	PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE -43		
	Department of Electronics and Telecommunication Engineering		
	ASSESSMENT YEAR: - 2021-22		CLASS: - TE- V
	Subject: - Advanced Java Programming		
Expt. No: 05	LAB Ref: ETC/2021-22/	ROLL NO: 32147	SUBMISION DATE:

Title: - Layout managers

Problem Statement: - Write a program in Java to illustrate different types of Layouts using AWT

1) Border Layout 2) Grid Layout

Objectives: -

To learn the concepts of different types Layout managers

Theory (Write Theory of the new concept demonstrated in this Assignment)

Layout means the arrangement of components within the container. In other way we can say that placing the the components at a particular position within the container. The task of layouting the controls is done automatically by Layout Manager.

Layout Manager

The layout manager automatically positions all the components within the container. If we do not use layout manager then also the components are positioned by the default layout manager. It is possible to layout the controls by hand but it becomes very difficult because of the following two reasons.

- It is very tedious to handle a large number of controls within the container.
- Oftenly the width and height information of a component is not given when we need to arrange them.

Java provide us with various layout manager to position the controls. The properties like size, shape and arrangement varies from one layout manager to other layout manager. When the size of the applet or the application window changes the size, shape and arrangement of the components also changes in response i.e. the layout managers adapt to the dimensions of appletviewer or the application window.
The layout manager is associated with every Container object. Each layout manager is an object of the class that implements the LayoutManager interface

AWT Layout Manager Classes:

Following is the list of commonly used controls while designed GUI using AWT.

1. BorderLayout :-

The class BorderLayout arranges the components to fit in the five regions: east, west, north, south and center. Each region is can contain only one component and each component in each region is identified by the corresponding constant NORTH, SOUTH, EAST, WEST, and CENTER.

Class declaration

Following is the declaration for java.awt.BorderLayout **class:**

```
public class BorderLayout
```

```
  extends Object
    implements LayoutManager2, Serializable
```

Field

Following are the fields for **java.awt.BorderLayout** class:

- **static String CENTER --** The center layout constraint (middle of container).
- **static String EAST --** The east layout constraint (right side of container).
- **static String NORTH --** The north layout constraint (top of container).
- **static String SOUTH --** The south layout constraint (bottom of container).
- **static String WEST --** The west layout constraint (left side of container).

Class constructors

S.N.	Constructor & Description
1	**BorderLayout()** Constructs a new border layout with no gaps between components.
2	**BorderLayout(int hgap, int vgap)** Constructs a border layout with the specified gaps between components.

2. **CardLayout** :-

The class CardLayout arranges each component in the container as a card. Only one card is visible at a time, and the container acts as a stack of cards.

Class declaration:-

Following is the declaration for java.awt.CardLayout class:

```
public class CardLayout
  extends Object
    implements LayoutManager2, Serializable
```

Class constructors

S.N.	Constructor & Description
1	**CardLayout()** Creates a new card layout with gaps of size zero.

2	**CardLayout(int hgap, int vgap)** Creates a new card layout with the specified horizontal and vertical gaps.

3. **FlowLayout :**

The class **FlowLayout** components in a left-to-right flow.

Class declaration:

Following is the declaration for java.awt.FlowLayout class:

```
public class FlowLayout
   extends Object
      implements LayoutManager, Serializable
```

Field:

Following are the fields for **java.awt.BorderLayout** class:

- **static int CENTER --** This value indicates that each row of components should be centered.
- **static int LEADING --** This value indicates that each row of components should be justified to the leading edge of the container's orientation, for example, to the left in left-to-right orientations.
- **static int LEFT --** This value indicates that each row of components should be left-justified.
- **static int RIGHT --** This value indicates that each row of components should be right-justified.
- **static int TRAILING --** This value indicates that each row of components should be justified to the trailing edge of the container's orientation, for example, to the right in left-to-right orientations.

Class constructors :

S.N.	Constructor & Description
1	**FlowLayout()** Constructs a new FlowLayout with a centered alignment and a default 5-unit horizontal and vertical gap.
2	**FlowLayout(int align)** Constructs a new FlowLayout with the specified alignment and a default 5-unit horizontal and vertical gap.
3	**FlowLayout(int align, int hgap, int vgap)** Creates a new flow layout manager with the indicated alignment and the indicated horizontal and vertical gaps.

4. GridLayout :

The class **GridLayout** arranges components in a rectangular grid.

Class declaration

Following is the declaration for java.awt.GridLayout class:

```
public class GridLayout
   extends Object
      implements LayoutManager, Serializable
```

Class constructors

S.N.	Constructor & Description
1	**GridLayout()** Creates a grid layout with a default of one column per component, in a single row.
2	**GridLayout(int rows, int cols)** Creates a grid layout with the specified number of rows and columns.
3	**GridLayout(int rows, int cols, int hgap, int vgap)** Creates a grid layout with the specified number of rows and columns.

5. GridBagLayout :

The class GridBagLayout arranges components in a horizontal and vertical manner.

Class declaration:

Following is the declaration for java.awt.GridBagLayout class:

```
public class GridBagLayout
   extends Object
      implements LayoutManager2, Serializable
```

Field:

Following are the fields for java.awt.BorderLayout class:

- **double[] columnWeights --** This field holds the overrides to the column weights.
- **int[] columnWidths --** This field holds the overrides to the column minimum width.
- **protected static int MAXGRIDSIZE --** The maximum number of grid positions (both horizontally and vertically) that can be laid out by the grid bag layout.
- **protected static int MINSIZE --** The smallest grid that can be laid out by the grid bag layout.
- **protected static int PREFERREDSIZE --** The preferred grid size that can be laid out by the grid bag layout.
- **int[] rowHeights --** This field holds the overrides to the row minimum heights.

- **double[] rowWeights --** This field holds the overrides to the row weights.

Class constructors:

S.N.	Constructor & Description
1	**GridBagLayout()** Creates a grid bag layout manager.

Diagram: -

Learning Outcomes: -		
	1	I have learnt different types Layout managers
	2	I have implemented the BorderLayout and GridLayout

Continuous Assessment			
RPP (out of 5)	**SPO (out of 5)**	**Total (Out of 10)**	**Sign**
			Date: -

#(RPP – Regularity, Punctuality, Performance), (SPO – Submission, Presentation, Oral)

Important Questions: -
1. What is function of layout manager?
2. Describe in detail about different layout in java GUI.
3. State various types of layout supported by JAVA. Which layout is default one?
4. Explain various fields in BorderLayout class.
5. Write methods and its description in GridLayout.