University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

SEMESTER I EXAMINATION – 2015/2016

COMP 20250

Introduction to Java

SAMPLE EXAM PAPER
PLEASE NOTE THAT NONE OF THESE QUESTIONS WILL BE AT THE FINAL EXAM

Prof. S. Dobson

Prof. P. Cunningham

Dr Eleni Mangina*

Time allowed: 2 hours

Instructions for candidates

Answer all three questions. The paper is marked out of 100.

.

1. Answer parts (a) to (c). **50 marks in total**

```
class BankAccount {
   private double balance;                                   // extract(i)

 public BankAccount() {                                      // extract(ii)
        balance = 0;
 }

 public BankAccount(double intitialBalance){                 // extract(iii)
        balance = initialBalance;
  }

  public double getBalance() {                               // extract(iv)
        return balance;                                      // extract(v)
   }

  public void withdraw(double amount){                       // extract(vi)
        balance = balance – amount;
   }

  public void deposit(double amount) {
              balance = balance + amount;                    // extract(viii)
   }

}
```

1.(a) Examine the above definition for a class BankAccount, as seen in lectures. For each of the following extracts from this definition, explain exactly what every part of the extract means, and explain what role the extract plays in the BankAccount class:

(i) private double balance;

(ii) public BankAccount()

(iii) public BankAccount(double intitialBalance)

(iv) public double getBalance()

(v) return balance;

(vi) public void withdraw(double amount)

(vii) balance = balance + amount;

1.(b) Assume the above class. Write a section of java code which does all of the following:

- creates two BankAccount objects, one with an initial balance of 50, the other with 0 initial balance
- stores the object with a balance of 50 in a variable called annesBankAccount and the object with 0 initial balance in a variable called johnsBankAccount
- calls methods to withdraw 25 from annesBankAccount and deposit 25 into johnsBankAccount
- prints out the balance of both bankaccounts after that withdrawal and deposit.

Answers for question 1.

Part 1.(a)

(viii) private double balance; A variable called "balance", which can hold double-format numbers (real numbers) and which can only be accessed by methods within the object or class

(ix) public BankAccount() A Constructor method for the class BankAccount. It is public so that objects in this class can be created from outside the class. It takes no parameters and so constructs a default bank account object

(x) public BankAccount(double intitialBalance) Another Constructor method for the class BankAccount. It is public so that objects in this class can be created from outside the class. It takes a parameter called initialBalance, which contains a double-type number. This is used to set the balance of the constructed account

(xi) public double getBalance() a public method called getBalance. Since it is public it can be used from outside objects of this class. The "double" means it return a double-type number whenever it is used.

(xii) return balance; the command that tells the method to return the content of the variable balance whenever the method is used.

(xiii) public void withdraw(double amount) a public method called withdraw. It takes a number called amount as a parameter, which is a double type number. This method does not return any value (that's what "void" means).

(xiv) balance = balance + amount; set the amount of the variable called "balance" in the current bankaccount object so it becomes equal to the previous value of balance plus whatever is in the amount variable.

Part 1.(b)

BankAccount johnsBankAccount = new BankAccount();

BankAccount annesBankAccount = new BankAccount(50);

johnsBankAccount.deposit(25);

annesBankAccount.withdraw(25);

System.out.println("annes balance is "+annesBankAccount.getBalance());

System.out.println("johns balance is "+johnsBankAccount.getBalance());

1.(c) Assuming some array of integers Y, write a snippet of code that creates an array of integers Z and then fills that array with the integers in Y but in reverse order (so that the last integer in Y is the first integer in Z, and so on).

Answer

```
//method signature not required as only a snippet is requested
//initialised an array "z" which is the same size as array "y"
int[] z = new int[y.length];
//"i" is initialised to the length of array "y"
int i=y.length-1;
//iterate for each element of the array "y"
for(int j =0; j<y.length; j++){
//copy the value of the element from the array "y" to the array "z"
    z[j]=y[i];
    //decrement "i"
    i--;
}
//return the array "z" which contains the values of array "y" in reverse order
return z;
```

2. Answer parts (a) and (b). 25 marks in total

2.(a) The following is a section of code from a program. You should examine the statements marked (i) to (v). Each of these statements is intended to assign a value to a variable. Some of these statements are correct, some are incorrect. Say, for each statement, whether it is correct or incorrect. If the statement is correct, say what the variable's value is after the statement. If the statement is incorrect, explain why it is incorrect.

```
String s = "20";                    //(i)
int i = s + 10;                     //(ii)
double d = 3.6;                     //(iii)
double x = 10;                      //(iv)
boolean b = x == d;                 //(v)
```

2.(b) Write a class whose main method takes two int numbers from the user (using repeated JOptionPanes). If the second int divides evenly into the first, the program should display the message:

```
integer ??? goes into integer ??? exactly ??? times.
```

where the first set of question marks would be replaced by the second integer, the second set of question marks by the first integer, and the third set of question marks by the number of times the second integer goes into the first.
If the second int DOES NOT divide evenly into the first, however, the program should display a different message, as follows:

```
integer ??? goes into integer ??? a total of ??? times, with ???
left over.
```

where the first set of question marks would be replaced by the second integer, the second set of question marks by the first integer, and the third set of question by the number of times the second integer goes into the first, and the final set of question marks would be replaced by the remainder obtained after dividing the first integer by the second.

Remember that the JOptionPane methods return Strings, not ints: your program will have to deal with this!

Answers for question 2.

Part 2.(a)

```
String s = “20”;            //(i)    correct
int i = s + 10;             //(ii)   incorrect; s is a string, and can’t go in an int
double d = 3.6;             //(iii)  correct
double x = 10;              //(iv)   correct
boolean b = x == d;         //(v)    correct:  x==d is a boolean!
```

Part 2.(b)

```
import javax.swing.JOptionPane;

public class Compare {

   public static void main(String[] args){
        String strOne = JOptionPane.showInputDialog(null, "enter one number");
        String strTwo = JOptionPane.showInputDialog(null, "enter another number");
        int one =  Integer.parseInt(strOne);
        int two =  Integer.parseInt(strTwo);
        int lower;
        int higher;
        if (one < two){
               lower = one;
               higher = two;
        } else  {
               lower = two;
               higher = one;
        }
        if (higher % lower == 0){
            JOptionPane.showMessageDialog(null, "integer "+lower+" goes into integer"
            +higher+" exactly "+higher/lower+" times");
        } else {
            JOptionPane.showMessageDialog(null, "integer "+lower+" goes into integer "
           +higher+ " a total of "+higher/lower+" times, with "+higher % lower +" left over");
        }
        System.exit(0);
   }
}
```

3. Answer parts (a) and (b). 25 marks in total

3.(a) Using either a `do` loop or a `while` loop, write a section of code which, on each cycle of the loop, uses a `JOptionPane` to ask the user for a word (a `String`). The loop should keep repeating as long as the user enters a word that is longer (contains more characters) than the words previously entered. If the user enters a word that is shorter than (or equal in length to) a previously-entered word, the program should stop. The first time the program asks the user for a word it should use the message "`enter a word`"; on all subsequent requests the program should give the message "`enter a longer word`".

Here is an example of how the program should work:

```
Program displays               User enters
Enter a word:                    Hello
Enter a longer word:             Greetings
Enter a longer word:             salutations
Enter a longer word:              Hi
That's not a longer word!
```

ANSWER:

Lines in blue would only be necessary if you were asked to write a program, in this case you are only asked for a **section of code** and so they are not necessary.

```
import javax.swing.JOptionPane;

public class LongerWordWhileLoop {

   public static void main(String []args){
        String previousInput = "";
        String input = JOptionPane.showInputDialog(null, "enter a word");
        while (input.length() > previousInput.length() ) {
              previousInput = input;
             input = JOptionPane.showInputDialog(null, "enter a longer word");
        }
        JOptionPane.showMessageDialog(null, "that's not a longer word!");
   }
}
```

3.(b) Explain the differences between a "while" loop and a "do" loop. Give an example of code using each type of loop.

A while loop has the test first, as in the example above. Do loop has the test after the body of the loop, as in

ANSWER:

Lines in blue would only be necessary if you were asked to write a program, in this case you are only asked for a **section of code** and so they are not necessary

```
import javax.swing.JOptionPane;

public class LongerWordDoWhileLoop {
   public static void main(String []args){
       String previousInput;
       String input = JOptionPane.showInputDialog(null, "enter a longer word");
       do {
            previousInput = input;
            input = JOptionPane.showInputDialog(null, "enter a longer word");
       } while(input.length() > previousInput.length());
       JOptionPane.showMessageDialog(null, "that's not a longer word!");

   }
}
```