Unit VII

Working with Graphics User Interface (Layout Managers and Menu)

Tutor: Pema Galey

Learning Outcomes

In this session, you will learn about:

- Window Fundamentals
- Work with Abstract Window Toolkit (AWT) control components
- Use layout managers
- Menu
- Images
- Create Swing components

Layout Managers

- A layout manager automatically arranges your controls within a window.
- 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 the **setLayout( )** method.

```
void setLayout(LayoutManager layoutObj)
```

- Here, ***layoutObj*** is a reference to the desired layout manager.
- Pass ***null*** If you wish to disable the layout manager and position components manually.

FlowLayout Manager

`FlowLayout` Manager:

- **FlowLayout** implements a simple layout style, which is similar to how words flow in a text editor.
- The direction of the layout is **left to right, top to bottom**. By default, components are laid out line-by-line beginning at the upper-left corner.
- The constructors for **FlowLayout**:

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

FlowLayout Manager

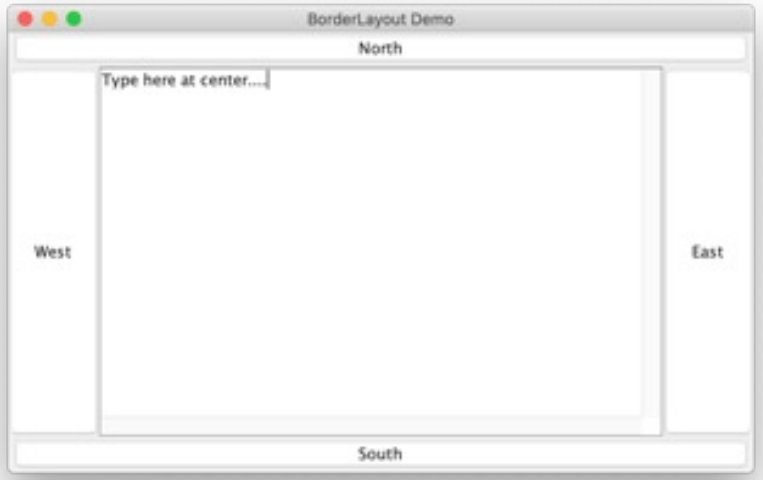
`FlowLayout` Manager:

- Alignments:
  - `FlowLayout.LEFT`
  - `FlowLayout.CENTER`
  - `FlowLayout.RIGHT`
  - `FlowLayout.LEADING`
  - `FlowLayout.TRAILING`

BorderLayout Manager

`BorderLayout` Manager:

- The **BorderLayout** class 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** is the default layout manager for **Frame**. Here are the constructors defined by **BorderLayout**:
  - `BorderLayout( )`
  - `BorderLayout(int horz, int vert)`

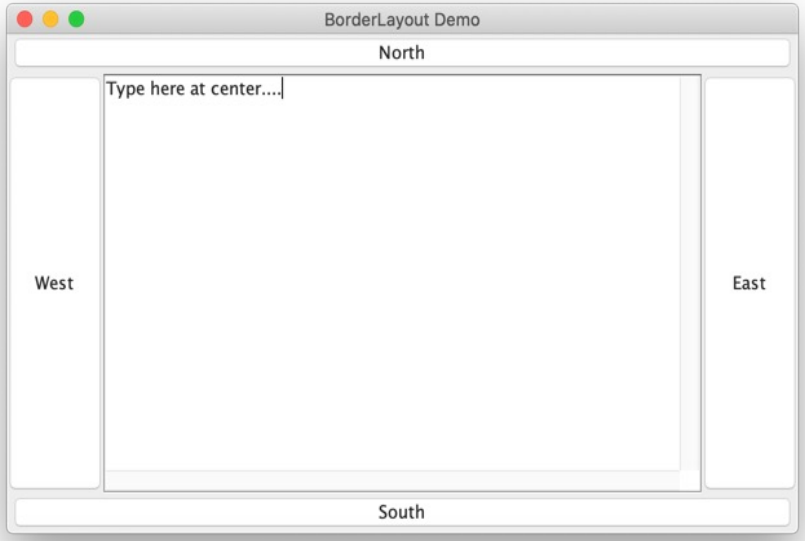


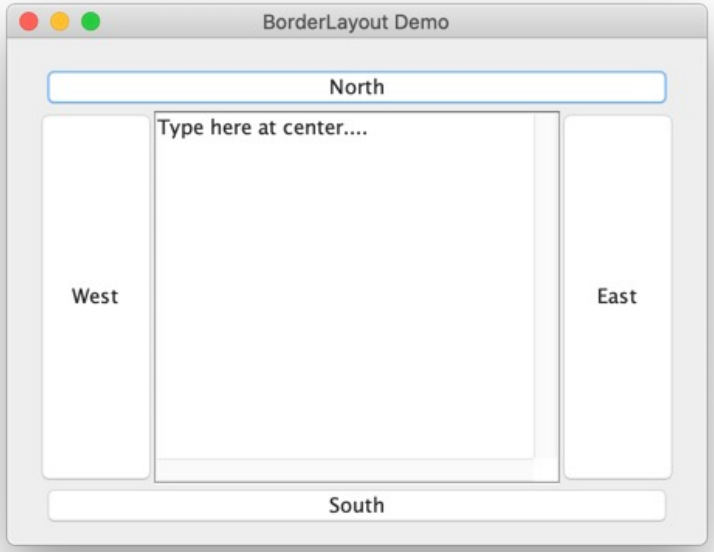
→void add(Component *compRef*, Object *region*)

BorderLayout Manager

Using **Insets**:

- To leave a small amount of space between the container that holds your components and the window that contains it.
  - `Insets(int top, int left, int bottom, int right)`





GridLayout Manager

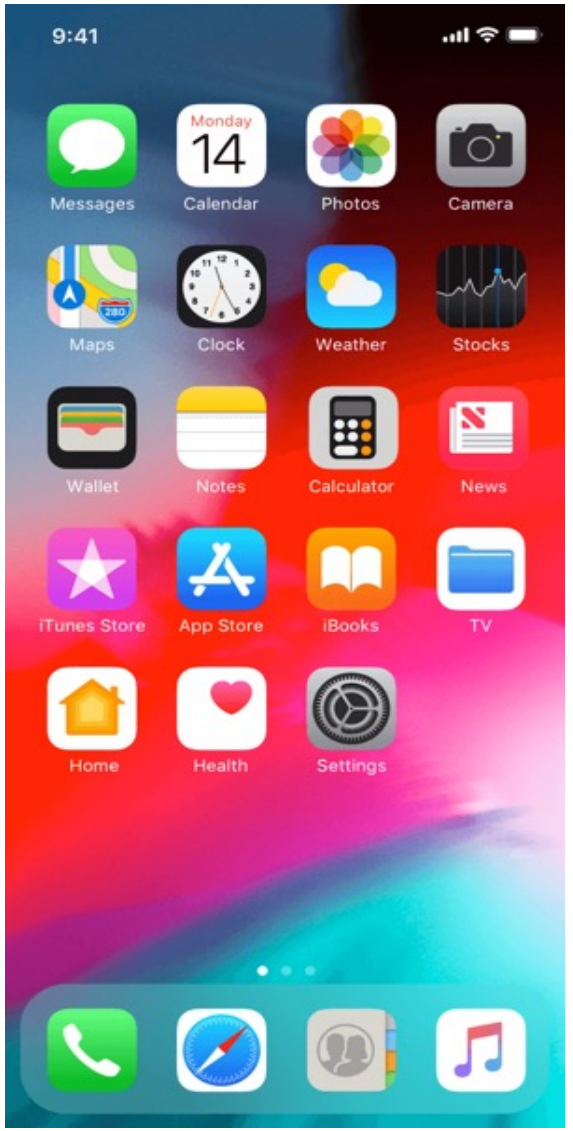
`GridLayout` Manager:

- **GridLayout** lays out components in a two-dimensional grid. When you instantiate a **GridLayout**, you define the number of rows and columns.
- The constructors supported by **GridLayout** are shown here:
  - `GridLayout( )`
    `GridLayout(int numRows, int numColumns)`
    `GridLayout(int numRows, int numColumns, int horz, int vert)`

GridLayout Manager

`GridLayout` Example:

- `new GridLayout(5,4)`



CardLayout Manager

`CardLayout` Manager:

- The **CardLayout** class 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.
- **CardLayout** constructors:
  - `CardLayout( )`
  - `CardLayout(int horz, int vert)`

GridBagLayout Manager

`GridBagLayout` Manager:

- Each component can be a different size, and each row in the grid can have a different number of columns.
- It's a collection of small grids joined together.
- **GridBagLayout** defines only one constructor, which is shown here:
  `GridBagLayout( )`

GridBagLayout Manager

`GridBagLayout` Manager:

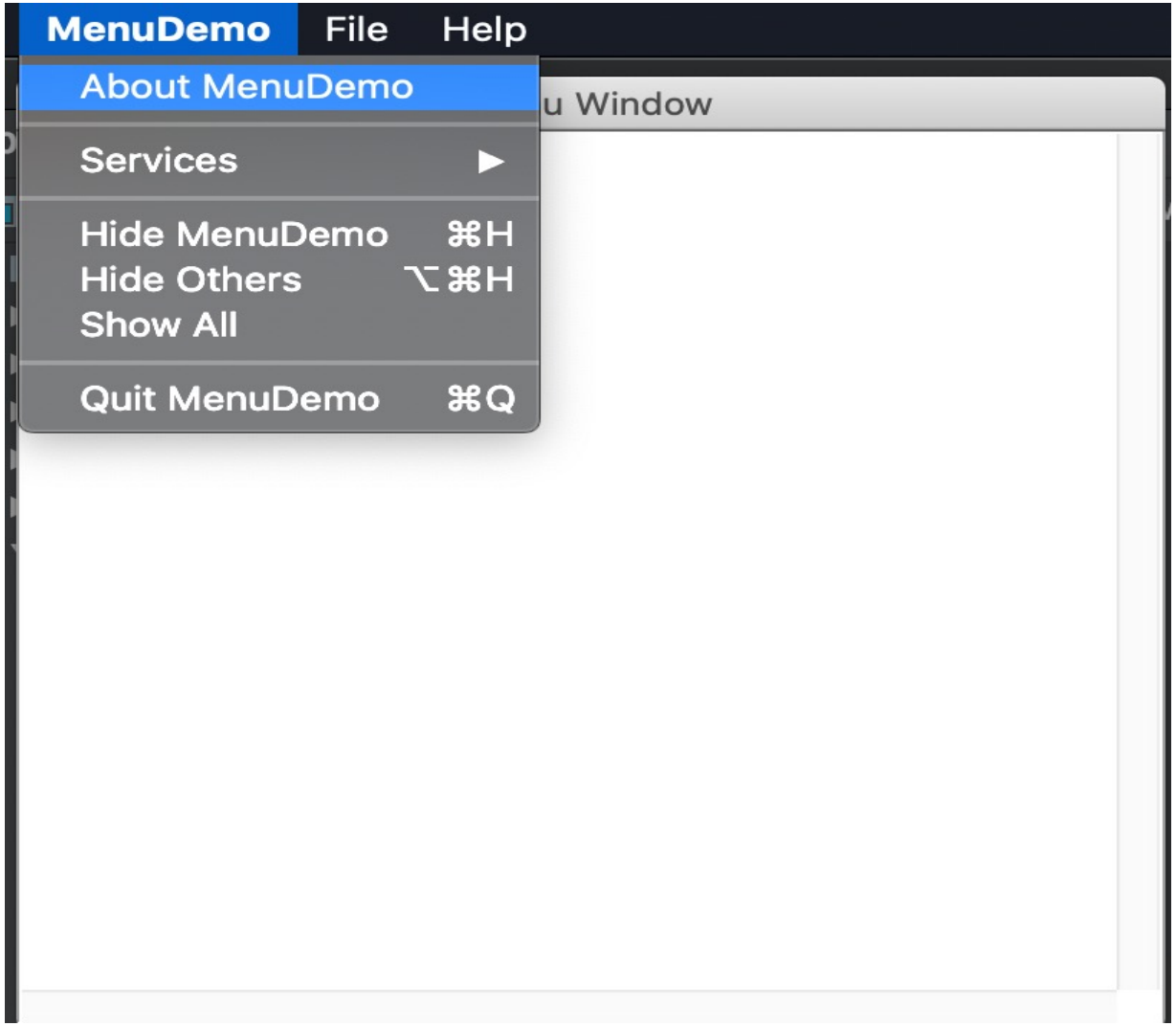
- **`GridBagLayout( )`**

Menu Bars and Menu

Menu Bars and Menu

- A top-level window can have a menu bar associated with it. A menu bar displays a list of top-level menu choices.
- Each choice is associated with a drop- down menu. This concept is implemented in the AWT by using the following classes: **MenuBar**, **Menu**, and **MenuItem**.
- `Menu( ) throws HeadlessException`
- `Menu(String optionName) throws HeadlessException`
- `Menu(String optionName, boolean removable) throws HeadlessException`

Menu Bars and Menu



Images

Images

- **Image** class is at **java.awt.image** package.
- **File Formats :** GIF, JPEG, PNG
- Three common operations with images:
  1. Creating an image,
  2. Loading an image, and
  3. Displaying an image.

1) The **createImage( )** method has the following two forms:

```
Image createImage(ImageProducer imgProd)
Image createImage(int width, int height)
```

Images

Loading an Image:

- Load one, either from a file on the local file system or from a URL.
- **ImageIO** class provides extensive support for reading and writing images.
- The method that loads an image is called **read( ):**

```
static BufferedImage read(File imageFile) throws
IOException
```

Images

Displaying an Image:

- Once you have an image, you can display it by using **drawImage( )**, which is a member of the **Graphics** class.
- It has several forms. The one we will be using is shown here:

```
boolean drawImage(Image imgObj, int left, int top,
ImageObserver imgOb)
```

- Using **read( )** and **drawImage( )**, it is to load and display an image.

Thank you!