Computer Science, Claremont McKenna College
CS51 - Introduction to Computer Science, Fall 2014

Problem Set 9 Due: 11:55 PM, April 12, 2015

General Instructions

Please carefully read and follow the directions exactly for each problem. Files and classes should be named exactly as directed in the problem (including capitalization!) as this will help with grading.

You should create your programs using your preferred text-editor, the Eclipse text editor, or jGrasp. Do not use a word processor such as Microsoft Word, WordPad, Google Docs, Apple's Pages, etc...

Your programs should be formatted in a way that's readable. In other words, indent appropriately, use informative names for variables, etc… Points will be deducted if your indenting style makes it difficult to follow your code. If you are uncertain about what is a readable style, look at the examples from class as a starting point for a reasonable coding style. .

Your programs should compile and run without errors. Please do not submit programs that do not compile! It's better to submit partial implementation that compiles as oppose to implementations that do not compile.

At the top of each file you submit, you should put your name and your email as comments.

This homework set should be submitted via Sakai using the Assignment Menu option (on the left pane). You should only submit the requested ".java" files. Do not submit ".class" files or Eclipse-specific project files. Finally, please do not submit files using Sakai's Dropbox Menu.

Turn in the following file(s):

- Point.java
- UsePoint.java
- Rectangle.java
- TestRectangle.java
- Message.java

Problem 1

In this warmup problem, you will recreate the `Point` class we looked at in class and add the following additional instance methods:

- `public int manhattanDistance(Point other)`

  Returns the "Manhattan distance" between the current Point object and the given `other` Point object. The Manhattan distance refers to the distance between two places if one can travel between them only moving horizontally or vertically as though driving on the streets of Manhattan. In our case, the Manhattan distance is the absolute value of the difference in x plus the absolute value of the difference in y.

- `public double slope(Point other)`

  Returns the slope of the line drawn between this Point and the other Point given as a parameter. The slope is undefined for points with identical x-coordinates so throw an `IllegalArgumentsException` in this case.

Supply a main method in `UsePoint.java` that tests your methods with several examples. Do not use a Scanner to query for input from the user.

Problem 2

Write a class called `Rectangle` that represents a rectangular two-dimensional region using x,y coordinates for the upper left corner and a width and a height. Add the following constructors, instance methods and any needed fields:

- `public Rectangle()`

  A default constructor that initializes all fields to zero

- `public Rectangle(int x, int y, int width, int height)`

  A constructor that initializes a new Rectangle object with the given x,y parameters as the top-left corner of the rectangle with the given width and height.

- `public Rectangle (Point p, int width, int height)`

  A constructor that initializes a new Rectangle object with the given Point object as the top-left corner of the rectangle with the given width and height.

- `public int getHeight() and public int getWidth()`

  Returns the height or width of the rectangle

- `public int getX() and public int getY()`

  Returns the x or y coordinate of the upper left corner of the rectangle..

- `public String toString()`

  Returns a String that contains information about the rectangle.

- `public boolean equals(Rectangle other)`

  Returns `true` if the `other` rectangle has the same x, y coordinates and the same width, width. Returns false otherwise.

- `public boolean contains(int x, int y)`
  `public boolean contains(Point p)`

  Returns true if the point defined by the paramters x,y or the Point p is inside the rectangle.

- `public Rectangle union(Rectangle rect)`

  Returns a new Rectangle object that represents the rectangular region that's the tightest bounding box that contains both the implicit parameter rectangle and the given Rectangle rect.

Test your implementations in a file called `TestRectangle.java` by calling <u>all</u> of the above methods and printing out informative messages indicating the results of each method call. Do not use a Scanner to query for input from the user.

Problem 3

In this problem, you will practice reading an example program to determine what instance a particular class has or needs to have. In this case, you are given the example program `UseMessage.java` that tests the methods of the `Message` class which you are to implement. The `Message` class represents an email message and your implementation should have the following fields (you decide whether they should be `public` or `private`):

- `String from`
- `String to`
- `String date`
- `String subject`
- `int length`
- `String body`

You should read `UseMessage.java` carefully to see what instance methods in the Message class need to be implemented. Most of the requirements for the methods to be implemented are straightforward to understand except for the boolean `isImportant` method. This method simply returns `true` if the `from` field contains the Strings "theDean" and either "cmc", "pitzer", "pomona", or "scripps"; otherwise, it returns `false`. Do not change UseMessage.java.

You are also given the file `sample_output.txt` that shows what the output of `UseMessage.java` looks like. Your output should match this file or be very similar to it in content.