In Java GUI, a container is a component that can contain other components, such as buttons, text fields, and labels. Containers provide a way to organize and group related components together. Examples of containers include JFrame, JPanel, JDialog, and JWindow.

On the other hand, a frame is a top-level container that provides a window for displaying other components. JFrame is the most commonly used frame in Java GUI programming. It provides a title bar, borders, and other decorations that make it look like a traditional desktop window. A frame can contain other components such as buttons, text fields, and labels.

In summary, a container is a component that can contain other components, while a frame is a top-level container that provides a window for displaying other components.

In Java Swing, getContentPane() is a method provided by the JFrame class that returns the content pane of the frame. The content pane is a container that is used to hold the visible components of the frame, such as buttons, labels, text fields, and other GUI components.

When we add components to a JFrame object, we actually add them to its content pane, which is an instance of Container class. The getContentPane() method allows us to get a reference to the content pane so that we can add or remove components from it.

Here's an example of how to use getContentPane() method to add a button to a JFrame:

```
import javax.swing.*;

public class MyFrame extends JFrame {

   public MyFrame() {
      // set the title of the frame
      setTitle("My Frame");

      // get the content pane of the frame
      Container contentPane = getContentPane();

      // create a button and add it to the content pane
      JButton button = new JButton("Click Me");
      contentPane.add(button);

      // set the size and visibility of the frame
      setSize(300, 200);
      setVisible(true);
   }

   public static void main(String[] args) {
      // create an instance of the frame
      MyFrame frame = new MyFrame();
   }
}
```

setBounds() is a method available in both **JFrame** and other **JComponent** classes like **JButton**. The **setBounds()** method is used to set the position and size of a component.
However, the **setBounds()** method on a **JFrame** is used to set the position and size of the **JFrame** itself on the screen, while **setBounds()** on a **JButton** or any other **JComponent** is used to set the position and size of that particular component within its container.
In a **JFrame**, **setBounds()** sets the size and position of the frame on the screen, including its title bar, border, and any other decorations. On the other hand, in a **JButton**, **setBounds()** sets the position and size of the button within its parent container, such as a **JPanel** or a **JFrame**.
Therefore, the **setBounds()** method on a **JFrame** is used to set the size and position of the frame itself, while **setBounds()** on other components like **JButton** is used to set the position and size of that particular component within its parent container.