this is the example you can add more Buttons and then you can run it and you can find it how it will appear.

(Refer Slide Time: 21:42)

### Class JComboBox

The object of **Choice** class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits **JComponent** class.



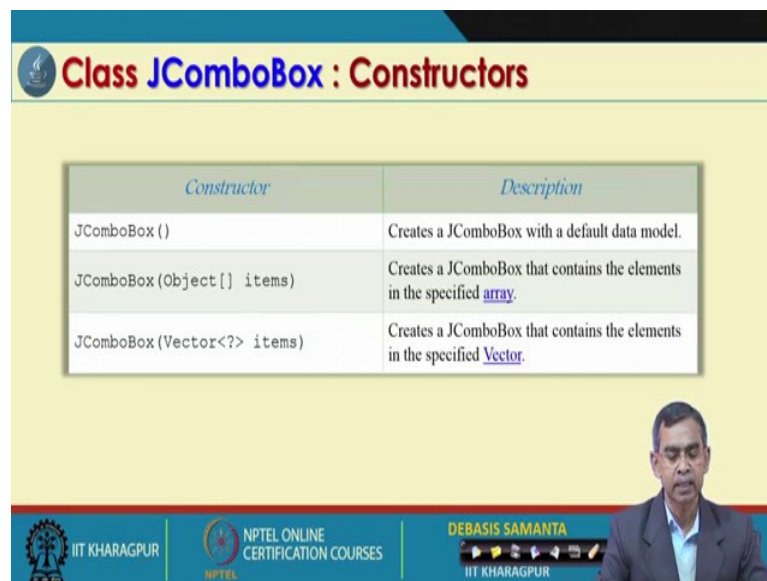
Below is the declaration for `javax.swing.JComboBox` class.

```
public class JComboBox extends JComponent implements ItemSelectable, ListDataListener, ActionListener, Accessible
```

DEBASIS SAMANTA  
IIT KHARAGPUR

And then JComboBox it is just similar to the list items there in AWT and here the in this example as we see 2 ComboBox; ComboBox1 and ComboBox2 is created and in the ComboBox1 there may be some items and in the ComboBox2 also some items. So, as many as ComboBox can be included in any frame or any effects also and then anyone items can be selected, which item you select that item will appear in the text field area in the main ComboBox slide.

(Refer Slide Time: 22:11)

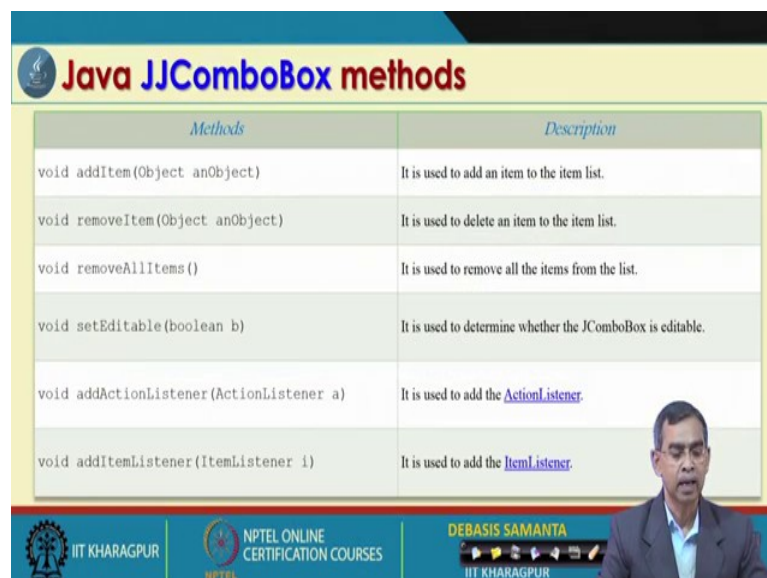


Constructor	Description
<code>JComboBox ()</code>	Creates a JComboBox with a default data model.
<code>JComboBox (Object[] items)</code>	Creates a JComboBox that contains the elements in the specified <a href="#">array</a> .
<code>JComboBox (Vector&lt;?&gt; items)</code>	Creates a JComboBox that contains the elements in the specified <a href="#">Vector</a> .

DEBASIS SAMANTA  
IIT KHARAGPUR

And these this has 3 constructors and then there are five methods declare in it.

(Refer Slide Time: 22:13)




Methods	Description
<code>void addItem (Object anObject)</code>	It is used to add an item to the item list.
<code>void removeItem (Object anObject)</code>	It is used to delete an item to the item list.
<code>void removeAllItems()</code>	It is used to remove all the items from the list.
<code>void setEditable (boolean b)</code>	It is used to determine whether the JComboBox is editable.
<code>void addActionListener (ActionListener a)</code>	It is used to add the <a href="#">ActionListener</a> .
<code>void addItemListener (ItemListener i)</code>	It is used to add the <a href="#">ItemListener</a> .

DEBASIS SAMANTA  
IIT KHARAGPUR

(Refer Slide Time: 22:17)

### Java JComboBox : An example

```
import javax.swing.*;
public class JComboBoxExample {
    JFrame f;
    JComboBoxExample() {
        f=new JFrame("JComboBox Example");
        String
        country[]={"India","Aus","U.S.A","England","Newzealand"};
        JComboBox cb=new JComboBox(country);
        cb.setBounds(50, 50,90,20);
        f.add(cb);
        f.setLayout(null);
        f.setSize(400,500);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new JComboBoxExample();
    }
}
```




DEBASIS SAMANTA  
IIT KHARAGPUR

This is a simple example for your practice, as you can run it then you can find it that this is a one JComboBox that will appear and in your program.

(Refer Slide Time: 22:27)

### Class JTable

The **JTable** class is used to display data in tabular form. It is composed of rows and columns.



First Name	Last Name	Sport	# of Years	Vegetarian
Kathy	Smith	Snowboarding	5	false
John	Doe	Rowing	3	true
See	Black	Knitting	2	false
Jane	White	Speed reading	20	true
Joe	Brown	Pool	10	false

OK

DEBASIS SAMANTA  
IIT KHARAGPUR

Now, Java Swing JTable this is a new one addition compared to the AWT it is it was not there in AWT, but it is here now this class JTable is basically showing how the entire table can be displayed.

So, if there is a table already stored in your machine in a memory then that table can be displayed here.



(Refer Slide Time: 22:45)

## Class JTable : Constructors

Constructor	Description
JTable()	Creates a table with empty cells.
JTable(Object[][] rows, Object[] columns)	Creates a table with the specified data.

DEBASIS SAMANTA  
IIT KHARAGPUR

But in I have just these are this for the creating JTable it has only 2 constructors.


(Refer Slide Time: 22:51)

## Java JTable : An example

```
import javax.swing.*;

public class TableExample {
    JFrame f;
    TableExample() {
        f = new JFrame();
        String data[][] = { {"101", "Amit", "670000"},
                            {"102", "Jai", "780000"},
                            {"101", "Sachin", "700000"} };

        String column[] = {"ID", "NAME", "SALARY"};
        JTable jt = new JTable(data, column);
        jt.setBounds(30, 40, 200, 300);
        JScrollPane sp = new JScrollPane(jt);
        f.add(sp);
        f.setSize(300, 400);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new TableExample();
    }
}
```



DEBASIS SAMANTA  
IIT KHARAGPUR

And then this is a simple program by which you can display one table here we have just created a table in program itself.

And as you see this table has three rows and then column heading is ID, NAME and then SALARY and the table will look like this as you see here. So, it is like this. So, here the table is just for an example here, but you can create a set of objects having the different




fields and then we can get the value and then we can put in the table and then table can be displayed.

(Refer Slide Time: 23:25)

## Class JList

The object of **JList** class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items. It inherits **JComponent** class.



Below is the declaration for `javax.swing.JList` class.

```
public class JList extends JComponent implements Scrollable, Accessible
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

(Refer Slide Time: 23:27)

## Class JList : Constructors

Constructor	Description
<code>JList()</code>	Creates a JList with an empty, read-only, model.
<code>JList(ary[] listData)</code>	Creates a JList that displays the elements in the specified array.
<code>JList(ListModel&lt;ary&gt; dataModel)</code>	Creates a JList that displays elements from the specified, non-null, model.

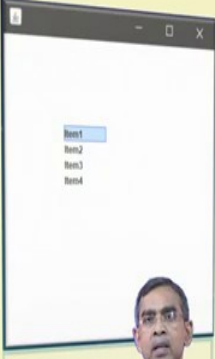
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

JList is also similar to the list actually. And we do not have to discuss much about because it is been in many way very similar to AWT list it has 3 constructor 4 methods.

(Refer Slide Time: 23:36)

### Java JList : An example

```
import javax.swing.*;
public class ListExample
{
    ListExample() {
        JFrame f = new JFrame();
        DefaultListModel<String> l1 = new DefaultListModel<>();
        l1.addElement("Item1");
        l1.addElement("Item2");
        l1.addElement("Item3");
        l1.addElement("Item4");
        JList<String> list = new JList<>(l1);
        list.setBounds(100,100, 75,75);
        f.add(list);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        new ListExample();
    }
}
```



DEBASIS SAMANTA  
IIT KHARAGPUR

And this is simple program you should practice it here actually it shows a list with 4 items it is defined there.

(Refer Slide Time: 23:45)

### Class JOptionPane

The **JOptionPane** class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. The **JOptionPane** class inherits **JComponent** class.



Below is the declaration for `javax.swing.JOptionPane` class.

```
public class JOptionPane extends JComponent implements Accessible
```

DEBASIS SAMANTA  
IIT KHARAGPUR

And JOptionPane it is also a new things which was not there in AWT. JOptionPane is basically is a popup dialog box or sometimes it is called the message box whenever you make some mistake automatically windows popup with a message box or window actually. So, if you want to create any window like this with certain messages, then this

is the right way and if you select then means you have accepted and if you select stop this view then you can do it.

(Refer Slide Time: 24:15)

### Class JOptionPane : Constructors

Constructor	Description
<code>JOptionPane()</code>	It is used to create a JOptionPane with a test message.
<code>JOptionPane(Object message)</code>	It is used to create an instance of JOptionPane to display a message.
<code>JOptionPane(Object message, int messageType)</code>	It is used to create an instance of JOptionPane to display a message with specified message type and default options.

DEBASIS SAMANTA  
IIT KHARAGPUR

So, this is the way the option pane can be created there are 3 constructors and then few methods are there.

(Refer Slide Time: 24:18)

### Java JOptionPane : An example

```
import javax.swing.*;
public class OptionPaneExample {
    JFrame f;
    OptionPaneExample() {
        f = new JFrame();
        JOptionPane.showMessageDialog(f, "Hello, Welcome to IIT Kharagpur.");
    }
    public static void main(String[] args) {
        new OptionPaneExample();
    }
}
```

Message  
Hello, Welcome to IIT Kharagpur.  
OK


DEBASIS SAMANTA  
IIT KHARAGPUR

And this is an example to create a simple option pane with a message hello welcome to IIT like. So, it is sometimes time to time we have to give our message to your user, then you can do user the option pane.

(Refer Slide Time: 24:31)

## Java JScrollBar

The object of **JScrollBar** class is used to add horizontal and vertical scrollbar. It is an implementation of a scrollbar. It inherits **JComponent** class.



Below is the declaration for `javax.swing.JScrollBar` class.

```
public class JScrollBar extends JComponent implements Adjustable, Accessible
```


IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And then JScrollBar bar is similar to the scrollbar bar that is there. So, is basically same thing horizontal and vertical scrollbar can be created.

(Refer Slide Time: 24:39)

## Java JScrollBar constructors

Constructor	Description
<code>JScrollBar()</code>	Creates a vertical scrollbar with the initial values.
<code>JScrollBar(int orientation)</code>	Creates a scrollbar with the specified orientation and the initial values.
<code>JScrollBar(int orientation, int value, int extent, int min, int max)</code>	Creates a scrollbar with the specified orientation, value, extent, minimum, and maximum.




IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

There are 3 constructor to create a Scrollbar.

(Refer Slide Time: 24:42)

### Java JScrollBar : An example

```
import javax.swing.*;
class ScrollBarExample
{
    ScrollBarExample() {
        JFrame f = new JFrame("Scrollbar Example");
        JScrollBar s = new JScrollBar();
        s.setBounds(100, 100, 50, 100);
        f.add(s);
        f.setSize(400, 400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        new ScrollBarExample();
    }
}
```



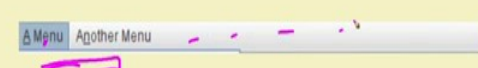
DEBASIS SAMANTA  
IIT KHARAGPUR

And here is a simple example by which one Scrollbar can be created it is a vertical scrollbar that we have created likewise horizontal Scrollbar, Scrollbar with more animation and everything can be also done.

(Refer Slide Time: 24:55)

### Class JMenuBar

The **JMenuBar** class is used to display menu bar on the window or frame. It may have several menus.



Below is the declaration for `javax.swing.JMenuBar` class.

```
public class JMenuBar extends JComponent implements MenuElement, Accessible
```

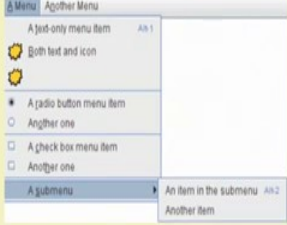
DEBASIS SAMANTA  
IIT KHARAGPUR

JMenuBar is very similar to JMenuBar is also a new things which was not there in AWT is basically as you see this is the MenuBar actually this is the MenuBar if you click it some other some Menu will appear like this one. So, the many MenuBar also can be included in your program and this is the idea about the JMenuBar.

(Refer Slide Time: 25:20)

## Class JMenu

The object of **JMenu** class is a pull down menu component which is displayed from the menu bar. It inherits the **JMenuItem** class.



Below is the declaration for `javax.swing.JMenuItem` class.

```
public class JMenuItem extends JMenuItem implements MenuElement, Accessible
```


IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

For this JMenuBar there is again few constructors are there. So, usually JMenuBar and JMenuItem related to each other.

(Refer Slide Time: 25:27)

## Java JMenuItem & JMenu : An example

```
import javax.swing.*;
class MenuExample {
    JMenu menu, submenu;
    JMenuItem i1, i2, i3, i4, i5;
    MenuExample() {
        JFrame f = new JFrame("Menu and MenuItem Example");
        JMenuBar mb = new JMenuBar();
        menu = new JMenu("Menu");
        submenu = new JMenu("Sub Menu");
        i1 = new JMenuItem("Item 1"); i2 = new JMenuItem("Item 2");
        i3 = new JMenuItem("Item 3"); i4 = new JMenuItem("Item 4");
        menu.add(i1); menu.add(i2); menu.add(i3);
        submenu.add(i4);
        menu.add(submenu); mb.add(menu);
        f.setMenuBar(mb);
        f.setSize(400, 400); f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[]) {
        new MenuExample();
    }
}
```



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

So, I will just give an idea about JMenuBar which is basically include some JMenu and there is also JMenuItem. So, this is an example as we see here it basically includes MenuItems 1 and i 2 and then it basically shows there and as you know for MenuItem there will be sub menu. So, it is sub menu under sub menu all these things. So, those things can be cleared. So, this program you can practice for your own understanding.

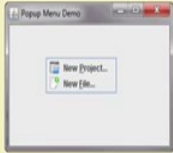


And then finally, the JPopupMenu; JPopupMenu is basically is a whenever something is appear the menu will popup like.

(Refer Slide Time: 26:02)

### Class JPopupMenu

JPopupMenu can be dynamically popped up at specific position within a component. It inherits the JComponent class.



Below is the declaration for `javax.swing.JPopupMenu` class.

```
public class JPopupMenu extends JComponent implements Accessible, MenuElement
```


IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And it is look like this. So, PopuMenu whenever you create it and then some menu will appear.

(Refer Slide Time: 26:07)

### Class JPopupMenu : Constructors

Constructor	Description
<code>JPopupMenu ()</code>	Constructs a JPopupMenu without an "invoker".
<code>JPopupMenu(String label)</code>	Constructs a JPopupMenu with the specified title.



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

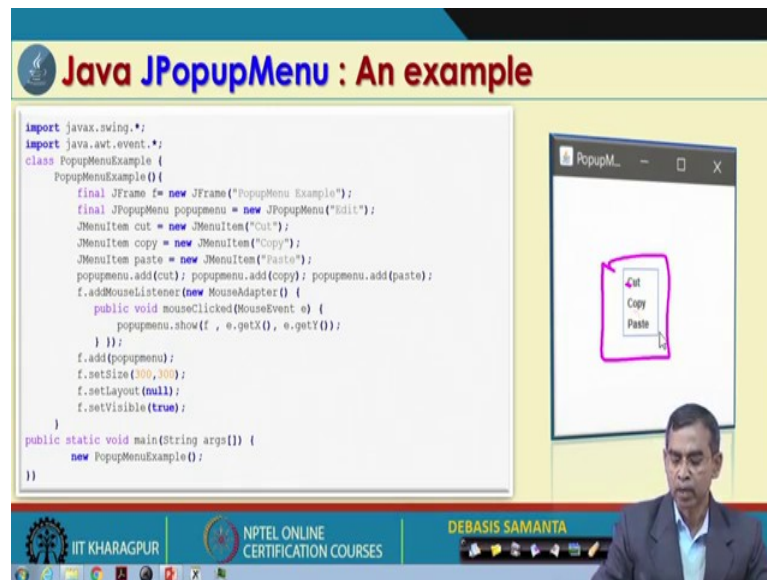
It has 2 constructor as we see here on constructor without any argument another is which string argument.



(Refer Slide Time: 26:15)

### Java JPopupMenu : An example

```
import javax.swing.*;
import java.awt.event.*;
class PopupMenuExample {
    PopupMenuExample() {
        final JFrame f = new JFrame("PopupMenu Example");
        final JPopupMenu popupmenu = new JPopupMenu("Edit");
        JMenuItem cut = new JMenuItem("Cut");
        JMenuItem copy = new JMenuItem("Copy");
        JMenuItem paste = new JMenuItem("Paste");
        popupmenu.add(cut); popupmenu.add(copy); popupmenu.add(paste);
        f.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                popupmenu.show(f, e.getX(), e.getY());
            }
        });
        f.add(popupmenu);
        f.setSize(300, 300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[]) {
        new PopupMenuExample();
    }
}
```



And here is a simple example that you can see this example as you see whenever some things clicked. So, it will basically a PopupMenu will appear this is the PopupMenu and then you can select anyone cut copy paste like.

As probably you know whenever you use some word or some other text reaching document and whenever you have to a select the edit and then usually cut copy paste all these things are there. So, it is just as the idea is very similar to that concept only. So, this is the PopupMenu and then JavaSwingCheckboxMenu.

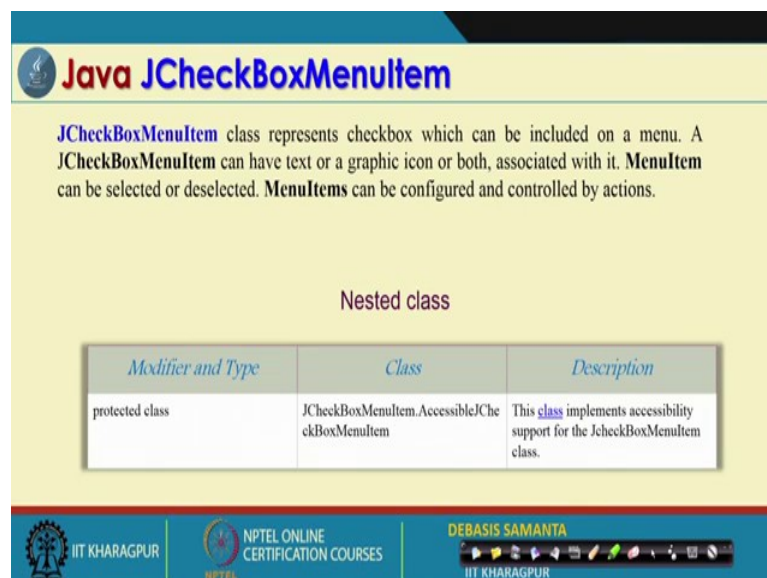
(Refer Slide Time: 26:52)

### Java JCheckBoxMenuItem

**JCheckBoxMenuItem** class represents checkbox which can be included on a menu. A **JCheckBoxMenuItem** can have text or a graphic icon or both, associated with it. **MenuItem** can be selected or deselected. **MenuItems** can be configured and controlled by actions.

Nested class

Modifier and Type	Class	Description
protected class	JCheckBoxMenuItem.AccessibleJCheckBoxMenuItem	This class implements accessibility support for the JCheckBoxMenuItem class.



CheckBoxMenuItem is very similar to the CheckBoxMenu CheckBoxConcept there is a AWT class.

(Refer Slide Time: 26:57)

Class JCheckBoxMenuItem : Constructors	
Constructor	Description
JCheckBoxMenuItem()	It creates an initially unselected check box menu item with no set text or icon.
JCheckBoxMenuItem(Action a)	It creates a menu item whose properties are taken from the Action supplied.
JCheckBoxMenuItem(Icon icon)	It creates an initially unselected check box menu item with an icon.
JCheckBoxMenuItem(String text)	It creates an initially unselected check box menu item with text.
JCheckBoxMenuItem(String text, boolean b)	It creates a check box menu item with the specified text and selection state.
JCheckBoxMenuItem(String text, Icon icon)	It creates an initially unselected check box menu item with the specified text and icon.
JCheckBoxMenuItem(String text, Icon icon, boolean b)	It creates a check box menu item with the specified text, icon, and selection state.

It is basically under the component class in the other.

(Refer Slide Time: 27:03)

Class JCheckBoxMenuItem : Methods		
Modifier	Method	Description
AccessibleContext	getAccessibleContext()	It gets the AccessibleContext associated with this JCheckBoxMenuItem.
Object[]	getSelectedObjects()	It returns an <u>array</u> (length 1) containing the check box menu item <u>label</u> or null if the check box is not selected.
boolean	getState()	It returns the selected-state of the item.
<u>String</u>	getUIClassID()	It returns the name of the L&F class that renders this component.
protected String	paramString()	It returns a string representation of this JCheckBoxMenuItem.
void	setState(boolean b)	It sets the selected-state of the item.

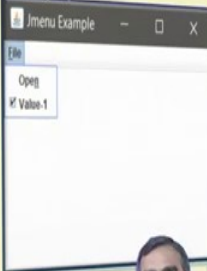
It has the so, many constructors usually little bit complex to use it.

(Refer Slide Time: 27:07)

### Java JCheckBoxMenuItem : An example

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import javax.swing.AbstractButton;
import javax.swing.Icon;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.MenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPopupMenu;

public class JavaCheckBoxMenuItemExample {
    public static void main(String args[]) {
        JFrame frame = new JFrame("JMenu Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JMenuBar menuBar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        fileMenu.setMnemonic(KeyEvent.VK_F);
        menuBar.add(fileMenu);
        JMenuItem menuItem1 = new JMenuItem("Open", KeyEvent.VK_O);
        fileMenu.add(menuItem1);
        JCheckBoxMenuItem caseMenuItem = new JCheckBoxMenuItem("Option 1");
        caseMenuItem.setMnemonic(KeyEvent.VK_C);
```



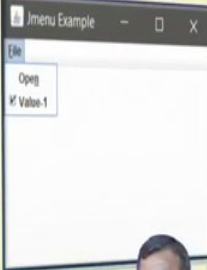
DEBASIS SAMANTA  
IIT KHARAGPUR

And then this is an example. So, this example we you can see here in this example you have added many thing button then JFrame Menu MenuBar MenuItem in the same example and also action in (Refer Time: 27:25) event is also used here. So, you can verify this algorithm this program and then you can check that how it is working. So, it is just look like a JCheckBox menu item file, and file open close save as all these things you can add it.

(Refer Slide Time: 27:38)

### Java JCheckBoxMenuItem : An example

```
fileMenu.add(caseMenuItem);
ActionListener aListener = new ActionListener() {
    public void actionPerformed(ActionEvent event) {
        AbstractButton aButton = (AbstractButton) event.getSource();
        boolean selected = aButton.getModel().isSelected();
        String newLabel;
        Icon newIcon;
        if (selected) {
            newLabel = "Value-1";
        } else {
            newLabel = "Value-2";
        }
        aButton.setText(newLabel);
    }
};
caseMenuItem.addActionListener(aListener);
frame.setJMenuBar(menuBar);
frame.setSize(350, 350);
frame.setVisible(true);
}
```



DEBASIS SAMANTA  
IIT KHARAGPUR

So, this is the part of the program that you can run it.

(Refer Slide Time: 27:41)

**? Questions to think...**

- Why Swing components are called lightweights, whereas AWT components are called heavyweight?
- How Swing is platform independent whereas AWT is not so?

DEBASIS SAMANTA  
IIT KHARAGPUR

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Anyway so, we have discussed about the Swing components, not the all components there are few more components are there and from this understanding probably you have heard about that the Swing is a new one compared to the AWT which was very old. And using the swing components we can built many other we can include many elements many components into our programs which was not possible in AWT. There are few more what components are also that we have yet to discuss. So, those things will be discussed in our next module.

Thank you very much.