**Homework 1**

**Due Date:** Homework 1 is due by 11:59 PM on Tuesday, September 26. Submit your work (the .java source code files ***ONLY***, not the compiled .class files!) through the “Homework 1” link on Blackboard. You may submit an unlimited number of times; we will only grade the last/latest submission attempt, but be sure to attach all of your files to each submission attempt. **Be sure to include your name and Stony Brook ID number in a comment at the beginning of each file that you submit.**

**Instructions**

The HomeworkOne.java file that accompanies this assignment has 4 methods that are (currently) nonfunctional. Follow the directions below to complete each method; **DO NOT MODIFY THE** main() **METHOD!** This assignment is worth a total of 40 points (10 points per method).

1. **Bank Fees**  
     
   A bank charges $10 per month plus the following check fees for a commercial checking account:   
   1. $0.10 each for 1-19 checks
   2. $0.08 each for 20-39 checks
   3. $0.06 each for 40-59 checks
   4. $0.04 each for 60 or more checks  
        
      (Note that the same fee is charged for all checks. If the customer writes 21 checks, all 21 checks are billed at the $0.08 rate.)

The bank also charges an additional $15 if the balance of the account falls below $400 (after applying the $10 fee but before any check fees are applied). Complete the getFees() method, which takes two arguments: a double representing the account balance, and an int representing the number of checks written during the past month. This method returns a double value representing the total amount that will be charged in fees for this month. You may assume that the account balance is always greater than 0. You may also assume that the number of checks is always non-negative (greater than or equal to 0).  
  
For example, getFees(1000.0, 14) would return 11.4 (the basic $10 fee plus 14 \* 0.1 for the checks). getFees(250.75, 50) would return 28.0 (the basic $10 fee, plus a $15 penalty for falling below $400.00, plus 50 \* 0.06 for the checks).

1. **Proper Divisors**  
     
   A *proper divisor* is an integer that evenly divides a source integer (i.e., the remainder is 0) **and** is strictly less than the source integer. For example, the proper divisors of 20 are 1, 2, 4, 5, and 10 (20 is also a divisor, but not a proper one). Their sum is 1 + 2 + 4 + 5 + 10, or 22.  
     
   Complete the addDivisors() method, which accepts a single int as its argument. This method uses a loop to compute the sum of the input integer's proper divisors, and then returns that sum as an integer value. You may assume that the input to this method is always a positive integer value.
2. **Alien Name Recognition**  
   The United Federation of Planets (UFP) is preparing to make first contact with a new alien race, the Bynars. Very little is known about this species except the following:  
     
   • The Bynar language is composed solely of sequences of 1s and 0s.  
   • If a Bynar’s name contains an odd number of 1s, he is a male; if it contains an even number of 1s, she is a female.  
     
   Diplomacy is critical during first contact. It would be disastrous if a UFP ambassador were to address a male Bynar as “miss” or a female Bynar as “mister.” Starfleet’s Diplomatic Corps has suggested that you can solve the problem by writing a Java method for the Universal Translator that will determine whether a given Bynar is male or female, by name alone.  
     
   Complete the getGender() method, which takes a single String argument that represents a Bynar name (you may assume that this String only contains the characters ‘1’ and ‘0’). If the name contains an odd number of 1s, the method should return the String “male”; otherwise, it should return the String “female.” Your method should not print anything.  
     
   **Hint:** There are multiple ways to approach this problem. You can use the modulo operator (%) to determine whether a given integer is even or odd. Alternately, you might use a boolean variable, flipping its value as you process the name.
3. **Index of Coincidence**  
     
   If two strings are superimposed on one another, then some letters may match one another. For example, in the two strings  
     
   **w**onderwhowrot**e**thebookonlov**e**  
   **w**eallliveinay**e**llowsubmarin**e**  
     
   there are three positions that contain the same letter: the first (w), the fourteenth (e), and the twenty-seventh (e). Of 27 possible positions, matches occur in three of them, or 11.1%. This percentage is called the *index of coincidence* for two strings. It is more than just a curiosity; during World War II, it was used to help decrypt enemy ciphers.  
     
   Complete the coincidence() method. This method takes two String arguments of equal length. The method computes and returns its arguments’ index of coincidence as a double value between 0.0 and 100.0. For normal written English, the index of coincidence has an average value of 6.6% (or 6.6 in our method); for random strings of letters, it is much lower, around 3.8% (3.8 from our method) on average.  
     
   You may assume that the String arguments supplied to your method are of equal length, and consist solely of lowercase letters (no spaces, digits, punctuation, or capital letters). Use a loop to examine the pairs of corresponding characters (i.e., the characters that appear at the same index in each String).

// CSE 114 Homework 1

//

// Fall 2017, Stony Brook University

//

// Name:

// SBU ID #:

import java.util.\*; // needed for the Scanner class

public class HomeworkOne

{

public static void main (String [] args)

{

Scanner sc = new Scanner(System.in);

// Test the getFees() method

System.out.print("Enter the starting account balance: ");

double bal = sc.nextDouble();

sc.nextLine(); // consume the extra newline character

System.out.print("Enter the number of checks that were written: ");

int checks = sc.nextInt();

sc.nextLine(); // consume extra newline

double fee = getFees(bal, checks);

System.out.println("You paid $" + fee + " in fees.\n");

// Test the addDivisors() method

System.out.print("Enter an integer to process: ");

int val = sc.nextInt();

sc.nextLine(); // consume any leftover newline characters

int sum = addDivisors(val);

System.out.println("Sum of proper divisors: " + sum + "\n");

// Test the getGender() method

System.out.print("Please enter the name of the Bynar you are speaking to: ");

String bName = sc.nextLine();

String gender = getGender(bName);

System.out.println("This Bynar is " + gender + "\n");

// Test the "index of coincidence" method

System.out.print("Enter a string to examine: ");

String s1 = sc.nextLine();

System.out.print("Enter another string to examine: ");

String s2 = sc.nextLine();

double index = coincidence(s1, s2);

System.out.println("These strings have an index of coincidence of " + index + "\n");

}

// COMPLETE THE METHODS BELOW FOR HOMEWORK 1

public static double getFees (double balance, int checks)

{

// Fill in this method for the homework

return -1.0; // CHANGE THIS LINE

}

public static int addDivisors (int input)

{

// Fill in this method for the homework

return -1; // CHANGE THIS LINE

}

public static String getGender (String name)

{

// Fill in this method for the homework

return null; // CHANGE THIS LINE

}

public static double coincidence (String first, String second)

{

// Fill in this method for the homework

return -1.0; // CHANGE THIS LINE

}

}