THIS PAPER IS NOT TO BE REMOVED FROM THE EXAMINATION HALLS

UNIVERSITY OF LONDON **CO2220 ZB**

BSc Examination

COMPUTING AND INFORMATION SYSTEMS and CREATIVE COMPUTING

Graphical Object-Oriented and Internet Programming in Java

Date and Time: Tuesday 3 May 2016 : 10.00 – 13.00

Duration: 3 hours

Candidates should answer **FOUR** questions only. Full marks will be awarded for complete answers to **FOUR** questions. All questions carry equal marks and full marks can be obtained for complete answers to **FOUR** questions. The mark for each part of a question is indicated in [ ] brackets.

You must answer **TWO** questions from **Section A** and **TWO** questions from **Section B**.

There are 100 marks available on this paper.

Only your first **TWO** answers for section A, and your first **TWO** answers for section B, in the order that they appear in your answer book, will be marked.

No calculators should be used.

PART A

Answer TWO questions from this section.

Question 1

(a) What will be the output of the following statement? [2 marks]

```
System.out.println((int)5.8);
```

(b) What will be the output of the following program? [4 marks]

```
public class FairCoinToss{
     /*this program simulates the result of tossing a fair
     coin a number of times*/

     public static int CoinToss1(int n){
          int sum = 0;
          for (int i = 0; i < n; i++){
               int r = (int)Math.random()*2;
               sum = sum + r;
          }
          return sum;
     }

     public static void main(String[] args){
          System.out.println(CoinToss1(1000));
     }
}
```

(A) The output will always be zero

(B) No output - the program will not compile

(C) The output will be 500, or a number close to 500

(D) None of the above

(c) Consider the following two classes, *Ant* and *Dec*:

```
public class Ant{

     private String s;

     public static void main(String[] args){
          Ant ant = new Ant();
     }
}

/*******************************************/

public class Dec{

     private String s;

     public Dec (String s){
          this.s = s;
     }

     public static void main(String[] args){
          Dec dec = new Dec();
     }
}
```

(i) One class will compile correctly and one will not. Can you identify which one will compile without errors? [2 marks]

(ii) Can you explain why the other class will not compile? [2 marks]

(d) Consider the following two classes, *Jack* and *Jill*:

```
public class Jack{

     private String location;

     public Jack (String location){
          this.location = location;
     }

     public String getLocation(){
          return location;
     }
}

/****************************************/

public class Jill extends Jack{

     private String thing;

     public Jill (String l, String t){
          super(l);
          this.thing = t;
     }
}
```

(i) Explain the purpose of the statement `super(l);` in the class *Jill* [3 marks]

(ii) Write a no-argument constructor for the *Jack* class that makes use of the existing constructor to set the *location* instance variable to “down the hill”. [4 marks]

(e) Consider the following class:

```
public class SearchArray{

static int num[] = {5, 96, 42, 7, 13, 101, 12};

     public static boolean linearsearch(int[ ] a, int find){
          int i;
          for (i=0; i<a.length && a[i]!=find; i++);
          return (a[i]==find);
     }
     public static void main (String[] args){
          System.out.println(linearsearch(num,7));
          System.out.println(linearsearch(num,12));
          System.out.println(linearsearch(num,5));
          System.out.println(linearsearch(num,800));
     }
}
```

When run, the output is as follows:

```
true
true
true
Exception in thread "main"
     java.lang.ArrayIndexOutOfBoundsException: 7
/*some more information about the exception*/
```

Can you explain why? [8 marks]

Question 2

(a) Why will class *Q2a* give a compilation error? [2 marks]

```
abstract class Rabbit{

     public abstract String hop();
}

class Q2a{

     public static void main( String[] args){
          Rabbit d=new Rabbit();
     }
}
```

(b) Consider the following:

```
public abstract class Q2b{

     public abstract boolean isEmpty();
     public abstract void enqueue(Object item);
     public abstract Object front();
     public abstract Object dequeue();

     public String toString(int n, int[] a){
          String s = "";
          for (int i = 0; i < n; i++) s += a[i] + " ";
          return s;
     }
}
```

Which of the following best describes what will happen? [4 marks]

(A) the *Q2b* class will produce a compilation error because the *toString()* method is not abstract

(B) the *Q2b* class will compile correctly

(C) the *Q2b* class will compile correctly but there will be a run-time error in any extending classes because of the non-abstract *toString()* method

(D) none of the above

(c) Consider the following definition:

```
interface Animal{
     abstract boolean isCarnivore();
}
```

(i) Define a (non-abstract) class *Dog* that implements *Animal*. [4 marks]

(ii) Suppose we have pet dogs which are like dogs but they also have an age. Write a class for pet dogs (include a constructor) and a method for getting the dog's age. [5 marks]

(d) Consider the following program:

```
class HeapQuiz{
     int id = 0;

     public static void main(String[] args){
          int x = 2;
          HeapQuiz[] hq = new HeapQuiz[5];
          while (x < 5){
               hq[x] = new HeapQuiz();
               hq[x].id = x;
               x++;
          }

          hq[1] = hq[3];
          hq[3] = hq[4];
          hq[4] = hq[0];
          hq[0] = hq[2];

/*1*/     System.out.println("0: "+hq[0].id);
/*2*/     System.out.println("1: "+hq[1].id);
/*3*/     System.out.println("2: "+hq[2].id);
/*4*/     System.out.println("3: "+hq[3].id);
/*5*/     System.out.println("4: "+hq[4].id);
     }
}
```

The program will compile, but when it is run one of the numbered `System.out.println()` statements in the main method will cause a `NullPointerException` exception.

Which one of the numbered `System.out.println()` statements will cause an exception to be thrown? [5 marks]

(e) Consider the following two classes:

```
public class Machine{
     private String name;
     private int pin;
     public Machine (String name, int pin){
          this.name = name;
          this.pin = pin;
     }
     public Machine (String name){
          this.name = name;
          pin = 00000000;
     }
     public void speak(){
          System.out.println("The machines are rising");
     }
}
/*****************************************************/
public class Robot extends Machine{
     public Robot (String name, int pin){
          super(name, pin);
     }
     public void speak(){
          System.out.println("I am your robot overlord");
     }
     public void broadcast (){
          System.out.println("the robot uprising has
               begun");
     }
     public static void main (String[] args){
          Machine rob1 = new Machine("Robbie",5126555);
          Machine rob2 = new Robot ("Hal", 8887132);
          Robot rob3 = new Robot ("",2);
          rob1.speak();
          rob2.speak();
          rob3.broadcast();
     }
}
```

(i) Where does overloading occur in the *Machine* and *Robot* classes? [2 marks]

(ii) Give the output of the main method of the *Robot* class. [3 marks]

Question 3

(a) Say which of the following statements are true, and which are false:

(A) In order to handle events a GUI must implement a listener interface. [5 marks]

(B) When a class needs more than one `JButton`, in order to implement different actions when different `JButtons` are clicked, the accepted solution is to implement the Listener interface (eg `ActionListener`) once for each button using inner classes.

(C) When a class needs more than one `JButton`, in order to implement different actions when different `JButtons` are clicked, the accepted solution is to have the Listener call back method (eg *actionPerformed()*) query the event source.

(D) Inner classes have access to their containing classes public and protected variables and methods **only.**

(E) Event sources (such as a `JButton`) can only be registered with one event handler.

(b) Consider the class *TwoButtonGUI* below:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TwoButtonGUI{

     JFrame frame;
     JButton button1, button2;

     public static void main (String[] args){
          TwoButtonGUI gui = new TwoButtonGUI();
          gui.go();
     }

     public void go(){
          frame = new JFrame();
          button1 = new JButton("Don't click me, click him!");
          button1.addActionListener(new ActionListener(){
               public void actionPerformed(ActionEvent arg0) {
                    button2.setText("CLICK ME NOT HIM!");
                    frame.repaint();
               }
          });
          button2 = new JButton("CLICK ME!");
          button2.addActionListener(new ActionListener(){
               public void actionPerformed(ActionEvent arg0) {
                    button1.setText("Yeah click him again!");
                    frame.repaint();
               }
          });
          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          frame.getContentPane().add(BorderLayout.EAST, button1);
          frame.getContentPane().add(BorderLayout.WEST, button2);
          frame.setSize(500,500);
          frame.setVisible(true);
     }
}
```

The program uses two anonymous classes to implement the *ActionListener* interface. Rewrite the class so that it uses inner classes to implement the *ActionListener* interface. The output of the class should remain the same. [8 marks]
NOTE: In your answer you only need to write down your rewritten *go()* method and your new inner classes.

(c) Consider the class *MovingCircleGUI* below. When the class is compiled and run the user will see a 500 x 500 `JFrame` displaying an orange circle at position (0, 0) (the top left corner) with diameter 100. The class uses an inner class, *CircleDrawPanel,* to place a draw panel onto the `JFrame`. The draw panel displays the orange circle.

```
import javax.swing.*;
import java.awt.*;

public class MovingCircleGUI{
     JFrame frame;
     public int x=0;
     public int y=0;
     public int diameter=100;
     CircleDrawPanel drawPanel;
     Color color = Color.orange;

     public static void main (String[] args){
        MovingCircleGUI gui = new MovingCircleGUI();
        gui.go();
     }

     //this method sets up the JFrame and adds the drawpanel to the
     frame
     public void go(){
        frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        drawPanel = new CircleDrawPanel();
        frame.getContentPane().add(BorderLayout.CENTER, drawPanel);
        frame.setSize(500,500);
        frame.setVisible(true);
     }

     class CircleDrawPanel extends JPanel{
        public void paintComponent (Graphics g){
             super.paintComponent(g);
             Graphics2D g2=(Graphics2D)g;
             g2.setColor(color);
             g2.fillOval(x,y,diameter,diameter);
        }
     }
}
```

(i) Change the inner class *CircleDrawPanel* so that when the user runs the program they will see the orange circle move slowly **across** the draw panel until it exits the panel. [7 marks]

NOTE: You only need to write the amended *CircleDrawPanel* class in your answer book.

(ii) Change your *CircleDrawPanel* class, so that once the circle has left the panel and can no longer be seen by the user, it returns to its starting position, and the animation restarts. [5 marks]

PART B

Answer TWO questions from this section.

Question 4

(a) Say which of the following statements are true, and which are false: [3 marks]

(A) All classes are serializable by default

(B) Static variables **cannot** be serialized

(C) The state of a transient variable is saved when its containing object is serialized

(b) Consider the *SL* and *SLUtils* classes below:

```
import java.util.ArrayList;
public class SLTest {
     public static void main(String[] args) {
          ArrayList<SL> list = new ArrayList<SL>();
          list.add(new SL("Juventus", "Serie A", 44));
          list.add(new SL("Arsenal", "Premier League", 45));
          list.add(new SL("Manchester City", "Premier League", 44));
          list.add(new SL("Real Madrid", "La Liga", 48));
          list.add(new SL("Bayern Munich", "Bundesliga", 54));
          SLUtils.displaySL(list);
     }
}
/*****************************************************************************************/
import java.io.*;
import java.util.ArrayList;
public class SLUtils {
  private SLUtils() {}
  public static void displaySL(ArrayList<SL> list) {
    String header = String.format("%-20s %-25s %s%n","Name","League",
           "Points");
    System.out.print(header);
    for (SL sl : list) System.out.println(sl);
  }
}
```

(i) The access level of the constructor in the SLUtils class is private. What will be the effect of this? [3 marks]

(ii) Consider the *SLTest* class below:

```
import java.util.ArrayList;

public class SLTest {
  public static void main(String[] args) {
    ArrayList<SL> list = new ArrayList<SL>();
    list.add(new SL("Juventus", "Serie A", 44));
    list.add(new SL("Arsenal", "Premier League", 45));
    list.add(new SL("Manchester City", "Premier League", 44));
    list.add(new SL("Real Madrid", "La Liga", 48));
    list.add(new SL("Bayern Munich", "Bundesliga", 54));

    SLUtils.displaySL(list);
  }
}
```

What will be the output of the *SLTest* class? Choose one of the following possible outputs, A, B or C: [4 marks]

(A)
```
Name            League                                Points
Juventus        Serie A                               44
Arsenal         Premier League                        45
Manchester City Premier League                        44
Real Madrid     La Liga                               48
Bayern Munich   Bundesliga                            54
```

(B)
```
Name            League          Points
Juventus        Serie A         44
Arsenal         Premier League  45
Manchester City Premier League  44
Real Madrid     La Liga         48
Bayern Munich   Bundesliga      54
```

(C)
```
Name                 League                    Points
Juventus             Serie A                   44
Arsenal              Premier League            45
Manchester City      Premier League            44
Real Madrid          La Liga                   48
Bayern Munich        Bundesliga                54
```

(iii) Assume that the following method is added to the *SLUtils* class.

```
public static ArrayList<SL> fromSerialized(String filename) {
   ArrayList<SL> teams = null;
   ObjectInputStream in;
   try {
        /* THIS LINE IS MISSING */
        teams = (ArrayList<SL>) in.readObject();
        in.close();
   }
   catch (FileNotFoundException e) {
        System.err.format("File not found! %s", e);
        return null;
   }
   catch (Exception e) {
        System.err.format("Error reading from file: %s", e);
        return null;
   }
   return teams;
}
```

The method is intended to deserialize an arraylist of *SL* objects. There is a line missing. State which of the following is the missing line: [4 marks]

(A)
```
in = new ObjectInputStream(new
FileInputStream(filename));
```

(B)
```
FileWriter y = new FileReader(new
FileInputStream(filename));
```

(C)
```
BufferedReader y = new BufferedReader (new
FileReader(filename));
```

(iv) Write a method for the *SLUtils* class with the signature: [4 marks]

```
public static void serializeToDisk(ArrayList<SL> teams)
```

The method will serialize the `ArrayList` deserialized by the *fromSerialized()* method.

(c) Consider the following class:

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

public class ShowFile{

     public static void main(String[] args) throws Exception{
          Scanner in = new Scanner(new FileReader(args[0]));
          while (in.hasNextLine()){
               System.out.println(in.nextLine());
          }
     }
}
```

It has been written to read in a file and write the contents to standard output. The file name is given by the user when running the file, eg `ShowFile SomeTextFile.txt`

There is a problem, in that if the *ShowFile* class is given a file that does not exist as a parameter, it will immediately end with a `FileNotFoundException`. The following three classes have been written to correct this, with the intention that if the user enters a file name that does not exist, the class enters a user interaction loop where the user is repeatedly asked for a file name, until they enter one that is valid.

Of the following three classes, one works as it should, one will not compile, and one, if run with an invalid file name, will enter an infinite loop, repeatedly asking the user to enter a file name.

Identify which one of *Showfile2*, *ShowFile5* and *ShowFile6* works, which one will not compile and which one has a user interaction loop that is infinite. [7 marks]

```
import java.io.File;
import java.io.FileReader;
import java.util.Scanner;

public class ShowFile2 {

     public static void main(String[] args) throws Exception{
          boolean isFile = false;
          File file = new File(args[0]);
          if (file.exists()) isFile = true;
          while (!isFile) {
               System.out.print("Enter filename: ");
               Scanner input = new Scanner(System.in);
               file = new File(input.nextLine());
               if (file.exists()) {
                    isFile = true;
                    input.close();
               }
          }
          Scanner in = new Scanner(new FileReader(file));
          while (in.hasNextLine()){
               System.out.println(in.nextLine());
          }
          in.close();
     }
}
```

```
import java.io.File;
import java.io.FileReader;
import java.util.Scanner;

public class ShowFile5 {

    public static void main(String[] args) throws Exception{
        boolean isFile = false;
        File file = new File(args[0]);
        if (file.exists()) isFile = true;
        while (!isFile) {
            System.in.print("Enter filename: ");
            Scanner input = new Scanner(System.out);
            file = new File(input.nextLine());
            if (file.exists()) {
                isFile = true;
                input.close();
            }
        }
        Scanner in = new Scanner(new FileReader(file));
        while (in.hasNextLine()){
            System.out.println(in.nextLine());
        }
        in.close();
    }
}
```

```
import java.io.File;
import java.io.FileReader;
import java.util.Scanner;

public class ShowFile6 {

    public static void main(String[] args) throws Exception{
        boolean isFile = false;
        File file;
        do {
            file = new File(args[0]);
            if (file.exists()) isFile = true;
            System.out.print("Enter filename: ");
            Scanner input = new Scanner(System.in);
            file = new File(input.nextLine());
        } while (!isFile);
        Scanner in = new Scanner(new FileReader(file));
        while (in.hasNextLine()){
            System.out.println(in.nextLine());
        }
        in.close();
    }
}
```

Question 5

(a) Consider the following class:

```
public class Zx{

        public static final int EXAMPLE;

        public String name;

        public Zx(String n){
                name = n;
        }
}
```

(i) When the *Zx* class is compiled what error will the compiler identify? [4 marks]

(ii) Why has the *EXAMPLE* variable been named with all capital letters? [3 marks]

(b) Consider the class *Steady*, below. When the *Steady* class is compiled, what error will the compiler find? [4 marks]

```
public class Steady{

        private double radius;
        final double circumference = 2.0;

        public Steady(double r){
                radius = r;
        }

        final double calcCircumference(){
                circumference = 2.0 * radius * 3.142;
                return circumference;
        }
}
```

(c) Consider the classes *Good* and *Better* below. What mistake will the compiler find in class *Better*? [4 marks]

```
final public class Good{

     private String name;

     public Good(String n){
          name = n;
     }

     void greeting(){
          System.out.println("Hello!");
     }
}

/**************************************/

public class Better extends Good{

     public Better(String n){
          super(n);
     }

     void greeting(){
          System.out.println("hello baby!");
     }
}
```

(d) Consider the *NewException* and *AbsVal* classes below:

```
public class NewException extends RuntimeException{
     public NewException (String msg){
          super(msg);
     }
     public NewException(){}
}
/**************************************/
public class AbsVal{
     static int onlyPositive (int x) throws NewException{
          if (x<0) throw new NewException();
          return x;
     }
     public static void main (String[] args){
          int z = 0;
          System.out.println("hello");
          z = onlyPositive(-2);
          System.out.println(z);
     }
}
```

The *onlyPositive()* method in the *AbsVal* class throws a *NewException* when given a negative number as a parameter.

(i) What would the output of the main method in the *AbsVal* class be if the program were compiled and run?

(A) The program would not compile [5 marks]

(B)
```
Exception in thread "main" NewException
```
/* some more info about the exception*/

(C)
```
hello
Exception in thread "main" NewException
```
/* some more info about the exception*/

(D)
```
hello
Exception in thread "main" NewException
```
/* some more info about the exception*/
```
0
```

(ii) Change the *AbsVal* class so that if the exception is thrown then the user will see the message `"<value of x> is a negative number, positive numbers only"` [5 marks]

For example, when given -2 as a parameter, the *onlyPositive()* method will throw a *NewException* as follows:

```
Exception in thread "main" NewException: -2
is a negative number, positive numbers only
```

Question 6

(a) Once a new thread has been made it moves between three states: name the three states. [3 marks]

(b) Thread programming has hazards, including the 'lost update' problem. What of the following best describes the lost update problem? [3 marks]

(A) It is when a thread collision happens and the thread loses the data on the top of its call stack

(B) It is when thread *x* has the key that thread *y* needs in order to continue, and thread *y* has the key that thread *x* needs

(C) It is when a thread 'wakes up' and continues operating on a value that it had read before going to sleep, not knowing that another thread has changed it

(D) It is when the thread scheduler fails to move a thread that has completed its *run()* method to the BLOCKED state

(c) Consider the *go()* method below:

```
public void go(){

     try{
          Socket s = new Socket("190.165.1.103",4242);
          InputStreamReader i=new
                  InputStreamReader(s.getInputStream());
          int c;
          while(true){
               c=i.read();
               System.out.print((char)c);
          }
     }

     catch(IOException ex){
          ex.printStackTrace();
     }
}// end of method
```

(i) The above method is listening for output from a machine. Which machine is it listening to? [2 marks]

(ii) Which port on the machine is it listening to? [2 marks]

(d) Consider the class *Ty* below:

```
public class Ty implements Runnable{

     public void run(){
          doMoreStuff();
     }

     synchronized public void doMoreStuff(){
          System.out.println("Thread t is in charge");
     }

     public static void main (String[] args){
          Runnable theJob = new Ty();
          Thread t = new Thread(theJob);
          t.start();
          try{
               Thread.sleep(100);
          }
          catch(Exception e){}
          System.out.println("Main rules!");
     }
}
```

Which one of the following would you most expect to be output when class *Ty* is run? [4 marks]

(A)
```
Main rules!
Thread t is in charge
```

(B)
```
Thread t is in charge
Main rules!
```

(C) Both of the above are equally likely

(e) Consider the following server program, *ThreadedServer*, and the class *Handler*.

```
import java.io.*;
import java.net.*;

class ThreadedServer{

  boolean keepGoing = true;

  public void go(){
    try{
      ServerSocket s = new ServerSocket(7005);
      while(keepGoing){
        Socket c = s.accept();
        Thread t = new Thread(new Handler(c));
        t.start();
      }
      s.close();
    }
    catch(IOException e){
      System.err.println("Error while starting: "+e.getMessage());
    }
  }

  public static void main(String[] args) throws Exception{
    new ThreadedServer().go();
  }
}
    /*************************************/
    class Handler implements Runnable{

         Socket socket;
         public Handler(Socket s){
              socket = s;
         }

         public void run(){
              System.out.println("Connection from: "+socket);
         }
    }
```

Say which of the following are true about the *ThreadedServer* program, and which are false: [4 marks]

(A) It assigns a thread to each new connection

(B) It assigns threads so that it can listen on more than one socket

(C) After accepting a connection to a client, it prints the client address to the screen

(D) After accepting a connection, it sends its own IP address to the client

(f) Consider the following three classes. What will the user see when the *P* class is compiled and run? [7 marks]

```
public class P{

    synchronized void f(){
        while (true) System.out.println("hello");
    }

    synchronized void g(){
        while (true) System.out.println("goodbye");
    }

    public static void main(String[] args){
        P it= new P();

        T1 t1 = new T1(it);
        T2 t2 = new T2(it);
        t1.start();
        t2.start();
    }
}

public class T1 extends Thread
{
    P x;

    T1(P y){
        x=y;
    }

    public void run(){
        x.g();
    }
}

public class T2 extends Thread{

    P x;

    T2(P y){
        x=y;
    }

    public void run(){
        x.f();
    }
}
```

END OF PAPER