

CISC 181 Spring 2014
Practice Set 2
Assigned: February 24
Due: March 2 at 11:55PM on Sakai

Practice sets are to be completed individually. You are free to consult other students to help complete the practice sets (see syllabus for collaboration policy). However, keep in mind that each practice set is designed to cover basic material on which you will be quizzed and tested.

This practice set is intended to cover the following major topics:

- Writing a JUnit Test
- Objects, Classes
- Encapsulation
- Static vs Instance Methods

Please consult chapters 8, 10, and 11 of your textbook, your class notes and the Transition Guide for additional examples that will help to solve these problems.

Note: For all practice sets you should create a project and add the JUnit library to it:

- Create a Java project named PS2
- Right click your project and click Properties (or go to menu Project->Properties). Choose Java Build Path on the left, and then the Libraries tab on the right. Choose Add Library... then JUnit, Next, choose version 3 and then Finish.

1. **[15 points]** Create a class named `Rectangle` to represent a rectangle. The class contains:
 - Two double data fields named `width` and `height` that specify the width and height of the rectangle.
 - A no-arg constructor that creates a default rectangle with 1 for both width and height.
 - A constructor that creates a rectangle with a specified width and height.
 - public accessor methods for each property
 - A method named `getArea()` that returns the area of the rectangle.
 - A method named `getPerimeter()` that returns the perimeter.
2. **[5 points]** Download the `TestRectangle.java` file from Sakai (make sure to right-click/Save As into your code folder -- do not copy paste the text). You may need to refresh Eclipse's view of your project files (right-click your Project in Eclipse and choose Refresh).

Run this program as a JUnit test in Eclipse (right click the file and choose Run As...-> JUnit test). You should see that one of the tests fails. Change it so it is successful. Note that the `assertEquals` takes 3 arguments, the expected value, a test value, and a

tolerance.

3. **[10 points]** Write a static method in your Rectangle class called makeGoldenRectangle that takes as a parameter a double representing the height and returns a newly created Rectangle that has that height and whose ratio of width to height is the golden ratio (http://en.wikipedia.org/wiki/Golden_ratio).
 - Write a new test in the TestRectangle class that tests several inputs of your makeGoldenRectangle static method. Since the result of makeGoldenRectangle is a Rectangle, you need to assign the result to a local variable and then use accessor methods to test each property. For example, a golden rectangle created with height=1.618 should be tested to see if it has a width=2.618 and a perimeter=8.472.
4. **[35 points]** Create a class named Account that contains:
 - A private String data field for a customer name
 - A private double data field for balance
 - A private double data field for monthly interest rate
 - A constructor that initializes all 3 data fields
 - An accessor method to getBalance
 - A mutator method to deposit a given double amount into the balance
 - A mutator method to withdraw a given double amount from the balance. This method should return false (without making a withdrawal) if the amount requested is more than the balance, otherwise it should return true indicating the withdrawal was successful.
 - A mutator method that will update the balance by adding monthly interest. For example, an account with a balance of 100 and a monthly interest rate of 0.005 should have a balance of 100.5 after adding monthly interest.
 - Download TestAccount.java file from Sakai and place it in your PS2 project src folder. Currently it tests the Account using a series of deposits, withdrawals, getBalance, and addMonthlyInterest method calls. Add 3 additional test methods that test each mutator method individually.
5. **[35 points]** Modify the Account class to support "credit" style accounts. If modified correctly, the earlier account tests will still succeed.
 - Add a new property for a balance limit that will be represented as a negative number. Add a new constructor that allows for all 4 properties to be initialized. Make the existing constructor call your new constructor with a default value of 0 for the limit.
 - Modify your withdraw method to allow a withdrawal as long as the balance would remain above the limit.
 - Add new test cases to your test_withdraw to test an Account that uses a negative limit to allow for negative balance.

- Add a method, `payoffBalance`, that takes another `Account` as a parameter, and pays off as much of the debt on that `Account` as possible. Uncomment the test in `TestAccount` and look at the expected results to help explain the logic of this method.

Turn in to Sakai your entire archived Eclipse PS2 project as follows:

1. Select project
2. File-> Export -> General -> Archive File
3. Click Browse to find place on your computer to place archive
4. Upload archive to Sakai

You should be uploading a single archive file and then clicking “submit”.