THIS PAPER IS NOT TO BE REMOVED FROM THE EXAMINATION HALLS

UNIVERSITY OF LONDON **CO2220 ZA**

BSc Examination

COMPUTING AND INFORMATION SYSTEMS and CREATIVE COMPUTING

Graphical object-oriented and internet programming in Java

Date and Time: Tuesday 2 May 2017: 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 **Part A** and **TWO** questions from **Part B**.

There are 100 marks available on this paper.

Only your first **TWO** answers for Part A, and your first **TWO** answers for Part 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) Consider the class *Boo*, and its child class, *SonOfBoo,* below. Note that *SonOfBoo* has FIVE numbered constructors.

```
public class Boo{
     private String s;
     private int i;

     public Boo(int i){
          this.i=i;
     }

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

     public Boo(String s, int i){
          this.s = s;
          this.i = i;
     }
}


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

class SonOfBoo extends Boo{
     private boolean b;

/*1*/     public SonOfBoo(String s, int i){
          this(i);
          super(s);
          }

/*2*/     public SonOfBoo(int i, boolean b){
          super(i);
          this.b = b;
          }

/*3*/     public SonOfBoo(int i){
          super("boo");
          }
```

```
/*4*/      public SonOfBoo(){
               super(42, "hello");
           }

/*5*/      public SonOfBoo(boolean b){
           super("hello");
           i = 42;
           }
     }
```

Three of the constructors in *SonOfBoo* will cause compilation errors.

(i) Identify the constructors in *SonOfBoo* that will cause compilation errors and give a brief (one or two sentence) explanation. [6 marks]
NOTE: You may identify constructors by their numbers in your answer.

(ii) Identify the constructors that are valid, and will not cause compilation errors. [2 marks]
NOTE: You may identify constructors by their numbers in your answer.

(b) Consider the following class, *PoorDog*:

```
class PoorDog{
     private int age;
     private String name;

     public int getAge(){
          return age;
     }

     public String getName(){
          return name;
     }

     public static void main (String[] args){
          PoorDog one = new PoorDog();
          System.out.println("Dog name is " +
             one.getName());
          System.out.println("Dog age is " + one.getAge());
     }
}
```

(i) The *PoorDog* class compiles and runs without error. Give the output when the main method of the class is executed. [3 marks]

(ii) Write a two parameter constructor for the class that allows the user to set the values of the *name* and *age* variables. [3 marks]

(iii) Assume that you have written a valid two parameter constructor for the *PoorDog* class, and have added it correctly to the program. When you recompile the class you get the following error:

```
error: constructor PoorDog in class PoorDog
     cannot be applied to given types;
          PoorDog one = new PoorDog();
                        ^
//some more information about the error

reason: actual and formal argument lists
differ in length
```

Explain the error. [3 marks]

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

```
public class Question {
     private String question;

     public Question(String question){
          this.question = question;
     }

     public String getQuestion(){
          return question;
     }
}
```

Extend the class to a *FreeQuizQuestion* class. The *FreeQuizQuestion* class has a `String` *answer* variable, a constructor, and a method to return the value of the *answer* variable. [8 marks]

Question 2

(a)

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

```
abstract class Triceratops{

     public abstract boolean
     runAwayFromTRex();
}

class Q2a{

     public static void main(String[] args)
     {
          Triceratops t=new Triceratops();
          t.runAwayFromTRex();
     }
}
```

(ii) 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? [3 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.

(iii) Given the following definition:

```
interface Mammal{
     abstract boolean isPlacental();
     abstract boolean isCarnivore();
}
```

Say which of the following classes will compile successfully and which will not. [3 marks]

```
abstract class Omnivore implements Mammal{
     abstract boolean isScavenger();
}

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

class Marsupial implements Mammal{
     public boolean isPlacental(){
                    return false;
     }
}

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

class Rabbit implements Mammal{

     public boolean isCarnivore(){
          return false;
     }

     public boolean isPlacental(){
          return true;
     }
}
```

(b)

(i) Consider the *ZZZ* class:

```
import java.util.*;

public class ZZZ{

     public static void main(String[] args{
          ArrayList  a = new ArrayList();
          a.add("hello");
          String s = a.get(0);
     }
}
```

The compiler finds the following error:

```
error: incompatible types: Object cannot be
converted to String
```

Identify the statement giving the error and write a correction so that the program will compile successfully. [3 marks]

(ii) Consider the class *YYY*:

```
import java.util.*;

class Machine{
     public Machine(){}
}

class Computer extends Machine{
     public Computer(){}
}

class LapTop extends Computer{
     public LapTop(){}
}

public class YYY{

     public static void main(String[] args){
       ArrayList <Machine> a = new ArrayList<Machine>();
       a.add(new Machine());
       a.add(new Computer());
       a.add(new LapTop());
       ArrayList <Computer> b =new ArrayList<Computer>();
       b.add(new Machine());
       b.add(new LapTop());
       b.add(new Computer());
     }
}
```

The *YYY* class will not compile. Considering the definitions of *Machine*, *Computer* and *LapTop* can you identify the statement causing the single compilation error? [3 marks]

(iii) For each of the following statements, say whether it is TRUE or FALSE [3 marks]

(A) *remove(), indexOf()* and *isEmpty()* are methods of the `ArrayList` class.

(B) The enhanced *for* loop can be used for iterating through what Java calls collections. It can be used with an `Array` because Java considers the class to be a collection.

(C) `ArrayLists` are preferred to arrays as they provide much faster access than `Arrays`.

(c) Consider the *NoisyDog* class

```
public class NoisyDog{

     public static void main (String[] args){
          Dog dog1 = new Dog("Fido", true);
          Labrador dog2 = new Labrador("Wuffles", true);
          dog1.howl();
          dog1.growl();
          dog2.growl();
     }
}
```

The *NoisyDog* class compiles and runs, with the following output:

```
oooooooooaaaawooool
grrrrr
grrrrr
grrr grrrrr GRRRRRRR
```

The *Dog* and *Labrador* classes used in the *NoisyDog* class are defined by the numbered code fragments below, when they are in the correct order. Put the numbered fragments into the right order, so that the *Dog*, *Labrador* and *NoisyDog* classes will all compile (assuming that they are in the same package) and *NoisyDog* will run successfully and give the output above. You should use only the numbered code fragments below in your answer, and should not add any other Java code. [8 marks]

In your answer you should follow the order:

1. instance variables
2. constructor
3. instance methods (may be in any order)

Please write the code fragments in full, and **not** just their numbers.

```
/*1*/      public class Labrador extends Dog{

/*2*/      System.out.println(" grrr grrrrr GRRRRRRR");
           }
     }

/*3*/      public class Dog{

/*4*/      public String getName(){
               return name;
           }

/*5*/      public void howl (){
              System.out.println("ooooooooooaaaawoooool");
     }

/*6*/      public void growl(){
               super.growl();

/*7*/      public boolean getOldDog(){
               return oldDog;
           }

/*8*/      }

/*9*/      public Labrador (String name, boolean old){

/*10*/     public Dog (String name, boolean isOld){

/*11*/     this.name = name;
           oldDog = isOld;
     }

/*12*/     super(name, old);
     }

/*13*/     private String name;
           private boolean oldDog;

/*14*/     public void growl(){
               System.out.println("grrrrr");
           }
```

Question 3

(a) Say which of the following statements are TRUE, and which are FALSE:

(A) An example of event handling is listening for events such as a clicked button in a GUI, and taking some action in response. [8 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 **all** of their containing classes variables, including private variables.

(E) Event sources (such as a `JButton`) can be registered with multiple event handlers.

(F) If a region is not specified then the single parameter *add()* method of *BorderLayout* defaults to the CENTER region.

(G) The BorderLayout manager has 4 regions.

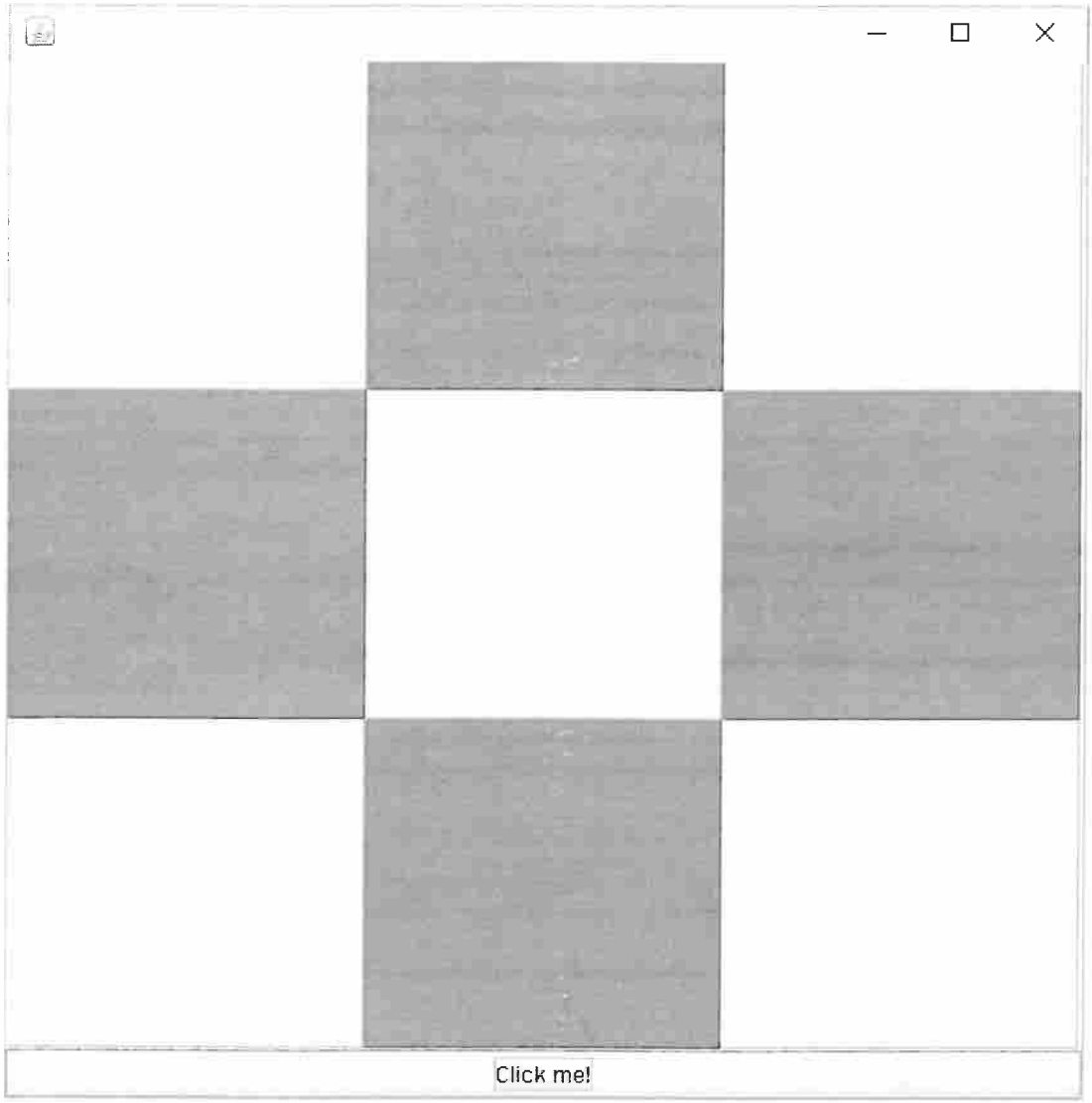
(H) Swing applications should **not** directly call the *paintComponent()* method.

(b) Consider the class *NineRectanglesGUI* class, and its output, below:

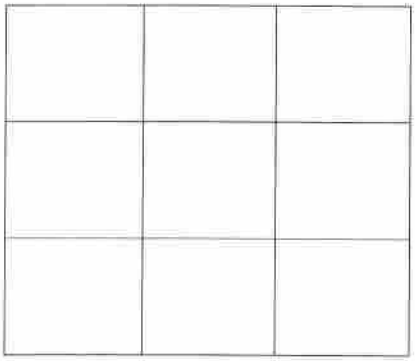
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class NineRectanglesGUI{
     JFrame frame;
     JButton rColorButton;
     RectangleDrawPanel dP;
     Color color1 = Color.red;
     Color color2 = Color.cyan;
     boolean color1Change = true;

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

     public void go(){
          frame = new JFrame();
          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          dP = new RectangleDrawPanel();
          frame.getContentPane().add(BorderLayout.CENTER, dP);
          rColorButton = new JButton("Click me!");
          rColorButton.addActionListener(new RColorListener());
          frame.getContentPane().add(BorderLayout.SOUTH,
              rColorButton);
          frame.setSize(600,600);
          frame.setVisible(true);
     }

     class RColorListener implements ActionListener{
               public void actionPerformed (ActionEvent e){
          }
     }
```

//NineRectanglesGUI class continues on next page

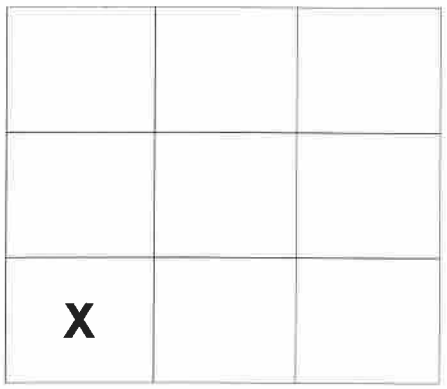
```
//NineRectanglesGUI class continued

     class RectangleDrawPanel extends JPanel{
          public void paintComponent (Graphics g){
               super.paintComponent(g);
               Graphics2D g2=(Graphics2D)g;
               int w = getWidth()/3;
               int h = getHeight()/3;
     /*1*/       g2.setColor(color2);
     /*2*/       g2.fill3DRect(0,0,w,h,true);
     /*3*/       g2.setColor(color1);
     /*4*/       g2.fill3DRect(0,h,w,h,true);
     /*5*/       g2.setColor(color1);
     /*6*/       g2.fill3DRect(w,0,w,h,true);
     /*7*/       g2.setColor(color2);
     /*8*/       g2.fill3DRect(w,h,w,h,true);
     /*9*/       g2.setColor(color2);
     /*10*/      g2.fill3DRect(0,2*h,w,h,true);
     /*11*/      g2.setColor(color2);
     /*12*/      g2.fill3DRect(2*w,0,w,h,true);
     /*13*/      g2.setColor(color1);
     /*14*/      g2.fill3DRect(2*w,h,w,h,true);
     /*15*/      g2.setColor(color1);
     /*16*/      g2.fill3DRect(w,2*h,w,h,true);
     /*17*/      g2.setColor(color2);
     /*18*/      g2.fill3DRect(2*w,2*h,w,h,true);
          }
     }
}
```



(i) Copy the schematic of the output (below) into your answer book, and write 'color1' or 'color2' in the squares corresponding to the rectangles of the output which have been filled with *color1* or *color2*. [3 marks]



(ii) Write in your answer book the two lines of code that are colouring, making and filling the rectangle indicated by an 'x' in schematic of the output below: [3 marks]
The statements have been numbered from 1 to 18 so that you may give your answer as the numbers corresponding to the statements.



(iii) Which one of the following would happen if the user were to run the program, and increase the size of the frame by dragging one of the corners while the program is running? As the frame increased in size would the filled rectangles: [3 marks]

(A) Stay the same size leaving areas of the frame unfilled.

(B) Grow with the size of the frame (so that the user will always be able to see nine complete rectangles).

(C) The user would not be able to increase the size of the frame while the program is running.

(D) None of the above.

(iv) You will note that in the *NineRectanglesGUI* class, the *actionPerformed(ActionEvent)* method of the *RColorListener* inner class has not been completed. [8 marks]
Complete the *actionPerformed(ActionEvent)* method so that when the *rColorButton* is clicked for the first time, *color1* changes to a random *Color*, while *color2* is unchanged. On each subsequent click, either *color1* or *color2* changes to a random *Color*, while the *Color* variable that was changed by the previous button click remains the colour that it was changed to. The user can keep on clicking the button for a random colour change indefinitely.

PART B

Answer TWO questions from this section.

Question 4

(a)

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

(A) `StringBuilder` or `StringBuffer` should be used instead of `String` when writing classes with a lot of `String` concatenation.

(B) `StringBuilder` objects are immutable.

(ii) Consider the SFA program below:

```
public class SFA{

     public static String formatter() {
          String a = "Come in number";
          int b = 11;
          String c = "over";
          /*missing statement*/
          return s;
     }

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

You will note that a statement has been removed and replaced with the comment "`/*missing statement*/`"

Given that the output of the program is:

```
Come in number 011 over
```

Say which one of the following statements is the missing statement: [3 marks]

(A) `String s = String.format("%-14s %03d %-5s", a,b,c);`

(B) `String s = String.format("%-28s %03d %-5s", a,b,c);`

(C) `String s = String.format("%-14s %04d %-5s", a,b,c);`

(D) `String s = String.format("%-28s %04d %-5s", a,b,c);`

(iii) What will be the output of the *StringThing* class when run? [3 marks]

```
class StringThing{
  public static void main(String[] args){
    String eg = "11,22,44";
    String[] result1 = eg.split(",");
    for (String token:result1) System.out.println(token);
  }
}
```

(A)
```
11,22,44
```

(B)
```
11
22
44
```

(C)
```
11,
22,
44
```

(b)

(i) Consider the following class:

```
public class Rx{

    public static final int FORINSTANCE;
    public int size;

        public Rx(int s){
            size = s;
    }
}
```

When the *Rx* class is compiled what error will the compiler identify? [3 marks]

(ii) Consider the classes *Happy* and *Happier*, below. When the *Happier* class is compiled, what error will the compiler find? [3 marks]

```
public class Happy{
     private String name;

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

     final void askToDance(){
       System.out.println("Shall we dance the Foxtrot?");
     }
}

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

public class Happier extends Happy{

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

     void askToDance(){
       System.out.println("Shall we dance the Rumba?");
     }
}
```

(iii) Consider the class *Ready* below. What mistake will the compiler find when compiling class *Ready*? [3 marks]

```
public class Ready{

     private int length, width;
     final int area = 2;

     public Ready(int l, int w){
          length = l;
          width = w;
     }

     final int calcArea(){
          area = width * length;
          return area;
     }
}
```

(c) Consider the *Employee* and the *PersonnelFromFile* classes, together with the contents of the *personnel.csv* file below:

```
public class Employee{

     private String name;
     private String jobTitle;
     private String teamName;
     private int age;
     private boolean permanent;

     public Employee(String name, String jobTitle, String teamName,
     int age, boolean permanent) {
          this.name = name;
          this.jobTitle = jobTitle;
          this.teamName = teamName;
          this.age = age;
          this.permanent = permanent;
     }

     public String toString() {
          return name + ", " + jobTitle + ", " + teamName + ", " +
               age +", " + permanent;
     }
}

/*************************************************************/
```

```
import java.util.ArrayList;
import java.util.Scanner;
import java.io.IOException;
import java.io.FileReader;

public class PersonnelFromFile{

     private static String filename = "personnel.csv";
     private static ArrayList<Employee> employees;

     public static void main (String[] args){
          employees = readEmployees();//test statements
          System.out.println(employees.get(1));
     }

     private static ArrayList<Employee> readEmployees() {
          Scanner in = null;
          ArrayList<Employee> workers = new ArrayList<Employee>();
          try {
               in = new Scanner(new FileReader(filename));
          }
          catch (IOException e) {
               System.err.println("Error reading from file.");
          }

          while(in.hasNextLine()) {
               String line = in.nextLine();
               Employee temp = parseEmployee(line);
               workers.add(temp);
          }
          in.close();
          return workers;
     }

     private static Employee parseEmployee(String line) {
          //some statements missing here
          return new Employee(name, jobTitle, teamName, age,
             permanent);
     }
}

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

     //contents of personnel.csv file, below (3 lines)
     A.N. Employee,Systems Analyst,Logistics,20,0
     A.N. Otherdude,Software Developer,Logistics,25,1
     A Dudess,Editor,Web Team,27,1
```

The *PersonnelFromFile* class has been written to read a comma-separated text file (personnel.csv), and convert each line into an *Employee* object. Information about an employee is stored in the csv text file in the order that instance variables are addressed in the constructor: ie Name, Job Title, Team Name, Age, Permanent status (ie are they a permanent member of staff or an agency worker). In the text file permanent status is recorded as 0 for false, and 1 for true.

Write the missing statements from the *parseEmployee(String)* method of the *PersonnelFromFile* class. [8 marks]

Question 5

(a)

(i) Java has two types of stream; *connection* is one, name the other. [2 marks]

(ii) Consider the following two statements:

```
FileWriter writer = new FileWriter("Foo.txt");

BufferedWriter writer = new BufferedWriter(new
     FileWriter("Foo.txt"));
```

Both statements represent ways of writing to a file. Which one would it be better to use in terms of the efficient use of system resources? Justify your answer. [3 marks]

(iii) Consider the following class:

```
import java.net.*;

public class MA{
     public static void main(String[] args ){
          try {
               InetAddress inetAddress =
                  InetAddress.getLocalHost();
               System.out.println(inetAddress.getHostAddress());
          }
          catch(UnknownHostException e){
               System.out.println(e);
          }
     }
}
```

What is it that the class will do when it is run? [3 marks]

(A) The class will find an IP address from a host name.

(B) The class will print the IP address of the current machine to standard output.

(C) The class will broadcast the IP address of the current machine.

(D) The class will resolve a host name from an IP address.

(b)

(i) What is saved when an object is *serialized*? Choose one from (A), (B) or (C) below: [2 marks]

(A) An object's state, given by its instance variables is saved. Any objects that are referenced by the instance variables, and in turn any further objects referenced by their instance variables etc are also saved.

(B) An object's static and instance variables are saved. Any objects that are referenced by the instance variables, and in turn any further objects referenced by their instance variables etc are also saved.

(C) The object's source code.

(ii) Can you say which of the following statements are true, and which are false? [4 marks]

(A) If a super class is **not** serializable then the sub class cannot be serializable.

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

(C) The state of a transient variable is **not** saved when its containing object is serialized.

(D) Serialization can throw runtime exceptions.

(iii) What is the purpose of a *serial version ID*? Choose one of the following statements: [3 marks]

(A) One issue with object deserialization is a possible change of class definition which may break the deserialization process. A serial version ID can solve this problem as it enables the JVM to assess if the class is compatible with the serialized object.

(B) A serial version ID is used in the serialization process to allow the JVM to keep track of the object's graph.

(C) A serial version ID is used in the deserialization process to ensure that the correct object is retrieved.

(c) Consider the *Point* and *PartC* classes, below:

```
import java.io.*;

public class Point implements Serializable{
     private int x;
     private int y;

     public Point(){
          x = 50;
          y = 90;
     }
}
```

```
import java.io.*;

public class PartC{

     static String s = "temp.ser";

     public static void serializeToFile(Point a) {
          try {
               FileOutputStream fos = new FileOutputStream(s);
               ObjectOutputStream oos = new ObjectOutputStream(fos);
               oos.writeObject(a);
               fos.close();
               oos.close();
          }
          catch (IOException ex) {
               ex.printStackTrace();
          }
     }

     public static void main (String[] args){
          Point a = new Point();
          Point b = new Point();
          serializeToFile(a);
          b=deserializeFromFile();
     }
}
```

The *PartC* class will not compile as it is, since the main method calls a *deserializeFromFile()* method that has not been written.

Write the *deserializeFromFile()* method. [8 marks]

Question 6

(a)

(i) Consider the *Exep* class:

```
class Exep{
     public static void main(String [] args){
          try{
               Integer.parseInt("aaa");
               System.out.println("Vala");
          }

          catch (Exception e){
               System.out.println("Tealc");
          }
     }
}
```

Give the output of the class. [2 marks]

(ii) Name the class that **all** exceptions sub-class. [2 marks]

(iii) Why are some Java exceptions checked and some unchecked? Choose one of the statements below that best describes the reason: [2 marks]

(A) Unchecked exceptions are usually due to faulty code logic, rather than unpredictable or unpreventable conditions that arise when the code is running. As such they should be taken care of by writing classes that are logically correct.

(B) Unchecked exceptions are a legacy issue from earlier versions of Java. They are now deprecated.

(C) Because exception checking is memory heavy, only those most critical to successful internet and network programming are checked.

(iv) Say which of the following five exceptions are checked, and which are unchecked: [2 marks]

```
ArithmeticException
ArrayIndexOutOfBoundsException
FileNotFoundException
NumberFormatException
UnknownHostException
```

(b)

(i) One word has been replaced with X's in both of the following numbered statements. For both statements give the number and the missing word. [2 marks]

(1) A thread may be XXXXXXX if, for example, it is waiting for data, is sleeping for a while, or is trying to access a locked object.

(2) A Thread is XXX when it has been instantiated but its *start()* method has not yet been called.

(ii) 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.

(iii) 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.

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

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

public class SourceViewer{

     public static void main (String args []) throws Exception{
          URL u = new URL ( args [0]);
          Reader r = new BufferedReader
               (new InputStreamReader(u.openStream()));
          int c = -1;
          while ((c = r. read())!= -1) {
               System.out.print((char)c);
          }
     }
}
```

Rewrite the *SourceViewer* class to handle exceptions. Your answer should have at least two *catch* blocks. [8 marks]

END OF PAPER