COMPUTER SCIENCE

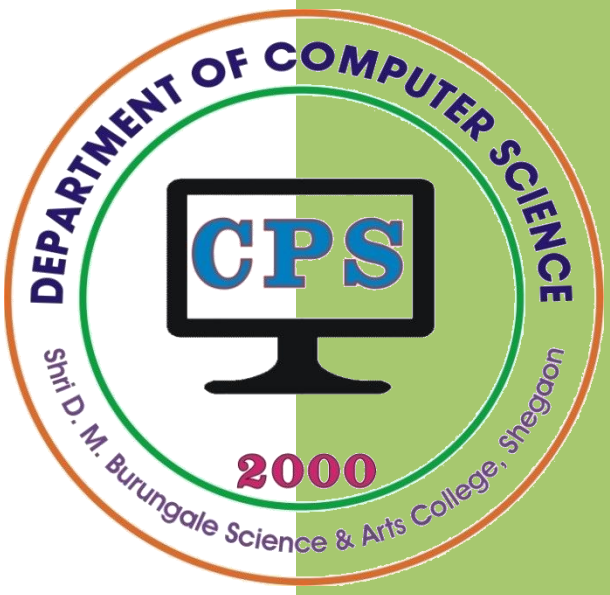
B. Sc. III

Semester-VI

2023-2024

6S : Advanced Java & VB.net

Unit-II : Java Applets



PROF. V. V. AGARKAR

Assistant Professor & Head

Department of Computer Science

Shri. D. M. Burungale Science & Arts College, Shegaon, Dist. Buldana

Unit II

Applet: Introduction to Applet, Applet life cycle, HTML applet tag with all attributes, Running the applet, Passing parameters to applets, Displaying using applet viewer, getDocumentBase() and getCodeBase() methods, Applet context, Applet vs Application, Graphics introduction, Graphic class, draw lines, circle, rectangle, ellipse.

Introduction to Applet:

Applets are small Java programs that can be embedded into a web page and are primarily used in internet computing. They can be transported over the internet from one computer to another and run using the Applet Viewer or any Web browser that supports Java. An Applet, like any application program, can perform arithmetic Operations, display graphics, play sounds, accept user input, create animation, and play interactive games. Applets are used to make the web site more dynamic and entertaining. An Applet is embedded in an HTML page using the `APPLET` or `OBJECT` tag and hosted on a web server.

Java applet has facilitated to create and use fully interactive multimedia web documents. A web page can now contain not only a simple text or a static image but also a Java applet which, when run, can produce graphics, sounds and moving images.

User can embed applets into web pages in two ways. One, user can write his own applets locally, stored in a local system and embeds them into web pages are known as "*local applet*". Second, user can download an applet developed by someone else from a remote computer system and then embed it into a web page is called "*remote applet*". Following are some important points about applets:

1. All applets are sub-classes (either directly or indirectly) of `java.applet.Applet` class.
2. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. `JDK` provides a standard applet viewer tool called `appletviewer`.
3. In general, execution of an applet does not begin at `main()` method.
4. Output of an applet window is not performed by `System.out.println()`. Rather it is handled with various `AWT` methods, such as `drawString()`.

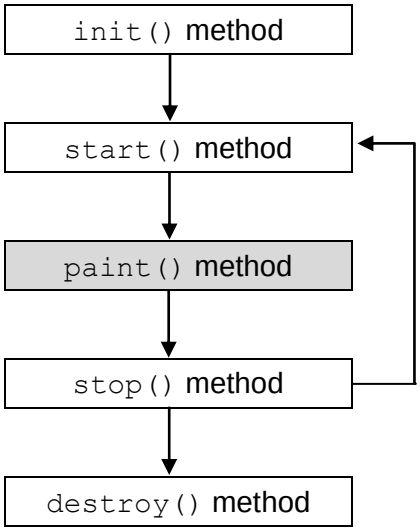
Applet life cycle

Each state of applet life cycle is represented by a method. That is, in its life of execution, the applet exists (lives) in one of the 5 states. These methods are known as "*callback methods*" as they are called automatically by the browser whenever required. Programmer just writes the methods with some code but never calls.

1. Applet is initialized.
2. Applet is started.
3. Applet is painted.
4. Applet is stopped.
5. Applet is destroyed.

DVB College, Shegaon

These methods are known as applet life cycle methods. They are defined in `java.applet.Applet` class except `paint()` method. The `paint()` method is defined in `java.awt.Component` class, an indirect super class of Applet.



[Fig.1 : An applet's Life Cycle]

`init()` method

The **`init()`** method is the first method in the life cycle of applet. When either the applet loaded into memory or the web page containing applet is opened by user for first time, the `init()` method is called. This method is often used to initialize variables, load images or sound files, set up the screen, get parameters out of the HTML file, and create objects the applet will need later. This method is called **only once** during the run time of the applet.

`start()` method

The `start()` method is called just after `init()` method. It is also called to restart an applet after it has been stopped. The `start()` method is called every time the browser displays the web page containing the applet. The `start()` usually does the "work" of the applet. It often starts threads, plays sound files, or does computation. Also, if a user leaves a web page and comes back, the applet resumes execution at `start()`.

`stop()` method

The `stop()` method stops the execution of the applet. The `stop()` method is called whenever the browser leaves the web page containing the applet. It should stop or suspend anything that the applet is doing. In this method the applet becomes temporarily inactive. An applet can come any number of times into this method in its life cycle and can go back to the active state whenever would like.

`destroy()` method

The `destroy()` method executes when the applet window is closed or when the tab containing the webpage is closed. The `stop()` method executes just before when `destroy()` method is invoked. The `destroy()` method removes the applet object from memory. This is the end of the life cycle of applet. This method is also called **only once** during the run time of the applet.

DMB College, Shegaon

From among above four methods the `init()` and the `destroy()` methods are executed exactly once, whereas, the `start()` and the `stop()` methods can be executed any number of times.

In addition to these four methods, java also has provided a method that is very handy while working with the applet. This method is:

`paint()` method

The `paint()` method is used to redraw the output on the applet display area. This method invoked immediately after the `start()` method, and also any time the applet needs to repaint itself in the browser. This method takes a `java.awt.Graphics` object as parameter. This is the place where the programmer can write his code of what he expects from applet like animation etc. This method is not a part of life cycle.

HTML `applet` tag with all attributes

The `applet` tag in HTML is used to *embed Java applets into any HTML document*. 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 will allow many applets on a single page. The `applet` tag always appears in `BODY` tag of an HTML document. Three attributes are required in the `APPLET` tag. Two of these attributes, `width` and `height`, specify the space the applet occupies on the screen. The third required attribute is `code`. The `code` attribute specifies the class file from which the applet is loaded. The general syntax of the `applet` tag is given below:

```
< 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>]
   . . .
</APPLET>
```

The `applet` tag was deprecated in HTML 4.0, and it's support has been completely discontinued starting from HTML 5. Alternatives available in HTML 5 are the `<embed>` and the `<object>` tags. It contains attributes that identify the applet to be displayed and, optionally, give the web browser hints about how it should be displayed.

Example :

```
<APPLET
CODE = "HelloJava.class"
 WIDTH = 400
  HEIGHT = 200>
</APPLET>
```

DMB College, Shegaon

The following Table-1 shows the attributes that are specific to the applet tag.

[**TABLE-1:** APPLET TAG ATTRIBUTES WITH DESCRIPTION]

Attribute	Value	Description
`code`	*URL*	`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.
`width`	*length*	Specifies the initial width of the applet's display area.
`height`	*length*	Specifies the initial height of the applet's display area.
`align`	*left right top middle bottom*	Specifies the alignment of <applet> element with respect to the surrounding content.
`alt`	*text*	Provides alternate text if the applet cannot be displayed.
`archive`	*URL*	Specifies a *comma-separated* list of URLs for archives containing classes and other resources that will be "preloaded".
`codebase`	*URL*	Specifies the URL of the directory where applets' .class files referenced by the code attribute are stored.
`hspace`	*pixels*	Specifies the amount of additional whitespace to be reserved on left and right side of the applet.
`name`	*text*	Specifies the name of the applet. Applets must be named in order for other applets on the same page to find them by name and communicate with them.
`vspace`	*pixels*	Specifies the amount of additional whitespace to be reserved on top and bottom side of the applet.

Running the applet

Java Applets are compiled by `javac` command. It cannot be run using `java` command. It can be run by using one of the following tools:

1. Java-enabled web browser (such as HotJava or Netscape)
2. Java `appletviewer`

1) Using Web Browser:

To view the applet in a web browser, a java enabled web browser is required. Here the entire web page containing applet is displayed. The following steps should be followed:

a) Compile the applet source code using `javac`.

b) Create an HTML file containing the `APPLET` tag, which have the compiled `.class` file as value to the argument `code`.

c) Open the HTML file in the web browser.

DMB College, Shegaon

2) Using appletviewer:

`Appletviewer` is a java tool provided to view applets. It is like a small browser provided to have a preview of applet as they would look in browser. The `APPLET` tag should be written in source code java program enclosing in comments. The following steps should be followed:

a) Write `<APPLET>` tag enclosed by comments in the source file of java.

b) Compile the applet source code using `javac`.

c) Use `appletviewer ClassName.java` to view the applet
(*or*
Use `appletviewer filename.html` to view the applet.)

Passing parameters to applets

Parameters specify extra information that can be passed to an applet from the HTML page. Java allows users to pass user-defined parameters to an applet with the help of `<PARAM>` tags. The `<PARAM>` tag has a `NAME` attribute which defines the name of the parameter and a `VALUE` attribute which specifies the value of the parameter. In the applet source code, the applet can refer to the parameter by its name to find its value. Inside the applet, you read the values passed through the `PARAM` tags with the `getParameter()` method of the `java.applet.Applet` class.

- **`PARAM` Tag**

The `<param>` tag is a sub tag of the `<applet>` tag. The `<param>` tag contains two attributes: `name` and `value` which are used to specify the name of the parameter and the value of the parameter respectively. The syntax of the `<param>` tag is:

```
<PARAM NAME = AttributeName VALUE = AttributeValue>
```

Example:

The param tags for passing name and age parameters looks as shown below:

```
<param name= ”Name” value= ”Smith”>
<param name= ”Age” value= ”25”>
```

- **The `getParameter()` method**

To retrieve a parameter's value, the `getParameter()` method of `applet` class is used. The syntax of it is:

```
public String getParameter(String name)
```

Where, the method takes a `String` argument `name`, which represents the name of the parameter which was specified with the `param` attribute in the `<applet>` tag and it returns the value of the name parameter (if it was defined) else null is returned.

Example:

```
name = getParameter(”Name”);
age  = getParameter(”Age”);
```

DMB College, Shegaon

The `getDocumentBase()` and `getCodeBase()` methods

In most of the applets, it is required to load text and images explicitly. Java enables loading data from two directories. The first one is the directory which contains the HTML file that started the applet (known as the *document base*). The other one is the directory from which the `class` file of the applet is loaded (known as the *code base*). These directories can be obtained as URL objects by using `getDocumentBase()` and `getCodeBase()` methods respectively. These URL objects can be concatenated with the string representing the name of the file that is to be loaded.

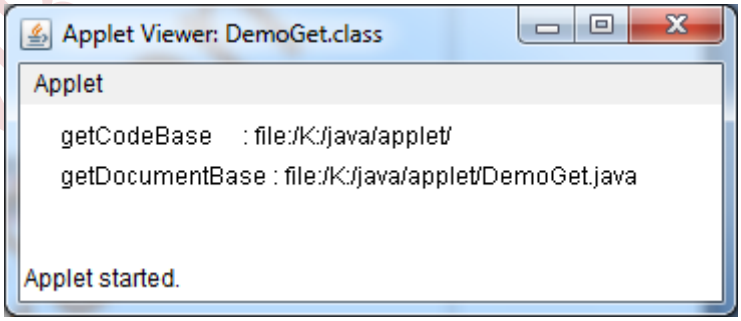
The `getDocumentBase()` method returns the complete URL of the ***.html file*** that loaded the applet. This method can also be used with the `getImage()` or `getAudioClip()` methods.

The **`getCodeBase()`** method returns the complete URL of the ***.class file*** that contains the applet. This method can also be used with the `getImage()` or `getAudioClip()` methods.

Example: Filename : `DemoGet.java`

```
import java.applet.Applet;
import java.awt.Graphics;
public class DemoGet extends Applet
{
    public void paint(Graphics g)
    {
       g.drawString("getCodeBase   : "+getCodeBase(), 20, 20);
       g.drawString("getDocumentBase:"+getDocumentBase(),20,40);
    }
}
/*
    <applet code="DemoGet.class" width=350 height=250>
    </applet>
*/
```

Output:



Applet context

Sometimes it is important to finding out information about the environment in which the applet resides. An applet runs inside a browser or the applet viewer. An applet can ask the browser to do things for it, for example, fetch an audio clip, show a short message in the status line, or show a different web page. The ambient browser can carry out these requests, or it can ignore them. In addition to the `Applet` class, `java.applet` also defines interface `AppletContext` that provide addition support for applet. The

DMR College, Shegaon

`AppletContext` interface is thought as a communication path between the applet and the ambient browser. The `AppletContext` provides the facility of communication between applet and browser and between two or more applets within same environment.

The `AppletContext` is the context (browser/ applet viewer) in which the applet runs. This enables your applet to access features of the browser that contains it. The `AppletContext` is an interface that lets you get information from the applet's execution environment. To communicate with the browser, an applet calls the `getAppletContext()` method.

The `AppletContext` interface contains several useful methods. Some of the frequently used methods are as follows:

Method	Description
`getAppletContext()`	method returns an object that implements an interface of type AppletContext
`Applet getApplet(String name)`	Returns the applet in the current context with the specified name (null if none exists)
`void showDocument(URL url)`	Shows a new web page in the browser, displacing the current page.
`Image getImage(URL url)`	Returns an image object that encapsulates the image specified by the URL
`AudioClip getAudioClip(URL url)`	Returns an AudioClip object that encapsulates the sound file specified by the URL.

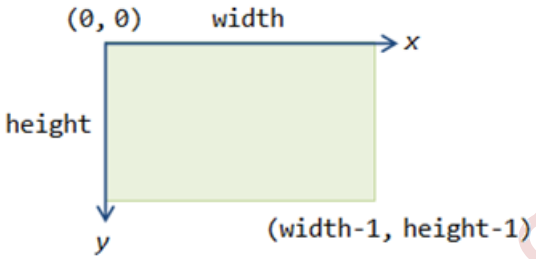
Java Application *vs* Java Applet

Java Application	Java Applet
Java Applications are the stand-alone programs which can be executed independently.	Java Applets are small Java programs which are designed to exist within HTML web document.
Java Applications must have `main()` method for the execution.	Java Applets do not need `main()` for execution.
Java Applications entry point is `main()` method.	Java applet entry point is `init()` method.
Java Applications do not need to extend any class unless required	Java Applets must extend java.applet.Applet class
Java Applications can execute codes from the local system	Java Applets Applications cannot do so
Java Applications has access to all the resources available in your system	Java Applets has access only to the browser-specific services

DMB College, Shegaon

Graphics Introduction

One of the most important features of Java is its ability to draw graphics. We can write Java applets that draw lines, figures of different shapes, images, and text in different fonts and styles. Every applet has its own area of the screen known as *canvas*, where it creates its display. The size of an applet's space is decided by the attributes of the `applet` tag. A Java applet draws graphical image inside its space using the coordinate system. Java's coordinate system has the origin (0, 0) in the upper-left corner. Positive *x* values are to the right, and positive *y* values are to the bottom. The values of coordinates *x* and *y* are in pixels. Java Graphics Coordinates System is shown in Fig. 2.



[Fig. 2 : Java Graphics Coordinates System]

`Graphics` class

Java's **Graphics** class includes methods for drawing many different types of shapes, from simple lines to polygons to text in a variety of fonts. To draw a shape on the screen, one of the methods is to be call that is available in the **Graphics** class. Table-2 shows the most commonly used drawing methods in the **Graphics** class. All the drawing methods have arguments representing end points, corners, or starting locations of a shape as values in the applet's coordinate system. To draw a shape, only need to use the appropriate method with the required arguments.

[**TABLE-2:** DRAWING METHODS OF THE GRAPHICS CLASS]

Method	Description
`clearRect()`	Erases a rectangular area of the canvas.
`copyArea()`	Copies a rectangular area of the canvas to another area.
`drawArc()`	Draw a hollow arc.
`drawLine()`	Draws a straight line.
`drawOval`	Draws a hollow oval.
`drawPolygon()`	Draws a hollow polygon.
`drawRect()`	Draws a hollow rectangle.
`drawRoundRect()`	Draws a hollow rectangle with rounded corners.
`drawstring()`	Displays a text string
`fillArc()`	Draws a filled arc.
`fillOval()`	Draws a filled oval.
`fillPolygon()`	Draws a filed polygon.
`fillRect()`	Draws a filed rectangle.
`fillRoundRect()`	Draws a filled rectangle with rounded corners.
`getColor()`	Retrieves the current drawing color.
`getFont()`	Retrieves the currently used font.

DMB College Shegaon

`getFontMetrics()`	Retrieves information about the current font.
`setColor()`	Sets the drawing color.
`setFont()`	Sets the font.

Drawing lines

The `drawLine()` method of the `Graphics` class is used to draw a straight line with current color between two points (x_1, y_1) and (x_2, y_2). It takes four arguments: the *x* and *y* coordinates of the starting point and the *x* and *y* coordinates of the ending point. The syntax is:

```
void drawLine(int x1, int y1, int x2, int y2)
```

Where, the four arguments

- **`x1`** – It takes the first point's *x* coordinate.
- **`y1`** – It takes first point's *y* coordinate.
- **`x2`** – It takes second point's *x* coordinate.
- **`y2`** – It takes second point's *y* coordinate

Example :

The following `MyLine` class draws a line from the point (20, 100) to (200, 100). Note that the `drawLine()` method is defined in the `Graphics` class.

```
// Draw line
import java.awt.*;
import java.applet.*;
public class MyLine extends Applet
{
      public void paint(Graphics g)
      {
            g.drawLine(20, 100, 200, 100);
      }
}
/*
<applet code="MyLine" width=300 height=200>
</applet>
*/
```

Drawing rectangle

Java `Graphics` class supports four types of drawing rectangles.

1. Right-angled hollow rectangle
2. Right-angled filled rectangle
3. Round-cornered hollow rectangles
4. Round-cornered filled rectangles

- `drawRect`

The `drawRect()` method is used to draw a hollow (outlined) rectangle. This method takes four arguments. The first two represent the *x* and *y* coordinates of the top-left corner of the rectangle, and the remaining two represent the *width* and the *height* of the rectangle. The syntax is:

DMB College, Shegaon

```
void drawRect(int x, int y, int width, int height)
```

- `fillRect`

  The `fillRect()` method is used to draw a filled (solid box) rectangle. This method takes four arguments. The first two represent the *x* and *y* coordinates of the top-left corner of the rectangle, and the remaining two represent the *width* and the *height* of the rectangle. The syntax is:

```
void fillRect(int x, int y, int width, int height)
```

- `drawRoundRect`

  Also a rounded rectangle (rectangles with rounded edges) can be drawn in java. The `drawRoundRect()` method is used to draw a hollow (outlined) rounded rectangle. This method is similar to `drawRect()` except that it can take two extra arguments representing the width and height of the angle of corners. These extra parameters indicate how much of corners will be rounded. The syntax is:

```
drawRoundRect(int x,int y,int width,int height,int arcWidth,int arcHeight)
```

- `fillRoundRect`

  The `fillRoundRect()` method is used to draw a filled (solid) rounded rectangle. This method is similar to `fillRect()` except that it can take two extra arguments representing the width and height of the angle of corners. These extra parameters indicate how much of corners will be rounded. The syntax is:

```
fillRoundRect(int x,int y,int width,int height,int arcWidth,int arcHeight)
```

Example :

The following `AppRect` class draws four rectangles that illustrates all four types of rectangles.

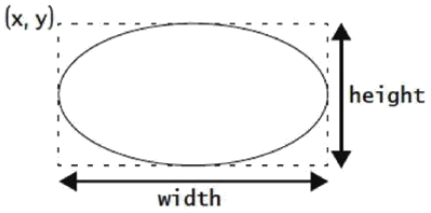
```
// Java Applet for 4 types of Reactangles
import java.awt.*;
import java.applet.*;
public class AppRect extends Applet
{
    public void paint(Graphics g)
    {
        g.drawRect(10, 10, 70, 70);
        g.fillRect(100, 10, 70, 70);
        g.drawRoundRect(10, 100, 70, 70,15,15);
        g.fillRoundRect(100, 100, 70, 70,15,15);
    }
}

/*
  <applet code="AppRect.class" width=400 height=200>
  </applet>
*/
```

Drawing circle and ellipse

The `Graphics` class does not have any method for circles or ellipses. However, the `drawOval()` method can be used to draw a circle or an ellipse. Ovals are just like

DMB College, Shegaon

imaginary rectangles with overly rounded corners as shown in Fig. 3. The `drawOval()` method takes four arguments: the first two represent the top-left corner of the imaginary rectangle and the other two represent the width and height of the oval itself. Note that if the width and height are the same, the oval becomes a circle. The oval's coordinates are actually the coordinates of an enclosing rectangle.



[Fig. 3 : Oval within an imaginary rectangle]

- **The `drawOval()` method**

The syntax is:

```
drawOval(int x, int y, int width, int height)
```

This draws a hallow oval that fits within the rectangle specified by the *x*, *y*, *width* and *height* arguments. The oval is drawn inside a rectangle whose upper left hand corner is at *(x, y)*, and whose *width* and *height* are as specified.

- **The `fillOval()` method**

The syntax is:

```
fillOval(int x, int y, int width, int height)
```

This draws a filled oval that fits within the rectangle specified by the *x*, *y*, *width* and *height* arguments. The oval is drawn inside a rectangle whose upper left hand corner is at *(x, y)*, and whose *width* and *height* are as specified.

Example :

```
// Java applet to draw Circle and Ellipse
import java.awt.*;
import java.applet.*;

public class AppOval extends Applet
{
      public void paint(Graphics g)
     {
     g.drawOval(10, 10, 50, 50);
     g.fillOval(100, 10, 75, 75);
     g.drawOval(190, 10, 90, 30);
     g.fillOval(70, 90, 140, 100);
      }
}

/*
  <applet code="AppOval" width=300 height=200>
  </applet>
*/
```

DMB College, Shegaon

The `drawString()` method

To display the text in applet the `drawString()` method is used. The syntax is:

```
void drawString(String str, int x, int y)
```

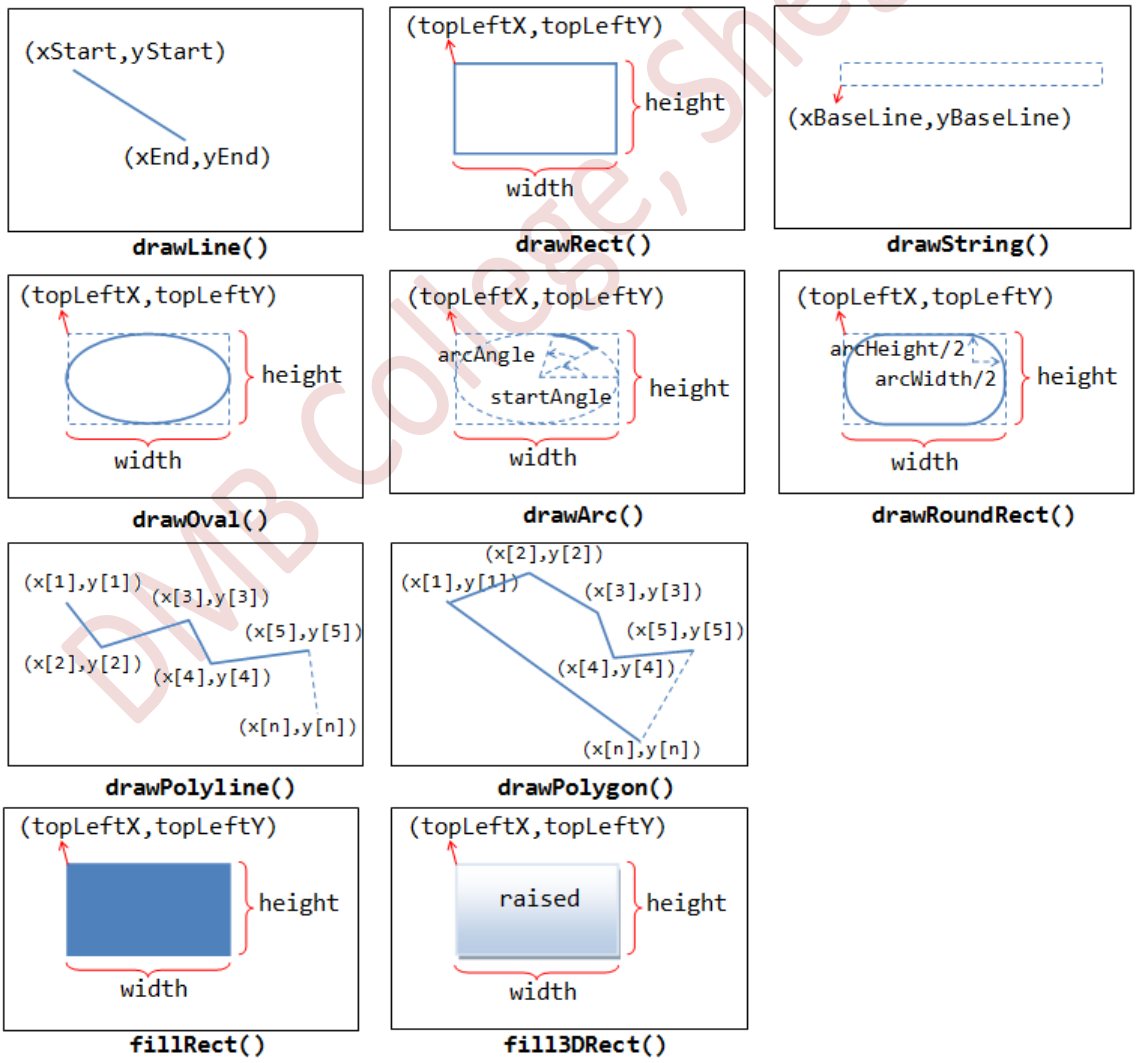
The `drawString()` method takes three arguments,

`string` : takes the string to be displayed.
`x` : takes the x co-ordinates where the string will be displayed on the screen.
`y` : takes the y co-ordinates where the string will be displayed on the screen.

The method draw string given as first argument on screen at the position (x,y) provided by second and third arguments.

Example :

```
public void paint(Graphics g)
{
      g.drawString("welcome",100,50);
}
```



■ ■ ■ ■ ■

DMB College, Shegaon

Sant Gadge Baba Amravati University, Amravati
B. Sc. Part THREE (Semester – VI) Examination
Questions Asked in Previous University Exams

• Summer-2022 (AY-2272)

4. A) Explain the following methods :
   (i) get DocumentBase( )
   (ii) get CodeBase( ) 6
   B) How to display using applet viewer? Explain the process. 6

OR

5. A) Write an applet program to draw multiple lines. 6
   B) Explain the applet life cycle in detail. 6

• Winter 2022 (AC-2131)

4. A) Explain the HTML APPLET tag with its attributes. 6
   B) Write a program to draw polygon. 6

OR

5. A) Explain :
   (i) drawRect( )
   (ii) drawLine( )
   (iii) fillOval( ) 6
   B) How to pass parameters to applets? Explain in detail. 6

• Summer 2023 (AD-1909)

4. A) Explain life cycle of an applet. 6
   B) What is applet context? Give the difference between Java Application & Applet. 6

OR

5. A) What is Graphics class? Explain. 6
   B) How to pass parameters to an applet? Explain. 6

• Winter 2023 (AE-1827)

4. A) What is Applet? Explain the applet life cycle in detail. 6
   B) Write a program to draw Rectangles using Java Applet and explain its graphics methods. 6

OR

5. A) Explain the steps of Applet initialization and termination. 6
   B) Explain the following with its arguments :
   (i) fillArc( ) (ii) fillOval( ) (iii) roundRect( )
   (iv) drawLine( ) (v) drawOval( ) (vi) drawCircle( ) 6

■ ■ ■ ■ ■

DMB College, Shegaon