Observer Pattern

In this lab, you will register an Observer on an object, change the object, and notice the change in the Observer.

# Objectives

In this lab, you will

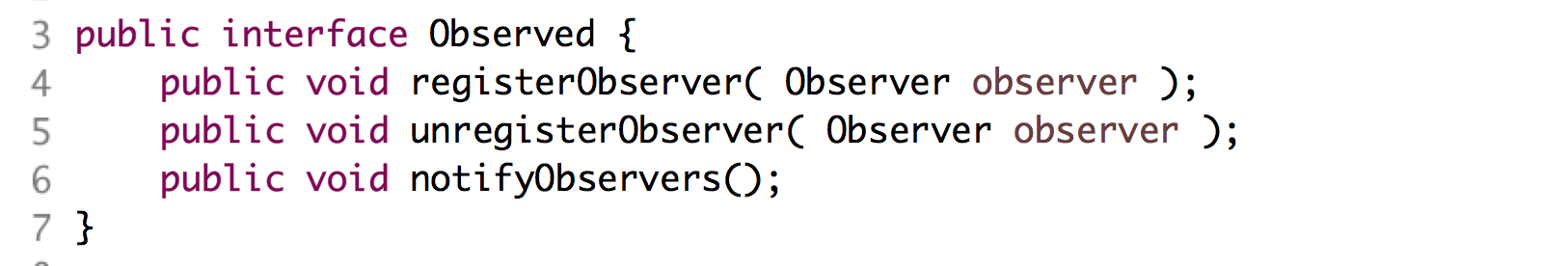
* Create an Observable object
* Create an Observer object
* Modify the Observable
* Notice the change in the Observer

# Exercise

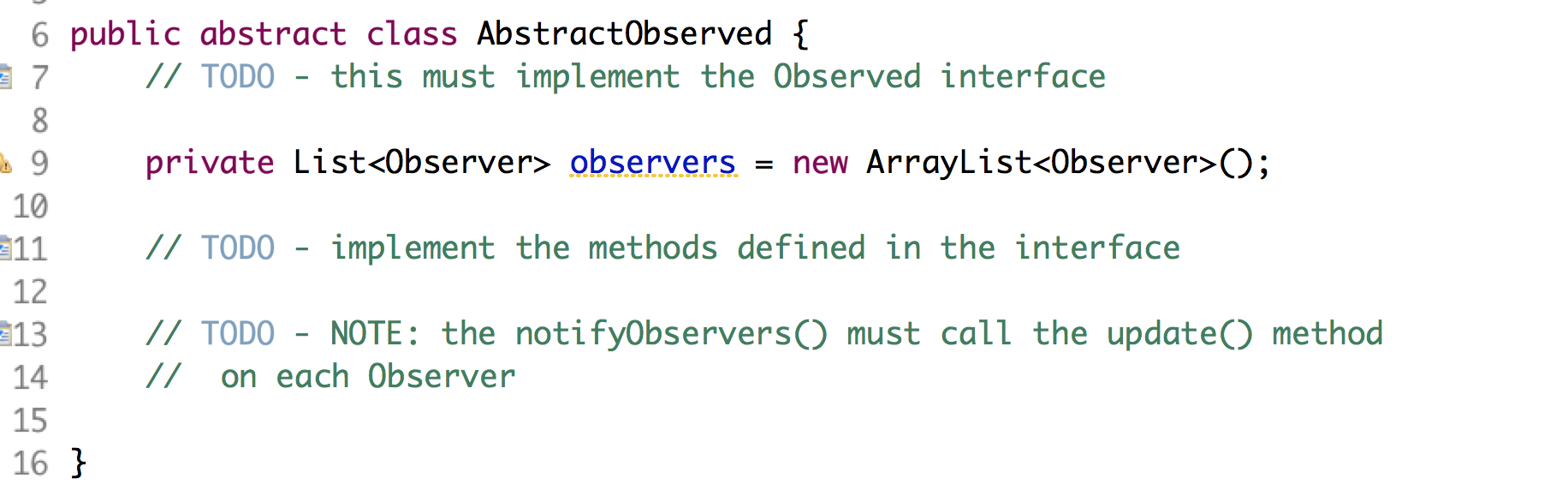
1. Using the Observer Pattern allows one part of a program to monitor another part’s state. When that state changes, the Observer part is notified.
2. In this example, we want the User objects to inform the UserObserver by calling the update() method, whenever the User state changes.
3. In Eclipse, in the exercises project, open the package com.paypal.patterns.Observer to view the project files.
4. Examine the client, Tester.java, as shown:



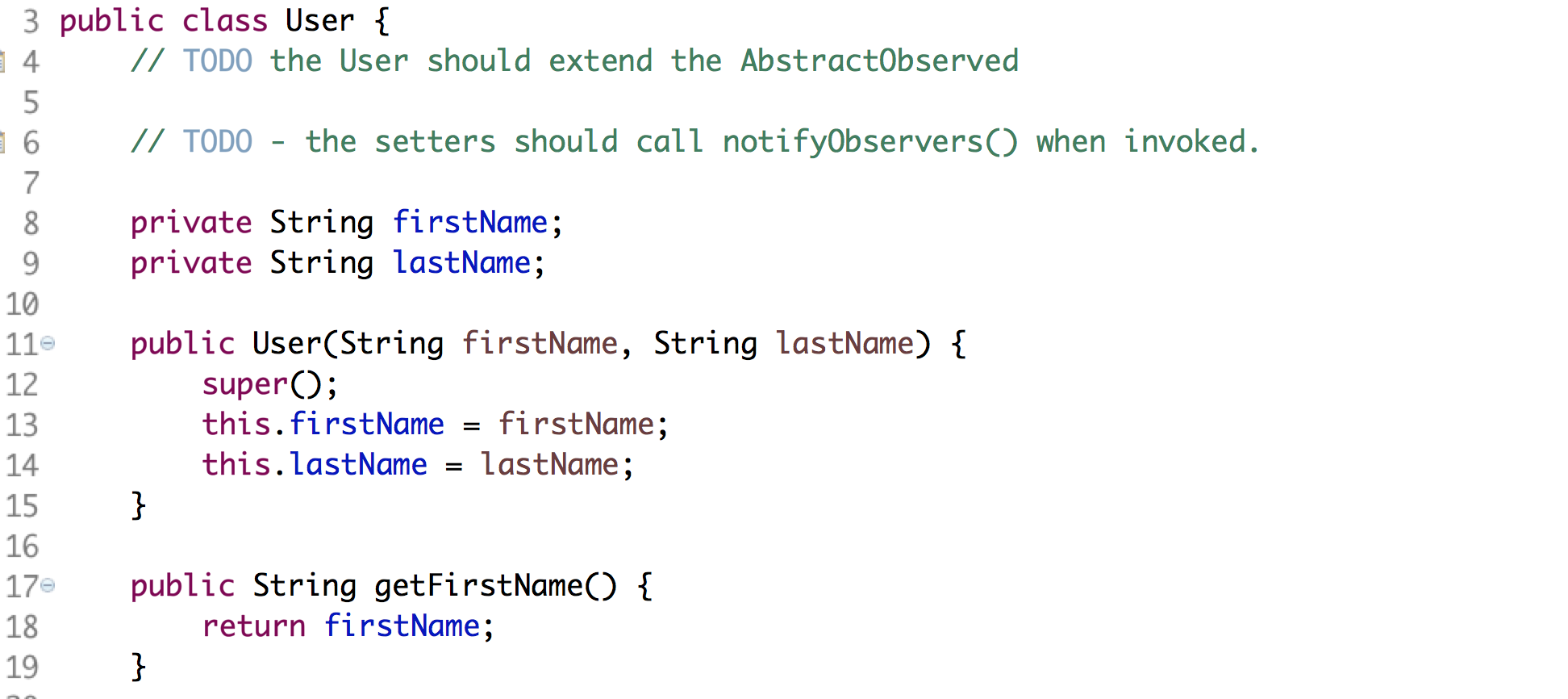
1. In lines 10-11, it creates two new User objects.
2. In lines 12-13, it registers those new User objects with the UserObserver with the commented lines.
3. In lines 15-19, it changes the state of the User objects. The UserObserver should be informed of the changes.
4. In this exercise, the User class is NOT being Observed. Our task is to make it observed so we can listen for changes.
5. Examine the Observered.java file as shown:



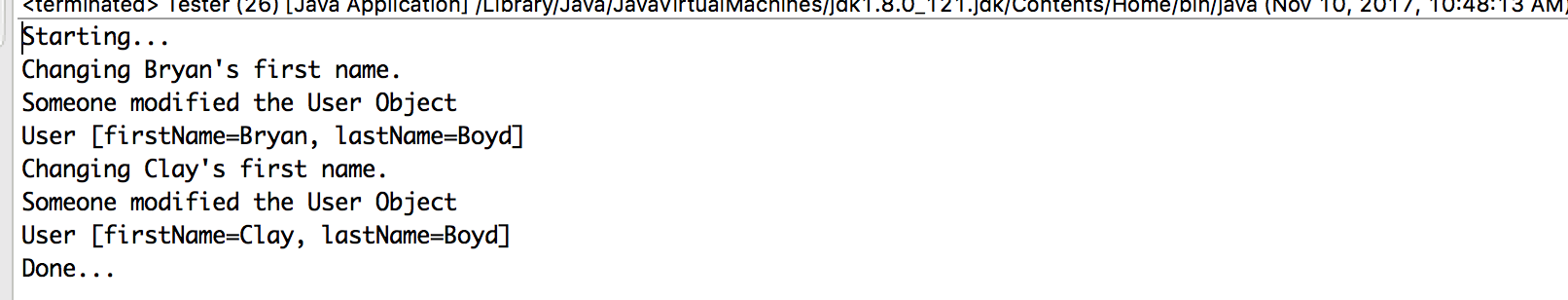
1. Realizations of the above interface define a List of Observers and a method to notify the Observer when a change occurs.
2. Edit the AbstractObservered.java file as shown below:



1. Write the code described above.
2. Now, we must make the User POJO into an Observed by sub classing the AbstractObserved above.
3. Edit the User.java file as shown below:



1. The above starts out as a simple POJO. Extend the class and modify the setters.
2. Uncomment the lines in Tester.java and execute the file as a Java application. You should see:



Congratulations. You have completed this lab.