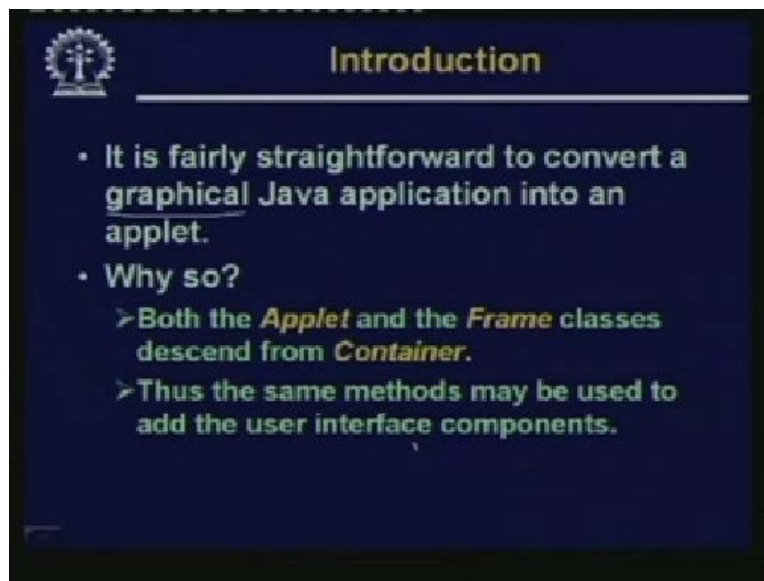


Internet Technology
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No #29
Java Applets – Part: 2

In this lecture we shall be continuing with our discussion on Java applets. If you recall in our earlier class we had mainly talked about the differences between Java applications and Java applets and we had also looked at some of the examples how a Java applet can be written, how an applet can be included or embedded from within an HTML page. So today the first thing that we look at is how to convert any arbitrary Java application to Java applet? Now this issue is important because we had mentioned earlier that we have Java applications which can be written to perform any arbitrary task, any sort of complex or sophisticated animation which can be done using that program. But in order to be used along with a web page in the internet scenario we need to convert that program into an applet which can be linked and also downloaded along with an HTML document.

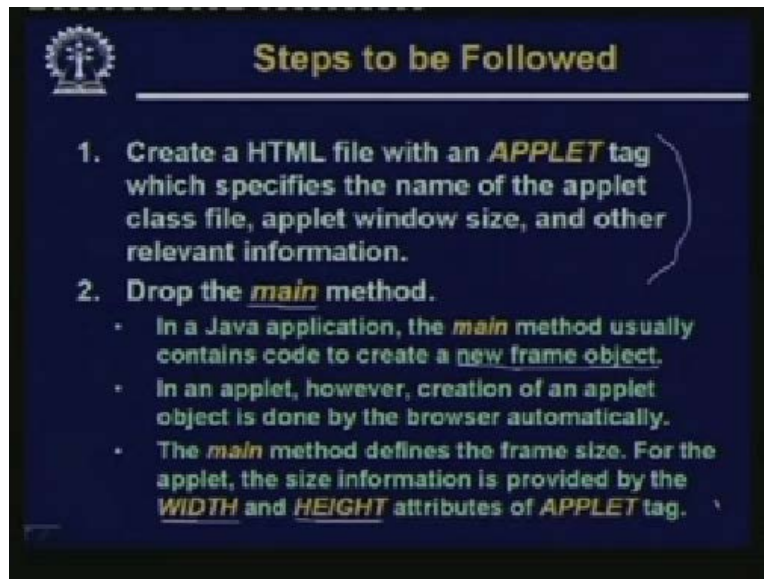
(Refer Slide Time: 02:01)



So let us see or look at this issue first. First thing to note is that it is not very difficult. Rather it is straightforward to convert an application to an applet. We call it graphical because most often than not we have graphical applications running on web pages. So the application that we want to convert is most often a graphical application; an application with graphics. Now the reason it is straightforward is that you know in Java, the classes that we use from which the objects are derived they are all instantiated or they are all inherited from other super classes or parent classes. Now in application or applet the classes that are used by default. They all decent from the same parent class. That is why the way they work, the way they function the methods they use they are somewhat quite similar. So you need have to make much change or much modification to an application to make it work as an applet. Specifically an Applet uses the applet class; an application uses the frame class. Both of which decent from the parent class called Container. So essentially what I have told that

because of this we can use the same methods and we use the same user interface component description in both the application and applet.

(Refer Slide Time: 03:47)



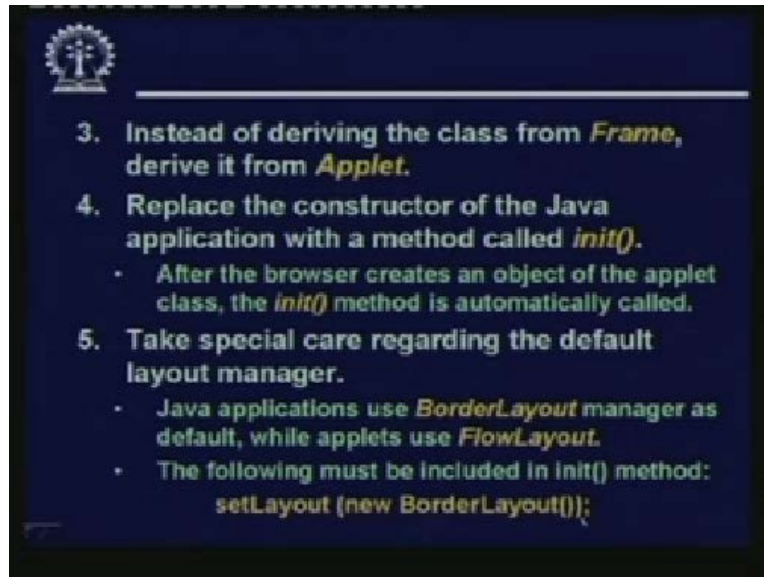
Let us now look specifically at the different steps we need to follow to carry out or affect this conversion. First thing of course this I have already mentioned that in order to convert an application to an applet you need to create an HTML file. Because an applet has to reside along with an HTML document; from the HTML document you need to provide a link to the applet which will be downloaded along with the HTML file. So the first thing you need to do is to have an HTML file with an APPLET tag which will specify the name of the applet byte code or the class file, the size of the applet window and other optional attributes which we had mentioned in our last class. Secondly one characteristic of a Java application is the existence of the main method.

Whenever you execute a Java application, by default it is the main method which executes first; from the main methods you can call other methods. So in an application the main method normally contains will contain some code to create a new frame object in which the different graphical item or user interface components would be included. So it is the main method which initializes the context under which the application has to run. But in an applet the situation is slightly different. We do not exclusively call an applet just like a Java application. So that there is no question of explicitly calling the main method. Rather the creation of the applet and the initiation or the beginning of execution is done automatically by the browser after the applet has been loaded successfully into the browser.

So, whereas for an application you need to explicitly type in a command to run it. For an applet it is automatic. It gets downloaded along with the HTML file and as soon as download completes the applet gets initiated or executed immediately by the browser. This does need any user intervention. The main method when it creates a new frame object it creates to, it defines the frame size for the applet. However here also you need to specify the size of the window in which the program has to run. But here this information is provided by the

WIDTH and the HEIGHT attributes which are present in the APPLET tag of the HTML document. This need be specified within the applet the size.

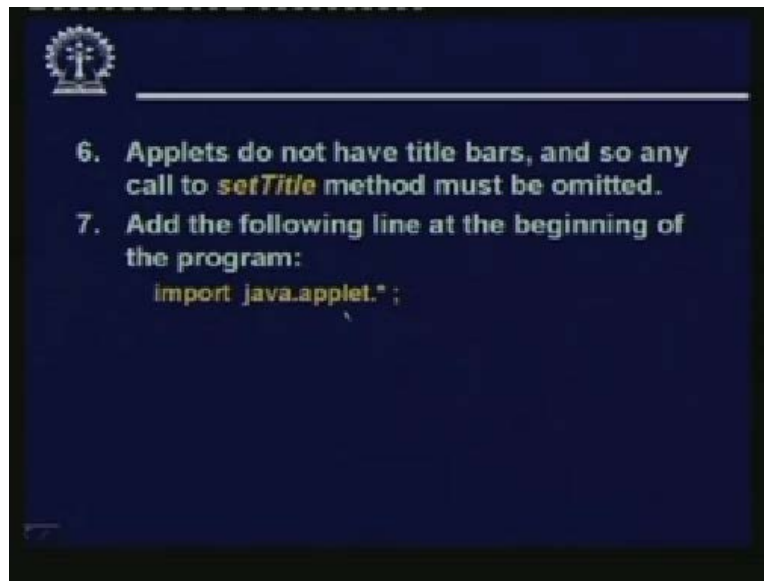
(Refer Slide Time: 06:59)



Third. An applet is derived from the class Applet. An application is derived from the frame. So when you are converting an application to an applet instead of deriving it from Frame you need to derive it from the Applet class. Fourthly you need to replace the constructor of the Java application with a method called init. Just to recall the constructor of a Java class represents a method having the same name as the class which is executed by default whenever an instance of that class is created. Whenever you create an object, the constructor class or the constructor method is automatically invoked and it is the responsibility of the constructor to make or create all the initializations before the other methods can be invoked. So in a Java application you have the constructor in applet you have an equivalent method called init. So the constructor function has to be replaced by a function called init. So whatever initializations you need to do which we earlier within the constructor, now it will go inside init.

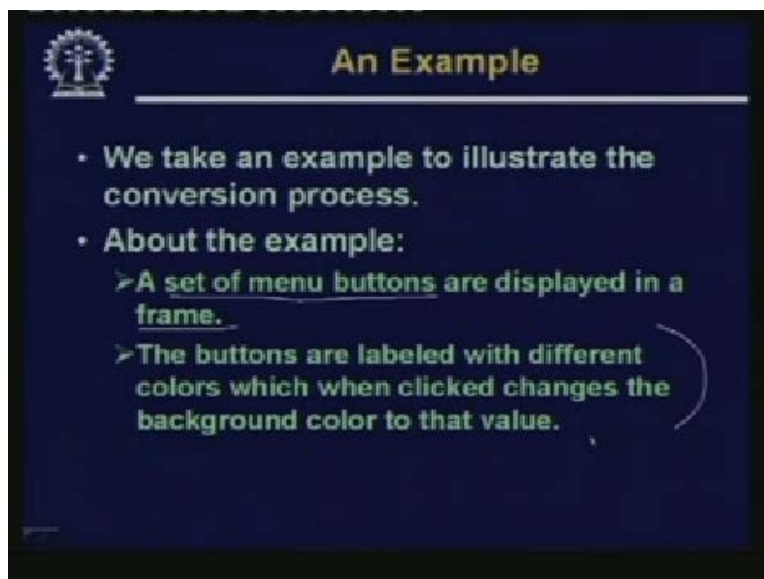
So an applet when it executes whenever the browser will create the object after creation the method will be automatically called. Next there is an issue regarding the layout manager. While any Java program you write, whatever graphics you create, that is created based on some library based on a layout manager. Java application by default BorderLayout manager while applet use the FlowLayout manager. So if you had an application which was written assuming the BorderLayout manager, when you convert it into an applet which by default uses the FlowLayout manager, you will have to explicitly specify that well. You do not use the default layout manager. But rather use the BorderLayout manager. This can be done by explicitly including the following statement in init method setLayout new BorderLayout. This will be overriding the default layout manager in the applet.

(Refer Slide Time: 09:45)



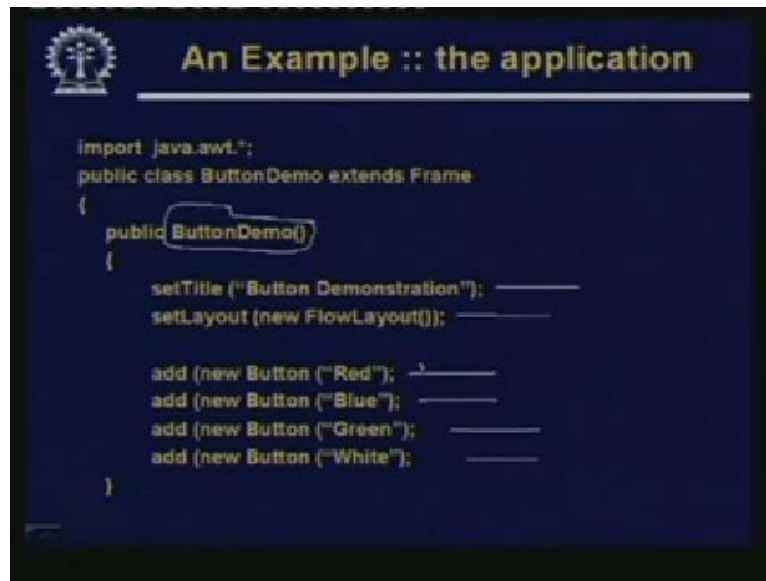
Applets have no concept of title bars. So if there are any setTitle calls in application, just remove them. And lastly you import the library java.applet.star at the beginning of the program to include methods that are applicable to applets.

(Refer Slide Time: 10:10)



Let us now look at an example which will illustrate the process of the conversion. Now the example that we take is as follows. The Java program will be creating a set of menu buttons which will be displayed inside a frame. The buttons will be labelled with different colors like Red, Blue, Green and White. And the purpose of the button is that if you click a particular button with a mouse the background color will get changed to that particular color. So this is what the program is given to you. We will start with an application then we shall see how the application can be converted into an applet. Let us look at the application first.

(Refer Slide Time: 11:04)



```
import java.awt.*;
public class ButtonDemo extends Frame
{
    public ButtonDemo()
    {
        setTitle (" Button Demonstration");
        setLayout (new FlowLayout());

        add (new Button ("Red"));
        add (new Button ("Blue"));
        add (new Button ("Green"));
        add (new Button ("White"));
    }
}
```

So an application name of the Java program or the Java class file here, we have assumed to ButtonDemo. This extends Frame. So by default it is a Frame class from descends. This is an example of the constructor object we had mentioned. So it is a method having the same name as the class ButtonDemo. So whenever there a method with the name same as the name of the class file, that is constructed or this is considered as the constructor and whenever you are running the application, this method will be called first. So here you see what is there in the button demonstration? First line there is a title Button Demonstration. This will appear at the top. You create or specify layout manager FlowLayout manager. You add four buttons on your screen, on your frame with labels Red, Blue, Green and White. So this new button, this will be creating a new instance of a button with the label Red and add will be creating that object button on the current frame being displayed. So this is the constructor method.

(Refer Slide Time: 12:41)



```
public boolean handleEvent (Event v)
{
    if (v.id == Event.WINDOW_DESTROY)
        System.exit (0);
    return super.handleEvent (v);
}
public boolean action (Event v, Object but)
{
    if (but.equals ("Red"))           setBackground (Color.red);
    else if (but.equals ("Blue"))     setBackground (Color.blue);
    else if (but.equals ("Green"))    setBackground (Color.green);
    else if (but.equals ("White"))    setBackground (Color.white);
    else return false;
    repaint ();
    return true; }
}
```

Then comes the event handler. So this is a public Boolean `handleEvent`. This is an event handler method. So if you are closing the window, this will generate the `WINDOW DESTROY` method. So this event handler simply says if the event die, if the id of the event `v dot id` is this, `Event.WINDOWDESTROYER` which is the type, then you exit with an exit code of zero. Otherwise if it is some other event you return to `super.handleEvent`, that means you are recursively calling the parent object from where this object was derived? So that you are saying that I will not handle the event myself. I will leave it to my parent to handle it. And this method action by default is called whenever some kind of an event or an action takes place. Like for example I am pressing the mouse button. So I want whenever I click on one of the buttons of the mouse, I need to change the color.

So it is this action method which will be invoked whenever I press the mouse button. Here if you look at the function it, takes as argument the Event and which Object was clicked or on which Object this Event took place. But is the type of object, you check if the idea or name of the object. This you can check using the equals function if. But `dot equals Red` is true which means you have clicked the Red button, then you call the method `setBackground` with `color.red`, you set the color attribute to red. Else if you click Blue then you set it to blue and so on. So if it is none of the valid buttons some other then you return false. `Repaint` is a function which will allow you to refresh the window so that whatever color you have selected the window will get refreshed with the new color so that the changed color in the background will be reflected immediately.

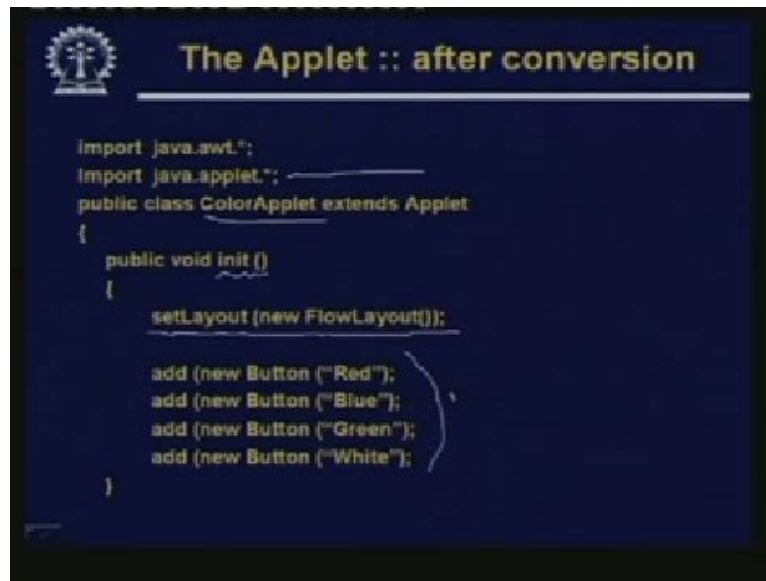
(Refer Slide Time: 15:15)



```
public static void main (String args [ ])
{
    Frame f = new ButtonDemo ();
    f.resize (300, 300);
    f.show ();
}
```

So this is how the application will look like. And finally you have the main method which simply creates frame `ButtonDemo` is the name of the class. It specifies the size of the frame 300 by 300 and `show` is a method which is called to actually display. The Java application on the created frame. So now the Java application is starting when you are invoking the method, `show` actually the Java program starts running. This is how the application is now we want to convert this application into an applet. Let us see.

(Refer Slide Time: 15:52)

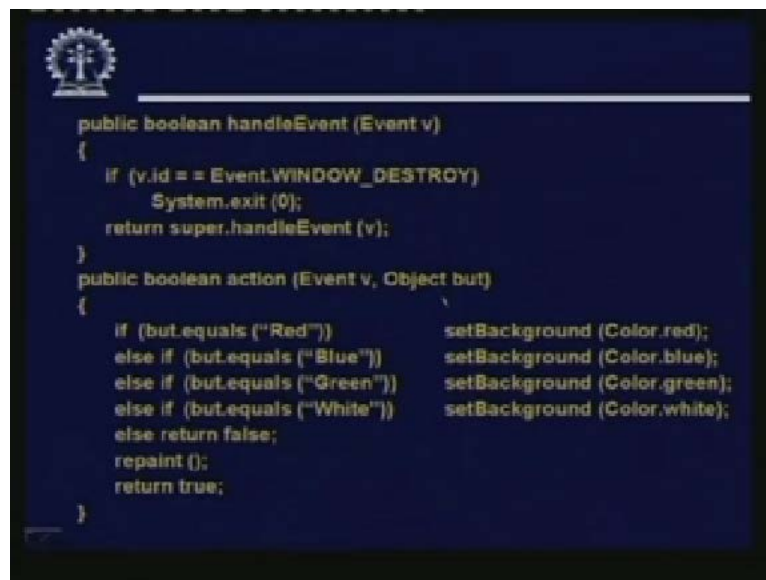


```
import java.awt.*;
import java.applet.*;
public class ColorApplet extends Applet
{
    public void init ()
    {
        setLayout (new FlowLayout());

        add (new Button ("Red"));
        add (new Button ("Blue"));
        add (new Button ("Green"));
        add (new Button ("White"));
    }
}
```

So after conversion, the applet version looks like this. The first thing I told, we have added this line `import java.applet.*` at the beginning. The name of applet you have given as `ColorApplet`. Now instead of the constructor method we have given or we have created a method called `init` which consists mainly the same set of statements which are there in the constructor of the application, four buttons are created. Only thing was that you explicitly set the layout manager here. This is what is done here.

(Refer Slide Time: 16:38)



```
public boolean handleEvent (Event v)
{
    if (v.id == Event.WINDOW_DESTROY)
        System.exit (0);
    return super.handleEvent (v);
}
public boolean actionPerformed (Event v, Object but)
{
    if (but.equals ("Red"))        setBackground (Color.red);
    else if (but.equals ("Blue"))  setBackground (Color.blue);
    else if (but.equals ("Green")) setBackground (Color.green);
    else if (but.equals ("White")) setBackground (Color.white);
    else return false;
    repaint ();
    return true;
}
```

The event handler and action they are same. There is no change out here. These methods are already existing in the object of the class from where a frame or an applet is derived. So you need have to change the event handler or action methods at all. They will remain the same. The only change is that you do not add any title to it.

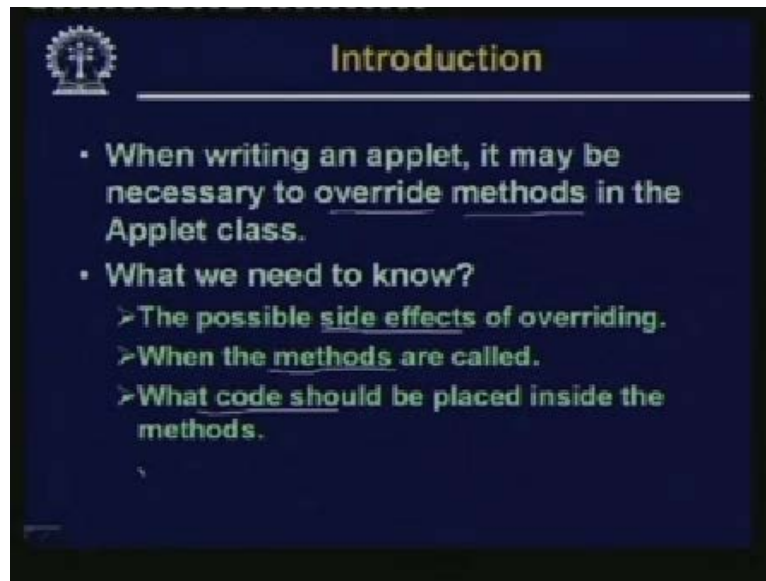
(Refer Slide Time: 17:08)



So now let us look at something called Applet Life Cycle. We have seen how an applet can be written and applet can be created. Starting from a Java application which perhaps is something which you have written to start with from an application. How to convert it to an applet? We have specified the different steps to be followed also illustrated the process with the help of a simple example. But even if it is some more complex example, you should be following more or less the same sequence of steps. Now let us look at some very specify methods that pertain to an applet. Only these are applet specify methods.

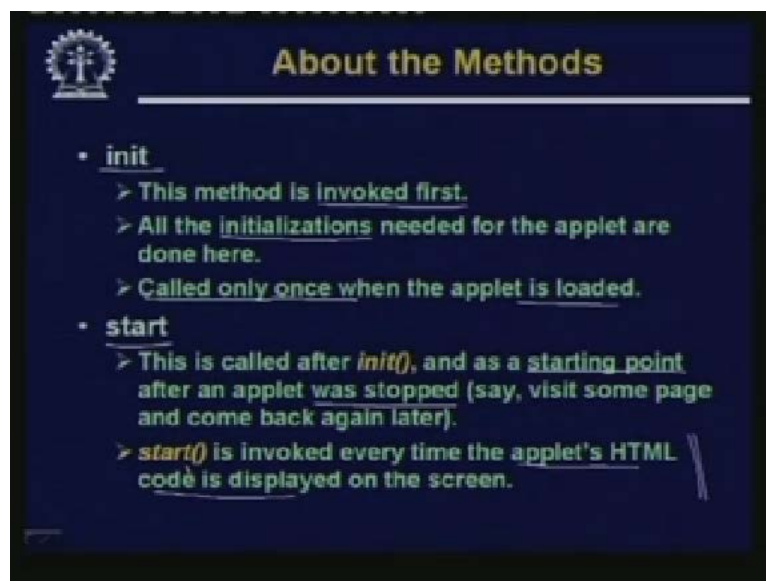
Now applet is supposed to be displayed as part of a browser. So there are certain events which are specifically, you can say relevant with respect to the browser. Like I can close a browser, I can open a browser; I can put one window on top of another window. So when I am closing the browser. Do I need to still continue running the applet or I should suspend the applet? Let it come up into my view again and it will again be started. So these are some issues and these leads to some different states of an applet and so called applet life cycle. So let us look at this so called applet life cycle now.

(Refer Slide Time: 18:42)



So when you are writing an applet as I had said there are methods that belong to the applet class which are already there by default. We sometimes write our own methods with the same names. That means we are overriding the existing methods. Now when you override the method, for example, there may be a method already existing called paint or repaint. I am writing the paint method again explicitly. So when I write I should be very clear about the side effects. If I change or modify an existing method which was there as part the derived class, what are the side effects I can expect to encounter. So the possible side effects I should know. I should know when the methods are invoked and what kind of codes should be placed in which method. Because I told in an applet there several types of standard methods available and each of the methods have a specific purpose. So unless you when they are called, why they are called? You will not be able to clearly tell that which code has to be put inside which method.

(Refer Slide Time: 20:18)



So let us look at the methods first which applet specific. Now in the example that we have seen we have already seen the init method. And we had told or mentioned that the init method is invoked whenever the applet is started. Whenever the applet is executed on by browser for the first time. So all the initializations that the applet requires with respect to the window, with respect to color everything, with respect to the buttons and the other GUI components you need, they will all have to be done by the init method. This init method you should understand, this is called only once when the applet is loaded. So whatever code you should put inside init, it should be the code which is executed only once at the beginning and not executed anytime in the future.

This you should understand. There is another method called start. The method start is invoked after you invoke init. So this method is either invoked after init. Or as the starting point or the resumption point of an applet after it was stopped and you want to restart it again. So a typically situation when this might happen is this. Suppose I was displaying an applet as part of a page web page on the browser. In the mean time I type in some URL and I go to some other page or I click a link and go to some other page. So now the applet is no more on the page. Now again from that new page I can press back, I can come back to my original page which contains the applet.

So now the applet has to be resumed. So it is at this point we can invoke the start method again. So the start method is invoked, every time we want to resume execution of the applet. So simplistically speaking, every time the applet window is refreshed or the code is displayed on the screen you have to invoke start. Say if a window was closed you are opening again. So again the code has to be displayed. You are going to some page; you are coming back to the page again. Then also you need to display. So under this situation where you need to display the HTML document again, so under these circumstances the start method will be invoked.

(Refer Slide Time: 23:12)

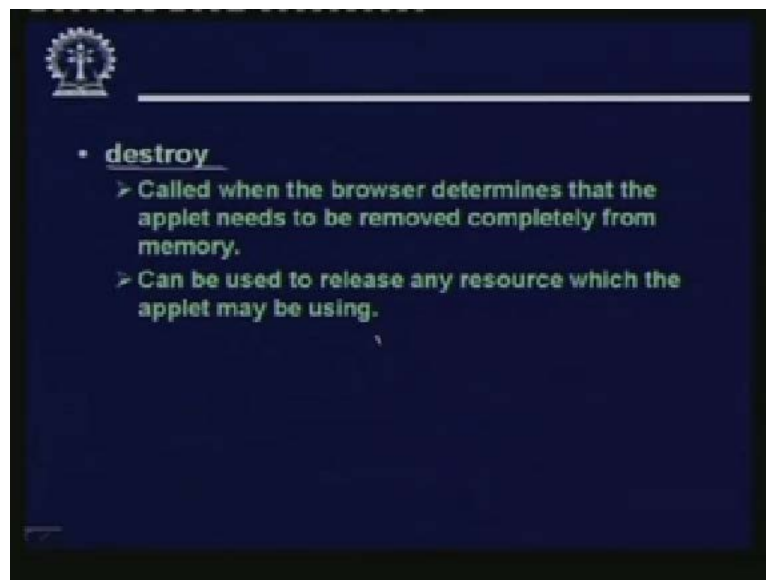


There is a method called paint. This will be called every time the window is damaged. So if you put one window on top of the applet window, again remove it bring it on top. So the

other window is damaged. The paint will be called which will be refreshing the window. And update is a similar function method which first fills the window with the background color and then calls paint. So normally you would be calling update it, will first initializing initialize the window with the default background color. Then it will refresh the graphics which is there using the method paint. Stop is a method which is used to temporarily stop the applet from running. This is typically called when the browser moves to some other document.

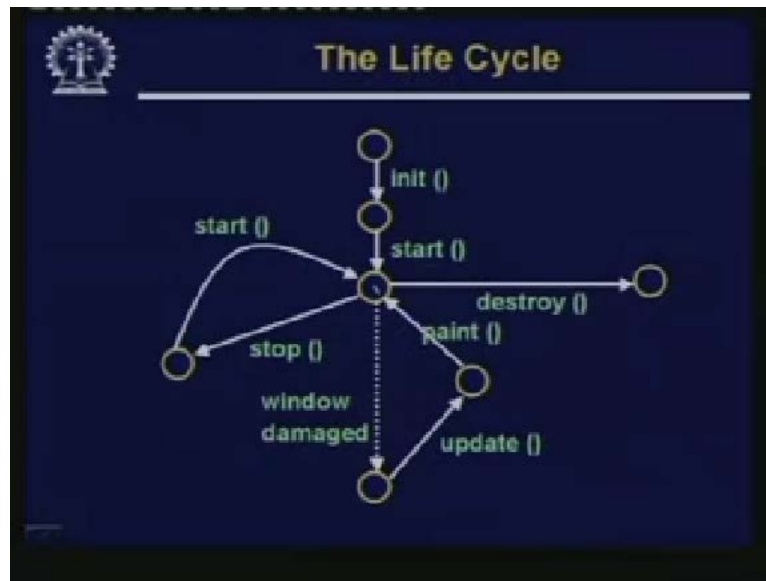
Suppose you type some other URL or click a link as I had said and some other document is displayed on the browser. So during this time, there is no point in continuing to run the applet which may be an animation applet. Animation thread which is a time consuming activity, it will be consuming CPU time. So if the applet at all is not visible to us. There is no point in continue running the applet there by consuming CPU time. Rather you suspend or stop the applet till it becomes visible again. So stop is used to temporarily suspend the applet. Now these activities can be restarted if we invoke start method again in the future.

(Refer Slide Time: 25:08)



And finally destroy is a method which is called, when you want to permanently remove the applet from the memory. Permanently remove means whatever resources the applet might be using including memory they will all be removed or relinquished. Now this destroy can be called when you are using the closing the browser window all together. Or if the applet itself there is an explicit button to close this application. Close this program if you click it. Then the destroy method will be invoked and applet will be removed from the memory totally. So these are the different methods and just what I said.

(Refer Slide Time: 25:55)



If you look at this diagram, this diagram summarises the same thing. The circular nodes are the states of an applet and each edge or the state transition is labelled by a method. So from the beginning whenever the applet starts the `init` method is invoked. So the applets start from the initialization state to an initialized state. Then the `start` method is invoked. Now the applet is running here. Now at this point, if you invoke `stop`, then it goes towards suspended state. From this suspended state, if you invoke `start` again it comes back to the active state. If you call `destroy` then the applet is removed from the memory. This is the terminate state and if while executing somehow the window gets damaged you go to a temporarily state which means active but damaged.

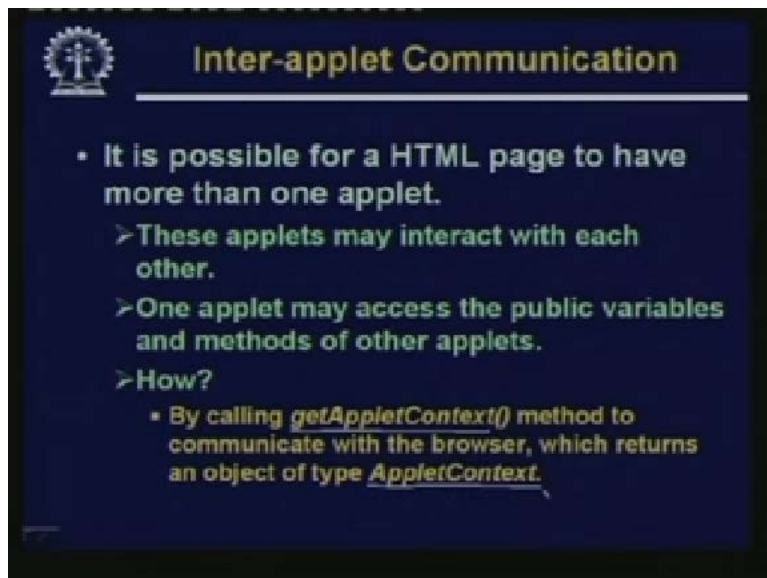
So whenever the window becomes prominent again you have to call `update` to initialize the background color. Then to call `paint` to redraw the objects which are present in the window again. So this simple state transition diagram summarizes the life cycle of an applet. What are the methods which can cause state transition? What are the typically functions of this method and when they are called? So with this understanding you will be in a position to know what piece of code should be included or put inside which part the applet.

(Refer Slide Time: 27:46)



Now let us look at the situation when we want to put multiple applets on the same page. Now earlier we have said that it is indeed possible to put more than one application in the same page. It is also possible for them to communicate among themselves. One applet can send some data to the other applet can send or can call a method and so on. Let us see through a simple example how this can be done.

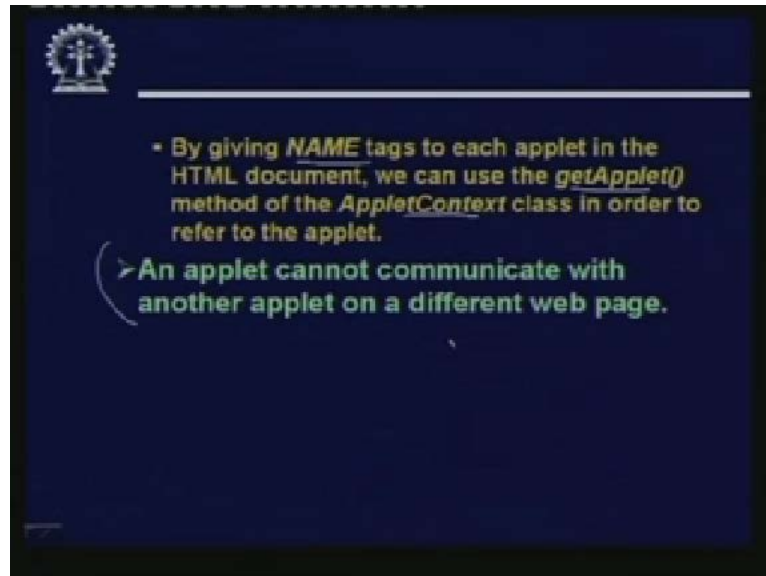
(Refer Slide Time: 28:20)



As I had said a HTML page can have more than one applet. Now we typically would want these applets to interact among themselves. Which means one applet may have access to the public variables and methods of the other applets. How we do this? We do this by invoking a method called `getAppletContext` which gives us a mean which gives the applet a mean to communicate with the browser. The browser will actually return when you make a call to this and object of type `AppletContext` which actually is a pointer or a means to access the other

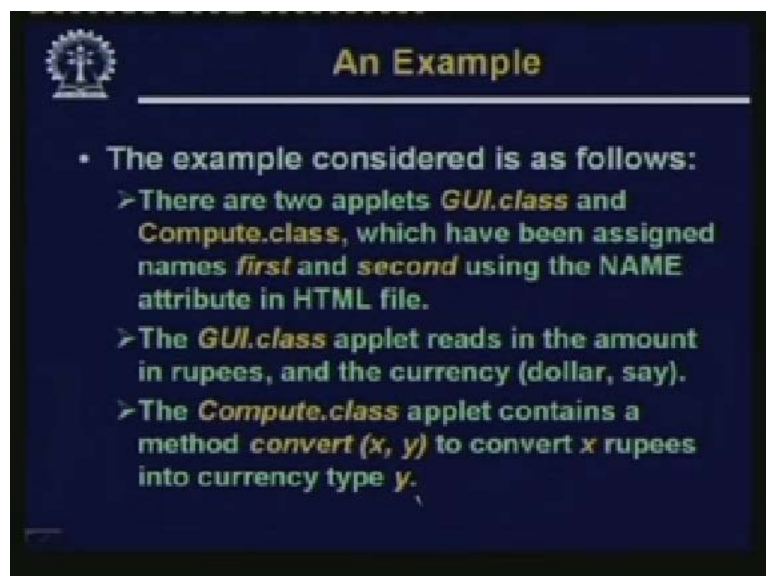
applet you wanted. From one applet, you can make a call to this method. You get an object of AppletContext type using which you can access the methods and the variables of the other applet.

(Refer Slide Time: 29:26)



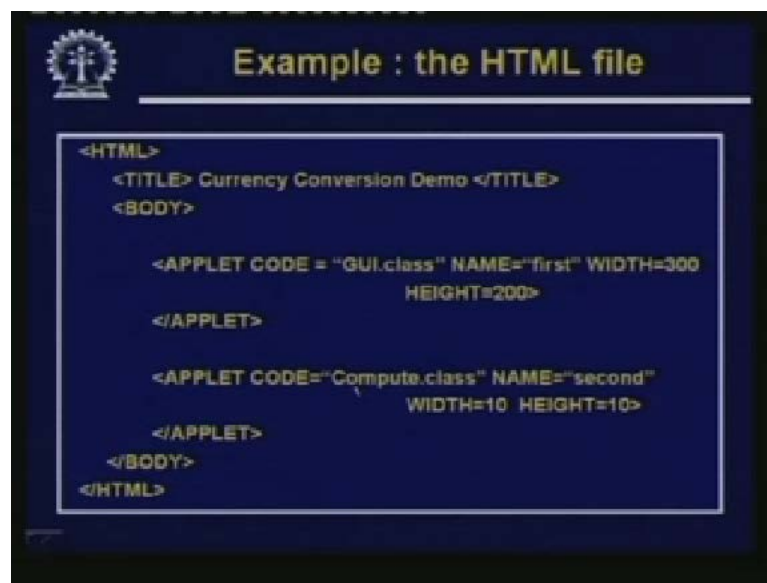
Now this you can do by using the NAME tag in the HTML document when you are including the applet. So the getApplet method of the AppletContext, also we shall see an example that how we can refer to an applet like this. But the constraint is that you can only carry out this kind of communication between applets on the same page. One applet located on one page cannot communicate with another applet on different web page. This is a constraint you will have to remember.

(Refer Slide Time: 30:06)



All write the example that we look at is this. There are two applets we consider. One is with a name GUI.class and other with a name compute.class. These two applets have been assigned the names first and second in the HTML file using the name attribute. The GUI class is what it does is it actually reads in the amount of money in Rupees. So by default it reads in a value which indicates the money in Rupees. And it also reads in a currency type dollar, euro or something else. The compute.class will actually read whatever data was fed in the GUI class including the amount and currency type and will contain a method called convert which will convert Rupee into the currency of the type specified. So the idea is like this. One applet simply allows the user to type in the amount in Rupees and currency type. The other has a method which can convert into the currency type specified. So the first applet will be calling the second applet with the parameters the user has entered and the result that is coming back will be used to display the result back on the screen. This is what this example will do.

(Refer Slide Time: 32:00)



```
<HTML>
<TITLE> Currency Conversion Demo </TITLE>
<BODY>

  <APPLET CODE = "GUI.class" NAME="first" WIDTH=300
                                HEIGHT=200>

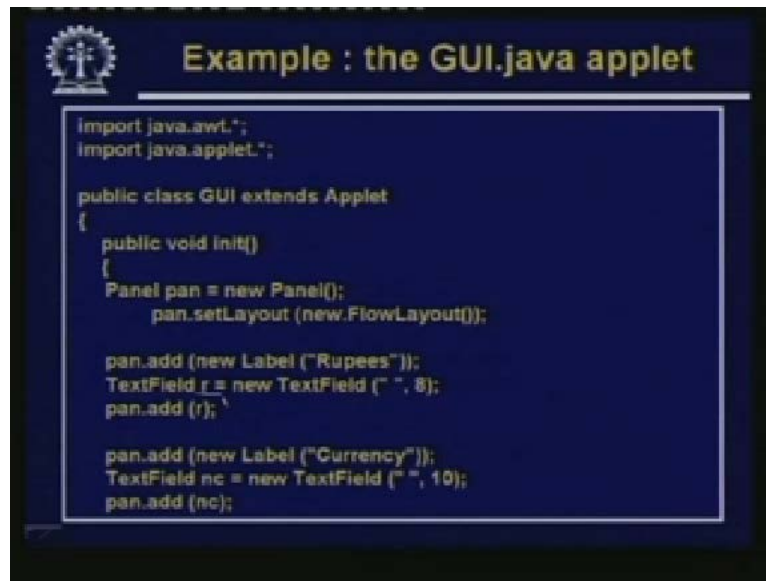
</APPLET>

  <APPLET CODE="Compute.class" NAME="second"
                                WIDTH=10 HEIGHT=10>

</APPLET>
</BODY>
</HTML>
```

First let us look at the HTML file. So the TITLE is currency conversion demo. There are two applets the first applet GUI class, the NAME that we have given is first. Some WIDTH and HEIGHT of this applet given. The second applet is not actually meant to be displayed on the screen that so the HEIGHT as a token WIDTH and HEIGHT are given small numbers 10 by 10, Compute.class the NAME is given as second. This is just a dummy example so no other HTML code is there. Now let us see what is there in the **do Java program** GUI class and Compute class.

(Refer Slide Time: 32:41)



```
import java.awt.*;
import java.applet.*;

public class GUI extends Applet
{
    public void init()
    {
        Panel pan = new Panel();
        pan.setLayout (new FlowLayout());

        pan.add (new Label ("Rupees"));
        TextField r = new TextField (" ", 8);
        pan.add (r);

        pan.add (new Label ("Currency"));
        TextField nc = new TextField (" ", 10);
        pan.add (nc);
    }
}
```

This is the GUI.java applet. In the GUI.java applet, what we have is this. First we have the init method. Now in the init method we create a new panel. On the panel we set the layout manager to FlowLayout. Then we add a little a label to the panel Rupees. Just alongside this we add a text field of size 8 and add the text field to the panel. Secondly we add another label called currency and another text field where the new currency is to be typed in. Here the default WIDTH is 10. This also we add. So there are two, you can say two form elements have been added in one of which I will be typing in an amount in the other I will be typing in the new currency name dollar or whatever.

(Refer Slide Time: 33:54)



```
pan.add (new Label ("Result of conversion:"));
    TextField result = new TextField (" ", 50);
    pan.add (result);

    add ("South", pan)
}

public boolean handleEvent (Event v)
{
    if (v.id == Event.WINDOW_DESTROY)
        System.exit (0);
    return super.handleEvent (v);
}
```

And then another field I am adding, this is the result; Result of conversion. So here the text field I am assuming of size 50. This also we are adding and all these panels you are adding to

the bottom of the panel south. Now the Event handler is similar to the example we had shown earlier. So nothing to say here.

(Refer Slide Time: 34:23)



```
public boolean action (Event v, Object arg)
{
    int x = Format atoi (r.getText());
    String y = nc.getText();
    Compute p = (Compute) getAppletContext . getApplet
("second");

    result.setText (r.getText().trim() + "Rupees = " + p.convert
(x, y) + nc.getText().trim());
}

private TextField r, nc;
}
```

And whenever some action takes place, whenever you click on something on the convert, then this method gets to execute. Here you see what we are doing, `r.getText`. This we are storing to `x`. If you go back you see the labels that you create the first label where you are typing in the Rupee, the name of the text window is `r`. So here we write `r.getText`. So we read the contents of that particular field store it into `x` after converting to integer, ASCII to integer conversion. String `y` equal to `nc`, `getText` in the other field `nc` it is already a string it is the currency type dollar baht. Or whatever I am typing there, so that as a string we are storing in `x` and here in the third line we are invoking a particular method of the other applet. So first we are getting an applet context; `compute` is the name of the second.

So we are creating an object of type `compute`. We are calling the `getContext` object. The `getApplet` method of that with the name of the second applet as parameter. So it returns a pointer to the second applet `p`. Now we can call. Here you see `p.convert`, `x y` we can now call the `convert` method of the second applet with the parameters `x` and `y` and the third panel element here which is `result` we are setting the result `setText` to a value. First the Rupee value `r` `getText` will trim any spaces will be trimmed deleted **concatenated to** Rupees equal to; you convert it plus, the amount you have typed in the `nc` part. So in this way you can actually, so you will be showing here as 1000 Rupees equal to say 25 dollars. Something like this will be displayed and `r nc` are private fields which are not to be accessed from outside.

(Refer Slide Time: 37:05)



Now the other applet the Compute.java looks like this. Where in the init function, it simply creates a panel, creates some labels. All these are optional. You can omit all these things because the second applet does not really require any kind of graphics. This is TextField result. All these are again optional, these are not needed.

(Refer Slide Time: 37:27)



What is needed is this method called convert. It takes two parameters. If the currency type equals, this string dollar then say you divide rupee by 45 and return it. If it equals the Thailand currency baht. Then you divide by “1.2” or if it equals any other currency xyz. You make the appropriate calculation. So in this way through if an if then else you can carry out any arbitrary currency conversion based whatever currency type you have typed in. So this simple example shows you how one applet can call some method of another applet, passing

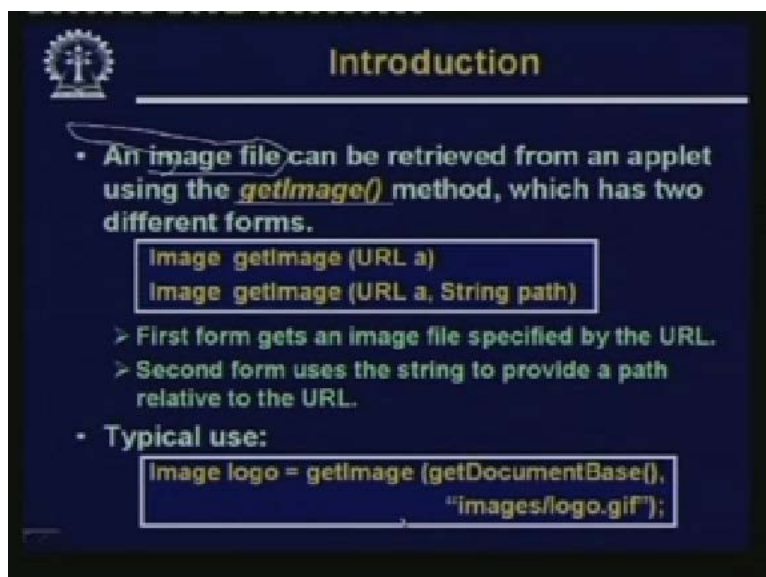
some arguments and getting the back the returned result and using it for some subsequent calculation.

(Refer Slide Time: 38:18)



So let us look at some more examples of applets. Some very specific things we first look at how we can image files from an applet. Because this is something which is quite popularly used we often find or see an op applet where some kind of a photo album is being shown cyclically automatically. So how we go on changing the image being displayed on the screen from a Java program applet. Let us see that.

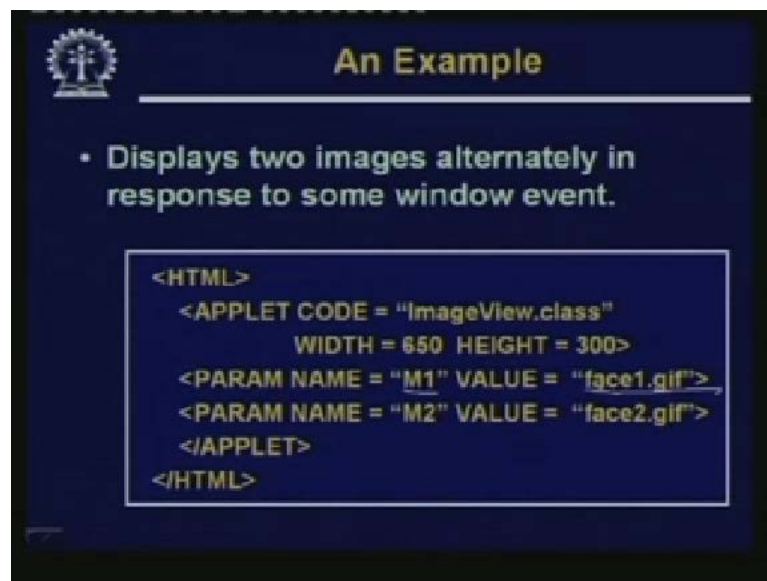
(Refer Slide Time: 38:55)



Now when you are trying to display an image or an applet, first thing is that you need to access the image file. So the image file you can retrieve using the `getImage` method. Now the `getImage` method has two forms where you directly specify the URL, where you can get an

image file specified by the URL. Secondly you give a relative path and the URL is subsequent to that string path. So the second form will be using this string to provide a path relative to the URL. For example in the URL path you give HTTP www.iitkgp.ac.in is the string path portion you give slash docx slash images slash a.jpg. So you can separate these two components out of an URL. So as to make your program look more structured instead of typing a very long URL's every time. So a typically use will be like this because the advantage is that this URL a need not be typed every time because if you call the method `getDocumentBase` then by default it will return the Base URL of the current document. So the string path can be specified like this, images slash logo.gif with respect to the Base Document. So this is a typical use of this function.

(Refer Slide Time: 40:45)



The slide, titled "An Example", features a dark blue background with a white logo in the top left corner. The title "An Example" is centered at the top in a yellow font. Below the title, a white bullet point states: "Displays two images alternately in response to some window event." In the center, a white-bordered box contains the following HTML code:

```
<HTML>
  <APPLET CODE = "ImageView.class"
           WIDTH = 650 HEIGHT = 300>
  <PARAM NAME = "M1" VALUE = "face1.gif">
  <PARAM NAME = "M2" VALUE = "face2.gif">
</APPLET>
</HTML>
```

Now here we show an example where two images are displayed alternately in response to some window event. Now window event means some event that is occurring on the window. So either where are clicking the mouse on the window or we are closing a window opening it again. So suppose initially, this on the window, the image 1 was displayed, I close, open it again; I see that now image tool is displayed. I close and open again. Again I see image 1 is displayed. So this will continue happening. So first the HTML file. Here in the HTML file you see here I have the APPLET the NAME of the code `ImageView.class` size I have specified the size will be approximately equal to size of the Image and the two image files are specified as parameters. So in order to make the applet program general these parameters are passed from HTML. So for the first one, the NAME of the parameter is M1 and value is `face1.gif`, for the second one, NAME is M2 and file NAME is `face 2.gif`.

(Refer Slide Time: 42:12)



```
import java.awt.*;
import java.applet.*;
public class ImageView extends Applet
{
    boolean flag = true;
    Image x, y;
    public void init ( )
    {
        x = getImage (getDocumentBase(), getParameter("M1"));
        y = getImage (getDocumentBase(), getParameter("M2"));
    }
}
```

Now let us look at the actual code. The code fairly simple. This is the class ImageView. So here we have Boolean flag which we have initializing it to true. This flag actually indicates that which of the two images will be displayed. The way the program is written is that if flag is true, then one image will be displayed. If flag is false, then the other image will be displayed. So flag is a Boolean variable which is used to keep track of the fact that which image is to be displayed next. So x and y are two variables of type image. This is the init method. This gets executed only once. So in this example what needs to be done exactly once is to get the two image files get access to the image files.

So you call this get image function with getDocumentBase. And if you look at the previous page in the PARAM with NAME M1, I am assigning the file name face one.gif. This is the parameter value. So here I am calling the get parameter method to retrieve the value of this parameter which will be face1.gif. Similarly in the second one it will be face2 gif. The two files we have specified, you actually open the two images and assigning it to variables x and y. This x and y are variables of type image. So in the init method we are actually opening two image objects assigning one of them to x and other to y. This is what we are doing here.

(Refer Slide Time: 44:15)



```
public void paint (Graphics g)
{
    if (flag)
    {
        g.drawImage (x, 0, 0, this);
        flag = false;
    }
    else
    {
        g.drawImage (y, 0, 0, this);
        flag = true;
    }
}}
```

Here this is the paint. So every time the window gets destroyed and it comes up, we need to call paint. So every time you close the window and open it again, this is a window event. Here this paint method will be called. So let us see what we are doing in paint. In paint, we are comparing that Boolean variable flag. If it is true, then we are drawing x using the draw image method. Parameters are x the name of the image objects the coordinate 0, 0 from where in the window the image has to be displayed. And this represents in which frame or window this means the current one. And we made flag to call separate so that next time when you come you go to else part. Similarly the else part you are displaying the image y and the then you are flag to true. So as this example shows you see this example also that if flag is true you come here this x. If flag is false you come here display y.

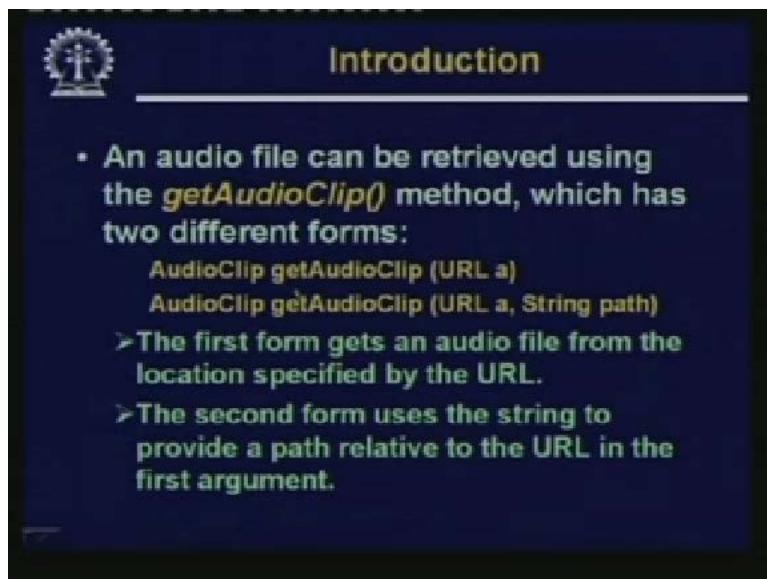
So every time the window is closed you open it again the method paint gets invoked and alternately once the image x other time image y would be displayed cyclically. So this is what we wanted. That, two images they will be displaying **simul** means display alternately. So this program you can generalize to handle any number of events. Any number of images which will be displayed one after the other or any particular order. Delay maybe different, whatever you can do using Java you can implement here. You can use some functions where you can specify the delays. All these things can be done.

(Refer Slide Time: 46:20)



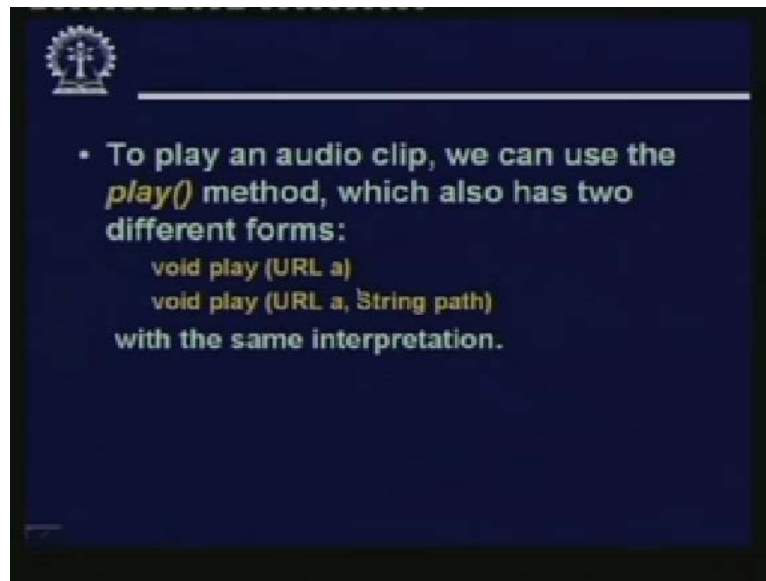
Lastly we will look at an example where we will show how we can include audio clips inside a Java applet. This is another which we require or use, quite often as part of Java applet which shows attractive web pages. So there are methods again existing for this.

(Refer Slide Time: 46:45)



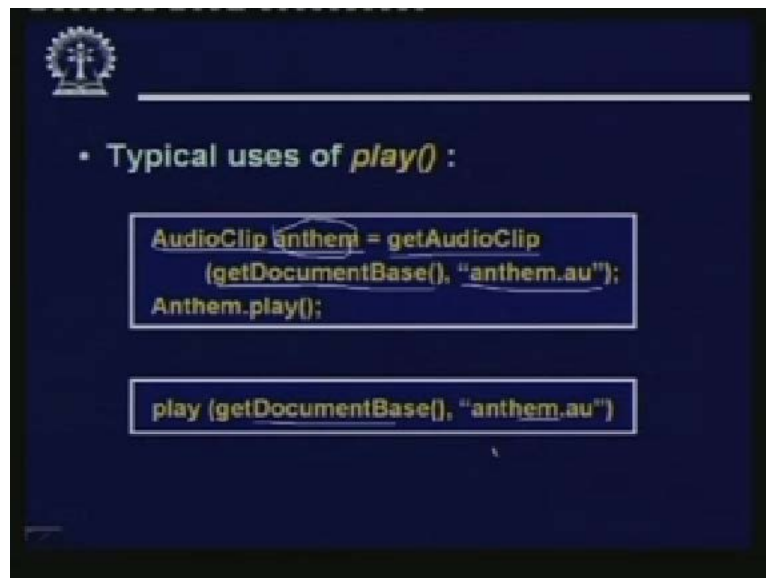
There is method called `getAudioClip` using which you can retrieve or audio object from a specified file specified URL. So just like image there are two alternate forms. The first form, you can specify the URL directly. In the second form, you can specify the base URL and the path relative to the base URL just like the image case. You have this `getAudioClip` method which you will invoke in one of these two ways.

(Refer Slide Time: 47:26)



To play the audio clip you have loaded, you can use the play method. Play method again has two parameters. You can either specify play URL or play URL. The base URL and the String path. Of course another form is there which I have not shown here you can call play without any parameter. This third form will by default play the audio image which was loaded previously using this `getAudioClip` method. So if you already have loaded a particular audio clip using `getAudioClip` you can just call `plat` subsequently. So the last audio clip loaded would get played by default or you can specify the clip id as a parameter.

(Refer Slide Time: 48:21)



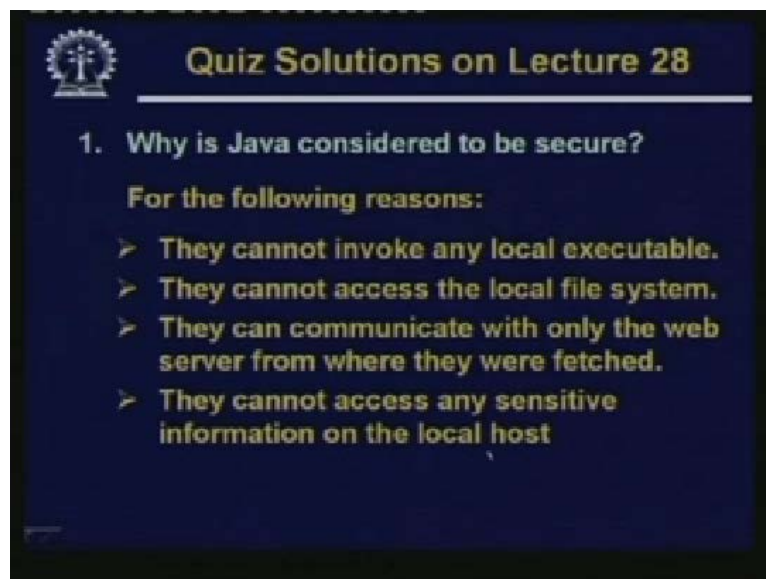
Some typical usages of the play function or as follows. Here you can either say, here the first one, you create a variable of type audio clip and you get an audio clip by calling this method `getAudioClip` with a call to the `getDocumentBase` which gives the base URL and the relative path you get `anthem.au`. And then with this returned variable object you call `play Anthem` and

then dot play. So the last one will start playing or you can directly invoke play also getDocumentBase and anthem.au. So in either of these two approaches you can play audio clips. Now in this same way we have just shown a couple of examples to show that from a Java when you are designing an applet, it is rather easy to include images audio clips as examples. Now in an applet which is supposed to be there as part of a web page.

These kinds of requirements are essential because they help you in designing very attractive web pages with multimedia contents. So in general you can have other kinds of contents also. Like you can have an mpeg video clip; you can have a streaming video clip; you can have any kind of content you can think of. Provided the appropriate library is available to you, which you can include as an object and you can call the methods of that object to load it to play it in whatever way you want. For example if you consider a video clip then the methods that may be present under them would not only just be play. It can be fast forward rewind position to a particular point, stop, pause, all these things. So all these methods will be very much media dependent and Java actually has full support for almost all kinds of media that are intended to be included as part of web pages.

So it helps you in designing web pages in a very convenient way. Now in our next lecture we shall be continuing our discussion and we shall show that how Java also can be used very easily for developing network applications. We shall see that how easy it is to write client-side programs in Java. You can write a Java client, you can write a Java server, they can be applets. One of them can be an applet; other can be a Java program running on the server. They can communicate among themselves and the amount of additional lines of code you need to write. For this is very small compared to an equivalent code you write in a language like C or C plus plus say for example. So Java is also very suitable for network or internet programming. This we shall see in our next class. So with this we come to the end of whatever we had to discuss today. So now let us look at some of the solutions to the questions we posed in our last class.

(Refer Slide Time: 52:21)

A presentation slide with a dark blue background and yellow text. In the top left corner is a circular logo with a gear and a cross. The title "Quiz Solutions on Lecture 28" is centered at the top. Below the title is a horizontal line. The main content is a list of reasons why Java is considered secure, starting with a question and followed by a list of four points.

Quiz Solutions on Lecture 28

1. Why is Java considered to be secure?

For the following reasons:

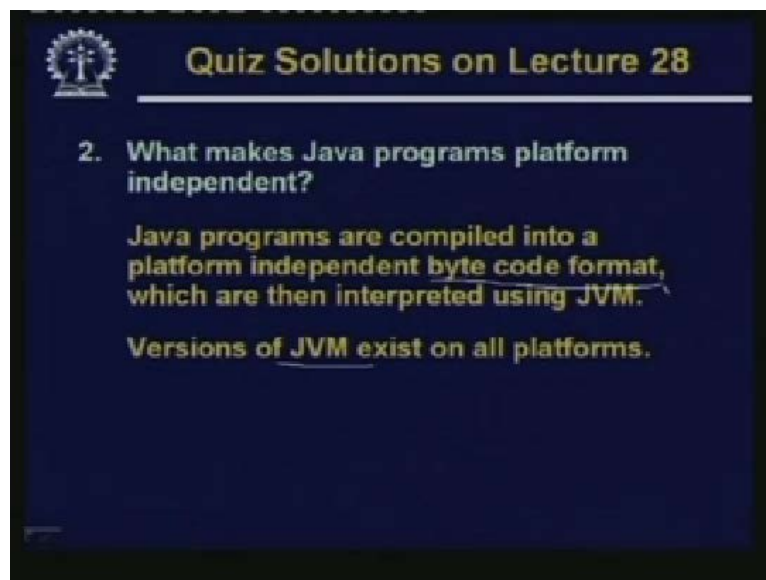
- They cannot invoke any local executable.
- They cannot access the local file system.
- They can communicate with only the web server from where they were fetched.
- They cannot access any sensitive information on the local host

The first question was:

Why do we consider Java to be secure?

Now this we had already mentioned in your last class. I am again repeating because this is very important concept to understand. Java programs are downloaded from some other web site and we would expect that when you are downloading and running them on my client machine. The Java program should not do any kind of harm or should not be able to retrieve any sensitive information that might be present on my local file system. So Java programs typically are not allowed to invoke any local executable. They are not allowed to access the local file system. They can communicate with only the web server from where the Java program was downloaded. They cannot communicate with any other server and moreover since they do not have access to the local file system they cannot access any sensitive information that are present on the local file system. Like for example user name email address or any other such information which are very personal centric which we do not want to share with others on the internet.

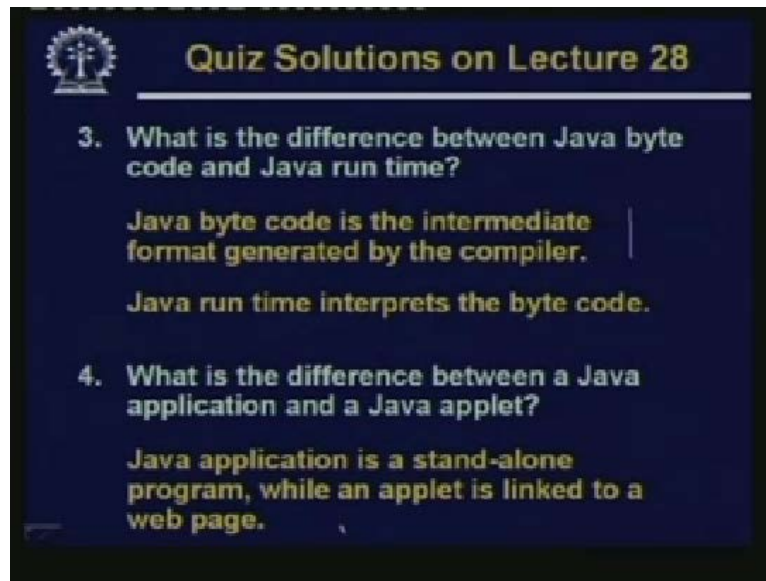
(Refer Slide Time: 53:43)



What makes Java programs platform independent?

Now this we had mentioned that basically the presence of the Java runtime or the Java Virtual Machine helps in making Java programs platform independent. Because Java programs are compiled into a platform independent byte code format. Byte code format is platform independent. You can freely copy them, you can freely download them over the internet byte code is the same on all machines. But it is the Java Virtual Machine which has to be there on your machine to make your machine Java ready or Java enabled. So if you have the JVM installed on your machine. You can download Java byte codes and run them on you are machine. And as I had said JVM or the Java runtime exist today on almost all computing platforms that are in existence.

(Refer Slide Time: 54:46)



The slide features a dark blue background with a white logo in the top left corner. The title "Quiz Solutions on Lecture 28" is centered at the top in a yellow font. Below the title, there are two numbered questions and their corresponding answers, also in yellow text.

3. What is the difference between Java byte code and Java run time?

Java byte code is the intermediate format generated by the compiler.

Java run time interprets the byte code.

4. What is the difference between a Java application and a Java applet?

Java application is a stand-alone program, while an applet is linked to a web page.

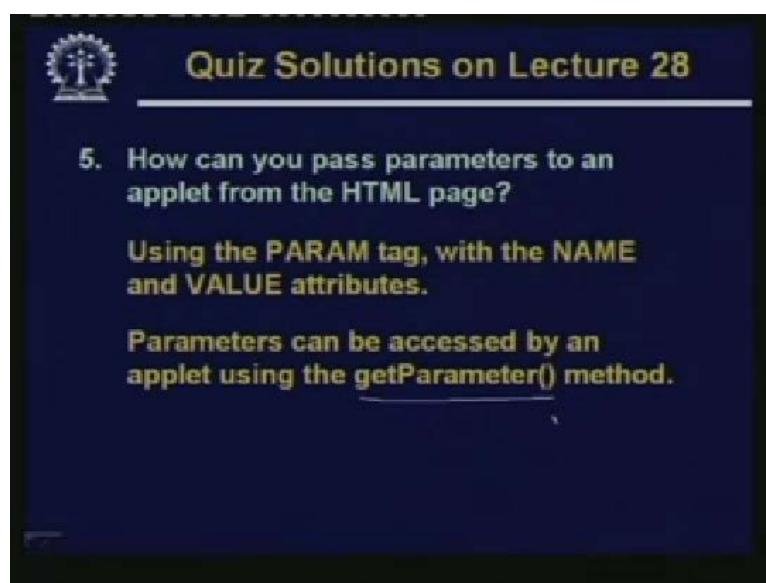
What is difference between Java byte code and Java runtime?

Java byte code is actually that intermediate code. This is the machine code of that hypothetical machine which is generated by the j compiler. On the other hand, Java runtime is the interpreter which executes the byte code.

What is the difference between a Java application and a Java applet?

This we have said repeatedly. Java application is a standalone program which typically is executed in a standalone fashion where as an applet is linked to a web page it cannot execute independently. It gets executed along with the web site in which it is executed along with the web page. So the web page comes along with the Java. Java applet which gets displayed along with the surrounding HTML code.

(Refer Slide Time: 55:46)



The slide features a dark blue background with a white logo in the top left corner. The title "Quiz Solutions on Lecture 28" is centered at the top in a yellow font. Below the title, there is one numbered question and its corresponding answer, also in yellow text.

5. How can you pass parameters to an applet from the HTML page?

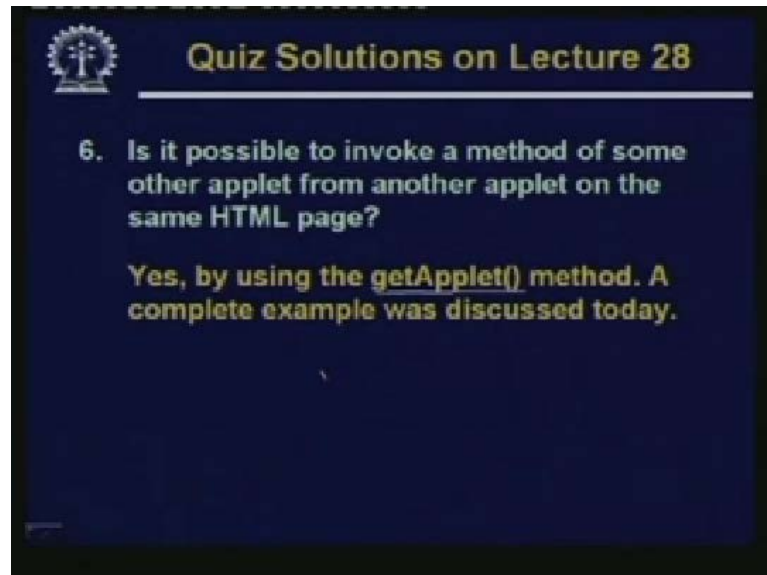
Using the PARAM tag, with the NAME and VALUE attributes.

Parameters can be accessed by an applet using the `getParameter()` method.

How can you pass parameters to an applet from the HTML page?

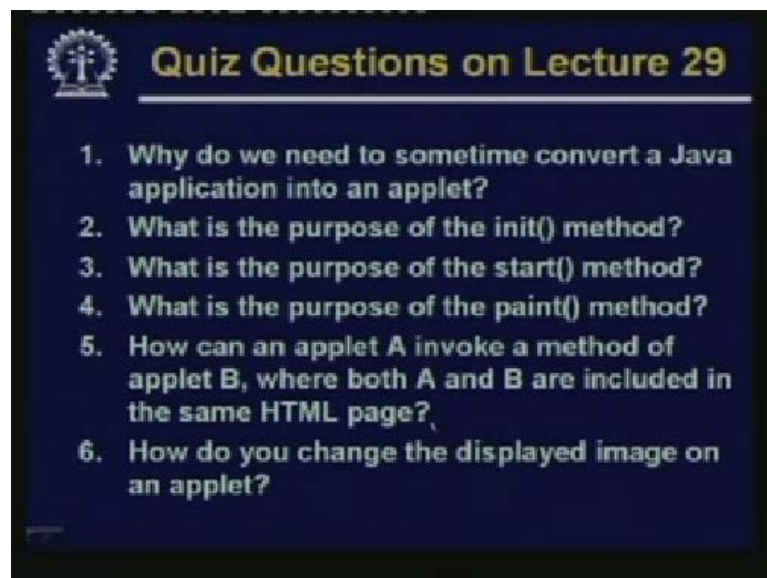
This we shall, we have seen already through examples that we can use the PARAM tag with the NAME and VALUE attributes to pass the parameters and the parameters can be accessed by invoking the getParameter method.

(Refer Slide Time: 56:06)



Lastly, is it possible to invoke a method of some other applet from another applet on the same page? Yes we have looked at an example we had seen that we can use the `getApplet` method to do so. The example which was discussed today illustrates this in detail. So now questions from today's lecture.

(Refer Slide Time: 56:33)



Why do we need to sometime convert a Java application into an applet?
What is the purpose of the `init` method?

What is the purpose of the start method?

What is the purpose of the paint method?

How can an applet A invoke a method of applet B where both A and B are included in the same HTML page?

How do you change the displayed image on an applet?

So as I had said, in our next lecture we shall be continuing our discussion on Java programming with special emphasis on how to write or create client-side applications using Java. Till then, good bye.