Adam Eloraby

Critical Response Questions – Chapter 10

---

1. Explain the difference between an event-driven application and a console-based application.

A **console-based application** runs in a straight line from start to finish and pauses for input when needed.
An **event-driven application**, such as one using a GUI, waits for user interactions like clicks or typing and reacts by executing specific event-handling code. The flow depends on user actions rather than a fixed program order.

2. Explain how code is executed in an event-driven application.

In an event-driven program, the code runs in response to **events**. Each interactive component (for example, a button) has an **event listener** that detects when the user performs an action. When the event occurs, the listener's method (like actionPerformed()) executes the corresponding block of code.

3. Can components be added directly to a frame? Explain.

No. Components should be added to the **content pane** of a frame, not directly to the JFrame itself.
This is because the content pane is the container where visible interface elements (like buttons and labels) are placed and managed by the layout manager.

4. Can a label respond to events? Explain.

Normally, **labels (JLabel) do not respond to events** because they are meant only to display information.
However, event listeners such as MouseListener can be added to a label to make it respond to clicks or movement if needed.

5. Why do you think a GUI needs to be run from an event-dispatching thread?

Swing applications use the **Event Dispatch Thread (EDT)** to handle user interactions and screen updates. Running GUI code on this thread ensures that events are processed in order and that interface updates are smooth and thread-safe. Without the EDT, display errors or unexpected behavior could occur.

6. What is the difference between a label and a button?

A **label (JLabel)** is used to show static text or images and is not interactive.
A **button (JButton)** displays text or an image too, but it is **interactive**—it can detect clicks and trigger actions through event listeners.

8. List three ways to control the layout of a content pane.

1. Using layout managers such as **FlowLayout**, **BorderLayout**, or **GridLayout**.
2. Setting the layout to **null** and manually positioning components using setBounds().
3. Using more advanced or external layout managers such as **GridBagLayout** or **MigLayout**.

13. What must first be done with numeric data typed in a text field before it can be used in a calculation?

Text field input is stored as a **String**. To use it in a calculation, it must be **converted to a numeric type**, such as int or double.
For example:

```
int num = Integer.parseInt(textField.getText());
double val = Double.parseDouble(textField.getText());
```

14. What is the value of num1 in the last statement below?

```
double num1;
Double num2 = new Double(3);
String num3 = "5";
num1 = num2.doubleValue() + Double.valueOf(num3).doubleValue();
```

Answer:

num1 = 3.0 + 5.0 = 8.0

15. An application prompts a user to select a name from a list of six names. Which is a better component choice: a text field or a combo box? Explain.

A **combo box** is the better choice. It allows the user to select from a fixed list of valid options, reducing typing errors and ensuring consistency. A text field would require manual input, which can lead to mistakes or mismatched names.