

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

Lecture - 40
A W T Programming-II



This is the next part in our AWT learning. We have discussed about few basic components and those are very very much useful to build our window program. Now we are going to discuss about layout manager in this module. So, there are different layouts that are need that needs to be I mean managed your windows and that we will discussed about it. So, so basically this discussion related to GUIs GUIs with layout managers.

(Refer Slide Time: 00:51)

Layout Managers

- It is a tedious process (particularly with a large number of components) to place components on the screen (it has to be specified explicitly).
- To tackle this, AWT includes the notion of **layout managers**: the layout of components in a container may be governed by a layout manager.
- A **Container** in Java is nothing but an applet which generally contains number of GUI attributes such as **Frame**, **Panel**, **Label**, **Button**, etc.
- Each **Container** object has a **layout manager** which is an instance of any class that implements the **Layout Manager** interface.
- Each component, such as a **Panel**, or a **Frame** has a default layout manager associated with it which may be changed by the Java developer when an instance of that container is created.
- Each layout manager keeps track of a list of **Components**. The layout manager is notified each time one adds a **Component** to a **Panel**, for example.
- Java has the following five predefined layout manager classes :

FlowLayout BorderLayout GridLayout CardLayout

 IIT KHARAGPUR  NPTEL ONLINE CERTIFICATION COURSES DEBASIS SAMANTA
IIT KHARAGPUR

Now, what are the different layout managers? Again, layout managers are defined in the AWT package and it can be used either explicitly or implicitly, if you do not mention it also some layout by default will be there. And mainly Java has the four different types of layout manager classes called the FlowLayout, the BorderLayout, GridLayout and then CardLayout and all this layout is basically to manage how the entire container, either is a frame or panel or a container whatever it is can be managed.

Now, we will discuss about each manager individually one by one with example and also we will see exactly what are the different constructors and methods are there to manage all these things. Let us first have the discussion on FlowLayout Manager.

(Refer Slide Time: 01:44)

FlowLayout Manager

The **FlowLayout** is used to arrange the components in a line, one after another (in a flow). It is the default layout of applet or panel.

Components Added by FlowLayout

The slide features a yellow background with a blue header. The title 'FlowLayout Manager' is in large, bold letters. Below it, a text box explains the purpose of FlowLayout. A diagram illustrates how components are added in a row and wrap to the next line. The bottom of the slide has a blue footer with logos for IIT Kharagpur and NPTEL, and a video feed of the presenter, Debasis Samanta.

So, what is the meaning of the FlowLayout manager? It is there. Now this manager helps the programmer to manage the different elements into the container. Now so how so if you can go on adding adding adding. So, how they can be added to the container? So, layout manager this FlowLayout manager tells you that if you add the first component then it will go here in this position.

Now, the next component we will go automatically here, then next here and once these all the this are the special field then the next we will come there. So, it is basically this row and then this one. So, row-wise and then this one. So, this way container we will go on automatically filling. Now here the idea is that you do not have to explicitly mention that whether this button will go there this button you just go on creating the button and then just add into this using the FlowLayout manager then FlowLayout manager automatically adjusts them.

But; however, you can say there is a gap between the different elements and everything that you can specify and there is a facility to do this things using this FlowLayout manager.

(Refer Slide Time: 03:00)

Page 30/30

Class FlowLayout : Constructor

Constructor	Description
FlowLayout()	Creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.
FlowLayout(int align)	Creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.
FlowLayout(int align, int hgap, int vgap)	Creates a check box with the specified label and sets the specified state.

Handwritten red arrows point from the descriptions to the right side of the slide.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA

Now, let us see what is the class definition for this manager it is and, so this manager has three constructors as we see here FlowLayout and then FlowLayout integer alignment; that means, which is the whether horizontal alignment or vertical the last example that I have seen that arrives horizontal element. And then vertical element means it maybe this then this also. So, this way also it is there otherwise this way is there.

So, there are different elements are there actually. Anyways; so there are different constructors are there and again if we see this is another constructor hgap and then vgap; that means. So, if this is the one component is added here another component. So, if this is the hgap, now if it is another component so this is basically vgap. So, all these things you can mention in your an I mean FlowLayout manager class constructor and then they can be automatically adjusted.

So, this is the different way the FlowLayout manager helps a programmer to maintain or to include the different components in them.

(Refer Slide Time: 04:11)

Class FlowLayout : Fields		
Modifier and Type	Method	Description
static int	CENTER	This value indicates that each row of components should be centered.
static int	LEADING	This value indicates that each row of components should be justified to the leading edge of the container's orientation, for example, to the left in left-to-right orientations.
static int	LEFT	This value indicates that each row of components should be left-justified.
static int	RIGHT	This value indicates that each row of components should be right-justified.
static int	TRAILING	This value indicates that each row of components should be justified to the trailing edge of the container's orientation, for example, to the right in left-to-right orientations.

So, this is the usefulness of this FlowLayout manager. Now let us have the methods and these are few fields are there center leading left-right trailing that is the different way that manager can be decided. Center means all is centered, left means left alignment all these are the usual meanings actually.

(Refer Slide Time: 04:24)

Creating a FlowLayout : An example

```
import java.awt.*;

public class MyFlowLayout{
    Frame f;

    MyFlowLayout() {
        f = new Frame();
        Button b1 = new Button("1");
        Button b2 = new Button("2");
        Button b3 = new Button("3");
        Button b4 = new Button("4");
        Button b5 = new Button("5");
        f.add(b1); f.add(b2); f.add(b3); f.add(b4); f.add(b5);
        f.setLayout(new FlowLayout(FlowLayout.RIGHT));
        //setting flow layout of right alignment
        f.setSize(300,300);
        f.setVisible(true);
    }

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

And here is an example let us see how we can use this FlowLayout manager in a program. We have defined one class MyFlowLayout is the name of the java program that we are going to create it. First, we create a frame and then we create 5 buttons here, all

these buttons are labeled as 1, 2, 3, 4, 5 like; so 5 buttons are created. Now, what is our objective? Our objective is to add this button into this frame.

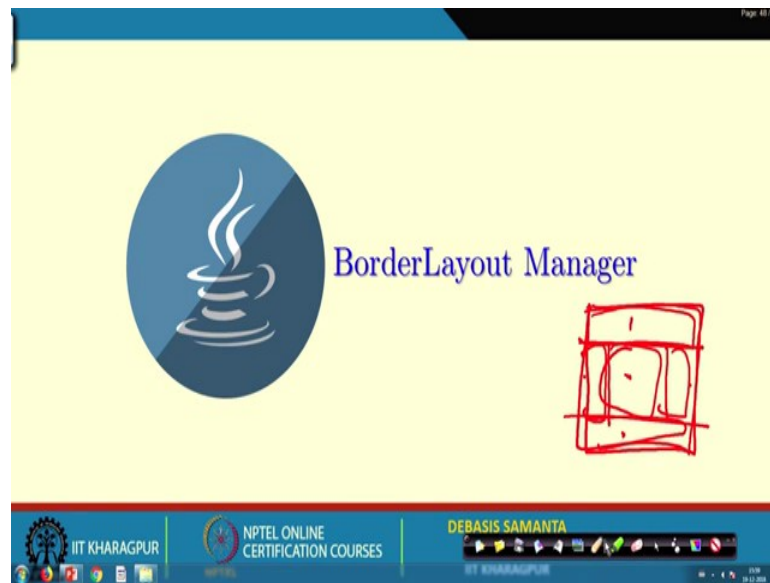
So, these are basically the statement by which we can add all this 5 buttons into the frame. Now here next point you see there is that this frame can be set with layout, this is the one method which is defined the component class and then component class and f f the f is the frame being the frame as a subclass of component which is the setLayout is accessible to it and then setLayout we just made the setLayout using this FlowLayout, FlowLayout RIGHT.

That means, it is right alignments, now as you see these basically the manager FlowLayout manager FlowLayout manager manages all the elements in the right alignment and right alignment, but in this direction, this is basically this direction like. And if you can add more then it will come again this then this and then this it is like. So, always right alignment way.

So, after f 5 is you can f 6 f 7 f 8 f 9 f 6 f 7 f 8 f 9, but align will right only. So, this way this is basically the role that the layout manager, in this case, FlowLayout manager can be played can play here and you see how the FlowLayout manager can be assigned into this one. Our all earlier example the default manager is basically FlowLayout manager we have not mentioned it, if you do not mention then default is there, but here we have mentioned FlowLayout with right alignment this is the second constructor that we have called it and this is basically setting the size of the frame and then make the visible true; so that the frame can be visible to the user.

So, this is the way the FlowLayout manager can be used in your program to manage the different classes are there. Now we will discuss about the applet in discuss applet there are so many buttons so many other component element is there. So, they all can be managed using this FlowLayout manager or some other manager may be other managers also there.

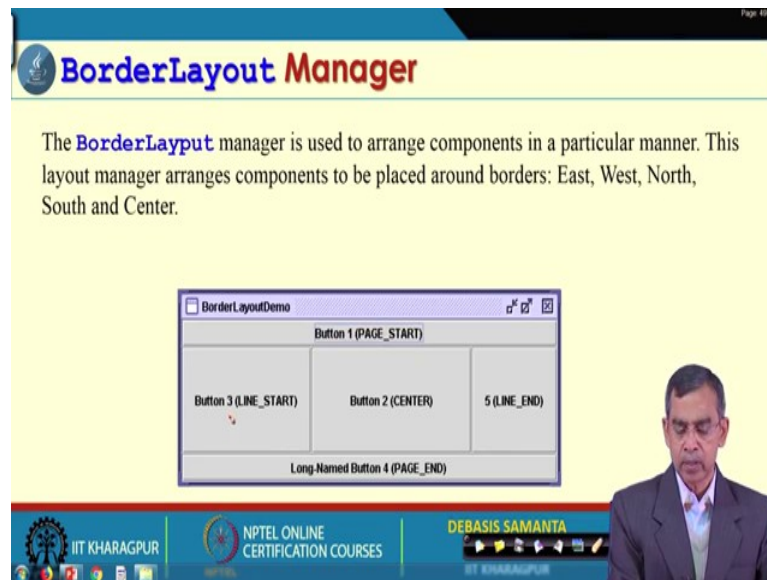
(Refer Slide Time: 06:58)



Now, let us see what are the other managers that can be considered now next example is called the BorderLayout manager. So, idea of the BorderLayout manager is basically if this is your entire if this is the entire say let it be frame or applet whatever it is there then BorderLayout means it basically has few borders, one this is the one border and one border, another border, another border, and center is there.

So, this is basically north, south, east, west, and center. So, this called the border nowhere the idea is that this layout manager we will place some component in this area then some component here some component here some component here; so whole the frame can be managed in this form actually. So, this is the idea about the BorderLayout manager. Now let us see how the BorderLayout manager can be invoked and where it is I mean how it the class it is for this one.

(Refer Slide Time: 07:54)



BorderLayout Manager

The **BorderLayout** manager is used to arrange components in a particular manner. This layout manager arranges components to be placed around borders: East, West, North, South and Center.

Button 1 (PAGE_START)

Button 3 (LINE_START)

Button 2 (CENTER)

Button 5 (LINE_END)

Long-Named Button 4 (PAGE_END)

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

So, it is basically a general look of the BorderLayout manager as I have told it. So, a button for example, includes button 1, button 2, button 3, button 4 and this 5; so this is basically the view of the BorderLayout manager actually.

(Refer Slide Time: 08:16)



Class BorderLayout : Constructor

Constructor	Description
<code>BorderLayout ()</code>	Creates a border layout but with no gaps between the components.
<code>JBorderLayout (int hgap, int vgap)</code>	Creates a border layout with the given horizontal and vertical gaps between the components.
<code>FlowLayout (int align, int hgap, int vgap)</code>	Creates a check box with the specified label and sets the specified state.

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

And the class definition so for the BorderLayout manager is concerned, it has two constructors the first is the default constructor and second constructor by which we can specify hgap and vgap; that means, that gap between the two borders we can say and then align also alignment can be mentioned.

(Refer Slide Time: 08:35)

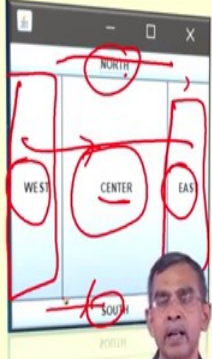
Class BorderLayout : Fields		
Modifier and Type	Method	Description
static String	CENTER	The center layout constraint (middle of container).
static String	EAST	The east layout constraint (right side of container).
static String	NORTH	The north layout constraint (top of container).
static String	SOUTH	The south layout constraint (bottom of container).
static String	WEST	The west layout constraint (left side of container).

So, these are the constructor that can be there these are the few fields are there center, east, north, south, west there are a 5 fields showing center means at the middle east means left side right west means right and north and south means top and bottom like.

(Refer Slide Time: 08:50)

Creating a BorderLayout : An example

```
import java.awt.*;
public class MyBorderLayout {
    Frame f;
    MyBorderLayout () {
        new Frame();
        Button b1 = new Button("NORTH");
        Button b2 = new Button("SOUTH");
        Button b3 = new Button("EAST");
        Button b4 = new Button("WEST");
        Button b5 = new Button("CENTER");
        f.add(b1, BorderLayout.NORTH);
        f.add(b2, BorderLayout.SOUTH);
        f.add(b3, BorderLayout.EAST);
        f.add(b4, BorderLayout.WEST);
        f.add(b5, BorderLayout.CENTER);
        f.setSize(300, 300); f.setVisible(true);
    }
    public static void main(String[] args) {
        new MyBorderLayout();
    }
}
```



Now, here is an example that you can see how the BorderLayout manager can be used to play there to develop one window and in this example as you see the frame is the frame which includes the 5 buttons those are the 5 buttons are declared here. All these 5 buttons

are labeled as north south east and west this is the button label like for example, this is the one-button actually, this is the one button, this is another button.

So, 5 buttons are there ok; so those 5 buttons are declared here and here you can see we did not decide the size of the buttons using set boundary like. So, if you do it then all this button can look like that, but here if you do not this the entire this portion is basically meant for this button this is another button which will be there and everything.

Now, let us see how all those buttons can be placed in which portion of this one. So, our next statement is basically f dot add; that means, we add this button b one into this frame using this BorderLayout dot north. So, this is basically to say that this button will go to the north position like this one, button b2 go to the south one and button b3 east button b4 west and then finally, 5 button is basically center and then finally, the size of the frame it is there and then visible these are usual statement there.

Now you can find the difference between the FlowLayout and BorderLayout. If you use the FlowLayout in this case for example; so, it will be like this and then go come here then this and then this like this one, but here actually it basically once you plan it you can place your component according to your own wish.

So, this is the idea about the BorderLayout manager and now let us have the discussion about the next manager, this next manager is called the GridLayout just this is also very important for one layout manager many ways similar to FlowLayout, but it has little bit different meaning also.

(Refer Slide Time: 11:00)

The **GridLayout** manager is used to arrange the components in rectangular grid. One component is displayed in each rectangle.

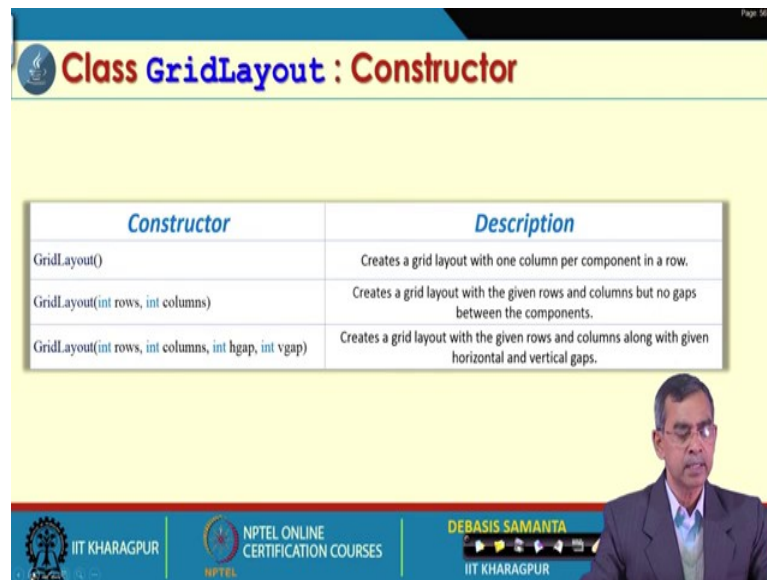
The slide features a screenshot of an 'Applet Viewer: GridLayoutDemo.class' window. The applet displays a 3x2 grid of buttons labeled B1 through B6. Below the grid, it says 'Applet started.' To the right of the applet window is a hand-drawn red grid with 3 rows and 2 columns. Below the grid, the text '3x2' and 'n x m' are written in red. The slide footer includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and DEBASIS SAMANTA.

A grid means in case of FlowLayout you can just go on adding but in case of GridLayout you just go on adding and then automatically it will adjust it and adjustment can be control maximum by hgap vgap and whatever it is there. But what is the idea about this GridLayout is that if it is like this we can make the entire container into a number of grids, then the number of grids same may be n cross m.

If n cross m it means that n number of rows and m number of columns are there. So, basically, the entire container will be pan like. So, n cross m; that means, so many things are there now again it is a difference from the BorderLayout is that in case of BorderLayout only 4, but here n cross m means n into m and. So, the number of per elements can be placed here and there now in this example as we see this is the applet which includes six buttons.

So, in first row two buttons, second-row another two and third-row two buttons in each row two buttons are there. So, it is basically we can say 3 cross two is the GridLayout size it is there. So, GridLayout is like this only now let us see how this GridLayout is defined in your component class.

(Refer Slide Time: 12:33)



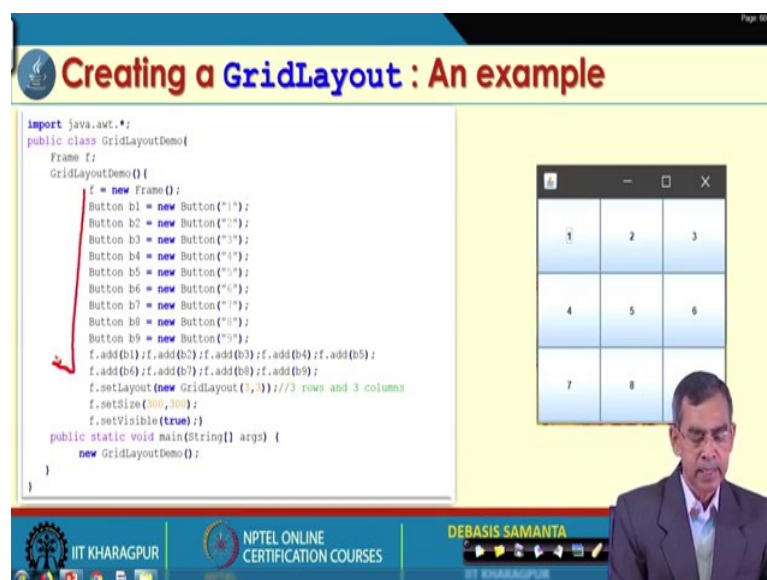
Class GridLayout : Constructor

Constructor	Description
GridLayout()	Creates a grid layout with one column per component in a row.
GridLayout(int rows, int columns)	Creates a grid layout with the given rows and columns but no gaps between the components.
GridLayout(int rows, int columns, int hgap, int vgap)	Creates a grid layout with the given rows and columns along with given horizontal and vertical gaps.

DEBASIS SAMANTA
IIT KHARAGPUR

So, there is a class called the GridLayout, this class has few constructors and the three constructors it is there in the GridLayout if it is the default constructor only one grid; that means, only one element can be placed there otherwise you can mention the row and columns in it and then hgap vgap can also be planned there. So, that you can make the I mean distance from each grid to its neighboring grids like.


(Refer Slide Time: 12:57)



Creating a GridLayout : An example

```
import java.awt.*;

public class GridLayoutDemo {
    Frame f;
    GridLayoutDemo() {
        f = new Frame();
        Button b1 = new Button("1");
        Button b2 = new Button("2");
        Button b3 = new Button("3");
        Button b4 = new Button("4");
        Button b5 = new Button("5");
        Button b6 = new Button("6");
        Button b7 = new Button("7");
        Button b8 = new Button("8");
        Button b9 = new Button("9");
        f.add(b1); f.add(b2); f.add(b3); f.add(b4); f.add(b5);
        f.add(b6); f.add(b7); f.add(b8); f.add(b9);
        f.setLayout(new GridLayout(3, 3)); // 3 rows and 3 columns
        f.setSize(300, 300);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new GridLayoutDemo();
    }
}
```



DEBASIS SAMANTA
IIT KHARAGPUR

And here is an example as you see this is basically an example to create 9 grids in the GridLayout form, and here the 9 grids contain the 9 buttons as you see here. So, we

create the 9 buttons and frame is here we add all this button into here. Now whenever you go on adding so it is basically going in that way, 1 2 3 just like a flow layout manager like and now see how we can do that. So, a `setLayout` new `GridLayout` three cross three these basically indicate that what will be the size of your grids in this case.

So, this is the most important point here to invoke the `GridLayout` manager and grid; so it is in this case. So, if you use the flow I mean `FlowLayout` then is an instead of `GridLayout`, `FlowLayout` almost is the same things it will come there, but in that case, the size and everything can be specified. Here no size needs to be specified for each component, rather we can just mention although we can for each we can mention the gap.

So, here you can `GridLayout` three cross three and then what is the gap between these and this we can mention between these and these, in that case then it will basically the smaller size also can be mentioned if the gap is specified like this one. So, there are more good looking is there, but we have a default gap means there will be no gap between two adjacents components. So, it is it will look like this.

And so these are the way the different buttons can be created they can be added into the frame and they can be added using the help of `GridLayout` manager and then the program can be executed as it, we can see it. So, this is the idea about the `GridLayout` example and then `CardLayout`, it is just like a panel, of course, the what exactly card is a card is basically one another we can say panel sort of thing, but there may be many cards can be added into the frame. So, that is the idea about the concept of `CardLayout`.

(Refer Slide Time: 15:07)

The **CardLayout** manager manages the components in such a manner that only one component is visible at a time. It treats each component as a card that is why it is known as **CardLayout**.

Applet Viewer: CardLayoutDemo.class
Applet
Card3
Applet started.

DEBASIS SAMANTA
IIT KHARAGPUR

So, the idea is to look like this. So, if this is the applet as it is so, the first Card3 is showing. Now, if we click it then it will go to the next card, Card4 Card5 like on.

(Refer Slide Time: 15:20)

Class CardLayout : Constructor

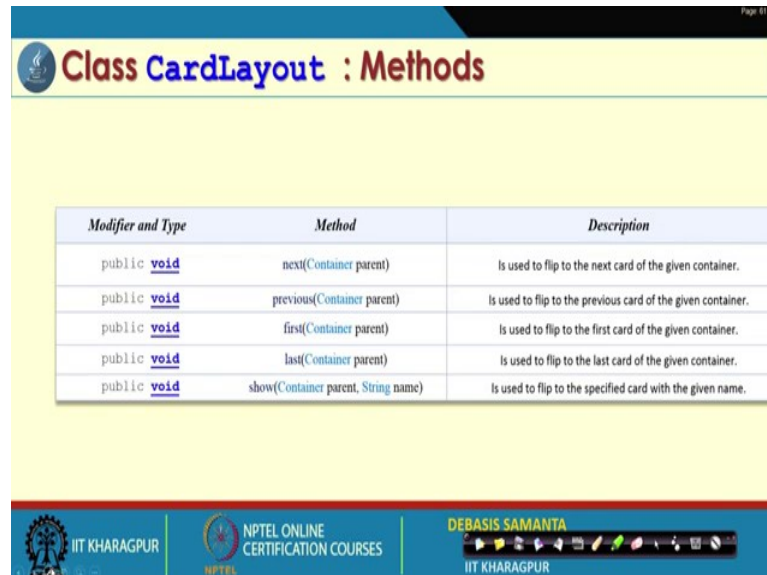
Constructor	Description
CardLayout ()	Creates a card layout with zero horizontal and vertical gap.
CardLayout (int hgap, int vgap)	Creates a card layout with the given horizontal and vertical gap.

DEBASIS SAMANTA
IIT KHARAGPUR

So, every card is basically one element in it there. Now, this layout is defined in the form of a class card layout which is there in the component subclass of the class component. And this has it has the two constructors as you see the default constructor the example that we have seen is the result of the default constructor otherwise we can mention the

hgap and vgap between the cards also so that the card will be reduced in its size like. So, the size of the card can be maintained using hgap and then vgap.

(Refer Slide Time: 15:57)



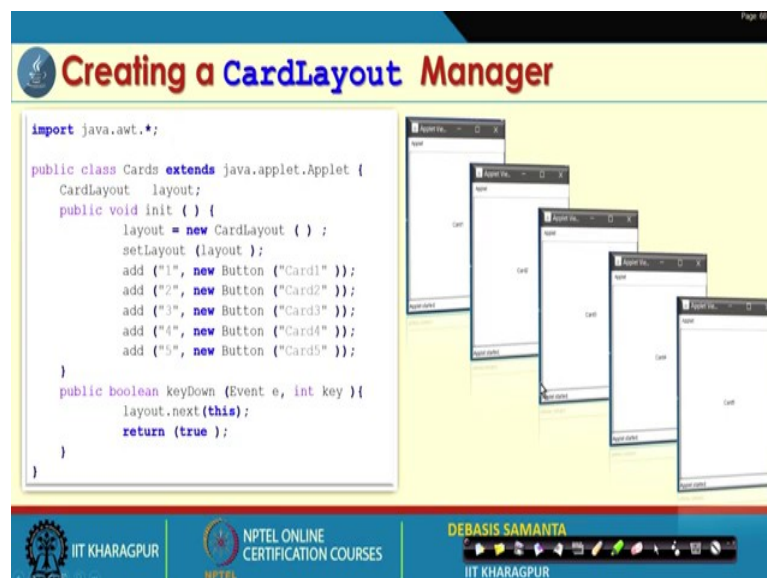
Class CardLayout : Methods

Modifier and Type	Method	Description
public void	next(Container parent)	Is used to flip to the next card of the given container.
public void	previous(Container parent)	Is used to flip to the previous card of the given container.
public void	first(Container parent)	Is used to flip to the first card of the given container.
public void	last(Container parent)	Is used to flip to the last card of the given container.
public void	show(Container parent, String name)	Is used to flip to the specified card with the given name.

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

Now so this is the few methods are also there next method past method previous method this one; that means, if we click it in which card it will go if we click by default is an next otherwise a previous if we can control previous and what are the methods are there.

(Refer Slide Time: 16:11)



Creating a CardLayout Manager

```
import java.awt.*;

public class Cards extends java.applet.Applet {
    CardLayout layout;

    public void init () {
        layout = new CardLayout ();
        setLayout (layout );
        add ("1", new Button ("Card1" ));
        add ("2", new Button ("Card2" ));
        add ("3", new Button ("Card3" ));
        add ("4", new Button ("Card4" ));
        add ("5", new Button ("Card5" ));
    }

    public boolean keyDown (Event e, int key ){
        layout.next(this);
        return (true );
    }
}
```

The slide also includes a visual representation of a CardLayout manager showing five overlapping windows, each containing a button labeled 'Card1' through 'Card5'.

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

Now, this is an example as you can see we have added cards into an applet like. So, it is an applet program and here the layout that we have the use the CardLayout. So, this is

basically to say that we want to follow the CardLayout as our manager for this container and the setLayout layout.

So, this is basically the method by which the CardLayout or the otherwise setLayout new CardLayout also can be used directly also here. Now here we add what we have added here is basically 4 5 buttons into the in the form of a card. So, these are the basic 5 different applets view actually here, in the first applet view we just go on clicking and then automatically this will be there I have used the three 5 different examples of course.

So, first, say suppose the card five. So, so the idea it is here is that Card5 is basically the button. So, the entire button is basically one card should look like this. So, here the entire button looks like a card, but instead of button some other things also can be included whatever it is there it can include some other applet also in it and. So, here actually we created 5 cards in the form of a button look like and they are added into this applet and then we declare on key down that is totally again different concept of event handling.

So, KeyDown is one method which is defined in event class, we and then that event class is apart of the component KeyDown is basically an abstract method we have to overwrite it. So, this KeyDown key KeyDown method is a basic event handling if we press an I mean mouse button so that is the KeyDown or not mouse button we can say KeyDown is related to the pressing a key.

So, if we press a key then event e will be generated; that means, pressing a key is your event and then it basically return which key is that the user has pressed it and then layout.next is basically here exactly saying that this layout has the next I mean if you press it then it will go to the next card. If you use instead of next previous if will go to the previous card like this one.

So, either previous or next whatever it is there and return true means it just simply a matter of return. So, key this is the KeyDown event, this event is automatically defined there and you just simply press it and then automatically this event will play it and then it will generate whatever the next action it is there. So, those methods are there and. So, this program actually it is if you run this so initially, it will Card1 then if you press a key then Card2 then Card3. Again if you press then Card4 and Card5, again after Card5 if you place it again then Card1 and so on so on.

So, it basically pops up look like; that means, the different cards will be pop up and then you can then display it. So, this card can contain few images or some other elements or some other things likewise. So, different cards can be planned to give the different facilities to your user look like. So, this is the way the CardLayout manager can be designed and then decided. And here is an ok; so interactive applet I just want to give an example about how the interactive applet can be created.

(Refer Slide Time: 19:40)

The slide is titled "Interactive applet" and features a Java code snippet on the left and a screenshot of a running applet window on the right. The code defines a class `InteractiveApplet` that extends `Applet`. It includes two text fields, `inputA` and `inputB`, and a `paint` method that calculates the sum of the values entered in these fields. The applet window on the right, titled "Applet View...", shows the graphical interface with two text input fields labeled "Enter two values:" and a label "The sum is: 1". A small status bar at the bottom of the window says "Applet started". The slide also includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and the name DEBASIS SAMANTA.

```
import java.applet.*;
import java.awt.*;
public class InteractiveApplet extends Applet {
    TextField inputA, inputB;
    public void init () {
        inputA = new TextField(10);
        inputB = new TextField(10);
        add(inputA); add(inputB);
        inputA.setText("0"); inputB.setText("1");
    }
    public void paint(Graphics g) {
        int x = 0; int y = 0; int z = 0;
        String s;
        g.drawString("Enter two values", 10, 50);
        try {
            s = inputA.getText(); x = Integer.parseInt(s);
            s = inputB.getText(); y = Integer.parseInt(s);
            z = x+y; s = String.valueOf(z); g.drawString("The sum is :", 10, 75);
            g.drawString(s, 100, 75);
        } catch (Exception e) {}
    }
}
```

Interactive applet let us see I can explain it like this. So, for example, this is the one applet, in this applet, this is the one text field this is the other text field. So, two text fields are there and this text fields initially by default it has the value 0 and it has the value 1. And we just display one thing like this another thing enter two values you can give a prompt.

So, if the user enters two values mean they can give the value from the keyboard typing the keys on the keyboard and then the value will be there and another value will be there. So, this is gone typing and then once it is typed enter please enter then. So, after typing enter then one the two values are filled there the automatically sum will be displayed this one. So, for example, if I (Refer Time: 20:37) is a 5 and if it is a 6 then is the eleven will be displayed like. So, this is the one simple example of an applet that we are going to have it.

Now, let us see what is the program for this a very simple program here. And so is an applet program so extends applet this is the name of the applet program. Now there are two text fields text field inputA and inputB as we have said. So, this is basically inputted a and input b like two text fields and then this is the init method this basically creates two text fields. So, input a is created a is a basically declared and they created instantiated and 8 is basically the size; that means, maximum 8 digit can be put into these areas like.

And then add input an additional input means. So, this input they text field input a and input b are the two text field added into this one and now this default is basically grid default is flowlayout. So, it is basically displayed like this one and then graphics we can just use the paint method to control the value and then take the event from the there, actually here the event also because typing some numbers is basically an event also corresponding to this one.

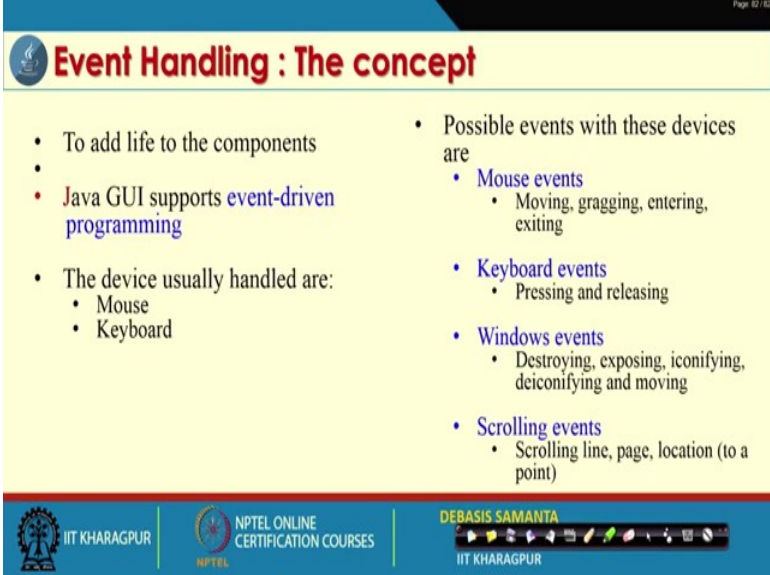
Now, here you see what the paint method we will do for you. So, we declare x and y are the two values, two-three different integer elements variables are declared x y z like and s is a string. Now here one important thing is that whatever the things it is there, java treat them as a string. So, it is also one string, if you type something it is a string actually and then that string needs to be converted integer. So, it is basically to input a get text basically the string and then string this is converted in integer by using integer.parseInt so x. So, x is basically whatever the user input here is stored as an integer now.

Similarly, other is y from this text filed area z is basically the sum and then we have to display it. So, we convert this z into the string using this formula, expression and then finally, g.drawString the sum is 10. So, this we will print and then the value will be printed here. So, this is a simple example which basically shows using the whatever the applet we know and using the GUI components the here we have only used two text in. So, this is interactive because we use it and then entre automatically it will take and the result will be displayed there that is all this is very simple once you see how many less course are there to do it everything it is there.

So, that is why this programming is called the lightweight programming, because you just may just put it and then done the job readily and then there is a lot of supports from the p is the AWT package they will give you and then implement everything. So, only this is very simple actually compared to other programming concepts it is there. So, this

is the one example about that interactive applets and event handling is a very important one concept that needs to be discussed thoroughly. In our next module, we will discuss about it, but I want to give a glimpse of the event handling concept it is there.

(Refer Slide Time: 24:00)



Event Handling : The concept

- To add life to the components
- Java GUI supports **event-driven programming**
- The device usually handled are:
 - Mouse
 - Keyboard
- Possible events with these devices are
 - **Mouse events**
 - Moving, dragging, entering, exiting
 - **Keyboard events**
 - Pressing and releasing
 - **Windows events**
 - Destroying, exposing, iconifying, deiconifying and moving
 - **Scrolling events**
 - Scrolling line, page, location (to a point)

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

These few slides are basically explained what exactly the event handling is because we have decided that whenever you click a mouse it becomes an event whenever you press a key it become an event if you drag a mouse it is also an event. So, it is basically called the event.

Now, if this event is occurred, if you click a mouse key mouse then and suppose if you click a button then the button click basically should generate certain facilities for the user. Say suppose if you click a button and then button is basically whenever click either it stops an applet or it basically play a video then you can automatically video we will come into the play and it will display it there. So, it is basically the event handling concept it is there. So, mainly two devices are used handling the key is basically mouse and keyboard those are the standard mouse and keyboard are basically means for generating an event. And then once the event is generated they can be handled by the corresponding program for each event that is there.

So, here is a programmer, we have to decide that if an event occurs then what will be the corresponding action that event we will produce to the user like. So, an event can be a source of the event is basically different component and at the background you have to

write the program that if that event is triggered by a mouse or keyboard then what will be the consequence it is actually there.

Now so far the different ways that the mouse event can be here we have listed few things. So, so for the mouse event is (Refer Time: 25:35) the mouse can be moved it can be dragging it can be entering it can be exiting keyboard event are pressing and releasing windows events is there its basically destroying when means is a cross button is selected is basically iconifying smalling. Or assuming and then it is basically moving also a window can be dragged by clicking this mouse also and then scrolling event is basically scroll bar page location and everything.

These are the few events are possible in our event handling mechanism or event add ability programming and regarding. So, many events we will discuss separately in details in our next module.

(Refer Slide Time: 26:12)

Event handling : An example

Applet Viewer: EventDemo.class

Applet

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	6

Applet started.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES DEBASIS SAMANTA

Now, here is a n example of how the event can be here, I can say it is a very simple example for illustration purpose only. Now you can see what we have done here it is an applet and in this applet what is the I mean container is there actually we see 1 2 3 4 these are the 15 buttons. So, 15 buttons are labeled as 1 2 3 4 like this one and all these layouts you can follow which layout manager I should follow it.

I have used it here as a GridLayout actually because it is in the form of a grid if it is a GridLayout then what is the size of the grid 4 cross 4 and here you see. So, out of 16 that the grid components the first is 1 2 3 4 to 15, these are basically 15 components are there and then the last is basically not the button it is basically called the text field.

So basically, we want to create a window that includes 15 buttons labeled as one two three 15 and one is this one. Now here we want to add some event handling. What is the event? If I mouse click the mouse here then whatever the mouse that is clicked it will automatically display here. So, if we click here this one, if we do not initially it will display with a star mouse-like nothing is clicked like.

So, here basically an initial view of the applet will be it is star and now if the user clicks here then automatically it will display 6 or if we user clicks here then it will display 8 like this one. So, this is just a simple other interactive windows like the user can interact with the interface and then the effect will be should be obtained to the user accordingly.

Now so, this is the event handling; I just want to have the code actually here right now and then the details about how the event handling needs more elaborate discussion. Now here is the code for this kind of programming, I hope you have understood exactly what is the thing we are going to do it.

(Refer Slide Time: 28:22)

The slide is titled "Event handling : An example" and features a Java code snippet for an applet. The code defines a class `EventDemo` that extends `Applet`. It sets a 4x4 grid layout and creates 15 buttons labeled from 1 to 15. A label is also added to the interface. A video inset in the bottom right corner shows a man, Debasis Samanta, speaking. The slide footer includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and the name DEBASIS SAMANTA.

```
public class EventDemo extends Applet {
    static final int n = 4;
    Label label;
    public void init() {
        setLayout (new GridLayout (n,n));
        setFont (new Font ("Helvetica", Font.BOLD, 24 ));
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                int k = i * n + j + 1;
                if (k < 16)
                    add (new Button (" " + k));
            }
        }
        label = new Label (" * ", Label.CENTER );
        label.setFont (new Font (" Times Roman", Font.ITALIC, 24 ));
        add (label );
    }
}
```

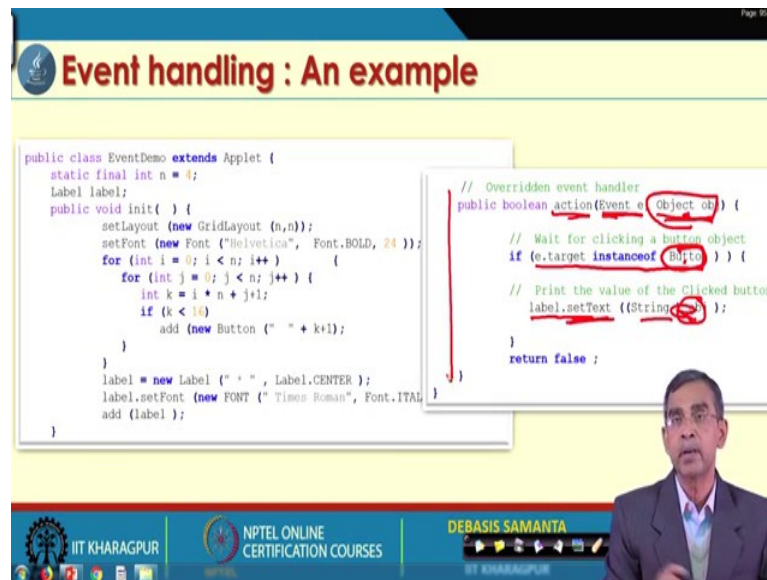
Now, the code is very pretty simple here as you see the code is very simple and first is that n is 4; that means, we have to create four cross four layouts it is like this. So, that is done by `setLayout GridLayout n cross n` this is; obviously, clear that we want to use the `GridLayout` as a manager for my layouts and then here actually `setFont`, we create the font as the Helvetica bold font 24; that means, all the fonts that we have already set to display in this case is actually here.

So, these are the fonts that we have discussed here. So, this is the these the font that we have discussed is Helvetica font actually anyways. So, this is like that we have discussed here `setFont` and then we have to arrange the different buttons there. Now here is the different button that can be arranged here, I have just make a loop that this one and automatically this is a tricky loop you can just follow yourself and you can understand that how the different button can be created with the label 1 to 15 and all those button can be placed in this 1 2 3 4 5 then 1 2 3 then 6 like this one this way only the button will be placed there.

And then the next the last place in the `GridLayout` will be placed by a label, and that label has this label center and default appearance is basically called the star. So, it is basically a label component is created there. So, it is a label and then these are labels also have the font `setFont times new roman font italic` that means that label that will appear in a different font it looks like this add label; so the label is added there.

Then next is the event to be handled because it is just simply the windows are created or we can say applet has been created, but not an event has been added here. So, our next part is there how the event can be added to it.

(Refer Slide Time: 30:27)



Event handling : An example

```
public class EventDemo extends Applet {
    static final int n = 4;
    Label label;
    public void init() {
        setLayout (new GridLayout (n,n));
        setFont (new Font ("Helvetica", Font.BOLD, 24 ));
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                int k = i * n + j + 1;
                if (k < 16)
                    add (new Button (" " + k));
            }
        }
        label = new Label (" ", Label.CENTER );
        label.setFont (new Font (" Times Roman", Font.ITALIC, 18 ));
        add (label );
    }
}

// Overridden event handler
public boolean action(Event e, Object ob) {
    // Wait for clicking a button object
    if (e.target instanceof JButton) {
        // Print the value of the Clicked button
        label.setText ((String) ob);
    }
    return false ;
}
```

DEBASIS SAMANTA
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, here is the event it is there. So, the event that we have planned here is the action, the action is basically again is an abstract method which is defined in the event class and which is there in the java.awt package is there we have overwritten it. Now Event e; that means, whatever the key placed is basically event e or mouse click it is Event e and then it is result written; that means, if we click a button say 5 then object obg is basically that this is the button 5 has been clicked.

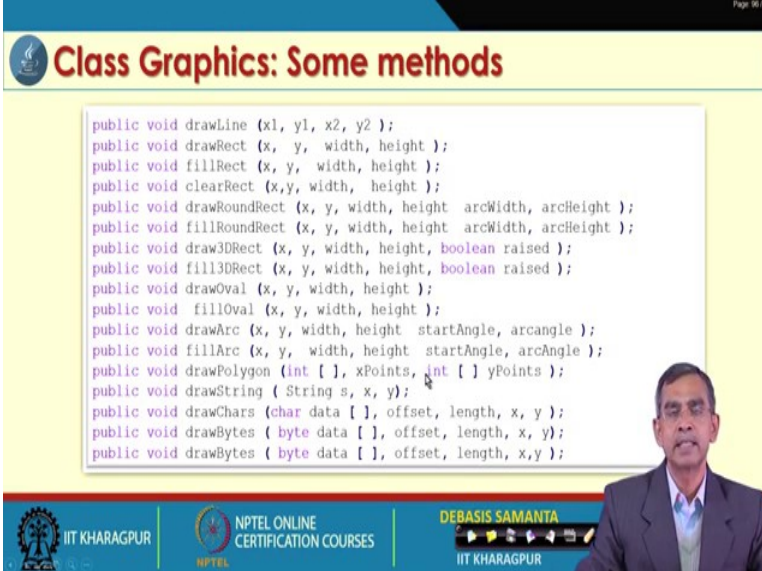
So, this basically the method that it will basically return it and if e.target instance of button; that means, if we create a button then label.setText string object here basically object means it is which button it is an object is already defined there here and then object is basically a string and then that string will be displayed there and this is very pretty simple the action method which has been overwritten here and that is all. And once it is done your actually that that program is ready for your working.

So, that program output as we have already shown it will give it. You can test of your own so that you can get more fun about it and accordingly instead of number you can say a b c also you can check it, the one also you can put in the button itself get these things and check it; so a lot of other manage what is called the experiment you can do with this concept.

So, this is about the layout manager is very important one concept and then there is a graphics also there. So, as I told you in case of component class there are many other

things or other AWT package that graphics is one essential component also to have the graphics. If you want to develop an application software look like say paintbrush then definitely graphics is to be considered there.

(Refer Slide Time: 32:21)



Class Graphics: Some methods

```
public void drawLine (x1, y1, x2, y2 );  
public void drawRect (x, y, width, height );  
public void fillRect (x, y, width, height );  
public void clearRect (x,y, width, height );  
public void drawRoundRect (x, y, width, height, arcWidth, arcHeight );  
public void fillRoundRect (x, y, width, height, arcWidth, arcHeight );  
public void draw3DRect (x, y, width, height, boolean raised );  
public void fill3DRect (x, y, width, height, boolean raised );  
public void drawOval (x, y, width, height );  
public void fillOval (x, y, width, height );  
public void drawArc (x, y, width, height, startAngle, arcAngle );  
public void fillArc (x, y, width, height, startAngle, arcAngle );  
public void drawPolygon (int [ ], xPoints, int [ ] yPoints );  
public void drawString (String s, x, y);  
public void drawChars (char data [ ], offset, length, x, y );  
public void drawBytes (byte data [ ], offset, length, x, y);  
public void drawBytes (byte data [ ], offset, length, x,y );
```

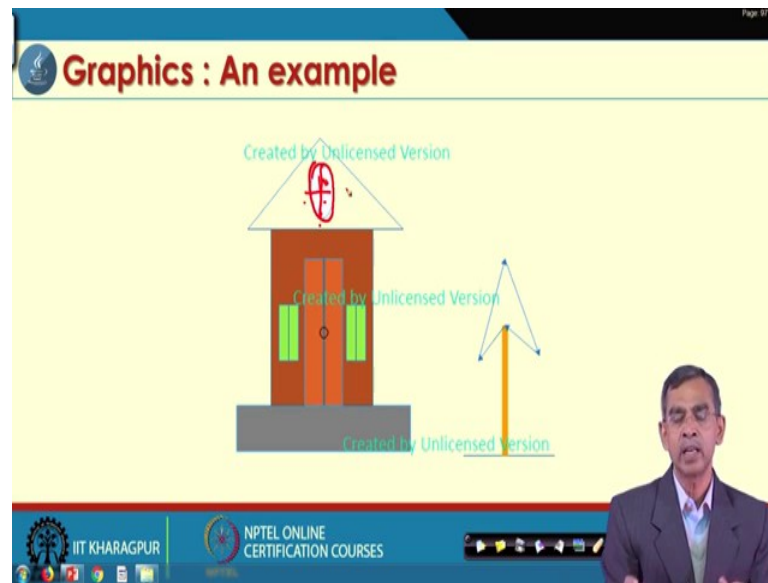
DEBASIS SAMANTA
IIT KHARAGPUR

And graphics have many methods are declared there, graphics is a very important one class there and it has many methods all these methods are basically how to draw different graphics objects.

For example how you can draw a rectangle, how you can draw an ellipse how you can draw polygon and how you can make the background of different, how the different styles can be whether dotted or different style can be fixed and there are many other things can be. Also, there are many other methods to position it zooming so many things are there I have listed few methods that are declared therein graphics class, it is basically a huge number of methods all methods are basically to handle the graphics related issues are there.

Now, if you have different methods of understanding and everything and then if I ask you to have a solution from your sides.

(Refer Slide Time: 33:12)



For example here; suppose we want to create this kind of thing are there it will display this kind of text as well as it will draw this kind of figure. So, in this figure, as you see this figure as one rectangle to be drawn another rectangle to be drawn, within this rectangle other small rectangle windows. And all these things are there the ovals ellipse will be there the rectangle will be there here is the one polygon will be drawn, another line will be drawn, the line has the different thickness, the different colors and another line will be also there.

So, basically, if you have the graphics methods and then different methods that are there using this method I can draw all these things, and only the thing is that coordinate needs to be specified. For example, say suppose this ellipse I want to draw here. So, ellipse is basically this is the coordinate of center and we have to mention these are the major axis and minor axis that is all.

So, center, major axis, minor axis and then background color and then these are the different line color and the style also they are many constructors for ellipse drawing; you just use it and then press it. Likewise for this rectangle also this is the position width and height if you position and then background line color and everything it will draw automatically. So, this way the different component can be placed and drawn and a picture look like this also we will appear to you.

I will be left it as an exercise for you; try to have of it is your own and you can enjoy it a lot, I believe. I am sure that with this you will be able to enjoy a lot. And really the AWT program is a lot of interesting facts are there we have just started about basic things and then in our next module we will discuss about event handling so for the AWT is concerned which is more important to understand also and more we have to develop our own skill for that.

And then all this AWT also can be down in a more advanced way using the swing also. So, all those things in our pipeline we will discuss first event handling, then we will be discussed about swing concept in our next lecture slides.

Thank you very much, thanks for your attention.