University of Glasgow

Tuesday 29 April 2014
09.30 am – 11.30 am
(Duration: 2 hours)

DEGREES OF MSc in Information Technology / Software Development

PROGRAMMING

(Answer all 6 questions.)

This examination paper is worth a total of 75 marks
(each of Questions 1-5 carries 10 marks, whilst Question 6 carries 25 marks)

For examinations of at least 2 hours duration, no candidate shall be allowed to leave the examination room within the first hour or the last half-hour of the examination.

THE USE OF CALCULATORS IS NOT PERMITTED IN THIS EXAMINATION

INSTRUCTIONS TO INVIGILATORS
Please collect all exam question papers and exam answer scripts and retain for school to collect. Candidates must not remove exam question papers.

This page is intentionally left blank

1. Consider two integer variables `a` and `b` and assume that `a<b` (the value of `a` is lower than the value of `b`), `a>0` (the value of `a` is greater than zero) and `b>0` (the value of `b` is greater than zero). Write two helper methods, `calculateLCM` and `calculatedGCD`, that perform the following tasks:

(a) `calculateLCM` finds the Least Common Multiple of `a` and `b` (the smallest non-zero integer that each of `a` and `b` divides without leaving a remainder). For example, if `a=3` and `b=6`, the Least Common Multiple is `12`.

[5]

(b) `calculateGCD` finds the Greatest Common Divisor of `a` and `b` (the largest non-zero integer that divides both `a` and `b` without leaving a remainder). For example, if `a=4` and `b=6`, the Greatest Common Divisor is `2`.

[5]

2. A program includes the following initializations:

```
int k=2, m=12, n=5;
double x=5.0;
String first = "John", last = "Stuart-Wilson";
```

(a) Consider the following statement:

```
System.out.println(String.format("The value of %d/%d is
%d",n,k,k/n)) ;
```

Write the console output produced by the statement.

[2]

(b) Consider the following statement:

```
System.out.println(String.format("The value of %.2f/%d is
%.2f",x,k,x/k)) ;
```

Write the console output produced by the statement.

[2]

(c) Consider the following statement:

```
System.out.println(String.format("Value one is %03d%nValue
two is %03d",k,m)) ;
```

Write the console output produced by the statement.

[2]

(d) Consider the following statement:

```
System.out.println(String.format("The name is
%s%s",first,last)) ;
```

Write the console output produced by the statement.

[2]

(e) Consider the following statement:

```
System.out.println(String.format("The name is:
%s%6.6s",first,last)) ;
```

Write the console output produced by the statement.

[2]

3. Explain the mistakes in the blocks of code below and write a correct version of the code that fulfils the expectations of the programmer:

(a) The goal of the programmer is to output the value of `j` after the conditional (assume that `test` is a boolean variable whose value has been determined previously):

```
int j = 4;
if (test) {
   int j = 5;
}
System.out.println(j);
```

[2]

(b) The goal of the programmer is to print the sum of `a` and `b` without changing the type of the variables:

```
int a = 3;
String b = "2";
int c = a+b;
System.out.println(c);
```

[2]

(c) The goal of the following method is to return the maximum of two input integer variables:

```
public void findMax(int i, int j) {
   int max;
   if (i>j) {
      max=i;
   } else {
      max=j;
   }
   return max;
}
```

[2]

(d) The goal of the programmer is to output a countdown from `10` to `1` inclusive:

```
for (int n=1; n<10; n--) {
   System.out.println(n) ;
}
```

[2]

(e) The goal of the programmer is to print an alert message on the console when the temperature is not in the interval `[25,30]` (the extremes are included):

```
int temp = 27, lowerLimit = 25, upperLimit = 30;

if ((temp >= lowerLimit) && (temp =< upperLimit))
   System.out.println("Alert!");
```

[2]

4. **(a)** Consider the following method:

```
public void myMethod(int n) {
    if (n <= 1)
        System.out.print(n);
    else {
        myMethod(n / 2);
        System.out.print(", " + n);
    }
}
```

Write the output from the following method calls:

(i) `myMethod(1);`
(ii) `myMethod(16);`
(iii) `myMethod(100);`

[3]

(b) Write a recursive method `writeNums` that accepts an integer parameter *n* and prints the first *n* positive integers in sequential order, separated by commas, to the console. For example, the following calls produce the following output:

Call	Output
`writeNums(5);`	`1, 2, 3, 4, 5`
`writeNums(11);`	`1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11`

You may assume that $n \geq 1$. Note that your method body must not use iteration (i.e., loops).

[4]

(c) Write a recursive method `starString` that accepts an integer parameter *n* and returns a `String` object of length 2^n (i.e., 2 to the *n*th power) containing stars (asterisks). For example:

Call	Output	Reason
`starString(0);`	`"*"`	$2^0 = 1$
`starString(1);`	`"**"`	$2^1 = 2$
`starString(2);`	`"****"`	$2^2 = 4$
`starString(3);`	`"********"`	$2^3 = 8$
`starString(4);`	`"****************"`	$2^4 = 16$

You may assume that $n \geq 0$. Note that your method body must not use iteration (i.e., loops).

[3]

5. A company has written a large class `BankAccount` with many methods. The headings of the constructor and key methods, and a key instance variable, are as follows:

```
public BankAccount(int startAmount)
// construct a BankAccount object using the initial balance
// startAmount in pennies

private int balance;
// stores the current balance in pennies

public void withdraw(int withdrawAmount)
// withdraw the amount in withdrawAmount in pennies

public void deposit(int depositAmount)
// deposit the amount in depositAmount in pennies

public int getBalance()
// return current balance in pennies
```

Write a new class `MinMaxAccount` whose instances can be used in place of a `BankAccount` object and include the ability to remember the minimum and maximum balances ever recorded for the account. You should ensure that the existing functionality of `BankAccount` is still provided, and additionally you should provide implementations of the following constructor and methods:

```
public MinMaxAccount(int startAmount)
// construct a MinMaxAccount object using the initial balance
// startAmount in pennies

public int getMin()
// return minimum balance in pennies

public int getMax()
// returns maximum balance in pennies
```

Assume that only the `withdraw` and `deposit` methods in `BankAccount` change an account's balance. Further, assume that the values of the integer parameters to all methods are positive.

[10]

6. Consider the following class:

```
public class Word {

   private String word;   // the actual word

   public Word (String s)
   {  word = s;
   }

   public String getWord()
   {  return word;
   }
}
```

(a) Suppose that `dictionary` is an array of `Word` objects, whose instance variables store `String` objects corresponding to actual (lower-case) words in the English language. Now suppose that `queryWord` is a `String` object. Implement the body of the following method:

```
public int occurs(Word [] dictionary, String queryWord,
                                      int maxIndex)
/* if queryWord is equal to the value of the instance
   variable of some index i of dictionary between
   positions 0 and maxIndex-1 inclusive, return the
   smallest value of i for which this is the case,
   otherwise return -1
*/
```

[5]

(b) Suppose that `sentence` is a `String` object containing a sentence of text, with punctuation symbols allowed. That is, each character of `sentence` is either (i) a letter of the alphabet, (ii) a space, (iii) a comma, or (iv) a full stop.

You are required to implement a simple spell-checker that will check whether each word in `sentence` belongs to `dictionary`, thus ignoring spaces and punctuation symbols.

If every word in `sentence` belongs to `dictionary`, a message to that effect should be reported. Otherwise, if some word does not belong to `dictionary`, one or more messages should be given that indicate the offending word or words (if more than one word in `sentence` is not in `dictionary`).

Here is an example: suppose that `sentence="This is a simple semtence, containing punctuaation."` and `dictionary` contains all words in the English language. Then the output should be as follows:

```
semtence is not in the dictionary
punctuaation is not in the dictionary
```

Words in `sentence` with upper-case characters should be converted to lower case using the `toLowerCase` method from the `String` class. For example:

`toLowerCase("This")` returns `"this"`, which can then be checked against `dictionary`.

You can assume that `sentence` and `dictionary` have been initialised, and you should just supply lines of code that are assumed to be within a hypothetical method (whose heading need not be given), in which `sentence` and `dictionary` are in scope.

[9]

(c) You are now required to find the word that occurs most frequently in `sentence` (in either lower or upper case, and ignoring punctuation symbols). For example, suppose that `sentence="The word with the most occurrences is the one that you would expect."` The output should be as follows:

```
The word with the most occurrences is "the".
```

If more than one word occurs most frequently, the output string should just contain any one among those words. In your solution, you should use the following class:

```
public class WordFreq extends Word
                      implements Comparable<WordFreq>
{
   private int frequency; // the frequency of the word

   public WordFreq (String s)
   {  super(s);
      frequency = 1;
   }

   public int getFrequency()
   {  return frequency;
   }

   public void incrementFrequency()
   {  frequency++;
   }

   public int compareTo(WordFreq other)
   {  if (frequency < other.getFrequency())
         return 1;
      else if (frequency == other.getFrequency())
         return 0;
      else
         return -1;
   }
}
```

Assume that the code in your solution to Part (c) follows on from the code in your solution to Part (b). Thus you may refer to variables from your solution to Part (b) that are within scope. You may also assume that the package `java.util` has been imported.

[11]