Objectives

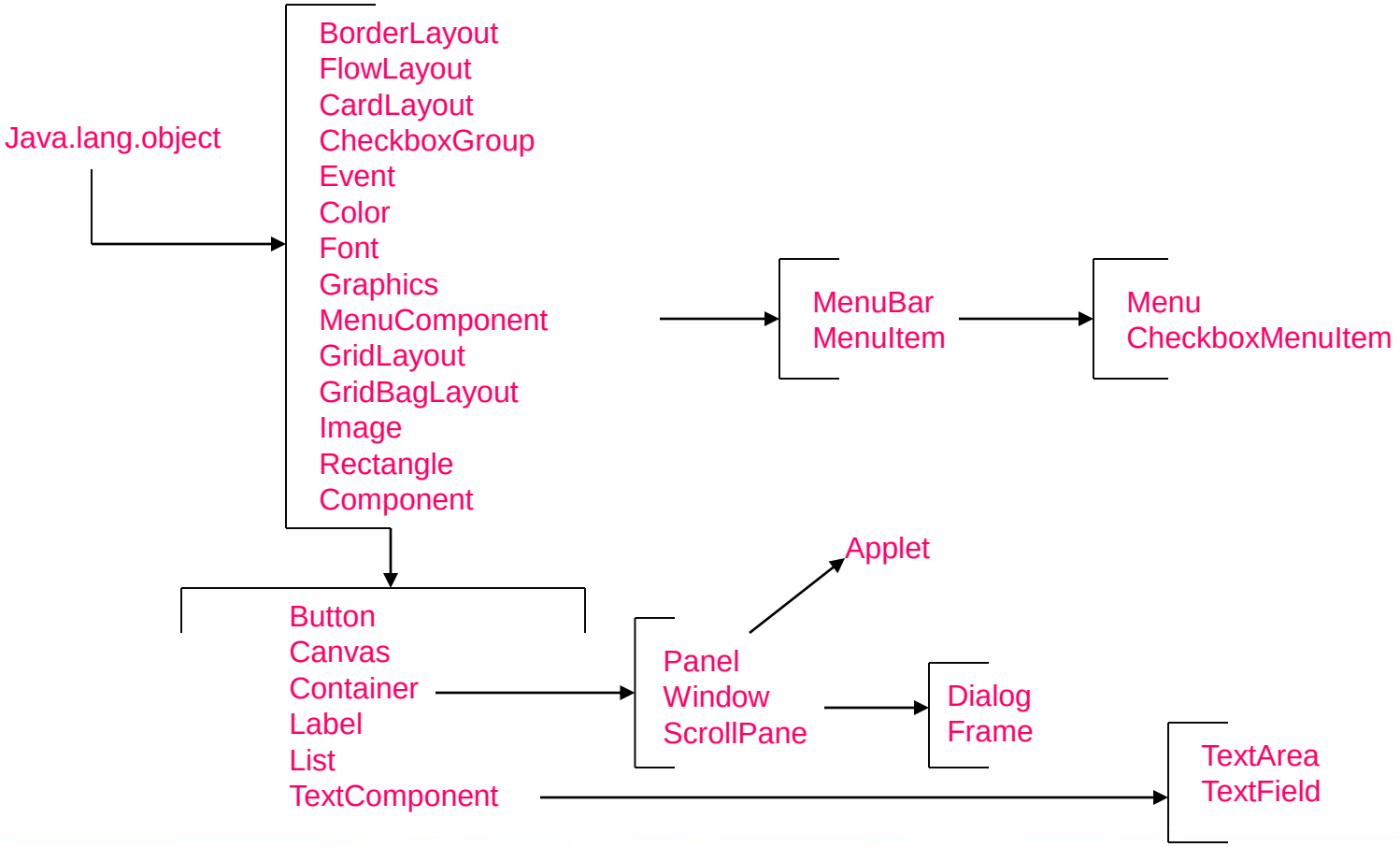
- In this session, you will learn to:
  - Describe the Abstract Window Toolkit (AWT) package and its components
  - Define the terms containers, components, and layout managers, and describe how they work together to build a GUI
  - Use the Frame and Panel containers appropriately
  - Add components to a container
  - Use various layout managers to achieve a desired dynamic layout

Abstract Window Toolkit

- Provides GUI components that are used in all Java applets and applications.
- Contains classes that can be composed or extended.
- Ensures that every GUI component that is displayed on the screen is a subclass of class `Component` or `MenuComponent`.
- Has `Container`, which is an abstract subclass of `Component` and includes two subclasses:
  - `Panel`
  - `Window`

The java.awt Package

- Basic overview of AWT package:



Containers

- The two main types of containers are:
  - `Window`
  - `Panel`
- A `Window` is a free floating window on the display.
- A `Panel` is a container of GUI components that must exist in the context of some other container, such as a `Window` or `Applet`.
- Add components with the `add()` method.

Components

- Java programming language supports various components:
  - `Button`
  - `Choice`
  - `Label`
  - `List`
  - `Scrollbar`
  - `TextComponent`, etc.
- The position and size of a component in a container is determined by a layout manager.
- You must use `setLocation()`, `setSize()`, or `setBounds()` on components to locate them in the container.

Frames

- Frames have the following characteristics:
  - Are a subclass of `Window`
  - Have title and resizing corners
  - Are invisible initially; use `setVisible(true)` to expose the frame
  - Have `BorderLayout` as the default layout manager
  - Use the `setLayout()` method to change the default layout manager

Frames (Contd.)

- An example to create a frame:

```
import java.awt.*;
public class FrameExample {
private Frame f;
public FrameExample() {
  f = new Frame("Hello Out There!");}
public void launchFrame() {
  f.setSize(170,170);
  f.setBackground(Color.blue);
  f.setVisible(true); }
public static void main(String args[]) {
  FrameExample guiWindow = new FrameExample();
  guiWindow.launchFrame(); }
}
```

Declaration of Frame Object

Initialization of Frame Object

Setting the size of Frame

Making Frame visible

Panels

- `Panel` provide a space for components.
- This enables subpanels to have their own layout manager.
- After creating `Panel`, it must be added to a `Window` or `Frame`.

Panels (Contd.)

- The following code snippet helps in creating a small yellow panel and adds it to a frame:

```
public Panel pan;
public Frame f;
f=new Frame( "I'm with panel");
pan = new Panel();
public void launchFrame()
{
f.setSize(200,200);
f.setLayout(null); // Use default layout
pan.setSize(100,100);
pan.setBackground(Color.yellow);
f.add(pan);
f.setVisible(true);
}
```

Declaring Panel Object

Initializing Panel Object

Setting the size of Panel

Giving the yellow color to the Panel

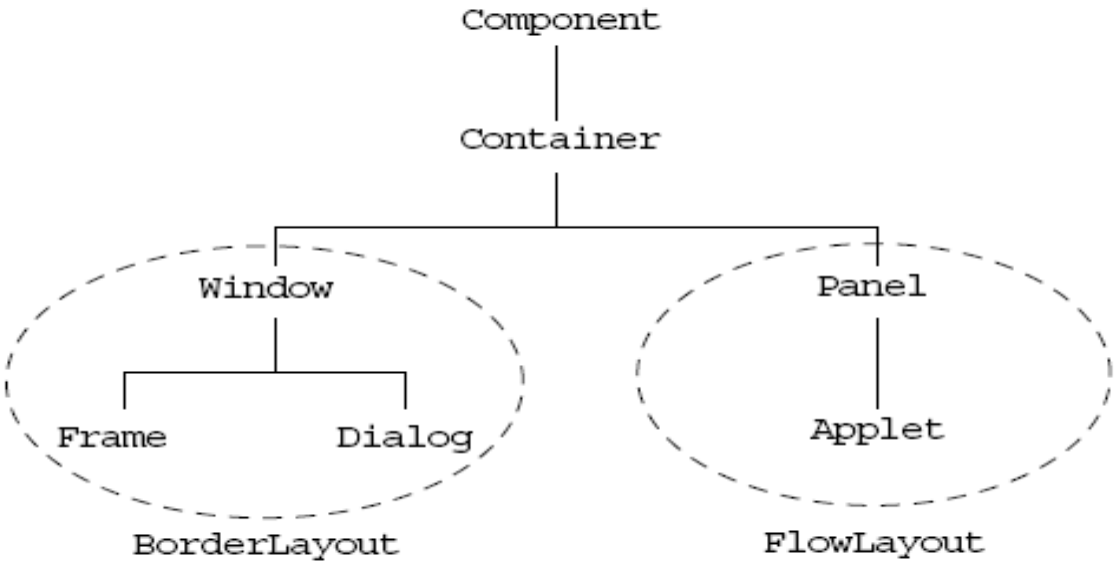
Adding Panel to the Frame

Layout Managers

- Layout managers are used to place the components on required positions.
- The following layout managers are included with the Java programming language:
  - `FlowLayout`
  - `BorderLayout`
  - `GridLayout`
  - `CardLayout`
  - `GridBagLayout`

Layout Managers (Contd.)

- The Default layout manager for `Window`, `Frame`, and `Dialog` class is `BorderLayout`.
- In the same way for `Panel` and `Applet`, `FlowLayout` is default layout manager.



Layout Managers (Contd.)

- The `FlowLayout` manager has the following characteristics:
  - Forms the default layout for the `Panel` class
  - Adds components from left to right
  - Alignment default is centered
  - Uses components preferred sizes
  - Uses the constructor to tune behavior

Layout Managers (Contd.)

- A simple example of FlowLayout:

```
public class LayoutExample
{
private Frame f;
private Button b1;
private Button b2;
public LayoutExample()
{
f = new Frame("GUI example");
b1 = new Button("Press Me");
b2 = new Button("Don't press Me");
}
```

Declaring the components

Initializing the components

Layout Managers (Contd.)

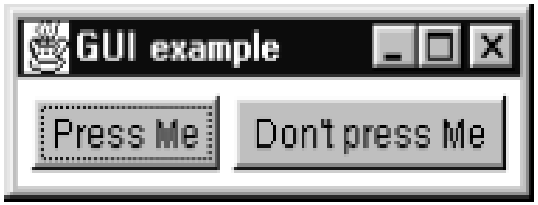
```
public void launchFrame()
{
f.setLayout(new FlowLayout());
f.add(b1);
f.add(b2);
f.pack();
f.setVisible(true);
}
public static void main(String args[])
{
LayoutExample guiWindow = new
LayoutExample();
guiWindow.launchFrame();
}
}
```

Setting FlowLayout

Adding components on the Frame

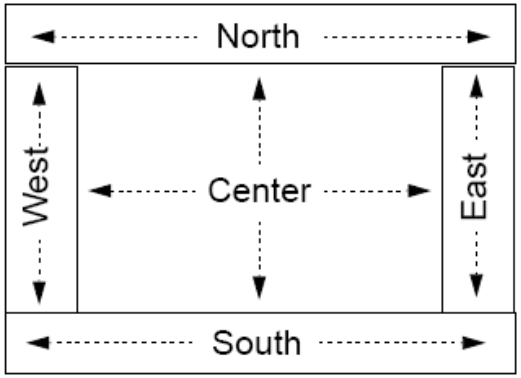
Layout Managers (Contd.)

- The preceding code will show the following output:



Layout Managers (Contd.)

- The `BorderLayout` manager has following characteristics:
  - Default layout for the `Frame` class
  - Components are added to specific regions
  - The resizing behavior is as follows:
    - North, South, and Center regions adjust horizontally
    - East, West, and Center regions adjust vertically



Layout Managers (Contd.)

- Using the constructor without any parameters constructs and installs a new `BorderLayout` with no gaps between the components:

```
setLayout(new BorderLayout());
```

- `BorderLayout` constructor specifiying `hgap` and `vgap` can be used to indicate the gaps between components:

```
BorderLayout(int hgap,int vgap);
```

- Components must be added to the named regions in `BorderLayout` manager:

```
f.add(button1,BorderLayout.NORTH);
```

Layout Managers (Contd.)

- The characteristics of `GridLayout` manager are:
  - Components are added from left to right, and from top to bottom.
  - All regions are sized equally.
  - The constructor specifies the rows and columns. For example:

    ```
    f.setLayout ( new GridLayout( 3,2));
    ```

    This statement using in Java Program will help to get the following output:

Demonstration

Lets see how to create a Java class that uses the AWT API to create a simple GUI front end.

Drawing in AWT

- `Graphics` class is an abstract class, which is used for drawing graphical figures.
- Every component has a `Graphics` object.
- The `Graphics` class implements many drawing methods.
- You can draw in any `Component` (although AWT provides the `Canvas` and `Panel` classes just for this purpose).
- Create a subclass of `Canvas` or `Panel` and override the `paint()` method.
- The `paint()` method is called every time the component is shown (for example, if another window overlapped the component and was then removed).

Drawing in AWT (Contd.)

- Example of `paint()` method:

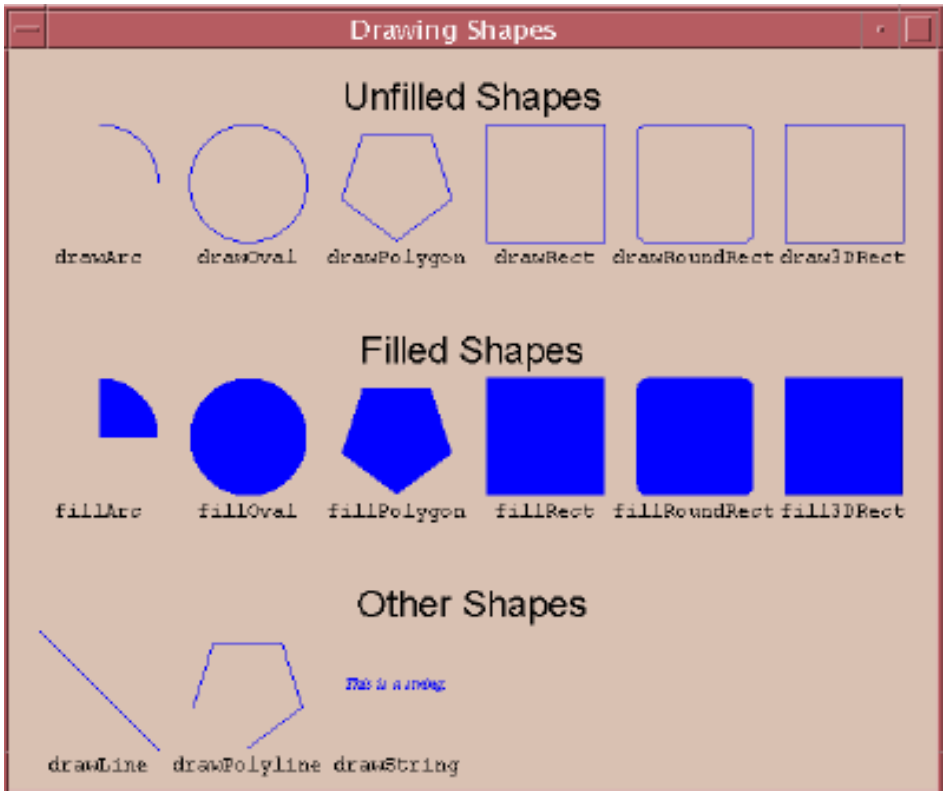
```
public void paint( Graphics g )
{
//display word "today" centred at x,y positions
FontMetrics fm = getFontMetrics(
g.getFont() );
String wording = "today";
int xadj = fm.stringWidth( wording ) / 2;
//position bottom left corner of the text
g.drawString( wording, x-xadj, y );
//draw a red line from x1,y1 to x2,y2
```

Drawing in AWT (Contd.)

```
g.setColor(Color.red );
g.drawLine(x1, y1, x2, y2);
//draw an pre-existing Image.imX, imY is top
left corner of the Image.
g.drawImage (
image,imX,imY,imWidth,imHeight,this);
} // end paint
```

Drawing in AWT (Contd.)

- Various Shapes Drawn by the `Graphics` Object:



Summary

- In this session, you learned that:
  - Abstract Window Toolkit provides GUI components that are used in all Java applets and applications.
  - `Window` and `Panel` are subclasses of container.
  - `Button`, `Choice`, `Label`, `List`, `Scrollbar`, `TextComponent` are various components supported by Java programming language.
  - `Frame` is subclass of `Window` and they are invisible until `setVisible(true)` method is not used to expose them.
  - Panel provides space for components and then panel need to be added to a `Frame` or `Window`.
  - Layout Managers are provided by Java language to place the components at any required positions.

Summary (Contd.)

- Java programming Language supports following Layout Managers:
  - `FlowLayout`
  - `BorderLayout`
  - `GridLayout`
  - `CardLayout`
  - `GridBagLayout`
- `FlowLayout` manager adds the components from left to right and it is default layout for `Panel`.
- `BorderLayout` manager adds the components to specific regions i.e North, South, East, West, and Centre.
- `GridLayout` manager adds components from left to right, and from top to bottom. The regions are sized equally as the specified number of rows and columns in the constructor.
- Graphical figures can be drawn by using `Graphics` class object.