UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR08014 INFORMATICS 1 - OBJECT-ORIENTED PROGRAMMING

Tuesday 5 th May 2015

14:30 to 17:30

INSTRUCTIONS TO CANDIDATES

1. **Note that all questions are compulsory.**
2. **Remember that a file that does not compile, or does not pass the simple JUnit tests provided, will get no marks.**
3. **This is an Open Book exam. You may bring in your own material on paper, and/or on a USB stick. No other electronic devices are permitted.**
4. **CALCULATORS MAY NOT BE USED.**

Convener: D. K. Arvind
External Examiner: C. Johnson

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. You are given, in file `Customer.java`, the code of a class `Customer` with the following public interface:

`public class Customer`	
`       Customer(String n)`	constructor
`String getName()`	
`String toString()`	

You will see that there are comments showing that certain things are not finalised yet. You will not modify this code, however: your task is to implement the subclass `PremiumCustomer`, representing a special kind of `Customer`.

(a) Define the class `PremiumCustomer`, extending `Customer`. It should have a private instance variable `freeGifts` of type `Map<String, String>`, representing that this `PremiumCustomer` will receive a certain free gift each time they order a certain item. For example, the key `"printer cartridge"` might map to the value `"chocolate bar"`. [*10 marks*]

(b) Write a public constructor for `PremiumCustomer` which takes a `String` representing the customer's name, and also a `Map<String, String>`. Pass the string to `Customer`'s constructor, and store the `Map` in `freeGifts`. [*10 marks*]

(c) Write a public instance method `giftsFrom`, taking a `String` representing the item ordered, and an `int` representing the quantity ordered, and returning no value. If the item ordered occurs as a key in `freeGifts`, print to standard output the appropriate free gift, then " x ", then the quantity. For example, the result of `giftsFrom("printer cartridge", 3)` might be that

```
chocolate bar x 3
```

gets printed to standard output. If the item ordered does not correspond to a free gift, `giftsFrom` must do nothing. [*10 marks*]

(d) Write a public instance method `toString`, taking no argument and returning a `String` representation of this object, exactly as follows. The string must begin with the string that is returned by `Customer`'s version of `toString` (even if the developers of that method decide to change it later, i.e., don't just duplicate the current behaviour). Then on a new line by itself, the string must have "Free gifts:". Then list each free gifts entry on a line by itself. The order in which these are listed does not matter. For each one, give the gift, " on ordering ", then the item that must be ordered to get the free gift. Here is an example of a possible result string:

```
Charles
Free gifts:
chocolate bar on ordering printer cartridge
biscuits on ordering box of paper
```

[*10 marks*]

(e) Write a public instance method `chocolateGifts`, taking no argument and returning an `int` representing the number of free gifts the customer is eligible for that involve chocolate. More precisely, the result is the number of values in the map `freeGifts` that contain the substring “chocolate”. [*10 marks*]

In summary `PremiumCustomer` has this public interface (omitting what is inherited from `Customer`), with behaviour as explained above:

```
public class PremiumCustomer
----------------------------------------------------------
       PremiumCustomer(String s, Map<String, String> fg)
  void giftsFrom(String s, int i)
String toString()
   int chocolateGifts()
----------------------------------------------------------
```

The file you must submit for this question is `PremiumCustomer.java`. Before you submit, check that it compiles and passes the basic JUnit tests provided, otherwise it will get 0.

2. This question relates to the Lucas sequence, which is related to the more famous Fibonacci sequence. However, you are not expected to have any prior knowledge of either. The 0th and 1st terms of the Lucas sequence are $L(0) = 2, L(1) = 1$ respectively. Thereafter, each term is given by adding the previous two. Thus the first few terms are

   2, 1, 3, 4, 7, 11, 18, 29

   (a) Implement a class `Lucas` as follows. Note that all its methods are static. Note also that the type `long` is used for elements of the Lucas sequence, because we may want integers bigger than will fit in a Java `int` value. You may need to know that literal values of type long are written using digits, followed by the letter L; thus 1L is legal as a literal long value, while 1 is not. Consult the online documentation if you need more information.

   Your class should have private static variables:

   - `lucas`, an array of `long`s, which will later store the sequence as far as it has been calculated – leave this uninitialised here;
   - `phiPlus`, a `double`, set to $\frac{\sqrt{5}+1}{2}$;
   - `phiMinus`, a `double`, set to $\frac{\sqrt{5}-1}{2}$. *[10 marks]*

   (b) Write a public static method `upto`, which takes an `int n`, initialises `lucas` appropriately, and places the first `n` Lucas numbers (that is, those with indexes 0,..., `n-1`) into `lucas`. It should return no result. Your code may assume that `n` is at least 2. *[10 marks]*

   (c) Write a public static method `primes`, which will work out which of the Lucas numbers stored in `lucas` are prime, and print this information to standard output. The function returns no value. The first line of output should be exactly:

   ```
   L(0)=2
   ```

   indicating that the Lucas number with index 0 is 2, and this is prime. Do not print any output for Lucas numbers that are not prime. Use a new line for each prime Lucas number. Do not put any spaces in your output. Recall that a prime number is an integer greater than 1 which is divisible only by 1 and itself. Your code for testing whether a number is prime does not need to be efficient, provided it is correct. (If you do this correctly, you will notice that every index of a prime Lucas number is 0, a power of 2, or itself prime – but you are not expected to prove this!) *[10 marks]*

   (d) Write a public static method `maxDiffClosedForm`, which will demonstrate that (up to rounding errors) there is a closed form for the Lucas numbers, namely,

   $$L(i) = \Phi^i + (-\phi)^i$$

where Φ is the value you stored in `phiPlus` and ϕ is the value you stored in `phiMinus`. Your function must return, as a `double`, the maximum of the absolute values of the differences between this expression for the Lucas number and the value you have stored in `lucas`. [*10 marks*]

(e) Finally, write a static method `main` with the usual header. When the command-line argument is an integer `n` between 2 and 42 inclusive, this method should:

i. use `upto` to calculate and store the first `n` Lucas numbers;

ii. use `primes` to print out the prime ones;

iii. use `maxDiffClosedForm` to calculate the maximum discrepancy between the closed form and the calculated form for Lucas numbers;

iv. print out the maximum difference to 15 decimal places. Print this number on a line by itself, with no explanatory text, e.g.

```
0.000000298023224
```

If the command-line input is an integer that is not between 2 and 42 inclusive, the program should do nothing. You are not required to handle the case where no integer is input. [*10 marks*]

The file you must submit for this question is `Lucas.java`. Before you submit, check that it compiles and passes the basic JUnit tests provided, otherwise it will get 0.