

## Chapter 11 Getting Started with GUI Programming

1. `java.awt.Component` is the root of all Java GUI component classes. A container class such as `JFrame` is a subclass of `Component`. `JComponent` is the root of Swing GUI component classes. You could add components into a button. It is legal, but a button should not be used as a container.

2. The AWT components are heavy weight while the Swing components are lightweight.
3. You use the constructor of the `JFrame` class to create a frame. Use the `setSize` method to set the size of the frame. Use the `getSize` method to find the size of the frame. if the statement `frame.setSize(400, 300)` and `frame.setVisible(true)` were swapped in the `MyFrameWithComponents` class, you have to resize the frame to display the components in the frame. This is because the `setVisible(true)` statement causes the container to be repainted, but the `setSize(400, 300)` doesn't. When you resize the window, all the components are repainted in the frame.

4. You can add a button to a frame.

**Answer:** True

You can add a frame to a panel.

**Answer:** False

You can add a panel to a frame.

**Answer:** True

You can add any number of components to a panel, to a frame, or to an applet.

**Answer:** True

You can derive a class from `JPanel`, `JFrame`, or `JApplet`.

**Answer:** True

5. Line 7 created an instance of `JFrame`. It should be `new Test()` instead.
6. Two errors: (1) constructor `TestDrawMessage` cannot have void. (2) `PaintComponent` should be `paintComponent`.
7. The layout manager provides a platform-independent way to place components in a GUI interface. The default layout manager for the content pane of a frame is `BorderLayout`. To add a component to a frame, you have to add it to the content pane of the frame.
8. Using `FlowLayout`, the components are arranged in the container from left to right in the order in which they were added. If one row becomes filled, a new row is

started. To use a FlowLayout manager, you need to set the layout in a container to FlowLayout, such as with `setLayout(new FlowLayout())`. There is no limit on the number of components that can be added to a FlowLayout container.

9. The GridLayout manager arranges components in a grid (matrix) formation with the number of rows and columns defined by the constructor. The components are placed in the grid from left to right, starting from the first row, then the second, and so on, in the order in which they were added. To use a GridLayout manager, you need to set the layout in a container to GridLayout, such as with `setLayout(new GridLayout())`. The number of components you can add to a GridLayout container is unlimited. (Please try adding more components into the container, to see what happens when the number of components exceeds the grid size. Interestingly, the column is enlarged.)
10. The BorderLayout manager divides the window into five areas: East, South, West, North, and Center. Components are added to a BorderLayout using `add(String, Component)`, where String is "East", "South", "West", "North", or "Center". You can use one of the following two constructors to create a new BorderLayout:

```
public BorderLayout(int hGap, int vGap)
public BorderLayout
```

You can add only one component into a section. If you need to add multiple components in a section, group the components in a panel, and add the panel into the section.

11. Use the `getBackground()`, `getForeground()`, `getFont()`, `setBackground(Color)`, `setForeground(Color)`, `setFont(Font)` methods to get and set background, foreground, and font of a component.  
  
Use the `setColor(Color)` method to set the color and the `setFont(Font)` method to set the color and font of a component . The `getColor()` and `getFont()` methods get the current color and font on a Graphics object.
12. Use the new `Color(int, int, int)` to create a color or use a standard color such as `Color.RED` to specify a color. Use new `Font(name, style, size)` to create a new font.
13. Use new `JPanel(LayoutManager)`
14. The default layout manager for a JPanel is FlowLayout. Components are added directly to the JPanel.
15. No. There is no `setTitle()` method in JPanel. This method is defined for Frame; Panel is used to organize components. The panel itself is invisible.

16. The y coordinate should increase and the x coordinate should remain unchanged.
17. The `paintComponent()` method is defined in the `Component` class. The Java runtime system invokes it to paint things on `JFrame`, `JApplet`, and `JPanel`. This method cannot be invoked by the system or by the programmer. The system automatically invokes it whenever the viewing area changes. The programmer invokes it through invoking the `repaint()` method. The programmer should never directly invoke the `paintComponent()` method.
18. See the section "Drawing Geometric Figures."
19. Draw a thick line from (10, 10) to (70, 30). You must draw several lines next to each other to create the effect of one thick line.

**Answer:**

```
for (int i = 0; i < 10; i++)  
    g.drawLine(10, 10 + i, 70, 30 + i);
```

Draw a rectangle of width 100 and height 50 with the upper-left corner at (10, 10).

**Answer:**

```
g.drawRect(10, 10, 100, 50);
```

Draw a rounded rectangle with width 100, height 200, corner horizontal diameter 40, and corner vertical diameter 20.

**Answer:**

```
g.drawRoundRect(10, 10, 100, 200, 40, 20);
```

Draw a circle with radius 30.

**Answer:**

```
g.drawOval(10, 10, 60, 60);
```

Draw an oval with width 50 and height 100.

**Answer:**

```
g.drawOval(10, 10, 50, 100);
```

Draw the upper half of a circle with radius 50.

**Answer:**

```
g.drawArc(10, 10, 100, 100, 0, 180);
```

Draw a polygon connecting the following points: (20, 40), (30, 50), (40, 90), (90, 10), (10, 30).

**Answer:**

```
int x[] = {20, 30, 40, 90, 10};  
int y[] = {40, 50, 90, 10, 30};  
g.drawPolygon(x, y, x.length);  
g.fillPolygon(x, y, x.length);
```

Draw a 3D cube like the one in Figure 8.28.

**Answer:**

```
public void paintComponent(Graphics g) {  
    //draw the front rect  
    g.drawRect(10, 60, 40, 40);  
  
    //draw the back rect  
    g.drawRect(30, 40, 40, 40);  
  
    //connecting the corners  
    g.drawLine(10, 60, 30, 40);  
    g.drawLine(10, 100, 30, 80);  
    g.drawLine(50, 60, 70, 40);  
    g.drawLine(50, 100, 70, 80);  
}
```