MODULE 5: APPLETS & SWINGS

Syllabus:

The Applet Class: Two types of Applets; Applet basics; Applet Architecture; An Applet skeleton; Simple Applet display methods; Requesting repainting; Using the Status Window; The HTML APPLET tag; Passing parameters to Applets; getDocumentbase() and getCodebase(); ApletContext and showDocument(); The AudioClip Interface; The AppletStub Interface; Output o the Console.

Beautiful thought: *"You have to grow from the inside out. None can teach you, none can make you spiritual. There is no other teacher but your own soul."* — *Swami Vivekananda*

APPLET

The Applet Introduction:

- Applets are small Java program/applications that are accessed on an Internet Server, transported over the Internet, automatically installed, and run as part of a Web document

- An applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included in a page. When you use a Java technology enabled browser to view a page that contains an applet, the applet's code is transferred to your system and executed by the browser's Java Virtual Machine (JVM)

Two Types of Applets

There are two varieties of applets. They are

1. Based on the Applet class: **Applet**

2. Based on the Swing Class Applet: **JApplet**

1. Based on the Applet class.

- These Applet uses the Abstract Window Toolkit(AWT) to provide the graphical user interface.
- This type of applet has been widely available since java was first created.

2. Based on the Swing Class Applet.

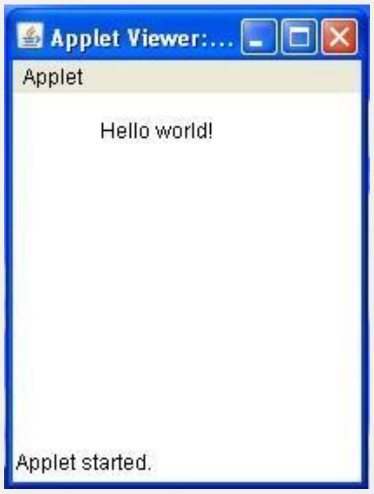
- This applet uses the swing class to provide GUI.
- Swing offers a rich and easier to use interface than AWT.
- Swing based applets are the most popular in practice.

Applet Basics

- The reason people are excited about Java as more than just another OOP language is because it allows them to write interactive applets on the web. Hello World isn't a very interactive program, but let's look at a webbed version.

```
import java.applet.Applet;
import java.awt.Graphics;


public class HelloWorldApplet extends Applet
 {
      public void paint(Graphics g)
      {
            g.drawString("Hello world!", 50, 25);
      }
  }
```

OUTPUT



```
Hello world!
```

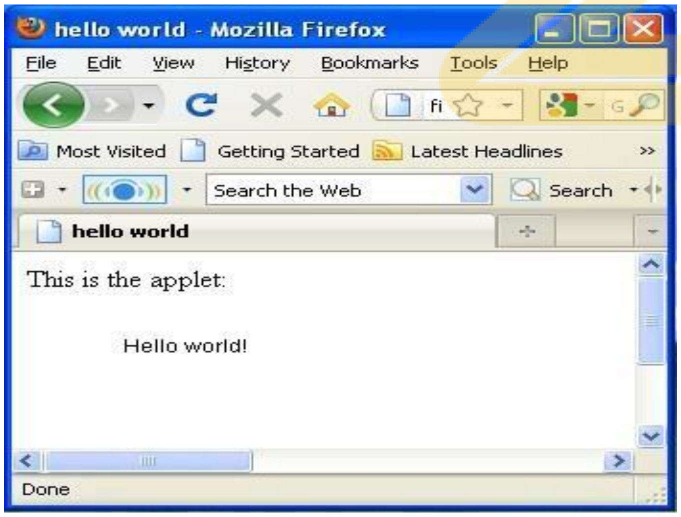
```
Applet started.
```

The Applet version of HelloWorld is a little more complicated than the HelloWorld application, and it will take a little more effort to run it as well.

First type in the source code and save it into file called HelloWorldApplet.java. Compile this file in the usual way. If all is well a file called HelloWorldApplet.class will be created. Now you need to create an HTML file that will include your applet. The following simple HTML file will do.

```
<html>
<head>
<title> hello world </title>
</head>

<body>
This is the applet:<P>
<applet code="HelloWorldApplet" width="150" height="50">
</applet>
</body>
</html>
```

OUTPUT



```
This is the applet:

        Hello world!
```

hello world - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines
Search the Web Search
hello world
Done

- Save this file as HelloWorldApplet.html in the same directory as the HelloWorldApplet.class file. When you have done that, load the HTML file into a Java enabled browser and see the output in the browser window.
- If the applet compiled without error and produced a HelloWorldApplet.class file, and yet you don't see the string "Hello World" in your browser chances are that the .class file is in the wrong place. Make sure HelloWorldApplet.class is in the same directory as HelloWorldApplet.html. Also make sure that your browsers support Java or that the Java plugin has been installed. Not all browsers support
Java out of the box.

The Applet Class

- An applet is a small program that is intended not to be run on its own, but rather to be embedded inside another application.
- The **Applet class** must be the super class of any applet that is to be embedded in a Web page or viewed by the Java Applet Viewer. The Applet class provides a standard interface between applets and their environment.

Method	Summary
Void **destroy**()	Called by the browser or applet viewer to inform this applet that it is being reclaimed and that it should destroy any resources that it has allocated.
AccessibleContext **getAccessibleContext**()	Gets the AccessibleContext associated with this Applet.
AppletContext **getAppletContext**()	Determines this applet's context, which allows the applet to query and affect the environment in which it runs.
String **getAppletInfo**()	Returns information about this applet.

AudioClip **getAudioClip**(URL url)	Returns the AudioClip object specified by the URL argument.
AudioClip **getAudioClip**(URL url,	Returns the AudioClip object specified by the URL and name arguments.

String name)	
URL **getCodeBase**()	Gets the base URL.
URL **getDocumentBase**()	Returns an absolute URL naming the directory of the document in which the applet is embedded.
Image **getImage**(URL url)	Returns an Image object that can then be painted on the screen.
Image **getImage**(URL url, String name)	Returns an Image object that can then be painted on the screen.
Locale **getLocale**()	Gets the Locale for the applet, if it has been set.
String **getParameter**(String name)	Returns the value of the named parameter in the
String[][] **getParameterInfo**()	HT Ret M ur L ns tag. inf ormation about the parameters than are understood by this applet.
Void **init**()	Called by the browser or applet viewer to inform this applet that it has been loaded into the
Boolean **isActive**()	system. Determines if this applet is active.
static AudioClip **newAudioClip**(URL url)	Get an audio clip from the given URL.
Void **play**(URL url)	Plays the audio clip at the specified absolute URL.
Void **play**(URL url, String name)	Plays the audio clip given the URL and a specifier that is relative to it.
Void **resize**(Dimension d)	Requests that this applet be resized.
Void **resize**(int width, int height)	Requests that this applet be resized.

Void **setStub**(AppletStub stub)	Sets this applet's stub.

void **showStatus**(String msg)	Requests that the argument string be displayed in the "status window".
void **start**()	Called by the browser or applet viewer to inform this applet that it should start its execution.
void **stop**()	Called by the browser or applet viewer to inform this applet that it should stop its execution.

Applet Architecture

- An applet is a window-based program, its architecture different from the console-based programs. There are two key concepts to understand the architecture they are

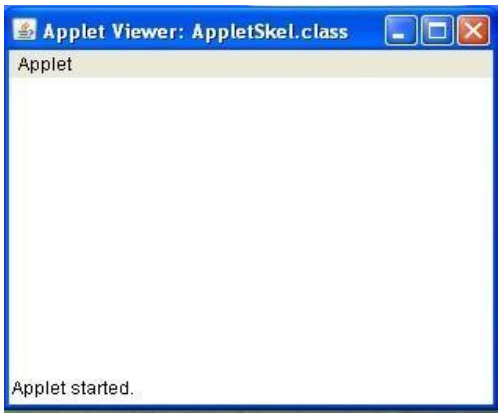
1. **Applets are Event driven**
    - An applet waits until an event occurs.
    - The *AWT* notifies the applet about an event by calling event handler that has been provided by the applet.The applet takes appropriate action and then quickly return control to *AWT*
    - All *Swing* components descend from the AWT Container class

2. **User initiates interaction with an Applet (*and not the other way around*)**

An Applet Skelton

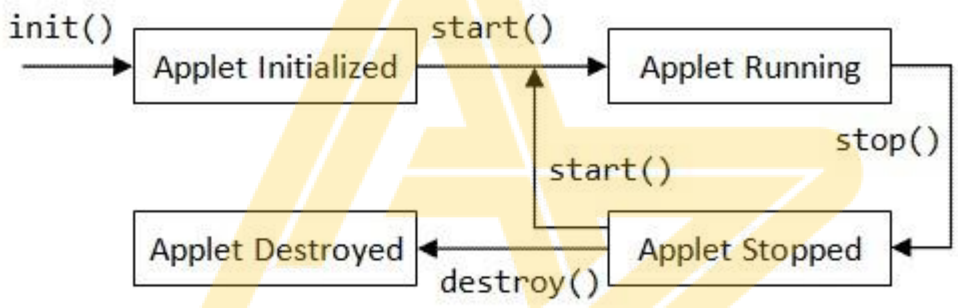
```
 // An Applet skeleton.
import         java.awt.*;
import javax.swing.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/

public class AppletSkel extends JApplet
{
     // Called first. public
      void init()
      {
          // initialization
      }


  /* Called second, after init(). Also called whenever the applet is restarted. */ public
      void start()
      {
          // start or resume execution
       }


  // Called when the applet is stopped. public
       void stop()
       {
              // suspends execution
        }


 /* Called when applet is terminated. This is the last method executed. */ public
       void destroy()
{                                                                          OUTPUT:
          // perform shutdown activities
```

```
    }

  // Called when an applet's window must be restored.
     public void paint(Graphics g)
     {
        // redisplay contents of window
     }
}
```



Applet started.

Applet Initialization and Termination/Applet Life Cycle



It is important to understand the order in which the various methods shown in the skeleton are called. **When an applet begins**, the AWT calls the following **initialization methods**, in this sequence:

1. **init( )**
2. **start( )**
3. **paint( )**

When an applet is terminated, the following sequence of method calls takes place:

1. **stop( )**

2. **destroy( )**

1. init( )

The **init( )** method is the first method to be called. This is where you should initialize variables. This method is called only once during the run time of your applet.

2. start( )

The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped. Whereas **init( )** is called once—the first time an applet is loaded **start( )** is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at **start()**.

3. paint( )

The **paint( )** method is called each time your applet's output must be redrawn. **paint( )** is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, **paint( )** is called.

The **paint( )** method has one parameter of type **Graphics**. This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

4. stop( )

The **stop( )** method is called when a web browser leaves the HTML document containing the applet when it goes to another page, for example. When **stop( )** is called, the applet is probably running. You should use **stop( )** to suspend threads that don't need to run when the applet is not visible. You can restart them when **start( )** is called if the user returns to the page.

5. destroy( )

The **destroy( )** method is called when the environment determines that your applet needs to be removed completely from memory. At this point, you should free up any resources the applet may be using. The **stop( )** method is always called before **destroy()**.

Simple Applet display methods

```
import java.applet.Applet;
import java.awt.Graphics;

 public class HelloWorldApplet extends Applet
 {
         public void paint(Graphics g)
         {
                 g.drawString("Hello world!", 50, 25);
         }
}
```

- Consider the above program to output a string to an applet, use drawString() this is a member of the Graphics class, this drawstring is called from within either update() or paint() as shown in the above program example .The general form of is

  drawString(String msg,int x, int y)

- The **msg** indicates that string to be output beginning **at x,y**. in java window the upper-left corner location is 0,0.the drawstring() method will not recognize newline character.
- To set the background color of an applet window use **setBackground**() and to set the foreground color for example the color in which text is shown use **setForeground**().these methods are defined by Component and they have the following general forms

  void setBackground(Color newColor) ,
  void setForeground(Color newColor)

The **newColor** specifies that new color. The class Color defines the constant shown below that can be used to specify colors.

```
Color.black      Color.lightGray Color.yellow

Color.blue       Color.magenta    Co lor.red
Color.cyan       Color.orange     Color.white
```

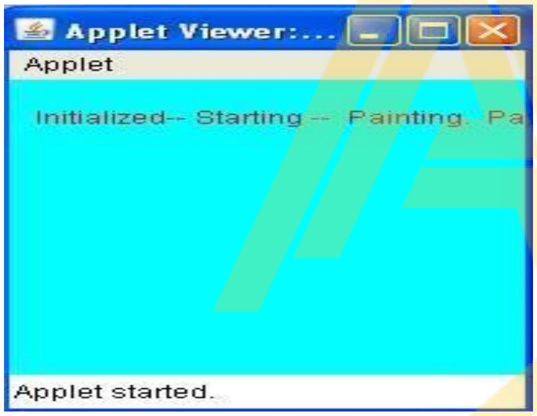
Color.darkGray Color.pink   Color.gray   Color.green **Example:**

```
    setBackround(Color.cyan);
    setForeground(Color.red)
```

Example Program:

```
/* This Applet sets the foreground and background colors and out puts a string. */
import java.applet.*; import java.awt.*;

public class Simple extends Applet
{
          String msg;

  // set the foreground and background colors. public
          void init()
          {
             setBackground(Color.cyan);
          setForeground(Color.red);
          msg = "Initialized--"; }
  // Add to the string to be displayed. public
  void start()
  {
        msg += " Starting --";
  }

  // Display the msg in the applet window. public
  void paint(Graphics g)
  {
        msg += " Painting.";
        g.drawString(msg, 10, 30);
   }
}
```

OUTPUT:



```
Initialized-- Starting -- Painting. Pa
```

```
Applet started.
```

Requesting repainting;

- The **repaint()** method is defined by the AWT. It causes the AWT run time system to call to your applet's update() method, which in its default implementation, calls paint(). Again for example if a part of your applet needs to output a string, it can store this string in a variable and then call repaint(). Inside paint(), you can output the string using drawstring().

The repaint method has four forms.

```
void repaint()
void repaint(int left, int top, int width, int height)
```

```
void repaint(long maxDelay)
```

```
void repaint(long maxDelay, int x, int y, int width, int height)
```

void repaint()

This causes the entire window to be repainted

void repaint(int left, int top, int width, int height)

This specifies a region that will be repainted. the integers left, top, width and height are in pixels. You save time by specifying a region to repaint instead of the whole window.

void repaint(long maxDelay)

void repaint(long maxDelay, int x, int y, int width, int height)

Calling repaint() is essentially a request that your applet be repainted sometime soon. However, if your system is slow or busy, update() might not be called immediately. This gives rise to a problem of update() being called sporadically. If your task requires consistent update time, like in animation, then use the above two forms of repaint(). Here, the maxDelay() is the maximum number of milliseconds that can elaspe before update() is called.

Using the Status Window

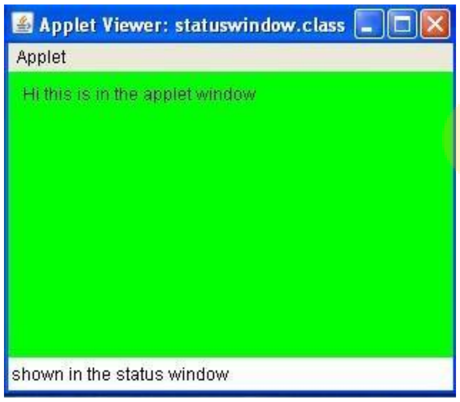
- If the user has chosen to show the Status Bar in their browser then messages can be put there from an applet.

The **showStatus()** method would do it for this applet, if the applet was running in a browser.

Example

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

public class statuswindow extends Applet
{
        public void init()
        { setBackground(Color.green);

        }
        public void paint(Graphics g)
        {
        g.drawString("Hi this is in the applet window",10,20);
        showStatus("shown in the status window");
        }
}
```

OUTPUT



```
Hi this is in the applet window
shown in the status window
```

The HTML APPLET Tag

- The APPLET tag is used to start an applet from both an HTML document and from an applet viewer.

- An applet viewer will execute each APPLET tag that it finds in a separate window, while web browsers like Netscape Navigator, Internet Explorer, and HotJava will allow many applets on a single page.

The syntax for the standard APPLET tag is shown here. Bracketed items are optional.

```
< APPLET
      [CODEBASE = codebaseURL]
       CODE = appletFile
      [ALT = alternateText]
      [NAME = appletInstanceName]
       WIDTH = pixels HEIGHT = pixels
      [ALIGN = alignment]
      [VSPACE = pixels] [HSPACE = pixels]
>

  [< PARAM NAME = AttributeName VALUE = AttributeValue>]
  [< PARAM NAME = AttributeName2 VALUE = AttributeValue>]
  . . .
  [HTML Displayed in the absence of Java]
</APPLET>
```

CODEBASE: CODEBASE is an optional attribute that specifies the base URL of the applet code, which is the directory that will be searched for the applet's executable class file (specified by the CODE tag).

CODE: CODE is a required attribute that gives the name of the file containing your applet's compiled **.class** file. This file is relative to the code base URL of the applet, which is the directory that the HTML file was in or the directory indicated by CODEBASE if set.

ALT The ALT tag is an optional attribute used to specify a short text message that should be displayed if the browser understands the APPLET tag but can't currently run Java applets. This is distinct from the alternate HTML you provide for browsers that don't support applets.

WIDTH AND HEIGHT: WIDTH and HEIGHT are required attributes that give the size (in pixels) of the applet display area.

ALIGN: ALIGN is an optional attribute that specifies the alignment of the applet. This attribute is treated the same as the HTML IMG tag with these possible values: LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, TEXTTOP, ABSMIDDLE, and ABSBOTTOM.

VSPACE AND HSPACE: These attributes are optional. VSPACE specifies the space, in pixels, above and below the applet. HSPACE specifies the space, in pixels, on each side of the applet. They're treated the same as the IMG tag's VSPACE and HSPACE attributes.

PARAM NAME AND VALUE: The PARAM tag allows you to specify appletspecific arguments in an HTML page. Applets access their attributes with the **getParameter( )** method.

Passing parameters to Applets;

- Parameters are passed to applets in **NAME=VALUE pairs in <PARAM> tags** between the opening and closing APPLET tags. Inside the applet, you read the values passed through the PARAM tags with the **getParameter() method** of the java.applet.Applet class.

The program below demonstrates this with a generic string drawing applet. The

applet parameter "Message" is the string to be drawn.

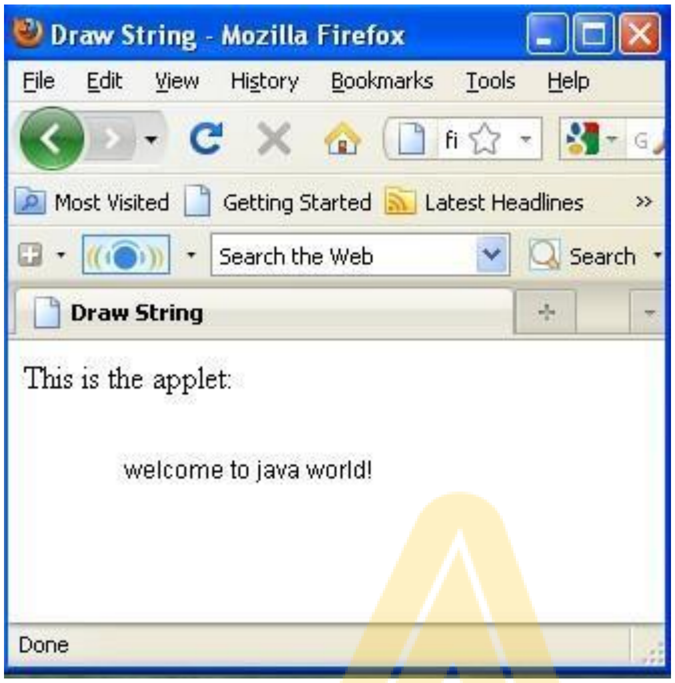
Example:

```
import java.applet.*; import java.awt.*; public
class DrawStringApplet extends Applet
{ private String defaultMessage = "Hello!";

        public void paint(Graphics g)
        {
             String inputFromPage =

             this.getParameter("Message"); if (inputFromPage ==

             null) inputFromPage = defaultMessage;

             g.drawString(inputFromPage, 50, 25);
         } }
```

You also need an HTML file that references your applet. The following simple HTML file will do:

```
 <HTML>
<HEAD>
<TITLE> Draw String </TITLE>
</HEAD>

<BODY>
This is the applet:<P>
<APPLET code="DrawStringApplet" width="300" height="50">
<PARAM name="Message" value="welcome to java world!">
This page will be very boring if your browser doesn't
understand Java.
</APPLET>
</BODY>
</HTML>
```

OUTPUT


Draw String - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines
Search the Web Search
Draw String
Done

This is the applet:

welcome to java world!

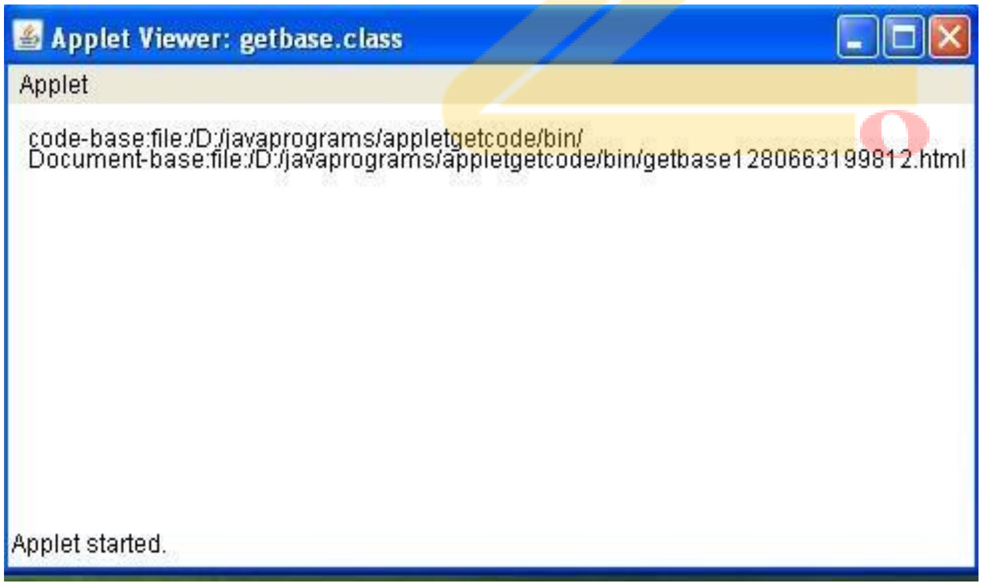
- You pass getParameter() a string that names the parameter you want. This string should match the name of a PARAM element in the HTML page. getParameter() returns the value of the parameter.
  - All values are passed as strings. If you want to get another type like an integer, then you'll need to pass it as a string and convert it to the type you really want.
- The PARAM element is also straightforward. It occurs between <APPLET> and </APPLET>. It has two attributes of its own, NAME and VALUE. NAME identifies which PARAM this is. VALUE is the string value of the PARAM. Both should be enclosed in double quote marks if they contain white space.

getDocumentbase() and getCodebase()

- We sometimes need to load media and Text with the help of Applets. We have the facility to load the data from the directory which holds the HTML file which started the applet and the directory from which the applet's class loaded. These directories are returned in the form of URL by **getDocumnetBase( ) and getCodeBase( ) methods.**

```
import java.awt.*; import
java.applet.*; import java.net.*; public
class getbase extends Applet
{
        public void paint(Graphics g)
        {
                String message;
           URL url=getCodeBase(); message="code-
                base:"+url.toString();
                g.drawString(message,10,20);
                url=getDocumentBase();
                message="Document-base:"+url.toString();
                g.drawString(message,10,30);

        }
}
```

OUTPUT



```
code-base:file:/D:/javaprograms/appletgetcode/bin/
Document-base:file:/D:/javaprograms/appletgetcode/bin/getbase1280663199812.html
```

Applet started.

AppletContext and showDocument()

- AppletContext is an interface which helps us to get the required information from the environment in which the applet is running and getting executed.
- This information is derived by **getAppletContext( )** method which is defined by Applet. Once we get the information with the above mentioned method, we can easily bring another document into view by calling **showDocument( )** method. The basic functionality of this method is that it returns no value and never throw any exception even if it fails hence needed to be implemented with utmost care and caution.

There are two showDocument( ) methods.

1. The **method showDocument(URL)** displays the document at the specified URL.
2. The **method showDocument(URL, where)** displays the specified document at the specified location within the browser window.

```
import  java.awt.*;  import  java.applet.*;
import      java.net.*;      public      class
contextdoc extends Applet
{
        public void start()
        {
                AppletContext ac=getAppletContext();
                URL url=getCodeBase();
                try
                { ac.showDocument(new URL(url+"demo.html"));
                }
                catch(MalformedURLException e)
                { showStatus("URL not found");

                }

        }
}
```

The AudioClip Interface:

- The AudioClip interface is a simple abstraction for playing a sound clip.

Multiple AudioClip items can be playing at the same time, and the resulting sound is mixed together to produce a composite.

It contains three methods:

- Play(): Starts playing this audio clip.
- Stop(): Stops playing this audio clip.
- Loop():Starts playing this audio clip in a loop.

After you have loaded an audio clip using **getAudioClip()**, you can use these methods to play it.

The AppletStub Interface

- The AppletStub interface provides a way to get information from the run-time browser environment.

Output to the Console

- The output to an applet window must be accomplished through GUI based methods such as **drawstring()** as illustrated in the above example applet programs, it is still also possible to use console output in your applet for debugging purpose.
- In an applet program where you call a method such as **System.out.println()** the output is not sent to your applet window instead it appears either in the console session or in the java console that is available in some br **JAVA**

Questions

1. List applet initialization and termination method? Write a java applet that set the background color cyan and foreground color red and output a string message “A simple Applet”?

(Jan 2013) 4 Marks

2. What are applets? Explain the different stages in the life cycle of applet? (Dec 2011)08Marks
3. How to embed applet inside the html page? Explain with an example program.
4. Explain getCodeBase() and getDocumentBase() methods.
5. Write a note on:
    a. showStatus().
    b. AppletContext and showDocument()
    c. AudioClip interface
6. Exaplin HTML Applet Tag attributes
7. With an example program explain how to pass parameters to Applet.
8. Explain Applet Skelton.