Here are the completed answers for questions **#1 – #6, # #13 – #15 from your worksheet:

---

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

An event-driven application responds to user actions or system events (like mouse clicks, key presses, or window events), controlling program flow based on interactions. A console-based application runs sequentially in a text-based command-line interface and processes input/output in a linear, step-by-step order.

---

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

In an event-driven application, the program waits for events in an event loop. When an event occurs, an event listener or handler responds by executing the associated code. Execution depends on user actions rather than a fixed sequence of instructions.

---

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

Yes, but in Swing, components are technically added to the frame's content pane, not directly to the `JFrame` itself. Calls like `frame.add(component)` are redirected to the content pane, which manages the actual placement of components.

---

4. Can a label respond to events? Explain.

A `JLabel` does not generate events by default because it is used for display purposes. However, event listeners (like `MouseListener`) can be attached to it, enabling it to respond to clicks or mouse movements.

---

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

A GUI must run on the event-dispatching thread to ensure thread safety. Swing components are *not thread-safe*, so keeping all component updates on one thread prevents race conditions and ensures consistent, predictable behavior.

---

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

- **Label (`JLabel`)**: Displays text or images, non-interactive by default.
- **Button (`JButton`)**: Interactive component that users can click, which generates an `ActionEvent` to perform tasks.

---

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

1. Use layout managers such as `BorderLayout`, `FlowLayout`, or `GridLayout`.
2. Nest panels (`JPanel`) with their own layout managers to create complex designs.
3. Use absolute positioning with `null` layout and `setBounds()` (not recommended in practice).

---

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

Data from a `JTextField` is read as a `String`. It must be converted into a numeric type using parsing methods like `Integer.parseInt()` or `Double.parseDouble()` before being used in arithmetic operations.

---

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

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

- `num2.doubleValue()` = 3.0
- `Double.valueOf("5").doubleValue()` = 5.0
- So, `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 (`JComboBox`)** is the better choice because it allows the user to select from a predefined list of names, ensuring valid input and preventing typing errors. A text field could lead to spelling mistakes or inconsistent formatting.
