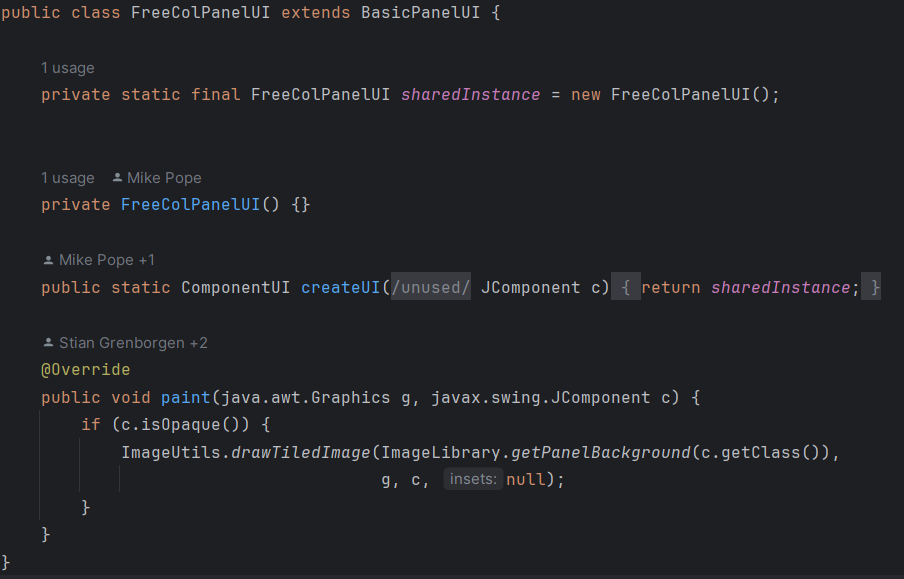
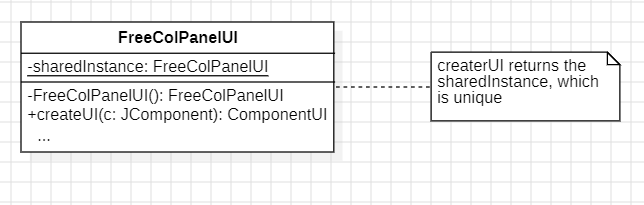
**DESIGN PATTERNS**

Singleton

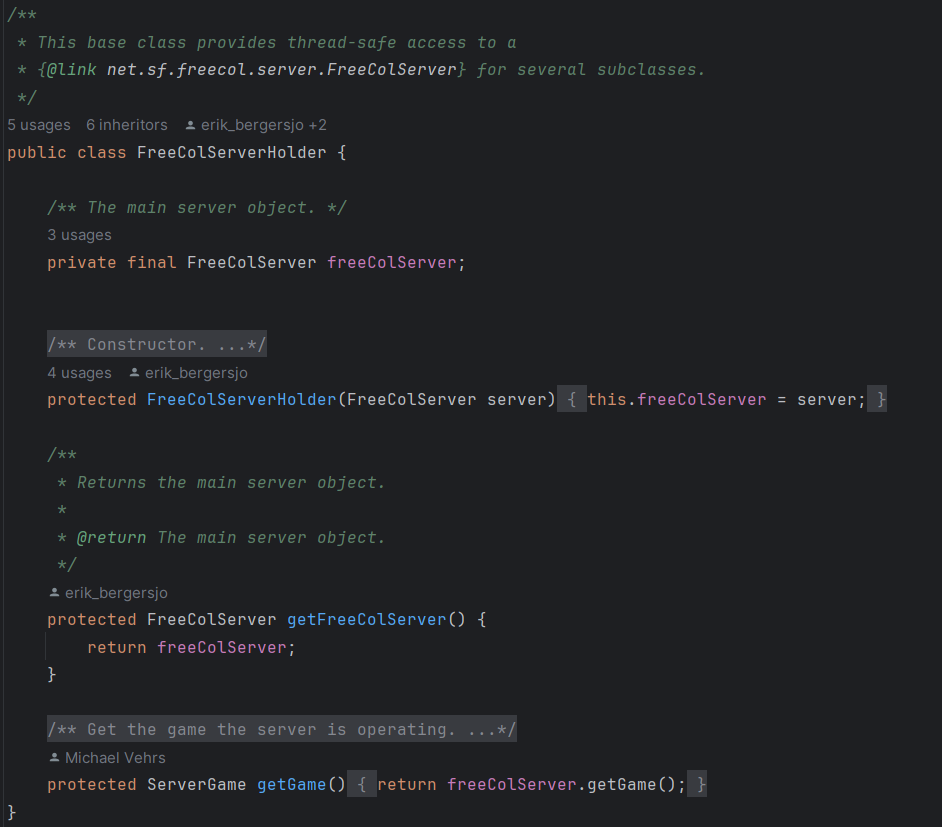
This design pattern is implemented on the class, net.sf.freecol.client.gui.plaf.FreeColPanelUI.java.

The class has a private constructor, “private FreeColPanelUI() {}”, which means that instances of this class can only be created from within the class itself. Then also there is a private static instance variable named “sharedInstance” that holds the single instance of the FreeColPanelUI class. It is final, so it cannot be reassigned once it's set.

The Class Diagram of this pattern is :

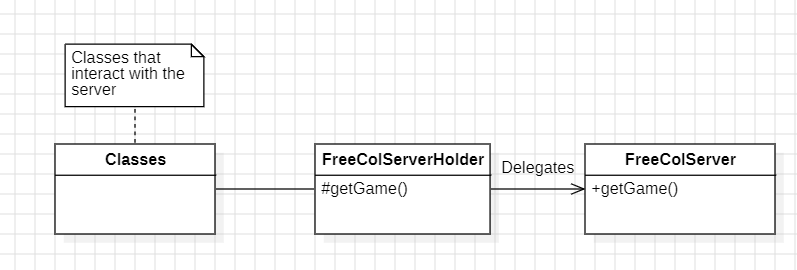


Proxy

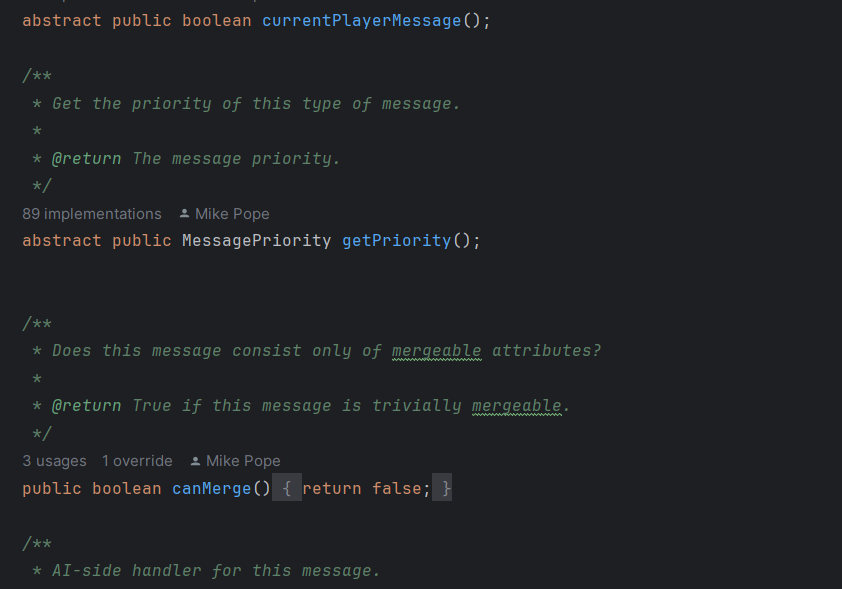
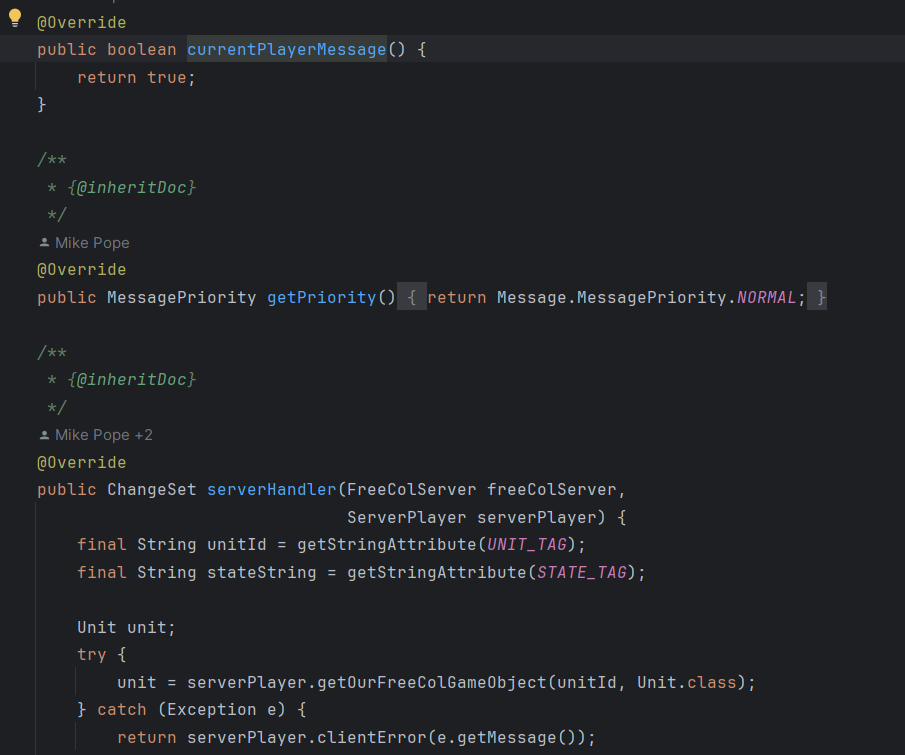


This design pattern is implemented in scr.net.sf.freecol.server.control.FreeColServerHolder

The class FreeColServerHolder acts as a proxy, and to be more specific, as a **protection proxy**.  
This is when a proxy class is used to control access to the real subject class, which in this case is FreeColServer.

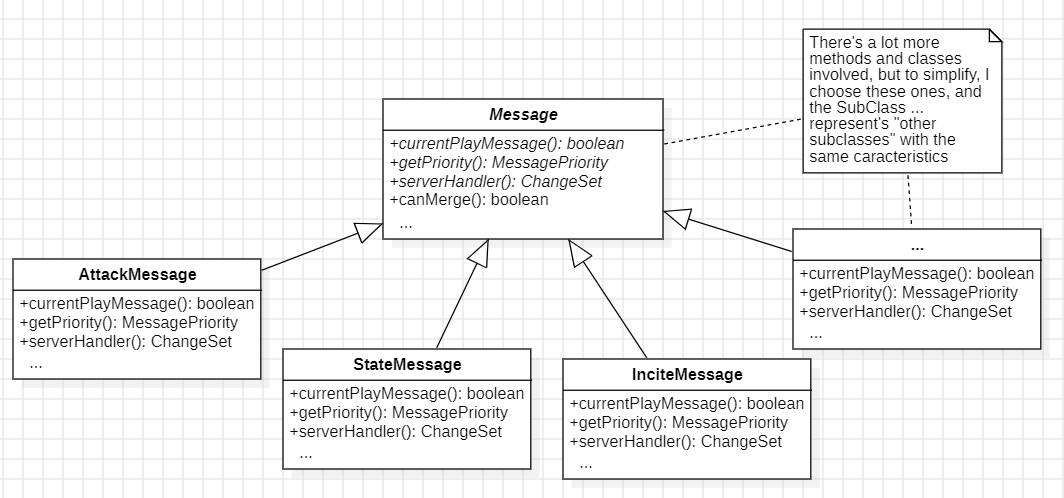
The Class Diagram of this pattern is:

Template

  
…  
  
… (the class keeps going but we’ll focus on a simpler example)  
From ChangeStateMessage.java  


This design pattern is found in scr.net.sf.freecol.common.networking.Message and it’s subClasses (for example scr.net.sf.freecol.common.networking.ChangeStateMessage.

The class Message is the general class that has both generic message related methods, and abstract methods to be implemented by the subclasses.

The Class Diagram for this pattern is (the real diagram would be too big, so I made a simpler and understandable version) :  


André Branco nº62482