THIS PAPER IS NOT TO BE REMOVED FROM THE EXAMINATION HALL

UNIVERSITY OF LONDON

CO1109 ZA

BSc, CertHE and Diploma EXAMINATION

COMPUTING AND INFORMATION SYSTEMS and CREATIVE COMPUTING

Introduction to Java and object-oriented programming

Friday 3 May 2019: 10.00 – 13.00

Time allowed: 3 hours

DO NOT TURN OVER UNTIL TOLD TO BEGIN

There are **SIX** questions on this paper. Candidates should answer **FOUR** questions. All questions carry equal marks and full marks can be obtained for complete answers to **FOUR** questions. The marks for each part of a question are indicated at the end of the part in [ ] brackets.

Only your first **FOUR** answers, in the order that they appear in your answer book, will be marked.

There are 100 marks available on this paper.

Calculators are not permitted in this examination.

UL19/0304

Question 1

(a)

(i) Consider the following four tasks:

(A) Asking the user to make a valid choice from a list of numbered menu items.

(B) Printing the last entry in an `Array`.

(C) Adding together all of the items in an `int Array`.

(D) Printing the following shape:

```
******
*    *
*   *
* *
**
*
```

Below are four possible ways of implementing each task. Match up each of the tasks (A) to (D) with one of the possible implementations below. Your answer should choose the most appropriate implementation for each task, and should use each one of the four possible implementations. [4 marks]

- nested `for` loops
- `while` loop
- `for` loop
- no loop, single statement needed.

(ii) Consider the `for` loop in the *NumbersDownToNew* class:

```
import java.util.Scanner;
class NumbersDownToNew{

     public static void main(String[] args){
          System.out.print("Enter Number>");
          Scanner in = new Scanner(System.in);
          int x=in.nextInt();
          for(int i=x;i>0;i--)System.out.println(i);
     }
}
```

In general, if the user enters *n*, then the output will be

```
n
n-1
n-2
.
.
.
3
2
1
```

For example, if the user enters 10, then the `for` loop will output:

```
10
9
8
7
6
5
4
3
2
1
```

Change the `for` loop so that the output will start at *n*-1 and continue down to, and including, 0. [2 marks]

(iii) Consider the class *Empty:*

```
class Empty{
     static int x;

     public static void main(String[] args){
          while(true) System.out.println("Hello world");
     }
}
```

What will be output when the class is run? [2 marks]

(A) No output, because the program will not compile.

(B) Infinite loop – the standard output will fill up with "Hello world", one to a line, and this will continue indefinitely.

(C) Run time error.

(D) None of the above.

(b) For each of the following loops, say how many asterisks (*) will be printed. If you think that the loop will continue indefinitely without stopping, write 'infinite loop' in your answer book for that part of the question. [9 marks]

(i) `for(int i=0;i<9;i++) System.out.print("*");`

(ii) `int j=18; while (j>6){System.out.print("*");j=j-2;}`

(iii) `for(int i=-2;i<4;i++) System.out.print("*");`

(iv) `int k=3; while (k>3){System.out.print("*");k=k+3;}`

(v) `for(int i=0;i>=0;i++) System.out.print("*");`

(vi) `for(int i=0;i<9;i=i+2) System.out.print("*");`

(vii) `int p=0; while(p<=8){System.out.print("*");p+=2;}`

(viii) `for(int i=5; i>=5; i++) System.out.print("*");`

(ix) `int m=0; while (true){System.out.print("*");}`

(c) Consider the *GameUserInterface* class, below:

```
import java.io.PrintStream;
import java.util.Scanner;

public class GameUserInterface {
  private static Scanner input = new Scanner(System.in);
  private static PrintStream output = System.out;
  private static boolean keepPlaying = true;

  public static void play() {
    showWelcomeMessage();
    mainLoop();
  }

  private static void showWelcomeMessage() {
    output.println("Welcome! Shall we play?");
    output.println();
  }

  private static void mainLoop() {
    //some statements missing here
  }

  private static void showMenu() {
    output.println("Choose one of the following:");
    output.println("1. Number Guessing Game");
    output.println("2. Word Game");
    output.println("3. Quit");
    output.println();
  }

  private static void askUserToChoose() {
    output.print("Enter your choice: ");
  }

  private static int getUserChoice() {
    String s = input.nextLine();
    return Integer.parseInt(s);
  }
```

```
  private static void executeChoice(int choice) {
    if (choice == 1) {
        output.println("Starting Number Guessing Game...");
        NumberGuessingGame.play(input, output);
        //assume above statement works as it should
    } else if (choice == 2) {
        output.println("Starting Word Game...");
        WordGame.play(input, output);
        //assume above statement works as it should
    } else if (choice == 3) {
        output.println("Bye!");
        keepPlaying = false;
    } else {
        output.println("I don't know what that is.");
    }
    output.println();
  }

  public static void main(String[] args) {
    play();
  }
}
```

Once the class is compiled and run it should behave as follows:

1. The user will see the following welcome message, menu and request for input:

```
Welcome! Shall we play?

Choose one of the following:
1. Number Guessing Game
2. Word Game
3. Quit

Enter your choice:
```

2. When the user enters a valid choice then the class gives the appropriate response, either starting a game or quitting.
3. When the user chooses to play a game, when the game ends the user is again shown the menu, and asked for input, so that they can choose to play another game or quit.
4. The user can play as many games as they want before deciding to quit.
5. If at any point when asked to make a valid menu choice, the user enters an invalid number, the class outputs:
   `I don't know what that is.`
   After this the user is again shown the menu and asked for input.

The *GameUserInterface* class does not behave as it should, because the *mainLoop()* method has some statements missing. Complete the *mainLoop()* method so that the class behaves as it should. In your answer assume that the statements in the method *executeChoice(int)* calling the *NumberGuessingGame* and *WordGame* classes will work as they should, because the classes exist, and the *GameUserInterface* class can access them. [8 marks]

Question 2

(a)

(i) If your program does not compile the best thing to do is: [2 marks]

(A) Correct the first error only and recompile.

(B) Correct the final error only and recompile.

(C) Correct all errors and recompile.

(D) None of the above.

(ii)

```
class QX{
     static String x;
     static String y;

     public static void main(String[] args){
          x = "hello";
          y = "world";
          swap();
          System.out.println(x+"  "+y);
     }

     static void swap (){
          x = y;
          y = x;
     }
}
```

What will be the output of the *QX* class above? [4 marks]

(A) `hello  world`

(B) `world  world`

(C) `world  hello`

(D) None of the above.

(iii) What error will the compiler find in the *PX* class below? [2 marks]

```
public class PX{
     int z;

     public static void main(String[] args){
          z = 5;
     }

     static void subtract(int x, int y){
          int temp = x - y;
     }
}
```

(A) `non-static variable z cannot be referenced from a static context`

(B) `error: missing return statement`

(C) No error – the class will compile.

(D) None of the above.

(b) For each of the following expressions state whether it type checks correctly or not. Write 'yes' in your answer book if it type checks correctly, and write 'no' if it does not. [9 marks]

(i) `Math.abs("Camelot");`

(ii) `Integer.parseInt(10);`

(iii) `int a = Math.abs("Camelot".length());`

(iv) `Integer.parseInt("350");`

(v) `"threefifty".compareTo("350");`

(vi) `int z = "threefifty".compareTo("350");`

(vii) `"boy".replace('b',"soup".charAt(0));`

(viii) `Math.max(("hello" + " dog").length(),5);`

(ix) `("boy".replace('b',"soup".charAt(0))).length();`

(c) Consider the *StringMethods* class below:

```
import java.util.Arrays;
import java.util.Random;

public class StringMethods{

     private static void echoString(String word) {
          System.out.println(word);
     }

     private static void two(String word) {
          int length = word.length() - 1;
          String s = "";
          char c;
          for (int i = length; i >= 0; i--) {
               c = word.charAt(i);
               s += c;
          }
          System.out.println(s);
     }

     private static void three(String word) {
          char[] array = word.toCharArray();
          char c;
          int to;
          int from;
          int length = array.length;
          Random rand = new Random();
          for (int i = 0; i < length; i++) {
               to = rand.nextInt(length);
               from = rand.nextInt(length);

               //swap characters
               c = array[to];
               array[to] = array[from];
               array[from] = c;
          }

          System.out.println(array);
     }
```

```
    private static void four(String word) {
        word = word.toUpperCase();
        String s = "";
        char c;
        for (int i = 0; i < word.length(); i++) {
            c = word.charAt(i);
            if(c =='A'|| c=='E'|| c=='I'|| c=='O'||c=='U'){
                s += c;
            }
        }

        System.out.println(s);
    }

    private static void five(String word) {
        word = word.toUpperCase();
        String s = "";
        char c;
        for (int i = 0; i < word.length(); i++) {
            c = word.charAt(i);
            if (!(c =='A'||c=='E'||c=='I'||c=='O'||c=='U'))
                s += c;
            }
        }

        System.out.println(s);
    }

    private static void six(String word) {
        int length = word.length() - 1;
        String s = "";
        char c;
        for (int i = 0; i < length; i++) {
            c = word.charAt(i);
            s += c + " ";//adding one space
        }
        c = word.charAt(length);
        s += c;

        System.out.println(s);
    }
}
```

There are 6 methods in the class named *echoString(String), two(String), three(String), four(String), five(String)* and *six(String)*. Only *echoString(String)* has a meaningful name, that tells you what it does – the method echoes the `String` parameter it is given by printing it to standard output. [8 marks]

Read the other 5 methods in the class, and give them names that reflect the actions that they will take on their `String` parameter, as you understand them to be.

Question 3

(a)

(i) Consider class *Bool*

```
public class Bool{

     public static void main(String[] args){

          boolean b = (6!=6);
          System.out.println(b);
     }
}
```

Which one of the following is true about class *Bool*? [2 marks]

(A) The class will not compile.

(B) The class will compile and output `true`.

(C) The class will compile and output `false`.

(D) None of the above.

(ii) What is the output of the following? [2 marks]

```
class Bool5{
  public static void main(String[] args){
     System.out.println(!(!(false)));
  }
}
```

(A) `true`

(B) `false`

(C) No output – the program will not compile.

(D) None of the above.

(iii) Consider the three statements below:

```
if (x > 10) System.out.println("true");
else if (y>10)
System.out.println("true");
else System.out.println("false");
```

Rewrite the statements so that there is only one `boolean` expression to evaluate, and the output will be the same. [4 marks]

(b)

(i) Consider the *testPressure(int)* method. It is designed to report on the pressure of a boiler. The value of the pressure starts at 0, which would effectively mean no pressure, and can climb above 100, which is considered to be dangerous. The limit for safe pressure is 100. Pressure values cannot be less than zero.

```
static void testPressure(int z){
  if (z > 100) System.out.println("danger!");
  if (z <= 100 && z > 60) System.out.println("pressure
     is high");
  if (z < 60 && z >= 30) System.out.println("within
     range");
  if (z < 30 && z > 0) System.out.println("pressure is
     low");
  if (z == 0) System.out.println("check systems");
}
```

Assume that the method compiles. Identify a logical error with the method. [3 marks]

(ii) Give the output of the *K2* class: [3 marks]

```
class K2{

     public static void main(String[] args){
          int x = 3/2;
          int y = 2;
          int z;
          z = 2*x +3*y;
          System.out.println(z+y);
     }
}
```

(iii) Say which two of the three numbered statements in the *Precedence* class below, will have the same output: [3 marks]

```
class Precedence{
     public static void main(String[] args){
     /*1*/     System.out.println(5*2+1);
     /*2*/     System.out.println(5*(2+1));
     /*3*/     System.out.println((5*2)+1);
     }
}
```

(c) Consider the *Shape* class, below:

```
public class Shape{
  public static void main(String[] args){
    int limit = 8;
    for (int i = 0; i <= limit; i++){
      for (int j = 0; j <= limit; j++){
        if(i==0||j==0||i==limit||j==limit||i==j||i+j==limit)
          System.out.print("*");
        else System.out.print(" ");//printing one space
      }
      System.out.println();
    }
  }
}
```

Give the output of the class. [8 marks]

Question 4

(a) Consider the following class:

```
import java.io.*;
public class filey{

  public static void bling(String s) throws
     Exception{
     BufferedReader in =new BufferedReader(new
     FileReader(s));
     int t=in.read();
     while (t!=-1){
          System.out.print((char)t);
          t=in.read();
     }
  }

  public static void main(String[] args) throws
     Exception{
     bling(args[0]);
  }
}
```

(i) Why is the `while` loop in the *bling(String)* method controlled by `t!=-1`? [2 marks]

(A) Because the *read()* method returns −1 when it detects that the file is closed.

(B) Because the *read()* method returns −1 when it detects the end of the file.

(C) Because *t* can never take the value −1 hence the `while` loop is infinite.

(D) None of the above.

(ii) If the *args[0]* variable contains the `String "filey.java"` what will the program do? [2 marks]

(A) The program will output nothing.

(B) The program will stop with a run-time error.

(C) The program will output the text contained in the file 'filey.java'.

(D) None of the above.

(iii) Assume that when *filey.java* is run, it is given a valid file name of a text file containing some text. What would be the output if the statement inside the `while` loop `System.out.print((char)t);` was changed to read: [2 marks]

```
System.out.print(t);
```

(A) The program will output the unicode value of each character in the file.

(B) The program will stop with a run-time error.

(C) The program will output the text contained in the file.

(D) None of the above.

(iv) What would happen if *filey.java* was run with the valid name of an empty file? [2 marks]

(A) The program would hang waiting for input.

(B) The program would end with an `InputMismatchException`.

(C) There would be no output as the condition `t!=-1` would be false at the beginning.

(D) None of the above.

(b) Consider the class *Qx*

```
import java.io.*;
class Qx{

  public static void main(String[] args) {
    try{
      FileReader in = new FileReader("silly.dat");
      int c;
      while ((c=in.read())!=-1)
         System.out.print((char)c);
    }
    catch (IOException f){
      System.out.println("unable to continue ");
    }
  }
}
```

(i) What will be the output of the class if the file *silly.dat* does not exist? [3 marks]

(ii) What is the name of the exception that would be thrown if the file *silly.dat* did not exist? [3 marks]

(iii) Name the exception that will be thrown by the following: [3 marks]

```
class T1{
     public static void main(String[] args){
          int x=Integer.parseInt("five");
     }
}
```

(c) The file *file1.txt* has the following contents:

```
10
200
450
100
90
```

The *Q4C* class below is intended to do the following:

1. Opens *file1.txt* to read from.
2. Reads in each number, adds it to a running total called *sum*.
3. Closes the file once it has read all the numbers.
4. Opens a file to write to called *file2.txt.*
5. Writes the value in the *sum* variable into the file.
6. Closes the file.
7. Handles any potential exceptions with try/catch.
8. If an exception occurs the catch block prints a message about it.

In the class below the body of the *readFromFileAndAdd()* method is missing. Complete the *readFromFileAndAdd()* method so that the class works as intended. [8 marks]

```
import java.io.*;
import java.util.Scanner;

public class Q4C {

     static int sum = 0;

     public static void main(String[] args) {
          readFromFileAndAdd();
          writeResultToNewFile();
     }

     public static void readFromFileAndAdd() {
          //statements missing here
     }

     public static void writeResultToNewFile() {
          try {
               PrintStream out =new PrintStream(new
                 FileOutputStream("file2.txt"));
               out.print(sum);
               out.close();
          } catch (Exception e) {
               System.out.println("Error: unable to continue");
          }
     }
}
```

Question 5

(a)

(i) Java has simple (primitive) variables and reference variables. Say which of the following are primitive variables: [2 marks]

```
Double
char
Array
int
```

(ii) Say whether the following two statements are TRUE or FALSE: [2 marks]

(A) When you make an assignment to a reference variable, the variable does not hold the value assigned, but instead points to it in memory.

(B) Variables in Java have unlimited scope, and hence can be accessed from anywhere in their class.

(iii) Say what the output of the *StringParam* class will be: [2 marks]

```
class StringParam{

     public static void main(String [ ]
     args){
          String s="hello";
          p(s);
          System.out.println(s);
     }

     static void p(String m){
          m="goodbye";
     }
}
```

(iv) Say what the output of the *ArrayParam* class below will be: [2 marks]

```
class ArrayParam{
     public static void main(String [ ]
     args){
          int[]a=new int[1];
          a[0]=1;
          p(a);
          System.out.println(a[0]);
     }

     static void p(int[ ] m){
          m[0]=7;
     }
}
```

(b)

(i) Consider the class *Very*

```
public class Very{

     public static void print(String line) {
          System.out.println(line);
     }

     public static void print(String[] lines) {
          for (int i = 0; i < lines.length; i++) {
               System.out.println(lines[i]);
          }
     }

     public static void main(String [] args){
          print("hello");
     }
}
```

The class has two methods, *print(String)* and *print(String[ ])*. [3 marks]
Say which one of the following is true:

(A) The program will have a run-time error because of a name clash with the two print methods.

(B) This is an example of method overriding and will not prevent the program from compiling and running.

(C) This is an example of method overloading and will not prevent the program from compiling and running.

(D) None of the above.

(ii) Consider the classes *BooleanToInt* and *FloatToChar*

```
public class BooleanToInt{

     public static void main(String [ ] args){
          System.out.println((int)true);
     }
}

public class FloatToChar{

     public static void main(String [ ]
     args){
          System.out.println((char)90.79);
     }
}
```

Say which one of the following is true: [3 marks]

(A) Both classes will compile.

(B) Both classes will not compile.

(C) The *BooleanToInt* class will compile and output `1`; the *FloatToChar* class will not compile.

(D) The *FloatToChar* class will compile and output a `char` (in this case `Z`); the *BooleanToInt* class will not compile.

(iii) Which one of the following is the most likely output of the *Chars1* class below? [3 marks]

```
public class Chars1{

     public static void main(String [ ] args){
          System.out.println('8'+'A');
          System.out.println((char)('8'+'A'));
     }
}
```

(A)
```
8A
y
```

(B)
```
121
y
```

(C) No output – compilation error.

(D) No output – runtime error.

(c) The following details about the Caesar Cypher are provided for information only. You do not need to use this information to answer the question. You only need to know that the Caesar Cypher will encrypt some plain text using a number (used to shift the alphabet) to effect the encryption.

A Caesar Cypher is a very simple method of encrypting text, supposedly used by the Emperor Julius Caesar. It works by numbering the alphabet, the first letter as 1, and the remaining following in order. So the English alphabet would have the first letter a numbered as 1 and final letter z = 26. The alphabet is then shifted by a certain amount, and text is encrypted using this new alphabet. For instance if the shift was 5, as in the table below, then f would be encrypted as a, since f is now numbered 1. Note that shifts are wrapped from the end of the alphabet to the beginning. Hence a is numbered 22 in the encrypted alphabet below.

Example of a Caesar Cypher shift of 5

a	b	c	d	e	f	g	. . .	v	w	x	y	z
1	2	3	4	5	6	7	. . .	22	23	24	25	26
v	w	x	y	z	a	b	. . .	q	r	s	t	u
22	23	24	25	26	1	2	. . .	17	18	19	20	21

In the above shift, *a* in plain text would be encrypted as *v* in cipher text.

Consider the class *CaesarCypherUserInterface*. The class when run should:

1. Ask the user for some text to encrypt.
2. Ask the user for the shift to use.
3. Encrypt the text given using the shift given.
4. Show the user the result of the encryption to the user.

It does not do this since the method *getPlainTextAndShiftFromUserEncryptAndShowResult*() has not been completed. The method has one statement, that calls the *encrypt(String, int)* method from the *CaesarCypher* class. You should assume that this class exists, that the *CaesarCypherUserInterface* class can access it, and that the method works as it should to encrypt the `String` parameter using the `int` parameter as the shift.

Complete the *getPlainTextAndShiftFromUserEncryptAndShowResult()* method. Each of the statements that you add to the method should include a call to one of the methods in the *CaesarCypherUserInterface* class. You do not need to write any further methods. In your answer you only need to give your rewritten *getPlainTextAndShiftFromUserEncryptAndShowResult()* method. [8 marks]

```
import java.io.PrintStream;
import java.util.Scanner;

public class CaeserCypherUserInterface{
    private static Scanner input = new Scanner(System.in);
    private static PrintStream output = System.out;

    private static int getNumberFromUser() {
        return Integer.parseInt(input.nextLine());
    }

    private static void
     getPlainTextAndShiftFromUserEncryptAndShowResult() {
        String encrypted = CaeserCypher.encrypt(text, shift);
    }//some statements missing from this method

    private static void askUserForTextToEncrypt() {
        output.print("Enter text to encrpyt: ");
    }

    private static String getTextFromUser() {
        return input.nextLine();
    }

    private static void askUserForShift() {
        output.print("Enter shift: ");
    }

    private static void showEncryptedResults(String encrypted) {
        output.println("Encrypted version: " + encrypted);
    }

    public static void main(String[] args) {
        getPlainTextAndShiftFromUserEncryptAndShowResult();
    }
}
```

Question 6

(a) Answer TRUE or FALSE to each of the following statements: [8 marks]

(A) A constructor must have the same name as the name of its containing class.

(B) Constructors have return types.

(C) A constructor is used to make an instance of its class.

(D) A class can have up to three constructors.

(E) The keyword `this` can be used in a constructor to distinguish the formal parameter from the field name.

(F) An instance method does not have the keyword static in its heading.

(G) `Person extends SentientBeing` means that the *Person* class can use the public instance methods of the *SentientBeing* class.

(H) An inheriting class can redefine the public instance methods of its parent class.

(b) The class *Birthday* has the following output from test statements in its main method:

```
The birthday of Adam is on 12 May 1960. Card sent: true
The birthday of Eve is on 16 Oct 1990. Card sent: false
```

Consider the following numbered (from 1 to 13) jumbled up fragments of program code from the class *Birthday*:

Using **all** the code fragments (and nothing else) assemble the fragments in the right order such that the *Birthday* class will compile and give the above output when run. [9 marks]

Your program should follow convention in that the order of appearance of the different elements should be: instance variables; constructor; instance methods; and finally, the main method. The order of instance methods is arbitrary, that is you may put them in any order you choose.

1. `private boolean isSent() {`

2. `public Birthday(String name, String birthday, boolean cardSent) {`

3. `return cardSent;`
`}`

4. `public class Birthday {`

5. `this.name = name;`
`this.birthday=birthday;`
`this.cardSent = cardSent;`
`}`

6. `private String name;`
`private String birthday;`
`boolean cardSent;`

7. `public static void printArray(Birthday[] b){`
`for(int i = 0; i<b.length; i++)`
`if (b[i] != null)`
`System.out.println(b[i]);`
`}`

8. `public String toString(){`
`String s = ("The birthday of "+name+" is on ");`
`s = s+(birthday+".");`
`s = s+(" Card sent: "+isSent());`

9. `printArray(birthdayList);`
`}`
`}`

10. `return s;`
`}`

11. `Birthday adam = new Birthday("Adam", "12 May 1960", true);`
`Birthday eve = new Birthday("Eve", "16 Oct 1958", false);`

12. `public static void main(String[] args) {`
`//test statements`

13. `Birthday[] birthdayList = new Birthday[10];`
`birthdayList[0] = adam;`
`birthdayList[1] = eve;`

(c) Consider the class *PopQuiz*, below:

```
import java.util.*;
import java.util.Random;
import java.util.Scanner;

public class PopQuiz {

  private Question[] questions;
  private Scanner in;
  public PopQuiz() {
    questions = getQuestions();
    in = new Scanner(System.in);
    quiz();
  }

  private boolean getAnswerFromUser(String question) {
    System.out.print("True or false? " + question + " ");
    boolean answered = true;
    do {
      String answer = in.nextLine();
      if (answer.startsWith("y") || answer.startsWith("t")) {
        return true;
      } else if (answer.startsWith("n")||answer.startsWith("f"))
     {
        return false;
      } else {
        answered = false;
        System.out.println("I don't know what to do with that");
        System.out.println("Answer true / false, or yes / no.");
      }
    } while (!answered);
    return false; //will never get here but compiler complains
     if it's not there
  }

  public void quiz() {
    int correctAnswers = 0;
    for (int i=0;i<3;i++) {
      boolean answer =
        getAnswerFromUser(questions[i].getQuestion());
      if (answer == questions[i].getAnswer())  correctAnswers++;
    }
    System.out.println("You scored " + correctAnswers+" out of
     3");
  }
```

```
  private Question[]  getQuestions() {
    Question[]a = new Question[3];
    Question q0 = new Question("Java is case sensitive", true);
    Question q1 = new Question("An instance method has the
     keyword static in its heading", false);
    Question q2 = new Question("A class can have no more than
     three constructors", false);
    a[0]=q0;
    a[1]=q1;
    a[2]=q2;

    return a;
  }

  public static void main(String[] args) {
    new PopQuiz();
  }
}
```

The class is using an object called *Question*. Write the *Question* class. [8 marks]

END OF PAPER