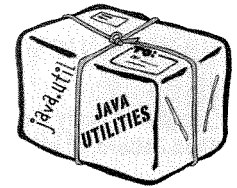


Swingin' with Frames



Up next on our Template Method safari... keep your eye out for swinging JFrame's!

If you haven't encountered JFrame, it's the most basic Swing container and inherits a `paint()` method. By default, `paint()` does nothing because it's a *hook*! By overriding `paint()`, you can insert yourself into JFrame's algorithm for displaying its area of the screen and have your own graphic output incorporated into the JFrame. Here's an embarrassingly simple example of using a JFrame to override the `paint()` hook method:

```
public class MyFrame extends JFrame {
    public MyFrame(String title) {
        super(title);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        this.setSize(300,300);
        this.setVisible(true);
    }

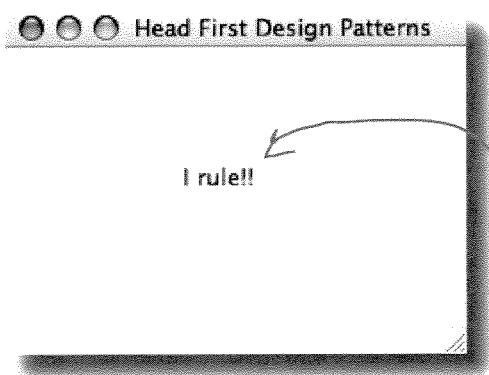
    public void paint(Graphics graphics) {
        super.paint(graphics);
        String msg = "I rule!!";
        graphics.drawString(msg, 100, 100);
    }

    public static void main(String[] args) {
        MyFrame myFrame = new MyFrame("Head First Design Patterns");
    }
}
```

We're extending JFrame, which contains a method `update()` that controls the algorithm for updating the screen. We can hook into that algorithm by overriding the `paint()` hook method.

Don't look behind the curtain! Just some initialization here...

JFrame's update algorithm calls `paint()`. By default, `paint()` does nothing... it's a hook. We're overriding `paint()`, and telling the JFrame to draw a message in the window.



Here's the message that gets painted in the frame because we've hooked into the `paint()` method.