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

Problem Set 3 Due: 11:55 PM, Feb 15, 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… 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.

This problem set assumes you have installed Java and an IDE (Eclipse or jGrasp) on your system. Please see the course web page on Sakai for installation instructions.

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):

```
Cone.java
FirstDigit.java
Harmonic.java
WraparoundSquare.java
```

Directions

In this problem set, you will practice writing: 1) methods with parameters and 2) methods that return a value. These methods will also utilize expressions and/or for loops. For each problem, you should also write a main method that calls and tests the specified methods. For problem 1, you should utilize a Scanner object in the main method to retrieve parameters input via the keyboard. You should print out informative text when querying the user for parameters. For problems 2 – 4, do not use a Scanner object. Instead, in your main method call your methods with the examples mentioned in the problems AND add some (1 or 2) extra test cases as well.

Problem 1

In a class called `Cone`, write a method called `coneSurfaceArea` that accepts a radius and height as parameters and returns the surface area of a cone with those dimensions For example:

- `coneSurfaceArea`(3.0, 4.5) should return: 79.24661417843355
- `coneSurfaceArea`(4.5, 6.0) should return: 169.64600329384882

You will need to use the Java-supplied `Math.sqrt` method and for Pi, use the Java-supplied static constant: `Math.PI`.

Problem 2

In a class called `FirstDigit`, write a method called `getFirstDigit` that accepts a positive integer number as a parameter and returns the first digit in that number as an integer. For example: `getFirstDigit(3572)` should return 3 and getFirstDigit(712398) should return 7. (Hint: consider using a for loop to repeatedly divide the number and consider when the for loop should stop.) Test your method in a main method by calling it several times with sample numbers as input and printing out the return value.

Problem 3

In a class called `Harmonic`, write a static method called `harmonicNumber` that accepts an integer parameter n and returns the n^{th} harmonic number. In mathematics, the *n*-th **harmonic number** is the sum of the reciprocals of the first *n* natural numbers:

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}.$$

Your method implementation should use a for loop to calculate the harmonic number. You may not use any other static methods in your implementation.

To test your method, write a main method that prints out the 4^{th} and 9^{th} harmonic number. Sample output:

```
The 4th harmonic number is 2.083333333333333
The 9th harmonic number is 2.8289682539682537
```

Problem 4

In a class called `WraparoundSquare`, write a method called `printWraparoundSquare` that accepts two parameters: a minimum and maximum integer. This method should print a square of lines of increasing numbers in a cyclic or "wraparound" fashion. The first line should start with the minimum and the following line should start with next higher number. On each line, the sequence of numbers should increase until the maximum is reached at which point, it should wraparound to the minimum and begin increasing again. For example:

printWrapSquare(3, 7) should produce:

```
34567
45673
56734
67345
73456
```

printWrapSquare(2, 8) should produce:

```
2345678
3456782
4567823
5678234
6782345
7823456
8234567
```

Hint: you should use nested for loops: one for loop to control printing the lines and perhaps multiple for loops to control printing each number in the line. You should also consider developing your method incrementally – get something similar but simpler working first and then modify it to match the requirements.