CHAPTER 1

■ ■ ■

# Introduction to Swing

## QUESTIONS AND EXERCISES

1. What is Swing?

**Answer:**

Swing is widget toolkit that provides graphical user interface (GUI) components to develop Java applications with a rich set of graphics such as windows, text fields, buttons, checkboxes, etc.

2. What is the use of the following classes in Swing applications: `Point`, `Dimension`, `Insets`, and `Rectangle`?

**Answer:**

- `Point`: An object of the `Point` class represents a location in a two-dimensional space. It encapsulates x and y coordinates of a point.
- `Dimension`: An object of the `Dimension` class wraps the width and height of a component. The width and height of a component are collectively known as its size. In other words, an object of the `Dimension` class is used to represent the size of a component.
- `Insets`: An object of the `Insets` class represents spaces that are left around a container. It wraps four properties named `top`, `left`, `bottom`, and `right`. Their values represent the spaces left on the four side of a container.
- `Rectangle`: An instance of the `Rectangle` class represents a rectangle. A `Rectangle` is defined by three properties: (x, y) coordinates of the upper-left corner, Width, and Height. You can think of a `Rectangle` object as a combination of a `Point` object and a `Dimension` object; the `Point` object holds the (x, y) coordinates of the upper left corner and the `Dimension` object holds the width and height.

3. Where in a `JFrame` do you add components: to the content pane or glass pane?

**Answer:**

You add components to a `JFrame` in its content pane.

4. Complete the missing code in the following snippet of code that obtains the reference of the content pane of a `JFrame`:

```
JFrame frame = new JFrame("My Frame");
Container contentPane = frame./* your code goes here */;
```

**Solution:**

```
JFrame frame = new JFrame("My Frame");
Container contentPane = frame.getContentPane();
```

5. What does the `pack()` method of the `JFrame` do?

**Answer:**

The `pack()` method of the `JFrame` class goes through all the components you have added to the `JFrame` and determines their preferred size. It sets the size of the `JFrame` just enough to display all the components. When you call the `pack()` method, you do not need to set the size of the `JFrame`. The `pack()` method will calculate the size of the `JFrame` and set it for you.

6. What is a layout manager and why is it important to use a layout manager in your Swing applications?

**Answer:**

A container uses a layout manager to compute the position and size of all its components. In other words, the job of a layout manager is to compute four properties (x, y, width, and height) of all components in a container. The x and y properties determine the position of a component within the container. The width and height properties determine the size of the component.

If specify the position and size of all components in your Swing application, your components will not be repositioned and resized when the container is resized. In addition, you will have to specify the size of the component for all platforms on which your application will run because different platforms render components a little differently. Suppose your application displays text in multiple languages. The optimal size for a `JButton`, say a `Close` button, will be different in different languages and you will have to calculate the size of the `Close` button in each language and set it, depending on the language the application is using. However, you do not have to take all of these into consideration if you use a layout manager. The layout manager will do these simple, though time-consuming, things for you. In short, a layout manager relives you of the pain of computing position and size of all Swing components for all different environments such as operating system, locale, screen resolution, etc.

7. What is the default layout manager of the content pane of a `JFrame`?

**Answer:**

`BorderLayout`

8. What is a `BorderLayout` and how does it organize its components?

**Answer:**

A `BorderLayout` is a layout manager. It divides a container's space into five areas: north, south, east, west, and center. When you add a component to a container with a `BorderLayout`, you need to specify to which of the five areas you want to add the component. The `BorderLayout` class defines five constants to identify each of the five areas. The constants are `NORTH`, `SOUTH`, `EAST`, `WEST`, and `CENTER`.

The five areas in a `BorderLayout` (north, south, east, west, and center) are fixed in direction and are not dependent on the orientation of components. Four more constants exist to specify areas in a `BorderLayout`. These constants are `PAGE_START`, `PAGE_END`, `LINE_START`, and `LINE_END`. The `PAGE_START` and `PAGE_END` constants are the same as the `NORTH` and `SOUTH` constants, respectively. The `LINE_START` and `LINE_END` constants change their positions depending on the orientation of the container. If the container's orientation is left to right, `LINE_START` is the same as `WEST`, and `LINE_END` is the same as `EAST`. If the container's orientation is right to left, `LINE_START` is the same as `EAST`, and `LINE_END` is the same as `WEST`.

A `BorderLayout` computes the size of the components based on the area in which they are placed. It respects the preferred height of the component in north and south. However, it stretches the component's width horizontally according to the available space in north and south. That is, it does not respect the preferred width of the components in north and south. It respects the preferred width of the components in east and west and gives them height necessary to fill the entire space vertically. The component in the center area is stretched horizontally as well as vertically to fit the available space. That is, the center area does not respect its component's preferred width and height.

9. Name the layout manager that displays all its components in a gird of cells giving them equal size.

**Answer:**

`GridLayout`

10. Suppose you add five components to a container having `CardLayout`. How many components will be visible in that container at any time?

**Answer:**

One

11. Create a Swing application with a `JFrame` and two buttons. The two buttons are labeled as "Top" and "Bottom". The "Top" button should be placed above the "Bottom" button. When you make the `JFrame` taller, the "Top" button should stay at the top of the `JFrame` and the "Bottom" button should stay at the bottom, thus creating a vertical gap between the two buttons. Use the `BoxLayout` to achieve this layout.

**Solution:**

```
// BoxedButtons.java
package com.jdojo.swing.intro.exercises;

import java.awt.Container;
import javax.swing.Box;
import javax.swing.JButton;
import javax.swing.JFrame;

public class BoxedButtons extends JFrame {
    private final JButton topBtn = new JButton("Top");
    private final JButton bottomBtn = new JButton("Bottom");

    public BoxedButtons(String title) {
        super(title);
        initFrame();
    }

    // Initialize the frame and add components to it.
    private void initFrame() {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container contentPane = this.getContentPane();

        Box vBox = Box.createVerticalBox();
        vBox.add(topBtn);
        vBox.add(Box.createVerticalGlue());
        vBox.add(bottomBtn);
        contentPane.add(vBox);
    }

    // Display the CustomFrame
    public static void main(String[] args) {
        BoxedButtons frame = new BoxedButtons("Vertically Boxed Buttons");
        frame.pack();
        frame.setVisible(true);
    }
}
```

12. What is the use of the `GridBagConstraints` class while using the `GridBagLayout` layout manager?

**Answer:**

An object of the `GridBagConstraints` class defines constraints for a component in a `GridBagLayout`. The constraints of a component are used to lay out the component. Some of the constraints include the component’s position in the grid, width, height, alignment inside the cell, etc. All constraints are defined as public instance variables in this class.

13. Use a `GridBagLayout` to display three buttons in a `JFrame`. The lower-right corner of the first button should touch the upper-left corner of the second button and the lower-right corner of the second button should touch the upper-right corner of the third button.

**Solution:**

```
// DiagonalButtons.java
package com.jdojo.swing.intro.exercises;

import java.awt.Container;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class DiagonalButtons extends JFrame {
    private final JButton btn1 = new JButton("Button 1");
    private final JButton btn2 = new JButton("Button 2");
    private final JButton btn3 = new JButton("Button 3");

    public DiagonalButtons(String title) {
        super(title);
        initFrame();
    }
    // Initialize the frame and add components to it.
    private void initFrame() {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container contentPane = this.getContentPane();
        contentPane.setLayout(new GridBagLayout());

        // Create a GridBagConstraints to set the constraints for each button
        GridBagConstraints gbc = new GridBagConstraints();

        gbc.gridx = 0;  gbc.gridy = 0;
        contentPane.add(btn1, gbc);

        gbc.gridx = 1;  gbc.gridy = 1;
        contentPane.add(btn2, gbc);

        gbc.gridx = 2;  gbc.gridy = 2;
        contentPane.add(btn3, gbc);
    }

    // Display the CustomFrame
    public static void main(String[] args) {
        DiagonalButtons frame = new DiagonalButtons("Diagonally Placed Buttons");
        frame.pack();
        frame.setVisible(true);
    }
}
```

14. What is absolute positioning when laying out Swing components in a container? How do you achieve it?

**Answer:**

Laying out components in a container at fixed positions is known as absolute positioning. You achieve absolute positioning in a Swing application by using no layout manager, which is achieved by setting the layout manager of a container to `null`. The following snippet of code shows how to set the layout manager of a content pane of a `JFrame` to `null`:

```
JFrame frame = new JFrame("No Layout Manager Frame");
Container contentPane = frame.getContentPane();
contentPane.setLayout(null);
```

If you add components to the content pane, you will need to set the position and size of those components in your code. It is not recommended to use absolute positioning in your Swing application, except for prototyping; your components may not be laid out correctly on all platforms and locales.

15. What is an event? What is the name of the event class that you need to handle when you are interested in capturing the clicked event of a `JButton`?

**Answer:**

An event in Swing is an action taken by a user at a particular point in time. For example, pressing a button, pressing a key down/up on the keyboard, and moving the mouse over a component are events. Sometimes the occurrence of an event in Swing (or any GUI-based application) is also known as "triggering an event" or "firing an event." When you say that a clicked event has occurred on a button, you mean that the button has been pressed using the mouse, the spacebar, or by any other means your application allows you to press a button. Sometimes you can use the phrase "clicked event has been triggered or fired on a button" to mean the same that the button has been pressed.

When an event occurs, you want to respond to the event. Taking an action in a program is nothing but executing a piece of code. Taking an action in response to the occurrence of an event is called *event handling*. The piece of code that is executed when an event occurs is called an *event handler*. Sometimes an event handler is also called an *event listener*.

An object of `ActionEvent` class in `java.awt.event` package represents the clicked event of a `JButton`.

16. Create a `JFrame` with a `JButton`. Label the JButton as "Exit". When the `JButton` is pressed, print a "Bye" message to the standard output and exit the application.

**Answer:**

```
// ByeFrame.java
package com.jdojo.swing.intro.exercises;

import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.Container;
import javax.swing.JButton;
import javax.swing.JFrame;

public class ByeFrame extends JFrame {
    private final JButton exitBtn = new JButton("Exit");

    public ByeFrame(String title) {
        super(title);
        initFrame();
    }

    // Initialize the frame and add components to it.
    private void initFrame() {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container contentPane = this.getContentPane();
        contentPane.add(exitBtn, BorderLayout.CENTER);

        exitBtn.addActionListener((ActionEvent e) -> {
            System.out.println("Bye");
            System.exit(0);
        });
    }

    // Display the CustomFrame
    public static void main(String[] args) {
        ByeFrame frame = new ByeFrame("Bye Frame");
        frame.pack();
        frame.setVisible(true);
    }
}
```