Adapter Pattern

In this lab, you will create an adapter to access a different API.

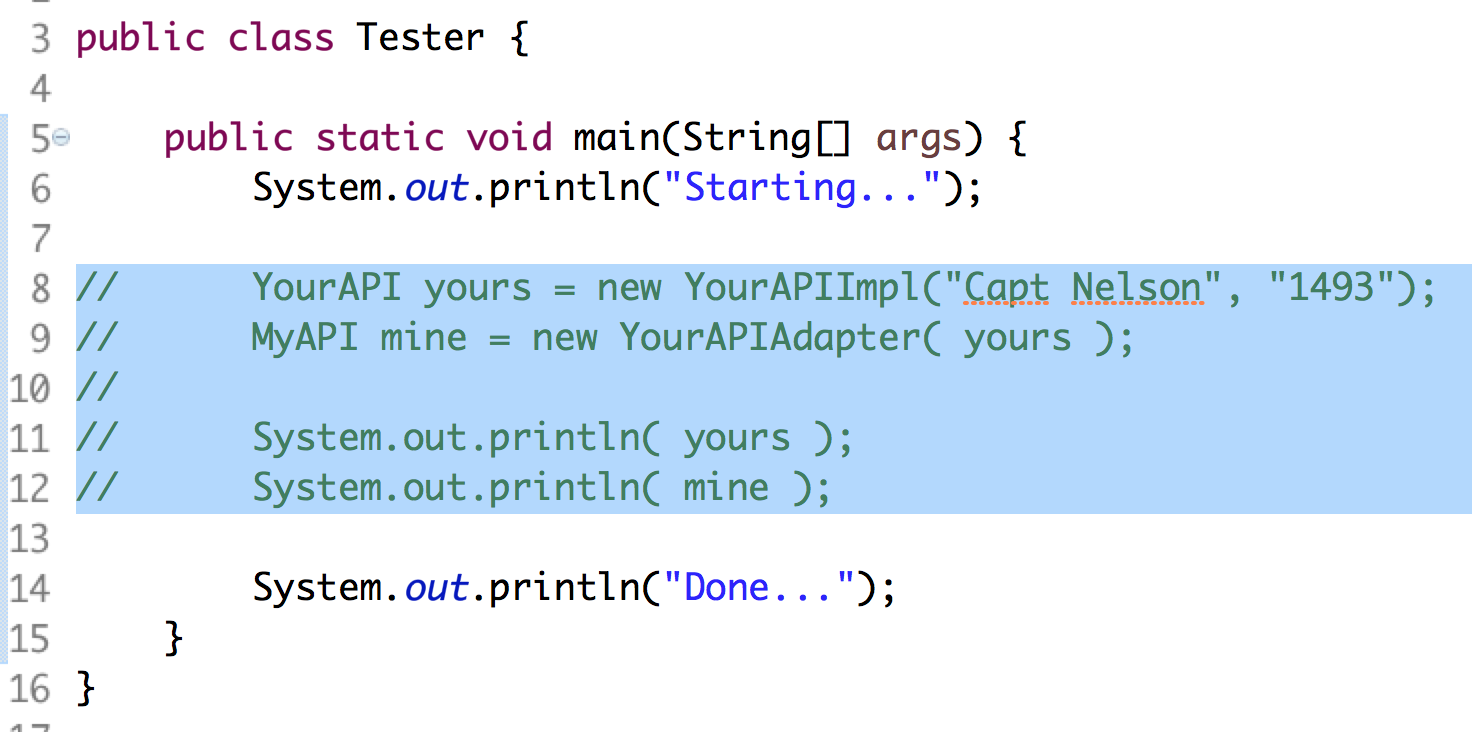
# Objectives

In this lab, you will

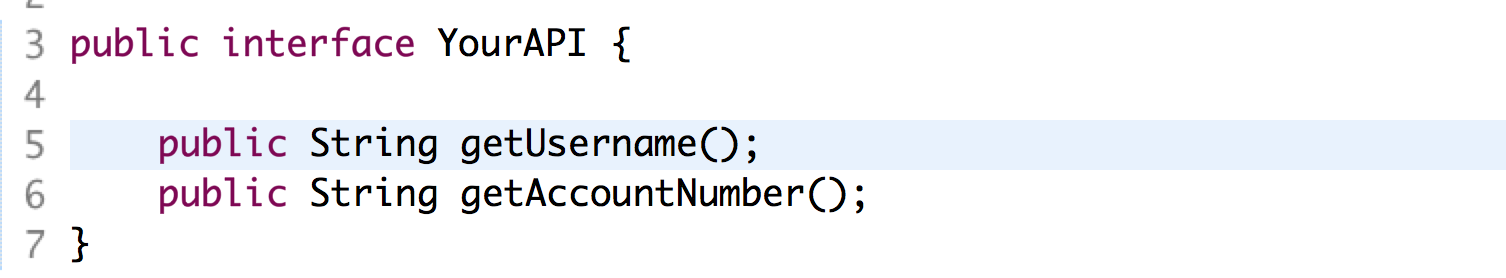
* Run the app

# Exercise

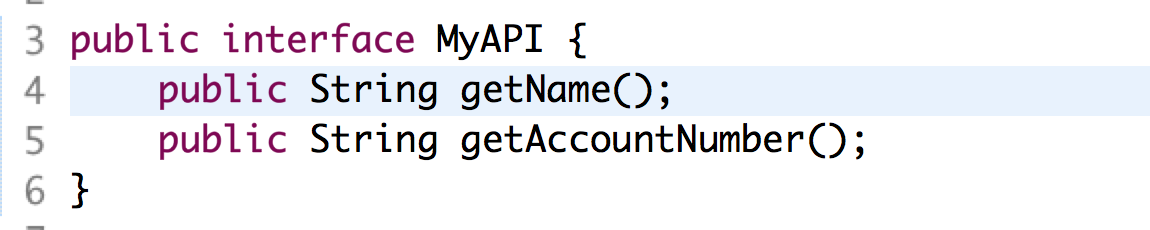
1. The Adapter pattern represents the interface code to change a target API to more nearly match a different API. This is extremely useful when different target APIs perform similar functions but use different methods to achieve their results.
2. In Eclipse, in the exercises workspace, open the package com.paypal.patterns.Adapter to view the project files.
3. Open Tester.java as shown below:



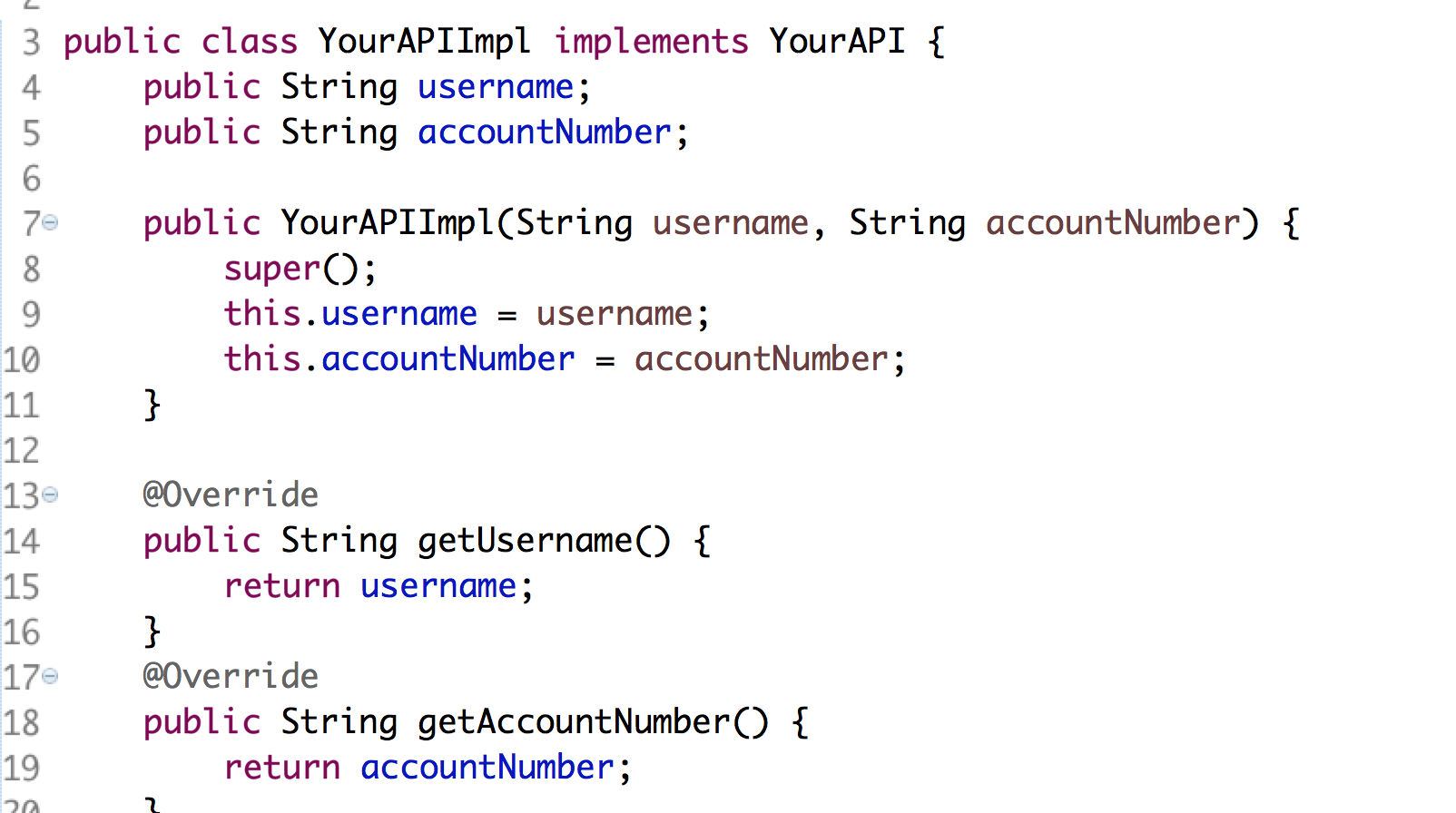
1. In the above, we get an instance of YourAPI and pass it to the adapter which returns an instance of MyAPI.
2. As shown in the following two images, the only difference between MyAPI and YourAPI is the method for retrieving the user name field.
3. Open YourAPI.java as shown below:



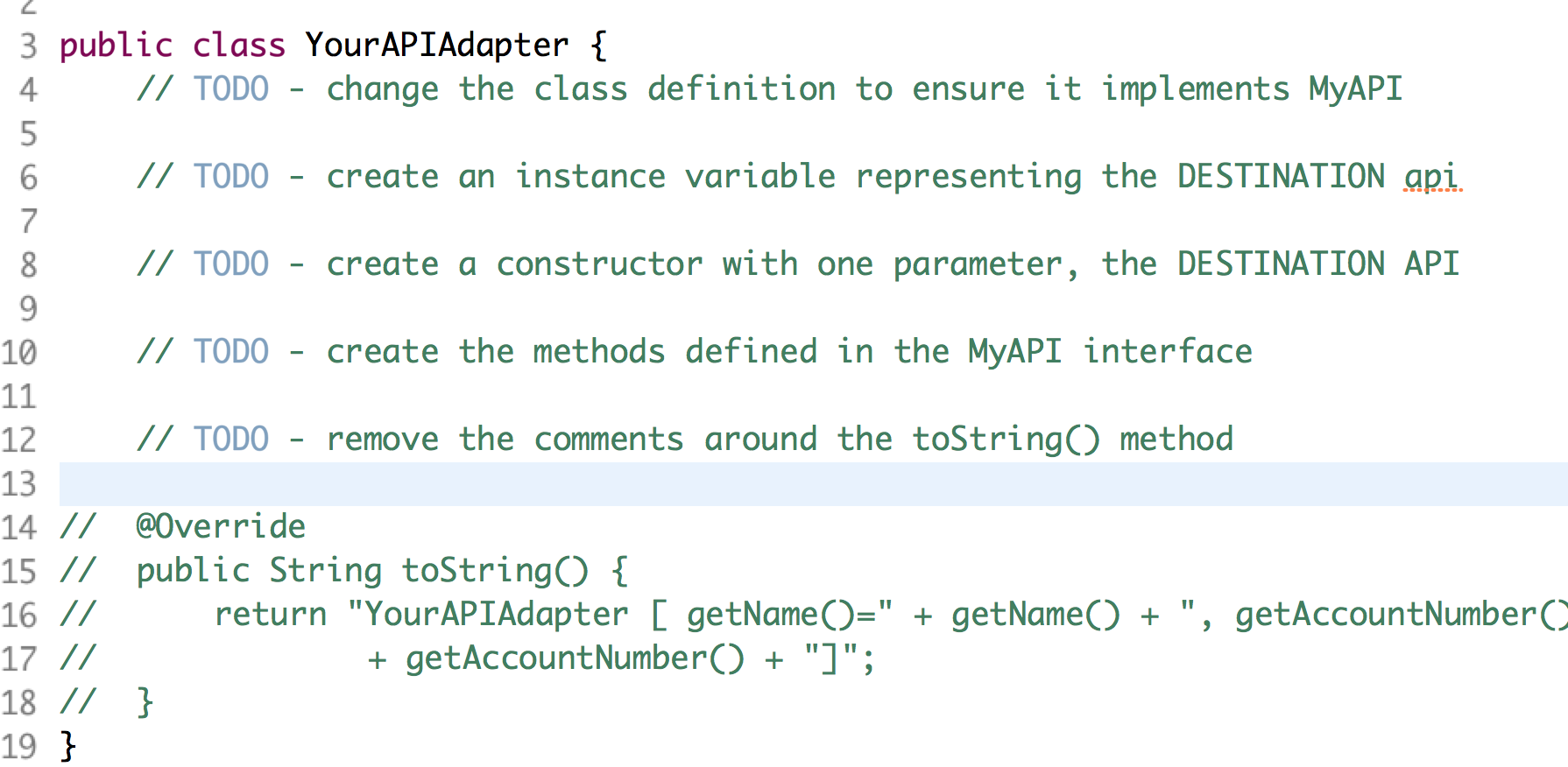
1. Open MyAPI.java as shown below:



1. Open YourAPIImpl.java as shown below:



1. The above shows the YourAPI code which is just a Java Bean.
2. Open YourAPIAdapter.java as shown below:



1. The adapter will intercept the calls to MyAPI and forward them to the YourAPI instance stored as an instance variable. Sometimes, this is called delegation.
2. From line 4 above, ensure the class implements the MyAPI interface.
3. From line 6, create an instance variable representing an instance of the YourAPI interface.
4. From line 8, create a constructor with one parameter, the instance of YourAPI. The caller will grab an implementation of YourAPI and pass it to this class.
5. From line 10, create the methods defined in the MyAPI interface. For each method, forward the calls to the YourAPI instance.
6. Remove the comments around the toString() method.
7. Remove the comments in the Tester.java file and execute it as a Java application.

Congratulations. You have completed this lab.