Façade Pattern

In this lab, you will create a Façade interface.

# Objectives

In this lab, you will

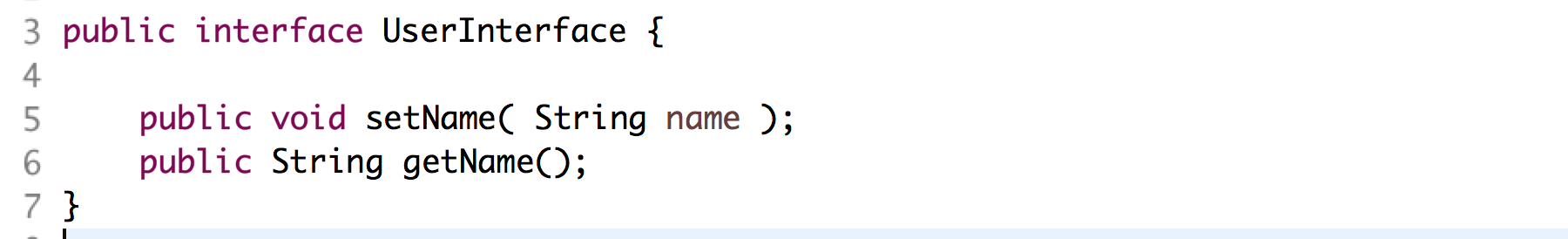
* Change the user name in an app
* Two different frameworks must change to change the name
* Create a Façade to change the user name
* Run the app

# Exercise

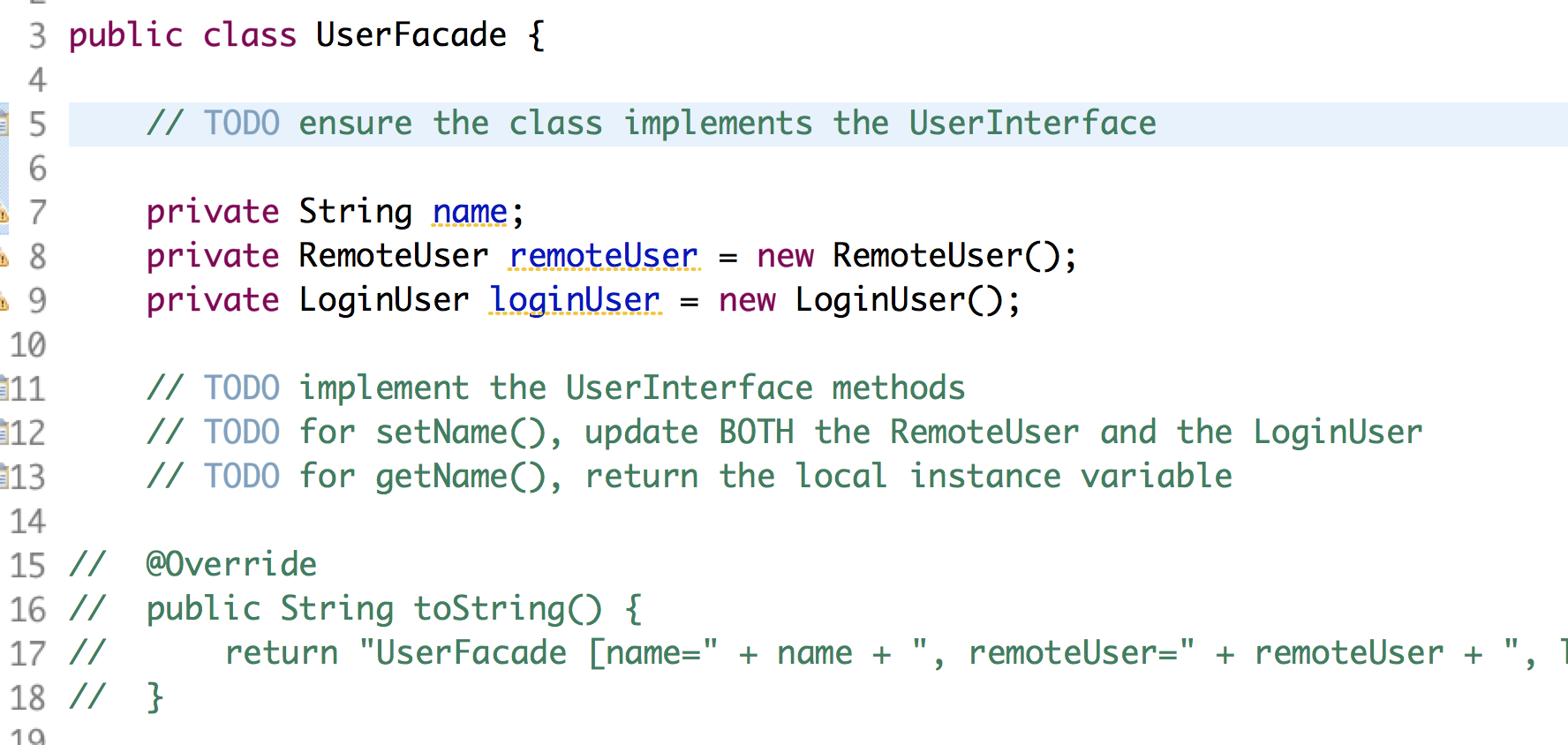
1. The Façade pattern is much like the Adapter pattern. Instead of changing the API to match a local one like the Adapter, the Façade pattern provides a simpler interface to other frameworks. In this example, the client wants to change the user name. The application requires changes in multiple places. The Façade pattern hides the complexity of the changes making the code much easier to reuse.
2. In Eclipse, in the exercises project, open the package com.paypal.patterns.Facade to view the project files.
3. Examine Tester.java shown below:



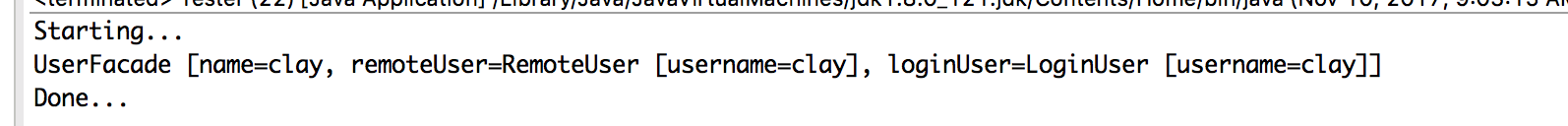
1. It creates the UserFacade and changes the user name. Very simple interface. In real life, we would probably go to a Factory somewhere to get the Façade. This allows the application to change the Façade implementation based on the environment or the role of the user.
2. In this case, we have two different representations of the user, the RemoteUser and the LoginUser. The Façade must set the user name in both.
3. The RemoteUser and the LoginUser are two simple Java beans around the username String. The Façade must call setUsername() on each of them when the client changes the name with Façade.setName().
4. Examine the UserInterface.java below:



1. The UserFacade implements the above interface.
2. Edit the UserFacade.java shown below:



1. On line 3 above, change the class definition to implement the UserInterface.
2. From lines 11-13, implement the UserInterface methods then uncomment the toString() method.
3. Uncomment the lines in Tester.java and execute the file as a Java application. You should see:



Congratulations. You have completed this lab.