A Presentation

On

Advanced Java Programming

By

Atul S. Chaudhari

M.E.(Computer), C-DAC Pune

Department of Computer Engineering
S. S. V. P. S's B. S. Deore Polytechnic, Dhule

Unit I
Abstract Window Toolkit(AWT)

(12 Marks)

Outline of Topics

1 AWT

2 Window Fundamentals

3 AWT Controls
- Label
- Button
- Checkbox
- Choice
- List
- TextField and TextArea
- Scrollbar

4 Layout Managers
- FlowLayout
- BorderLayout
- GridLayout
- CardLayout
- GridBagLayout

5 MenuBar, Menus and MenuItems

6 Dialog Boxes

7 FileDialog

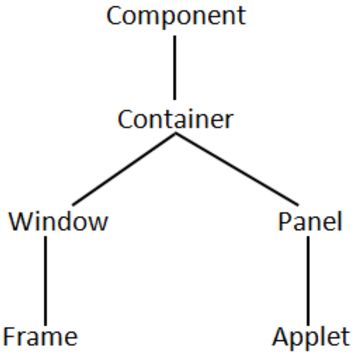
8 References

AWT: Abstract Window Toolkit

- AWT stands for Abstract Window Toolkit
- AWT contains numerous classes and interfaces that allow you to create and manage windows.
- AWT is an API to develop GUI or window-based applications in java
- AWT classes and interface are present in java.awt package

Window Fundamentals

AWT class hierarchy



Component

Container

Window

Panel

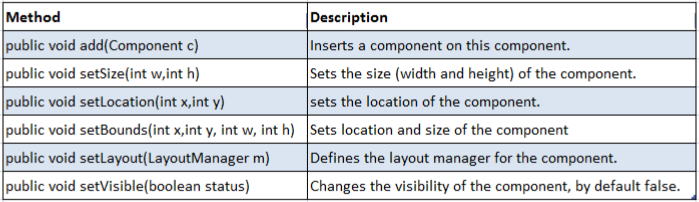
Frame

Applet

Window Fundamentals

Component class

- Component class is present at the top of AWT hierarchy
- Component is an abstract class that encapsulates all the attributes of a visual component
- All user interface elements that are displayed on the screen and that interact with the user are subclasses of Component class
- It defines over a hundred public methods that are responsible for managing events



Method	Description
public void add(Component c)	Inserts a component on this component.
public void setSize(int w,int h)	Sets the size (width and height) of the component.
public void setLocation(int x,int y)	sets the location of the component.
public void setBounds(int x,int y, int w, int h)	Sets location and size of the component
public void setLayout(LayoutManager m)	Defines the layout manager for the component.
public void setVisible(boolean status)	Changes the visibility of the component, by default false.

Window Fundamentals

Container class

- Container class is a subclass of Component class
- The Container is a component that can contain another components like buttons, textfields, labels etc.
- A container is responsible for laying out(i.e. positioning) any components that it contains.

Window Fundamentals

Panel class

- Panel class is a concrete subclass of Container
- It doesnt add any new methods; it simply implements Container
- Panel is the superclass for Applet.
- When screen output is directed to an applet, it is drawn on the surface of a Panel object
- Panel is a window that does not contain a title bar, menu bar, or border.
- Other components can be added on Panel by its add( ) method.
- Once these components have been added, you can position and resize them manually using the methods defined by Component class

Window Fundamentals

Window class

- The Window class creates a top-level window.
- A top-level window is not contained within any other object; it sits directly on the desktop.
- Generally, we dont create Window objects directly. Instead, we use a subclass of Window called Frame.

Window Fundamentals

Frame class

- It is a subclass of Window and has a title bar, menu bar, borders, and resizing corners.
- It can have other components like button, label, textfield etc.
- Frame class is used to create application window(standalone application)

Steps to create GUI using AWT

1. Create window (Applet Window or Application Window)
2. Set the layout of the window
3. Create different AWT controls
4. Add AWT controls on window
5. Set location and size of AWT controls

Creating Applet Window

- To create applet window we need services of two classes
  1. **Applet class:** It provides life cycle methods of an applet (init(), start(), stop(),destroy())
  2. **Graphics class:** It provides different methods to perform output operations on an applet

Creating Applet Window

Steps to develop and execute an applet:

1. Building an applet code (.java file)
2. Creating executable applet (.class file)
3. Design web page containing <applet> tag (.html file)
4. Executing an applet

Creating Application Window

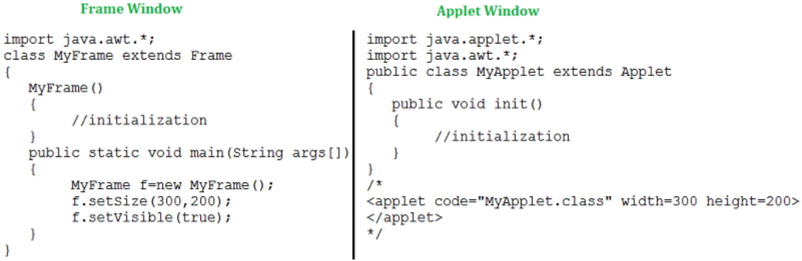
Frame class

- Frame class is used to create Application Window
- There are two ways to create a frame in AWT:
  1. By extending Frame class
  2. By creating the object of Frame class
- **Constructors:**

  Frame( )
  Frame(String title)
- **Methods:**

  void setSize(int w, int h)
  void setVisible(boolean vflag)
  void setTitle(String title)

Frame and Applet Window



Frame Window

```
import java.awt.*;
class MyFrame extends Frame
{
   MyFrame()
   {
        //initialization
   }
   public static void main(String args[])
   {
        MyFrame f=new MyFrame();
        f.setSize(300,200);
        f.setVisible(true);
   }
}
```

Applet Window

```
import java.applet.*;
import java.awt.*;
public class MyApplet extends Applet
{
   public void init()
   {
        //initialization
   }
}
/*
<applet code="MyApplet.class" width=300 height=200>
</applet>
*/
```

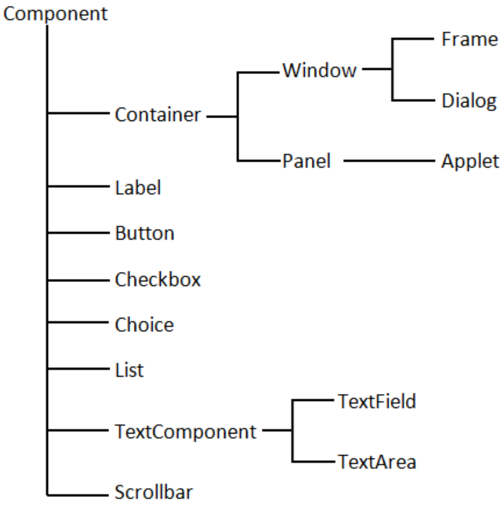
AWT Controls

AWT supports following controls:

- Label
- Button
- Checkbox
- Choice
- List
- TextComponent-TextField and TextArea
- Scrollbar

These controls are subclasses of Component class

AWT Controls



Label

- It is used to display a single line of read only text.
- A label is an object of Label class.
- It contains a string, which it displays.
- Labels are passive controls that do not support any interaction with the user.

Constructors:

Label( )
Label(String str)
Label(String str, int align)

Here, align is constant must be one of these values
Label.LEFT, Label.RIGHT, Label.CENTER

Methods:

void setText(String str)
String getText( )
void setAlignment(int align)
int getAlignment()

Button

- Push button is a component that contains a label.
- Push buttons are objects of Button class.
- Push button is active control that generates an event when it is pressed.

Constructors:

```
Button( )
Button(String str)
```

Methods:

```
void setLabel(String str)
String getLabel( )
```

Checkbox

- A check box is a control that is used to turn an option on or off.
- It consists of a small box that can either contain a check mark or not.
- There is a label associated with each check box that describes what option the box represents.
- You change the state of a check box by clicking on it.
- Check boxes can be used individually or as part of a group.
- Check boxes are objects of the Checkbox class.

Constructors:

```
Checkbox( )
Checkbox(String str)
Checkbox(String str, boolean on)
Checkbox(String str, boolean on, CheckboxGroup cbg)
```

Methods:

```
void setLabel(String str)
String getLabel( )
void setState(boolean on)
boolean getState()
```

CheckboxGroup

- CheckboxGroup class is used to create a set of mutually exclusive check boxes
- Only one check box in the group can be checked at any one time.
- These check boxes are often called as radio buttons.
- To create a set of mutually exclusive check boxes, first define the group to which they will belong and then specify that group when you construct the check boxes.

Constructor:

CheckboxGroup( )

Methods:

Checkbox getSelectedCheckbox( )
void setSelectedCheckbox(Checkbox ch)

Choice

- Choice class is used to create single selection pop-up list of items from which the user may choose
- Choice control is a form of menu.
- When inactive, a Choice component takes up only enough space to show the currently selected item.
- When the user clicks on it, the whole list of choices pops up, and a new selection can be made.
- Each item in the list is a string that appears as a left-justified label in the order it is added to the Choice object.

Constructor:

```
Choice( )
```

Methods:

```
void add(String name)
String getSelectedItem( )
int getSelectedIndex( )
int getItemCount( )
void select(int index)
void select(String name)
String getItem(int index)
```

List

- List class is used to create multiple selection scrollable list of items from which the user may choose
- List object can be constructed to show any number of choices(items) in the visible window.
- It can also be created to allow multiple selections.

Constructors:

```
List( )
List(int rows)
List(int rows, boolean multiselect)
```

Methods:

```
void add(String name)
void add(String name, int index)
String getSelectedItem( )
int getSelectedIndex( )
int getItemCount( )
void select(int index)
String getItem(int index)
String[ ] getSelectedItems( )
int[ ] getSelectedIndexes( )
```

TextField

- TextField class is used to create single-line text-entry area, usually called an edit control.
- Text fields allow the user to enter strings and to edit the text using the arrow keys, cut and paste keys, and mouse selections.
- TextField is a subclass of TextComponent.

Constructors:

```
TextField( )
TextField(String str)
TextField(int numChars)
TextField(String str, int numChars)
```

Methods:

```
String getText( )
void setText(String str)
String getSelectedText( )
void select(int startIndex, int endIndex)
void setEditable(boolean canEdit)
void setEchoChar(char ch)
```

TextArea

- TextArea class is used to create multiline-line text-entry area
- TextArea is a subclass of TextComponent.

Constructors:

```
TextArea( )
TextArea(String str)
TextArea(int numLines, int numChars)
TextArea(String str, int numLines, int numChars)
TextArea(String str, int numLines, int numChars, int sBars)
```

Here, sBars must be one of these values
SCROLLBARS__BOTH, SCROLLBARS__HORIZONTAL__ONLY,
SCROLLBARS__NONE, SCROLLBARS__VERTICAL__ONLY

Methods:

```
String getText( )
void setText(String str)
String getSelectedText( )
void select(int startIndex, int endIndex)
void setEditable(boolean canEdit)
void append(String str)
void insert(String str, int index)
void replaceRange(String str, int startIndex, int endIndex)
```

Scrollbar

- Scroll bars are used to select continuous values between a specified minimum and maximum.
- Scroll bars may be oriented horizontally or vertically.
- Each end of Scroll bar has an arrow.
- The current value of the scroll bar is indicated by the slider box (or thumb).
- If we click on arrows, current value of scrollbar is changed by 1 unit in the direction of arrow(Unit increment)
- If we click on background space which is on either side of thumb then current value of scrollbar is changed by some increment larger than 1 unit (Block increment)
- By default unit increment of scroll bar is 1 unit
- By default block increment(page up or page down increment) of scroll bar is 10 unit

Scrollbar

Constructors:

Scrollbar( )
Scrollbar(int style)
Scrollbar(int style, int initialValue, int thumbSize, int min, int max)

Here, style must be one of two values
Scrollbar.VERTICAL or Scollbar.HORIZONTAL

Methods:

void setValues(int initialValue, int thumbSize, int min, int max)
int getValue( )
void setValue(int v)
int getMinimum( )
int getMaximum( )
void setUnitIncrement(int newIncr)
void setBlockIncrement(int newIncr)

Layout Managers

- Layout manager automatically arranges your controls within a window by using some type of algorithm
- Each Container object has a layout manager associated with it.
- A layout manager is an instance of any class that implements the LayoutManager interface.
- The layout manager is set by using setLayout( ) method
  Syntax:
  void setLayout(LayoutManager obj)
- If you wish to disable the layout manager and position components manually, pass null for obj
- Following are different layout managers:
  1. FlowLayout
  2. BorderLayout
  3. GridLayout
  4. CardLayout
  5. GridBagLayout

FlowLayout

- FlowLayout is the default layout manager for panel and applet.
- It arranges the components similar to how words flow in a text editor.
- It arranges the components starting with top-left corner of window, in the direction left-to-right and top-to-bottom.
- By default horizontal and vertical space between the components is 5 pixels.

Constructors:

```
    FlowLayout( )
    FlowLayout(int how)
    FlowLayout(int how, int horz, int vert)
```

Here, valid values for the parameter **how** are:
FlowLayout.LEFT
FlowLayout.RIGHT
FlowLayout.CENTER

BorderLayout

- BoderLayout is the default layout manager for top level windows (Frame, Dialog).
- It has four narrow, fixed-width components at the edges and one large area in the center.
- The four sides are referred to as north, south, east and west.
- The middle area is called the center.
- BorderLayout defines the following constants that specify the regions:
  1. BorderLayout.NORTH
  2. BorderLayout.EAST
  3. BorderLayout.WEST
  4. BorderLayout.SOUTH
  5. BorderLayout.CENTER
- When we add components, we will use these constants with the following form of add( ), which is defined by Container:
  void add(Component compObj, Object region)

Constructors:

BorderLayout( )
BorderLayout(int horz, int vert)

GridLayout

- GridLayout lays out components in a two-dimensional grid.
- When we instantiate a GridLayout, we have to define the number of rows and columns.

Constructors:

```
    GridLayout( )
    GridLayout(int numRows, int numColumns)
    GridLayout(int numRows, int numColumns, int horz, int vert)
```

CardLayout

- CardLayout is unique among the other layout managers it stores several different layouts.
- Each layout can be thought of as being on a separate index card in a deck that can be shuffled so that any card is on top at a given time.
- This can be useful for user interfaces with optional components that can be dynamically enabled and disabled upon user input.
- You can prepare the other layouts and have them hidden, ready to be activated when needed.
- The deck is typically an object of class Panel.
- The cards that form the deck are also typically objects of type Panel

CardLayout contd...

Steps to create CardLayout:

1. Create the object of panel that act as deck
2. Set layout of this panel(deck) as CardLayout
3. Create object of panel that act as card and add required components on it
4. Repeat step 3 for each card
5. Add all cards on deck using add( ) method having syntax:
   void add(Component panelObj, Object name)
6. Finally add deck panel on window

Constructors:

CardLayout( )
CardLayout(int horz, int vert)

Methods:

void first(Container deck)
void last(Container deck)
void next(Container deck)
void previous(Container deck)
void show(Container deck, String cardName)

MenuBar, Menus and MenuItems

- A top-level window can have a menu bar associated with it.
- To create menu bar in AWT following classes are used:
  1. MenuBar
  2. Menu
  3. MenuItem
- A menu bar contains one or more Menu objects.
- Each Menu object contains a list of MenuItem objects.
- Each MenuItem object represents something that can be selected by the user.
- Since Menu is a subclass of MenuItem, a hierarchy of nested submenus can be created.

MenuBar, Menus and MenuItems contd...

Constructors:

MenuBar:

MenuBar( )

Menu:

Menu( )
Menu(String optionName)
Menu(String optionName, boolean removable)

MenuItem:

MenuItem( )
MenuItem(String itemName)
MenuItem(String itemName, MenuShortcut keyAccel)

CheckboxMenuItem

- It is also possible to include checkable menu items.
- CheckboxMenuItem class is used to create checkable menu items.
- It is subclass of MenuItem class.

Constructors:

CheckboxMenuItem( )
CheckboxMenuItem(String itemName)
CheckboxMenuItem(String itemName, boolean on)

Methods:

boolean getState( )
void setState(boolean checked)

Dialog Boxes

- Dialog boxes are primarily used to obtain user input.
- They are always child windows of a top-level window.
- They are similar to frame window except that they don't have menubar
- There are two types of dialog boxes:
  1. Modal dialog Box
  2. Modeless dialog Box

1. **Modal dialog Box:** When a modal dialog box is active, all input is directed to it until it is closed. Means we cannot access other parts of your program until you have closed the modal dialog box.
2. **Modeless dialog Box:** When a modeless dialog box is active, input focus can be directed to another window in your program. Thus, other parts of your program remain active and accessible.

Constructors:

Dialog(Frame parentWindow, boolean mode)
Dialog(Frame parentWindow, String title, boolean mode)

FileDialog

- Java provides a built-in dialog box that lets the user specify a file.
- To create a file dialog box, create an object of class FileDialog.
- Actually it is the standard file dialog box provided by the operating system.

Constructors:

```
    FileDialog(Frame parent)
    FileDialog(Frame parent, String boxName)
    FileDialog(Frame parent, String boxName, int how)
```

Here, the parameter **how** has one of two values:
FileDialog.LOAD
FileDialog.SAVE

Methods:

```
    String getDirectory( )
    String getFile( )
```

References

H. Schildt, *Complete Reference Java*. McGrawHill Education, New Delhi.

Holzner and Steven, *Java 2 Programming Black Book*. Dreamtech Press, New Delhi.

K. L. Solutions, *Java Sever Programming Tutorial Java EE6 Black Book*. Dreamtech Press, New Delhi.

"Website." [Online]. Available: https://www.javatpoint.com/java-tutorial

"Website." [Online]. Available: https://www.tutorialspoint.com

Thank You...