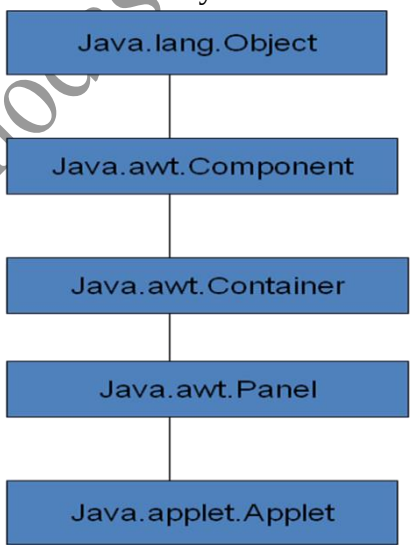
4 Applets, Layout Manager

Topics Covered

1. Introduction to Applet
2. Applet Lifecycle
3. Implement and Executing Applet with parameters
4. Graphics class
5. FlowLayout
6. BorderLayout
7. GridLayout
8. GridBagLayout
9. CardLayout
10. BoxLayout
11. SpringLayout
12. NoLayout

Introduction to Applet

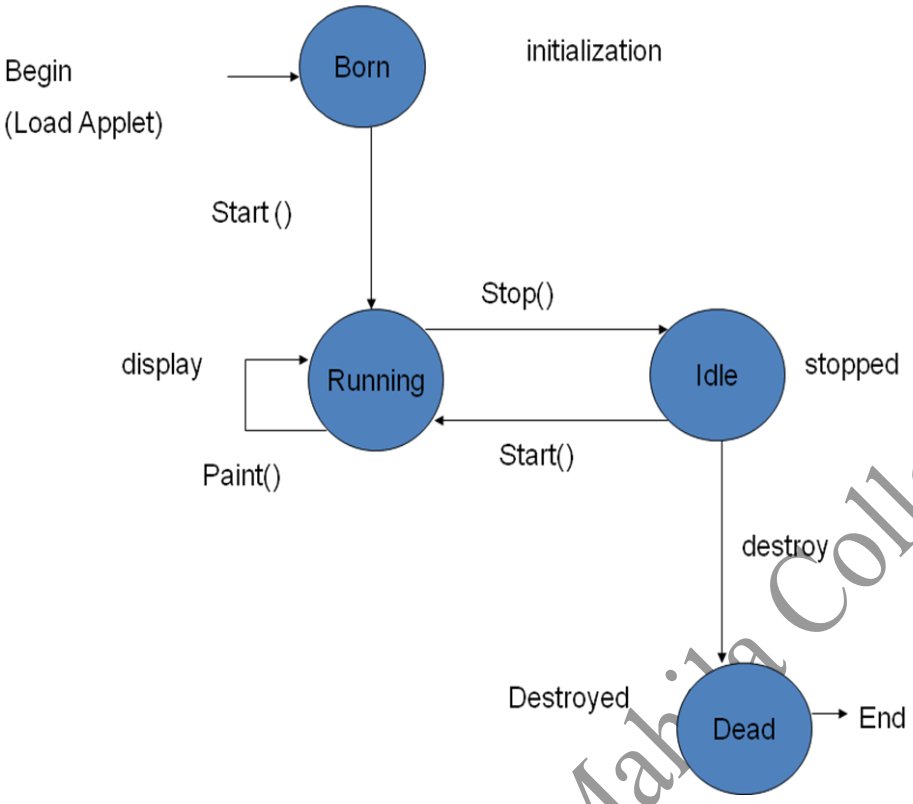
- Applet is a one type java program.
- The Applet class is contained in the java.applet package.
- All applets are subclasses of Applet.
- Applet is a java program and it runs in the viewer and browser.
- Here, following is the Applet class hierarchy.



Applet Life Cycle

- Java applet inherits a set of default behaviours from the Applet class.
- Applet has the following state.

  1)Born state
  2)Running state
  3)Idle and stopped state
  4)Dead State



1)Initialization State

- Applet enters the initialization state when it is first loaded.
- This is achieved by calling the init() method of Applet class. The applet is born.
- At this stage, we may do the following, if required.
  - Create objects needed by the applet
  - Setup initial values
  - Load images or fonts
  - Set up colors
- The initialization occurs only once in the applet's life cycle.
- We must declare init() method in the born state.

```
public void init()
{
    ----------
    ----------(Action)
    ----------
}
```

2)Running state

- Applet enters the running state when the system calls the start() method of Applet class.
- This occurs automatically after the applet is initialized.
- For example, we may leave the web page containing the applet temporarily to another page and return back to the page. This again starts the applet running.
- Start() method may be called more than once.

```
public void start()
{
        ------------
        ------------(Action)
        ------------
}
```

3)Idle or Stopped state

- An applet becomes idle when it is stopped from running.
- Stopping occurs automatically when we leave the page containing the currently running applet.
- We can also do so by calling the stop() method explicitly.
- If we use a thread to run the applet, then we use a thread to run the applet, then we must use stop() method to terminate the thread.

```
public void stop()
{
        ------------
        ------------(Action)
        ------------
}
```

4)Dead state

- An applet is said to be dead when it is removed from memory.
- This occurs automatically by invoking the destroy() method when we quit the browser.
- Like initialization , destroying stage occurs only once in the applet life cycle.

```
public void destroy()
{
        ------------
        ------------(Action)
        ------------
}
```

5)Display state

- Applet moves to the display state whenever it has to perform some output operations on the screen.
- This happens immediately after the applet enters into the running state.
- The paint() method is called to complete this task.
- Every applet have a paint method.

```
public void paint(Graphics g)
{
        -------------
        -------------(Display statements)
        -------------
}
```

Applet tag

- The <APPLET….>tag supplies the name of the applet to be loaded and tells the browser how much space the applet requires.
- The ellipsis in the tag <APPLET..> indicates that it contains certain attributes that must specified.
- The <APPLET> tag specifies the minimum requirements to place the hellojava applet on a web page as given below:
  <APPLET code=hellojava.class width=400 height=200>
  </APPLET>
- This HTML code tells the browser to load the compiled java applet hellojava.class, which is in the same directory as the HTML file.

Passing parameters to Applet

- We can supply user-defined parameters to an applet using <PARAM> tag.
- Each <PARAM> tag has a name attribute such as color, and a value attribute such as red.
- Inside the applet code, the applet can refer to that parameter by name to find its value.
- For example, we can change the color of the text displayed to red by an applet by using a <param..> tag as follow:

```
<applet…..>
<param name=color value=red>
</applet>
```

- Passing parameters to an applet code using <param> tag is something similar to passing parameters to the main function using command line arguments.
- To setup and handle parameters,we need to do two things
  1)Include appropriate <param…>tags in the HTML document.
  2)Provide code in the applet to pass these parameters.
- Parameters are passed on an applet when it is loaded. We can define init() method in theapplet to get hold of the parameters defined in the <param> tag.
- This is done using the getParameter() method which takes one string argument representing the name of the parameter and returns a string containing the value of a parameter.

Graphics class

- The graphics class includes methods for drawing many different types of shapes.
- To draw a shape on the screen, we may call one of the methods available in the graphics class.
- All drawing methods have arguments representing starting point, ending point and corners.

Methods	Description
clearRect()	Erases a rectangular area of the canvas
copyArea()	Copies a rectangular area of the canvas to another area.
drawArc()	Draws a arc
drawLine()	Draws a Line.
drawOval()	Draws an Oval

drawPolygon()	Draws a Polygon
drawRect()	Draws a rectangle.
drawRoundRect()	Draws a round rectangle.
drawString()	Displays a string
fillArc()	Draws a filled Arc.
fillOval()	Draws a filled Oval.
fillPolygon()	Draws a filled Polygon.
fillRect()	Draws a filled Rectangle.
fillRoundRect()	Draws a filled Round Rectangle.
getColor()	Retrieves the currently drawing color.
getFont()	Retrieves the currently used font.
setColor()	Sets the drawing color.
setFont()	Sets the font.

Line Method

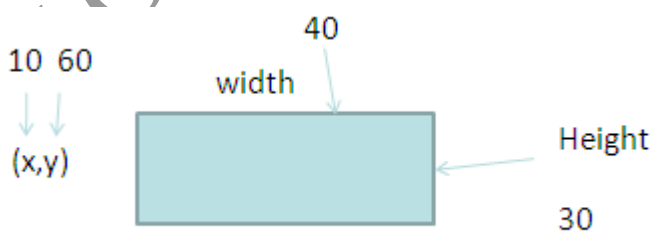
- The simplest shape we can draw with the graphics class is a line.
- The drawLine () methods takes two pairs of coordinates (x1 , y1) and (x2 , y2) as arguments and draws a line between them.
- Ex

  g.drawLine (10,10,50,50)
- The g is the graphics object passing to the paint () method

Rectangle Method

- We can draw a rectangle using the drawRect () method.
- This method take four arguments
- The first two represents the x and y coordinates on the top left corner of the rectangle ,and the remaining two represent the width and the height of the rectangle.

  g.drawRect(10,60,40,30);



- We can draw a solid box by using the method fillRect ()
- We can also draw rounded rectangles using the methods drawRoundRect ()and fillRoundRect ().
- These function takes extra two arguments which represent width and height of the angle.

  g.drawRoundRect (10,100,80,50,10,10)

Circles and Ellipses

- The graphics class does not have any method for circles or ellipses.
- The drawOval () method can be use to draw a circle or an ellipse.

- The drawOval methods take four arguments.
- The first two represent the top left corner of the oval and the other two represent the width and height of the oval.

Arcs

- An arc is a part of an oval
- An oval is series of arcs that are connected together in an orderly manner
- The drawArc() is used to draw Arcs which takes six arguments.
- The first four are the same as the arguments of the oval and the last two represent the srarting angle of the arc and the number of degrees around the arc.
- Java considers the three O'Clock position as zero degree position and degrees increase in anti-clock wise direction.

Polygons

- Polygons are shapes with many sides.
- A polygon may be considered a set of lines connected together.
- The end of the first line is the beginning of the second line,the and of the second is the beginning of the third and so on.
- We can draw polygon more conveniently using the drawPolygon () method.
- This method takes three arguments
  1)An array of integers containing x coordinates.
  2)An array of integers containing y coordinates.
  3)An integer for the total number of points.

Flow Layout

- It is default layout manager.
- Components are laid out from the upper left corner,left to right and top to bottom.
- When no more components fit on a line, the next one appears on a new line.
  - FlowLayout()
  - FlowLayout(int align)
  - FlowLayout(int align,int h gap,int v gap)
- Here align is the alignment left, right and center.
- H gap is the horizonatal distance and v gap is the vertical distance.

Border Layout

- This class implements a common layout style for top-level windows.
- It has four narrow, fixed width components at the edges and one large area in the center.
- The four side are referred as north, south, east and west.
- The middle area is called the center.
  - BorderLayout()
  - BorderLayout(int horz,int vert)
- The horz and vert specify the horizontal and vertical distance between the components.

Card Layout

- The cardlayout has some special capabilities that other layout do not have.
- It creates layout like playing cards.

- Assume that more that one card on one another so that only top card is visible at a time.
- But you can shuffle the cards to see other cards.
- This can be useful for user interface with optional components that can be dynamically enabled and disabled upon user input.
    - CardLayout()
    - CardLayout(int horz,int vert)
- The horz and vert specify the horizontal and vertical distance between the components.

Grid Layout

- This class automatically arranges component in a grid.
- All components are created with equal size.
    - GridLayout()
    - GridLayout(int numRows,int numColumns)
    - GridLayout(int numRows,int numColumns,int horz,int vert)
- The first constructor is default constructor which will create only one column grid.
- Here, rows and columns specify the number of row and columns of the grid.
- The horz and vert specify the horizontal and vertical distance between the components.

GridBag Layout

- It is very flexible layout manager.
- It is an extension of GridLayout.
- In GridLayout all the cells have same height and width which is not necessary in the GridBagLayout.
- This can be done by specifying constraint.
- To specify constraint you have to create an object of GridBagConstraitns.
    - GridBagLayout()
    - GridBagConstraint()

The GridBagConstraint are as follow:
gridx and gridy

- gridx specifies the horizontal position and the gridy specifies the vertical position of the component.

Gridwidth and gridheight

- Th gridwidth specifies the width and the gridheight specifies the height of the cell in number of rows and in number of columns.

Fill

- It specifies what to do if the component size is larger than th size of the cell.

Ipadx and ipady

- It specifies that how many space will remain around the component in the cell.

Insets

- It specifies the spacing between the edges of the container an the components.

BoxLayout

- The BoxLayout is a general purpose layout manager which is an extension of FlowLayout.
- It places the components on top of each other or places them one after another in a row.
- Its constructor is.
    - BoxLayout(Container obj, int placement)
- Here, the obj is the object of a Container such as a panel and the placement can be one of the following:

BoxLayout.PAGE_AXIS : It places the component like a stack i.e. on top of each other.
BoxLayout.LINE_AXIS : It places the component in a line ,i.e. one after another.

SpringLayout

- The SpringLayout is a very flexible layout that has many features for specifying the size of the components.
- You can have different size rows and/or columns in a container in SpringLayout you can define a fixed distance between the left edge of one component and right edge of another component.
- The constructor for SpringLayout is:
    - SpringLayout()
- You can put constraint by the following methods
    - Void putConstraint(String edge1,Component obj1,int distance,String edge2,component obj2)

The edge are
SpringLayout.NORTH
SpringLayout.SOUTH
SpringLayout.EAST
SpringLayout.WEST

NoLayout Manager

- To give the absolute position to the component this layout is used.
- To set the position you have to use setBounds() method.
- To set the NoLayout follow the steps.

1) set the container's layout manger to null(Call setLayout(null))
set the bounds of the component(Call setBounds())