Managing Layouts

- Once you have created the UI components of an application, it is important to arrange them properly to enhance the look and feel of the application.
- To arrange the components in a container, you can use the various layout classes, such as `FlowLayout` and `BorderLayout`.
- These layouts use relative positioning to place the components on the container.
- The `setLayout`() method is used to set the layout of the containers.

Managing Layouts (Contd.)

- Some of the layout manager classes are:
  - FlowLayout
  - BorderLayout
  - GridLayout
  - GridBagLayout
  - BoxLayout
  - GroupLayout

Using FlowLayout

- FlowLayout places components in a sequence one after the other component in a row.
- By default, the components are aligned in the center and are placed from left to right and top to bottom.
- The orientation of the components can be changed by using the setComponentOrientation() method.
- The setComponentOrientation() method can accept the following values:
  - ComponentOrientation.LEFT_TO_RIGHT
  - ComponentOrientation.RIGHT_TO_LEFT
- The FlowLayout class provides the various constructors that can be used to create an instance.

Using FlowLayout (Contd.)

- Some of the constructors are:
  - `FlowLayout()`
  - `FlowLayout(int align)`
  - `FlowLayout(int align, int hgap,int vgap)`

Using BorderLayout

- `BorderLayout`:
  - Divides the container into five regions: `NORTH`, `SOUTH`, `EAST`, `WEST`, and `CENTER`.
  - Arranges and resizes the components to place them in the five regions.
- It is the default layout of the frame container.
- By default, the components are added in the `CENTER` region.
- You can specify the region for the border layout using the following fields:
  - `BorderLayout.NORTH`
  - `BorderLayout.SOUTH`
  - `BorderLayout.EAST`
  - `BorderLayout.WEST`
  - `BorderLayout.CENTER`

Using BorderLayout (Contd.)

- The `BorderLayout` class provides the various constructors that can be used to create an instance.
- Some of the constructors are:
  - `BorderLayout()`
  - `BorderLayout(int hgap, int vgap)`

Using GridLayout

- `GridLayout` divides the container into rows and columns.
- The intersection of the row and the column is called a cell.
- The cell can contain only one component.
- The `GridLayout` class provides the various constructors that can be used to create an instance of the `GridLayout` class.
- Some of the constructors are:
  - `GridLayout()`
  - `GridLayout(int rows, int cols)`
  - `GridLayout(int rows, int cols, int hgap, int vgap)`

Using GridBagLayout

- `GridBagLayout`:
  - Places components in a grid of rows and columns, allowing specified component to span multiple rows or columns.
  - Places components within the cell of a grid and uses the components' preferred sizes to determine the size of the cells.
- The `GridBagLayout` class provides the `GridBagLayout()` constructor.
- `GridBagLayout` is associated with the `GridBagConstraints` class.
- The `GridBagCosntraints` object specifies the location within the container.
- It considers each component's minimum and preferred sizes to place them.

Using GridBagLayout (Contd.)

- Some of the commonly used instance variables of the `GridBagConstraints` class are:
  - `gridx`
  - `gridy`
  - `fill`
  - `anchor`
  - `gridwidth`
  - `gridheight`
  - `weightx`
  - `weighty`

Introduction to Java

Using BoxLayout

- `BoxLayout` arranges multiple components either vertically, such as a stack, or horizontally, such as a row.
- `BoxLayout` works with the following axis parameters:
  - `X_AXIS`
  - `Y_AXIS`
  - `LINE_AXIS`
  - `PAGE_AXIS`
- The following constructor can be used to create an instance of the `BoxLayout` class:
  - `BoxLayout(Container target, int axis)`

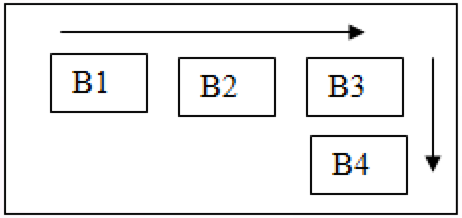
Introduction to Java

Using GroupLayout

- `GroupLayout` is used to group various UI components in order to position them in a container.
- Each group may contain any number of elements, where an element can be:
  - A group
  - The UI components
  - A gap
- `GroupLayout` defines a layout for both, horizontal axis and vertical axis.
- It uses the following two types of arrangements:
  - Sequential
  - Parallel

Using GroupLayout (Contd.)

- The following figure shows a sequential group of three components, B1, B2, and B3, along the horizontal axis, and a parallel group of two components, B3 and B4, along the vertical axis.



- The `GroupLayout` class provides the following constructor that can be used to create an instance of the GroupLayout class:
  - `GroupLayout(Container host)`

Introduction to Java

Using GroupLayout (Contd.)

- If you do not want to use any of the predefined layout managers, you can create a container without a layout manager.
- You can use the `setLayout()` method of the container and pass `null` as an argument.
- If the layout of the container is set to null, you need to explicitly define the position and size of the component.
- In order to explicitly define the position and size of the component, you must use the following method:

```
void setBounds(int x, int y, int width, int height)
```

Introduction to Java

Just a minute

- Which one of the following layouts arranges the components according to the sequential and parallel arrangements?
  - `FlowLayout`
  - `GroupLayout`
  - `BorderLayout`
  - `GridLayout`

Just a minute (Contd.)

- Solution:
  - `GroupLayout`

Just a minute

- Which one of the following layouts allows you to place the components only in five regions?
  - BorderLayout
  - BoxLayout
  - FlowLayout
  - GridBagLayout

Just a minute (Contd.)

- Solution:
  - `BorderLayout`

Activity 7.1: Managing Layouts

- Problem Statement:
  - Peter has been assigned a task to redesign the Hangman game to provide it a graphical interface. For this, he decides to design the interface for the menu and game windows. The menu window will be used as a start-up screen with three options: play game, view instructions, and exit game. The game window will be used to play the Hangman game that will display three panels:
    - The first panel should contain a label that has the text, Guess a Country Name.
    - The second panel should contain an area to display the letters guessed by the user.
    - The third panel should contain a keypad that will be used by the user to select a letter.

Activity 7.1: Managing Layouts (Contd.)

- Problem Statement (Contd.):
  - In addition, he has been asked to ensure that the UI for the application should have an attractive look and feel. Further, the controls should be arranged in the window with proper alignment. Help Peter to achieve the preceding requirement.

Activity 7.1: Managing Layouts (Contd.)

- Solution: To perform the activity, refer the steps given in the embedded document.

Microsoft Word Document

Introduction to Java

Summary

- In this session, you learned that:
  - In a CUI application, a user needs to remember all the commands to work with the application.
  - The GUI provides a graphical way of interacting with the application.
  - Java provides the various UI components, such as `JFrame`, `JPanel`, `JButton`, and `JLabel`, of the `javax.swing` package.
  - The most commonly used containers and components are:
    - `Jframe`
    - `Jdialog`
    - `Jpanel`
    - `JTabbedPane`
    - `JMenuBar`
    - `JMenu`

Summary (Contd.)

  - `JMenuItem`
  - `JLabel`
  - `JTextField`
  - `JTextArea`
  - `JCheckBox`
  - `JRadioButton`
  - `JList`
  - `JComboBox`
  - `JButton`
  - `JOptionPane`
- You need to import the classes and the interfaces available in the `javax.swing` package to create the UI of the applications.
- To arrange the components in a container, you can use the various layouts, such as `FlowLayout` and `BorderLayout`.

Introduction to Java

Summary (Contd.)

- The `setLayout()` method is used to set the layout of the containers.
- Java provides the various layouts. Some of the layout manager classes are:
  - `FlowLayout`
  - `BorderLayout`
  - `GridLayout`
  - `GridBagLayout`
  - `BoxLayout`
  - `GroupLayout`
- You can use the `setLayout()` method of the container and pass `null` as an argument to create a container without a layout manager.