FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE	**COMPUTER SCIENCE 2A** CSC02A2
CAMPUS	AUCKLAND PARK CAMPUS (APK)
FINAL SUMMATIVE ASSESSMENT OPPORTUNITY SSA	JULY
DATE: 2021-07-15	**SESSION:** 08:00 - 10:00
ASSESSOR(S):	**MR. A. MAGANLAL**
	MR. S. SITHUNGU
MODERATOR:	**MR. R. MALULEKA**
DURATION: 120 MINUTES	**MARKS:** 100

Please read the following instructions carefully:

1. You must complete the assessment **by yourself** within the prescribed time limits.
2. No communication concerning the assessment is permissible during the assessment session except with **ACSSE** staff members.
3. You are bound by all university regulations including, but not limited, to assessment, plagiarism, and ethical conduct.
4. You may not directly take any code from any source, including your own previous submissions. All code must be written by yourself during the assessment.
5. If you **do not have access to a computer** then you can do a *pen and paper submission*:
   - Write *cleanly* and *legibly*.
   - Make use of CamScanner app to create a PDF from your written work.
6. **All answers** must be in a *single* **PDF** file. Make sure your details appear at the top of the first page of the PDF file.
7. Complete the **Honesty Declaration: Online Assessment** and upload it as part of your submission. The completed **Honesty Declaration** is required for a submission to be eligible to be marked.
8. Your answers to the question (in a single PDF file) together with the declaration must be submitted in a zip archive named in the following format:
   `SURNAME_INITIALS_STUDENTNUMBER_CSC2A_2021_FSAO_SSA.zip`
9. Additional time for submission is allowed for as per the posted deadlines on EVE.
10. This paper contains **9** question(s).
11. This paper consists of **5** page(s) excluding the cover page.

QUESTION 1: Java Overview

JavaDoc is a unique documentation mechanism available in **Java**.

(a) **Discuss** how writing ***JavaDoc comments*** are different to normal *code comments*. **[06]**

(b) **Discuss** the process of compiling **JavaDoc**. Your discussion must include the command-line tool(s) and the inputs and outputs of the tool. **[04]**

Total: 10

QUESTION 2: Elementary Java Programming

(a) **List** two (2) *advantages* of ***using methods*** in a program as opposed to having code defined in the ***main*** method. **[04]**

(b) In Java **String**s are *immutable*. **How** can a mutable string be obtained in Java? **[02]**

(c) **Provide** ***Java*** *source code* to reference the ***standard error stream*** of a Java application. **[01]**

(d) **Analyse** the following **Java** code segment and answer the questions that follow: **[03]**

```
1 long 1LongNumber = 5;
2 double doubleNumber = 42;
3 float $cash = 1337.37;
4 boolean true = false;
5 Object[] o_b_j_e_c_t_s =  null;
6 short __ = 1;
7 long 2ndLong = 6;
```

Java has strict rules for identifiers. Some of the identifiers in the code segment above break those rules. **Indicate** the line where the ***problem*** occurs and what the error is on that line.

Total: 10

QUESTION 3: Text Processing and Persistence

(a) **Can** a **Scanner** be used to ***read binary data***? **Provide** a *reason* for your answer. **[03]**

(b) When writing objects using an **ObjectOutputStream** some data members cannot be written. **Which** *data members* cannot be ***serialized***? **[02]**

(c) **Provide** a *single* ***regular expression*** that matches *all t-shirt size, styles and colours* in the following format. **[05]**

- XXL sleeveless YELLOW 44”
- XS basic RED 34”
- XL vneck MAGENTA 42”
- M graphic BLUE 38”
- L basic CYAN 40”
- S vneck GREEN 36”

Total: 10

~~ Assessment continues on the next page. ~~

QUESTION 4: Object Orientation

Analyse the following **Java** code segment and answer the questions that follow:

```java
 1 interface Steerable {
 2   void steer();
 3 }
 4 class Vehicle implements Steerable {
 5   public void steer()  { System.out.println("Steering..."); }
 6   /* remainder of class omitted */
 7 }
 8 class Driver implements Steerable{
 9   private Vehicle v;
10   public Driver(Vehicle v) { this.v = v; }
11   public void steer()  { v.steer(); }
12   /* remainder of class omitted */
13 }
14 public class VehicleSim {
15   public static void main(String... args) {
16     Vehicle v1 = new Vehicle();
17     Driver d1 = new Driver(v1);
18     Driver d2 = new Driver(v1);
19     d1.steer();
20     d2.steer();
21   }
22 }
```

Identify five (5) unique **Java** ***programming constructs/principles*** in the code above. **List** the line number and the ***programming constructs/principles***, for example: *Line 99 - Exception Handling*.

Total: 10

QUESTION 5: Graphical User Interfaces

Compare the following ***Java GUI frameworks***: AWT and JavaFX. Your comparison must include both *similarities* and *differences* between the frameworks.

Total: 10

~~ Assessment continues on the next page. ~~

QUESTION 6: Advanced Java Programming

(a) **Analyse** the following **Java** code segment and answer the questions that follow:

```java
public void method1(File myfile) {
  try (Scanner txtin = new Scanner(myfile))  {
    /* read from file */
  }
  catch (IOException e) {
      /* Handle error */
  }
}

public void method2(File myfile) throws IOException
{
  try (Scanner txtin = new Scanner(myfile))  {
    /* read from file */
  }
}
```

i. **Is *IOException*** an *unchecked exception*? **[01]**

ii. **Which *method*** (method1 or method2) is the correct way to read frpm a file? **Provide** a ***reason*** why you would this method. **[04]**

(b) With regards to multi-threading in **Java**, **discuss** the *similarities* and *difference* between when a **Thread *yield***s compared to when a **Thread *sleep***s. **[05]**

Total: 10

QUESTION 7: Design Patterns

The SelfDriveSoft company is working on software for remotely controlling Wi-Fi enabled vehicles. SelfDriveSoft wants their software to make the process of controlling a vehicle well designed and abstracted, and as such, the senior developer is looking into various kinds of software design patterns which would allow for efficient creation and execution of commands. The senior developer also identified that it is important to be able to undo certain operations as well as maintain a log of past operations in case they need to be repeated. The following design patterns have been proposed so far and you have been asked to help choose the most appropriate design pattern.

- Façade Design Pattern
- Proxy Design Pattern
- Command Design Pattern

(a) **Which *design pattern*** do you think would best fit the *requirements*? **[02]**

(b) **Provide** reasons for your choice of design pattern. **[04]**

(c) **Discuss** the ***limitations*** that would need to be considered before implementing the design pattern. **[04]**

Total: 10

~~ Assessment continues on the next page. ~~

QUESTION 8: UML

Utopian Arts Games is working on their latest adventure game. The team is looking into designing the game in such a way that different types of **Items** can be created automatically while the game is being played. You advised the team that the **Abstract Factory Design Pattern** is a plausible approach since it allows for the efficient creation of different kinds (families) of objects in real-time. You now need to provide a UML diagram showing how the design pattern makes this possible. The classes for the different kinds of objects that need to be supported are shown below in the form of Java code.

```java
public interface AbstractEntityFactory{
  public Item produceWater();
  public Item produceRock();
  public Item produceGrass();
  public Item produceGround();
}
public class ConcreteEntityFactory implements AbstractEntityFactory{/*code ommitted*/}
public interface AbstractEntityProduct{
  public void initialise();
}
public abstract class Item implements AbstractEntityProduct{
  public abstract void animate();
}
public class Water extends Item{/*code ommitted*/}
public class Rock extends Item{/*code ommitted*/}
public class Grass extends Item{/*code ommitted*/}
public class Ground extends Item{/*code ommitted*/}
```

Provide a UML class diagram for the **Abstract Factory Design Pattern** applied to the problem stated above.

Total: 15

~~ Assessment continues on the next page. ~~

QUESTION 9: Cold code

Provide Java source code for the following problem. You can assume that all relevant packages have been imported.

The **Utopian Soccer Tournament (UST)** has finally come to an end after the finals took place in Utopia City. Now **UST** wants to provide fans with useful insights relating to the tournament. In order to do so, the data for each **Game** must be read into the analytics software. There are two types of **Games**: **GroupStage** and **Knockout**. Some of the code available is shown below:

```java
public class Game {
  private int ID;
  private String time;
  private String participants;
  private String result;
  public Game(int ID, String time, String participants, String result) {
    /*code ommitted*/
  }
  /*rest of class ommitted*/
}
public class GroupStage extends Game {
  private char group;
  public GroupStage(int ID, String time, String participants, String result, char
  ↪  group) {
    /*code ommitted*/
  }
  /*rest of class ommitted*/
}
public class Knockout extends Game {
  private boolean penalties;
  private String venue;
  public Knockout(int ID, String time, String participants, String result,
  ↪  boolean penalties, String venue) {
    /*code ommitted*/
  }
  /*rest of class ommitted*/
}
```

Create a static ***getKnockoutGames*** method. The ***getKnockoutGames*** method will read **File** handle passed as a parameter and return **ArrayList** of **Knockout** games. **Knockout** games will always have an even ***gameID***. The **UST** are trying to keep up to date with the advances in **Java** and have requested you use ***Automatic Resource Management*** in the method. The format of each type of game is shown below:

GroupStage format

```
ID,TIME,PARTICIPANTS,RESULT,GROUP
```

Knockout format

```
ID,TIME,PARTICIPANTS,RESULT,PENALTIES,VENUE
```

Total: 15

~~ THE END ~~