UNIVERSITY OF JOHANNESBURG

FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE CSC2A10
OBJECT ORIENTED PROGRAMMING

CAMPUS AUCKLAND PARK CAMPUS (APK)

EXAM JUNE 2014

DATE 09/06/2014 **SESSION** 8:30-10:30

ASSESOR(S) MR DT VAN DER HAAR
MR A MAGANLAL

INTERNAL MODERATOR DR ID ELLEFSEN

DURATION 2 HOURS **MARKS** 100

NUMBER OF PAGES: 7 PAGES

INSTRUCTIONS: ANSWER ALL THE QUESTIONS

REQUIREMENTS: AN EXAM BOOK FOR THE STUDENT

SECTION A - Theory

QUESTION 1

Java Basics

(a) **Name** the programming language **class** the source code below belongs to and list two **advantages** it has over other programming classes. [3]

```
1  class Fruit implements Clonable{
2
3  private String fruitName;
4  private float weight;
5
6  public Fruit (){
7  }
8
9  public setFruitName (String name){
10   this.fruitName = name;
11 }
12
13 public getFruitName (){
14   return fruitName;
15 }
16
17 public setFruitWeight (double weight ){
18   this.weight = weight;
19 }
20
21 public getFruitWeight (){
22   return weight;
23 }
24
25 public String toString (){
26   return String.format ("%s:%.2f", fruitColor, weight);
27 }
```

(b) By analysing the above source code, what are **differences** we can see between this language and C++? [4]

(c) What is a pure-virtual function? **List** 1 reason why using it can cause problems in programs. [3]

[10]

QUESTION 2

Elementary Java Programming

(a) In Java, what is the storage size of a **short** and **float**? What makes the data types in Java **different** from C++? [4]

(b) Briefly **describe** what the dangling else problem is and describe **how** it can be avoided. [4]

(c) Briefly explain **pass-by-sharing** within the context of methods in Java. [2]

[10]

QUESTION 3

(a) Briefly describe **variable length parameter** lists. Provide a short Java **source code** segment that demonstrates this concept. [3]

(b) What is a **Lliffe** vector? [2]

(c) Classify the **Bubble Sort** algorithm in terms of: [3]

- Computational complexity
- Memory usage
- Iterative versus recursive approach

[8]

QUESTION 4

Text Processing and Persistence

(a) What **occurs** in the Java source code below? What is this **called**? [3]

```
1  public class Test{
2    public void static main(String[] args){
3      String s1 = "Test";
4      String s2 = "Test";
5
6      if (s1==s2){
7        System.out.println("Proof!");
8      }
9      else{
10       System.out.println("Failed");
11     }
12   }
13 }
```

(b) Aside from the *Scanner* class, name another stream class that can be used for **text** reading. [1]

(c) Provide a **regular expression** that provides the following match result (you do **not** have to provide Java source code): [4]

```
<html>
<title>This is a test!</title>
<a1>That shows how only certain tags are matched</a1>
<applet code="Test.class"
 width = "640"
 height = "480"/>
<p>in an html file</p>
<a href="index.html>but ignores block and complex tags</a>
<a4>Done!</a4>
</html>
```

[8]

QUESTION 5

Object Orientation

(a) List 3 **requirements** a class must conform to in order for it to be an immutable class. [3]

(b) Dynamic binding is required in order to achieve polymorphism. What is **dynamic binding**? [2]

(c) List 3 **properties** of abstract classes. [3]

[8]

QUESTION 6

Graphical User Interfaces

(a) What is a layout manager and why is it **useful**? Name 2 **examples** that can be found in Java. [4]

(b) Analyse the Java source code below and list any **problems** found. [4]

```
1  public class Test extends javax.swing.JFrame{
2
3    public Test(){
4      javax.swing.JButton btn = new JButton("Test");
5      addActionListener(new Listener);
6      add(new javax.swing.JLabel("Click Test"));
7      setSize(100,200);
8      setVisible(false);
9    }
10
11   public static void main(String[] args){
12     new test();
13   }
14
15   class Listener {
16     public void actionPerformed(ActionEvent e){
17       System.out.println("Test has been clicked");
18   }
19 }
```

[8]

QUESTION 7

Advanced Java Programming

(a) Briefly explain what an **assertion** is and where it is used. [2]

(b) Describe the 3 **wild card formats** supported for generics in Java. [3]

(c) Name 3 **states** an applet in Java can transition/move into in the applet life cycle. [3]

[8]

QUESTION 8

Design Patterns

(a) Briefly describe how the following **design patterns** work: [4]

1. Facade
2. Adapter
3. Visitor
4. Singleton

(b) An online shoppping company, which sells *Products* explicitly caters to *Gamers*, has approached you to provide them with the ability to notify *Gamers* when *Video Games* are available in the country. Using best practices (such as appropriate object encapsulation and design patterns) provide a class diagram that would help the company to their *Customer*'s needs. [6]

[10]

SECTION B - Practical

QUESTION 9

Provide Java source code for a `paintComponent` function which will draw a blue rectangle of size 50 x 50.

[10]

QUESTION 10

Provide Java source code for a function to read from a binary file called *code.bin*. The file contains a integer followed by a string followed by two more integers. Display each piece of data to the standard output stream. Remember to include any necessary exception handling. You can assume that all necessary packages have been imported.

[10]

QUESTION 11

Read the code below and provide the missing code (in the segments labelled as A to G).

```
1  import java.util.ArrayList;
2  import java.util.Random;
3
4  class RacingTeam implements __(A (2 marks))__
5  {
6    private final Random  randGen=new Random();
7    private final String  teamName;
8    private long  time;
9
10   public RacingTeam(String name)
11   {
12     this.teamName = name;
13   }
14
15   public String getName()
16   {
17     return teamName;
18   }
```

```
19
20    public long getTime()
21    {
22      return time;
23    }
24
25    @Override
26    public __(B (1 marks))__ run()
27    {
28      System.out.format("%s starting engine%n", name);
29      time = (randGen.nextInt(30) + 1) * 10;
30      try
31      {
32        // Simulate racing time
33        Thread.__(C (2 marks))__(time);
34      }
35      catch (__(D (2 marks))__ e)
36      {
37        e.printStackTrace();
38      }
39      System.out.format("%s finished in %d%n", name, time);
40    }
41
42  }
43
44  public class BigRace
45  {
46
47    public static void main(String[] args)
48    {
49      ArrayList<RacingTeam> teams = new ArrayList<>();
50      teams.add(new RacingTeam("Ferrari"));
51      teams.add(new RacingTeam("Force India"));
52      teams.add(new RacingTeam("Lotus"));
53      teams.add(new RacingTeam("Marussia"));
54      teams.add(new RacingTeam("McLaren"));
55      teams.add(new RacingTeam("Red Bull"));
56      teams.add(new RacingTeam("Sauber"));
57      teams.add(new RacingTeam("Toro Rosso"));
58      teams.add(new RacingTeam("Williams"));
59
60      RacingTeam winner = null;
61      ArrayList<__(E (1 marks))__> threads = new ArrayList<>();
62
63      for (RacingTeam team : teams)
64      {
65        Thread t = new Thread(__(F (1 marks))__);
66        threads.add(t);
67        t.start();
68      }
69
70      for (Thread t : threads)
71      {
72        try
73        {
74          t.join();
75        }
76        catch (InterruptedException e)
77        {
78          e.printStackTrace();
79        }
80      }
```

```
81      long time = Long.MAX_VALUE;
82      for (RacingTeam team : teams)
83      {
84        if (team.getTime() < time)
85        {
86          winner = team;
87          time = __(G (1 marks))__;
88        }
89      }
90
91      System.out.println("The winner is \t" + winner.getName());
92
93    }
94
95  }
```

[10]

The End!