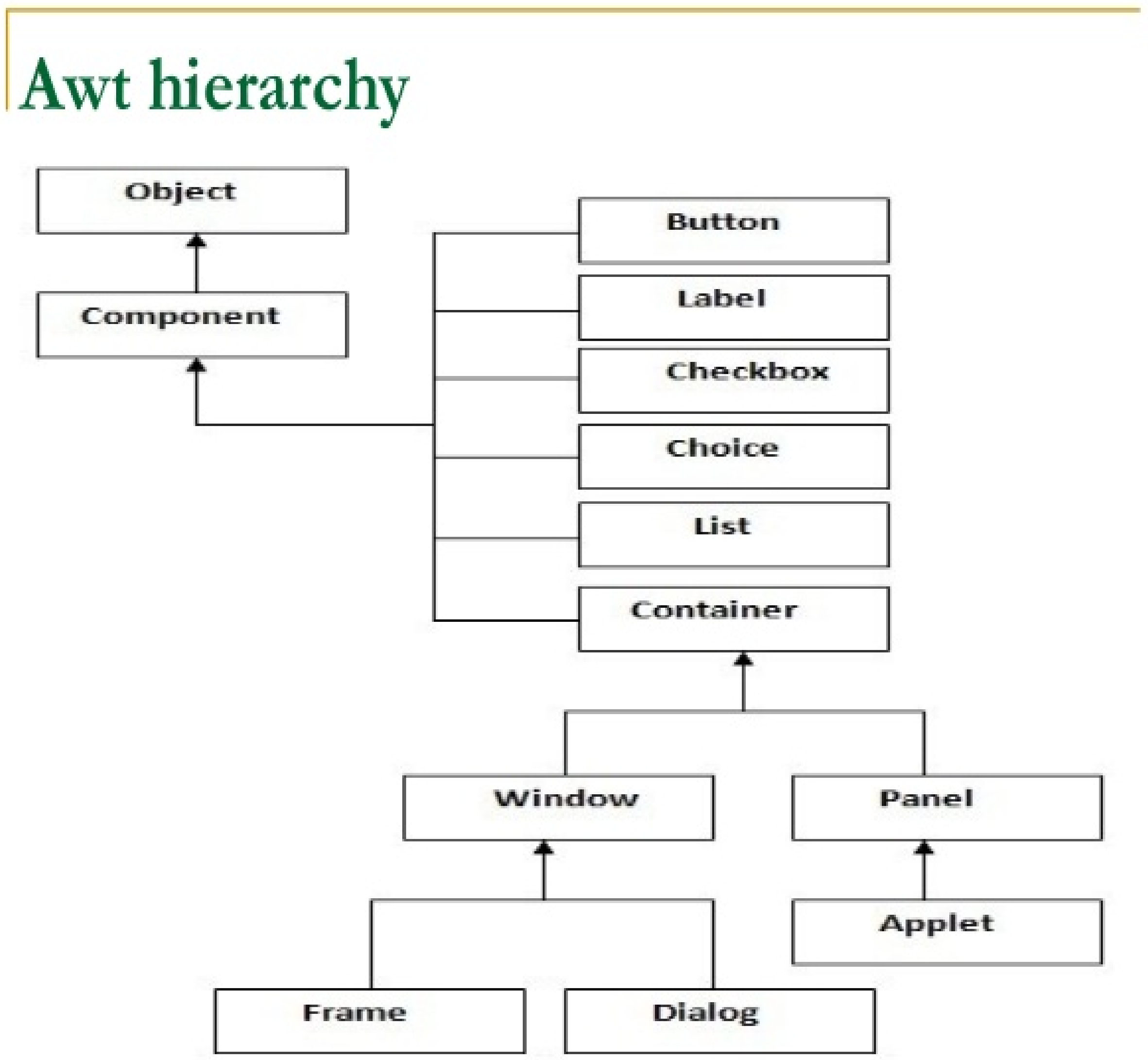
ABSTRACT WINDOW TOOLKIT

- AWT contains various classes and methods that allow you to create and manage windows.
- The AWT contains the foundation for Swing



AWT Classes:

Button	Create a push button control
Checkbox	Creates a check box control
CheckboxGroup	Creates a group of checkboxes
Choice	Creates a pop-up list
Color	Manages colors in a portable platform-independentfashion
Component	An abstract superclass for various AWT components
Container	A subclass of Component that can hold other components
Dialog	Creates a dialog window
Font	Encapsulates a type font
Frame	Creates a standard window that has a title bar, resize corners and a menu bar.
Graphics	Encapsulates the graphics context
GridLayout	The grid layout manager,displays components in a two dimensional grid
Label	Creates a label that displays a string
List	Creates a list from which the user can choose
Menu	Creates a pulldown menu
MenuBar	Creates a menubar
Panel	Simplest concrete subclass of Container
Scrollbar	Creates a scrollbar control
TextArea	Creates a multiline edit control

TextComponent	A super class of TextArea and TextField
TextField	Creates a single line edit control
Window	Creates a window with no frame,no menu bar , and no title

Component

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

Container

- Subclass of Component
- Has additional methods that allow other Component objects to be nested within it
- Responsible for laying out i.e positioning of components

Panel

- Concrete subclass of Container
- Simplest Container class
- Provides space in which an application can attach any other component including other panels.
- Superclass of Applet
- Is a window that doesnot contain a title bar menu bar or border
- That's why we don't see them when applet is run from a browser

Window

- The window class creates a top-level window which is not contained within any other object ; it sits on the desktop
- So we don't create Window object directly and use a subclass of Window called Frame

Frame

- Encapsulates what is commonly thought as a window
- Subclass of Window and has a title bar, menu bar, borders, and resizing corners

Working with Graphics

Drawing Lines:

void drawLine(int startX,int startY,int endX,int endY)

Drawing Rectangles

void drawRect(int top,int left,int widthe,int height)
void fillRect(int top,int left,int widthe,int height)
void drawRoundRect(int top,int left,int widthe,int height,int xDiam,int yDiam)

void fillRoundRect(int top,int left,int widthe,int height,int xDiam,int yDiam)

Drawing Ovals

void drawOval(int top,int left,int widthe,int height)
void fillOval(int top,int left,int widthe,int height)

Drawing Arcs

void drawArc(int top,int left,int widthe,int height,int startangle,int sweepangle)
void fillArc(int top,int left,int widthe,int height,int startangle,int sweepangle)

Drawing Polygons

void drawPolygon(int x[],int y[],int nmpts)
void fillPolygon(int x[],int y[],int nmpts)
Setting colors can be done as g.setColor(Color.red)
If u wish to see the overridden images then use
g.setXORMode(Color.black)

<u>EXAMPLE PROGRAM</u>

```
import  java.awt.*;
import java.applet.*;
/*<applet code="Applet2" width=500 height=600>
</applet>
*/
public class Applet2 extends Applet
{
  public void paint(Graphics g)
{

  g.drawLine(20,30,100,200);

  g.drawRect(10,10,60,50);
  g.fillRect(400,100,60,50);
 g.drawRoundRect(190,10,60,50,15,15);
 g.fillRoundRect(700,900,140,100,30,40);

 g.drawOval(150,150,50,50);

 g.fillOval(200,200,75,50);

 g.drawArc(150,40,70,70,0,75);
g.setColor(Color.blue);
 g.fillArc(100,40,70,70,0,75);
g.setColor(Color.red);
 int xpo[]={30,200,30,200,30};
 int ypo[]={30,30,200,200,30};
 int num=5;
 g.drawPolygon(xpo,ypo,num);
g.setColor(Color.cyan);
 int xpo1[]={300,200,300,200,300};
 int ypo1[]={300,300,200,200,300};
 int num1=5;
g.setXORMode(Color.white);
 g.fillPolygon(xpo1,ypo1,num1);
}
}
```

AWT Controls :

- Controls are components that allow a user to interact with your application in various ways for ex: control like a push button,labels,checkbox etc
- Layout manager automatically positions components within a container… if not specified the default layout manger will be used

Control Fundamentals

- To include a control in a window you must add it to the window by creating an instance of the desired control and then adding it to a window by calling add() defined by Container.
- Forms of add()
  Component add(Component cobj)
  - To remove a control from a window call remove() defined in Container
    Void remove(Component cobj)
  - You can remove all the controls by calling removeAll()
  - Except for labels , which are passive , all controls generate events when they are accessed by the user.

LABELS

- A label is object of type Label and contains a string which it displays

- Constructors are
  Label() throws HeadlessException
  Label(String str) throws HeadlessException
  Label(String str,int how) throws HeadlessException
- First version creates empty label
- Second creates a label that contains the string specified and is left-justified
- The third creates a label that contains the string specified and the alignment specified by 'how'.
- You can set or change the text in a label by using setText() and can obtain the current label by getText() method.
  Void setText(String str)
  String getText()
- You can set alignment using setAlignment() and obtain the current by getAlignment()
  void setAlignment(int how)
  int getAlignment()

<u>PROGRAM</u>

```
import  java.awt.*;
import java.applet.*;
/*<applet code="LabelDemo" width=500 height=600>
</applet>
*/
public class LabelDemo extends Applet
```

```
{
 public void init()
{

Label one=new Label("ONE");
Label two=new Label("TWO");
Label three=new Label("THREE");
add(one);
add(two);
add(three);
}}
```

<u>BUTTONS</u>

- a push button is a component that contains a label and that generates an event when it is pressed, are objects of type Button.
- 2 contructors
- Button() throws HeadlessException
  ---Creates an empty button and label can be set using setLabel()
  ---void setLabel(String str)
- Button(String str) throws HeadlessException
  --creates a button that contains str as a label
  ---can retrieve its label by getLabel()
  --String getLabel()
- Each time a button is pressed, an action event is generated that is sent to any listeners registered; each listener implements

ActionListener interface that defines actionPerformed() method which is called whenever an event occurs.

- The label can be obtained by calling getActionCommand() on ActionEvent object passed to actionPerformed().

PROGRAM

```
import  java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*<applet code="ButtonDemo" width=500 height=600>
</applet>
*/
public class ButtonDemo extends Applet implements
ActionListener
{
 String msg="";
Button yes,no,maybe;
  public void init()
{

yes=new Button("YES");
no=new Button("NO");
maybe=new Button("MAYBE");
add(yes);
add(no);
add(maybe);
yes.addActionListener(this);
no.addActionListener(this);
maybe.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
 String str=ae.getActionCommand();
 if(str.equals("YES"))
msg="U PRESSED YES";
else if(str.equals("NO"))
msg="U PRESSED NO";
msg="U PRESSED UNDECIDED";

repaint();
}
public void paint(Graphics g)
{
g.drawString(msg,30,200);
}
}
```

<u>CHECKBOXES:</u>

---A checkbox is a control that is used to turn an option on or off.

--Constructors:

1. Checkbox() throws HeadlessException

---creates check box whose label is initially blank and to set the label setLabel() is used
---void setLabel(String str)
2. Checkbox(String str) throws HeadlessException
----creates checkbox with the label as str and the state of the checkbox is unchecked
3. Checkbox(String str,Boolean on) throws HeadlessException
----allows to set the initial state of the checkbox ; if variable on is true , then the checkbox is initially checked otherwise is cleared.
4. Checkbox(String str,Boolean on,CheckboxGroup cb) throws HeadlessException
----creates checkbox with label as str and group specified by cb ; if not a part of this group then cb will be null.

---to get current label
String getLabel()
----to set the state call setState()
void setState(boolean o); if o is on then checked else cleared

---each time checkbox is selected or deselected an itemevent is generated which is sent to the listeners registed with it ; all these listeners shud implement ItemListener interface which defines itemStateChanged() method.

PROGRAM

```
import  java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*<applet code="CheckboxDemo" width=500 height=600>
</applet>
*/
public class CheckboxDemo extends Applet implements
ItemListener
{
 String msg="";
Checkbox  winXP,solaris,mac;
  public void init()
{
winXP=new Checkbox("Windows XP",null,true);
solaris=new Checkbox("Solaris");
mac=new Checkbox("Mac Os",true);
add(winXP);
add(solaris);
add(mac);
winXP.addItemListener(this);
solaris.addItemListener(this);
mac.addItemListener(this);
}
public void itemStateChanged(ItemEvent ae)
{
 repaint();
}
public void paint(Graphics g)
{
msg="windows XP   :  "+winXP.getState();
g.drawString(msg,30,200);
msg="Solaris   :   "+solaris.getState();
g.drawString(msg,30,250);
msg="mac OS  :   "+mac.getState();
g.drawString(msg,30,300);

}
}
```

<u>CHECKBOXGROUP</u>

---to create a set of mutually exclusive check boxes in which one and only one checkbox in the group can be checked at anyone time ….these are often called "radio button"
---we can determine which checkbox in a group is currently selected by calling getSelectedCheckbox() and can set a checkbox by calling setSelectedCheckbox()
--Checkbox getSelectedCheckbox()
--void setSelectedCheckbox(Checkbox which)

<u>PROGRAM</u>

```
import  java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*<applet code="CBGroup" width=500 height=600>
```

```
</applet>
*/
public class CBGroup extends Applet implements ItemListener
{
 String msg="";
Checkbox  winXP,solaris,mac;
CheckboxGroup cbg;
  public void init()
{
cbg=new CheckboxGroup();
winXP=new Checkbox("Windows XP",cbg,true);
solaris=new Checkbox("Solaris",cbg,true);
mac=new Checkbox("Mac Os",cbg,false);
add(winXP);
add(solaris);
add(mac);
winXP.addItemListener(this);
solaris.addItemListener(this);
mac.addItemListener(this);
}
public void itemStateChanged(ItemEvent ae)
{
 repaint();
}
public void paint(Graphics g)
{
msg=cbg.getSelectedCheckbox().getLabel();
g.drawString(msg,30,200);
}
}
```