Chapter 10 CRT

1. Difference between event-driven and console-based applications
   An event-driven application reacts to things that happen—like clicks, key presses, or window actions. The program's flow depends on these interactions instead of following a single, fixed sequence. In contrast, a console-based application runs from start to finish in a straight line, taking input and giving output through text only.
2. How code executes in an event-driven application
   In an event-driven program, the code doesn't just run top to bottom. Instead, it waits for something to happen, such as a user pressing a button. When that event occurs, a specific piece of code called an event handler runs to respond. The order of execution depends on what the user does.
3. Adding components to a frame
   Technically, Swing components are added to the content pane of a JFrame, not directly to the frame itself. However, when you use a command like `frame.add(component)`, Java automatically sends it to the content pane, so it looks like you're adding it to the frame directly.
4. Can a label respond to events?
   By default, a JLabel doesn't generate events because it's mainly used to show text or images. However, you can attach a listener, such as a MouseListener, if you want it to respond to clicks or mouse movement.
5. Why the GUI must run on the event-dispatching thread
   All Swing GUI operations need to run on the event-dispatching thread (EDT). This keeps updates and user interactions synchronized. If Swing components were updated from different threads, it could cause unpredictable behavior or even crash the program.
6. Difference between a label and a button
   A JLabel is used to display information like text or images and isn't interactive by default. A JButton, on the other hand, is something the user can click to perform an action—it actively triggers an event when pressed.
7. Ways to control the layout of a content pane

- Use layout managers such as `BorderLayout`, `FlowLayout`, or `GridLayout`.
- Combine multiple panels, each with its own layout, to organize complex interfaces.
- Use absolute positioning with `null` layout and `setBounds()` (though this approach is less flexible and generally not recommended).

13. Using numbers from a text field in a calculation
    Data typed into a JTextField is stored as a String, so it must be converted to a number before being used. This is usually done with methods like `Integer.parseInt()` or `Double.parseDouble()`.
14. Value of num1 in the last statement

```
java

double num1;

Double num2 = new Double(3);

String num3 = "5";

num1 = num2.doubleValue() + Double.valueOf(num3).doubleValue();
```

Here,
`num2.doubleValue()` = 3.0
`Double.valueOf("5").doubleValue()` = 5.0
So, `num1 = 3.0 + 5.0 = 8.0`

15. Best component for selecting a name
    A combo box (JComboBox) is a better choice because it provides a drop-down list of predefined names. This prevents typing mistakes and ensures the input is one of the available options, while a text field could lead to misspellings or inconsistencies.