- **Command-line applications**, interact with the user through simple prompts and feedback but they lack the **rich user experience**
- With GUI, the user is not limited to responding to prompts in a particular order and receiving feedback in one place
- Three kinds of objects are needed to create a GUI in Java
  - components
  - events
  - listeners

- **Component** – defines a screen element used to display information or allow the user to interact with the program
  - A **container** is a special type of component that is used to hold and organize other components
- Java components and other GUI-related classes are defined primarily in two packages
  - java.awt
  - javax.swing

- **buttons** – of course...
- **text fields** – input from keyboard
- **check boxes** – a button that can be toggled on or off using the mouse (indicates a boolean value is set or unset)
- **radio buttons** – to provide a set of mutually exclusive options
- **sliders** – specify a numeric value within a bounded range
- **combo boxes** – select one of several options from a “drop down” menu
Every container is managed by a `layout manager` object that determines how the components in the container are arranged visually.

- Every container has a default layout manager, but we can replace it if desired.
- The layout manager is consulted when needed, such as when the container is resized or when a component is added.
- A layout manager determines the size and position of each component.
- Every layout manager has its own rules and properties governing the layout of the components it contains.
- For some layout managers, the order in which you add the components affects their positioning.

A `BorderLayout` organizes components into five areas (North, South, East, West, and Center).

A `GridLayout` organizes components into a grid of rows and columns.

A `FlowLayout` organizes components from left to right, starting new rows as necessary.

A `BoxLayout` organizes components into a single row or column.

A `TabbedLayout` organizes components into one area such that only one area is visible at a time (think of a tabbed browser window).

```java
public class BorderPanel extends JPanel {
    public BorderPanel() {
        // Set up the layout and background color
        setLayout(new BorderLayout());
        setBackground(Color.green);
        JButton b1 = new JButton("BUTTON 1");
        // Code for the other 4 buttons here
        add(b1, BorderLayout.CENTER);
        add(b2, BorderLayout.NORTH);
        add(b3, BorderLayout.SOUTH);
        add(b4, BorderLayout.EAST);
        add(b5, BorderLayout.WEST);
    }
}
```

```java
public class GridPanel extends JPanel {
    public GridPanel() {
        // Set up the layout and background color
        setLayout(new GridLayout(2,3));
        setBackground(Color.green);
        JButton b1 = new JButton("BUTTON 1");
        // Code for the other 4 buttons here
        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
    }
}
```
The way components are grouped into containers, and the way those containers are nested within each other, establishes the *containment hierarchy* for a GUI.

For any Java GUI program, there is generally one primary (top-level) container, such as a frame or applet.

The top-level container often contains one or more containers, such as panels.

These panels may contain other panels to organize the other components as desired.

---

**Event** – an object that represents some occurrence in which we may be interested
- Often correspond to user actions (mouse button press, key press)
- Most GUI components generate events to indicate a user action related to that component
- Program that is oriented around GUI, responding to user events is called *event-driver*

**Listener** – an object that “waits” for an event to occur and responds in way when it does (aka: event handler)
- In designing a GUI-based program we need to establish the relationships between the **listener**, the **event** it listens for, and the **component** that generates the event.

---

To create a Java program that uses a GUI, we must:
- **set up** ("instantiate") the necessary **components**,
- **implement listener** classes that define what happens when particular events occur, and
- **Establish relationship between listeners and components** that generate the events of interest.