UNIVERSITY
OF
JOHANNESBURG

## FACULTY OF SCIENCE

### ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

| | |
|---|---|
| **MODULE** | CSC2A10<br>**COMPUTER SCIENCE 2A** |
| **CAMPUS** | AUCKLAND PARK CAMPUS (APK) |
| **EXAM** | JUNE |

| | |
|---|---|
| **DATE:** 2016-05-28 | **SESSION:** 08:30 - 10:00 |
| **LECTURER(S):** | **MR. A. MAGANLAL**<br>**MR. B. GREAVES** |
| **MODERATOR:** | **DR. DT. VAN DER HAAR** |
| **DURATION:** 120 MINUTES | **MARKS:** 100 |

Please read the following instructions carefully:

1. Answer **all** the questions.
2. Answer only in the examination books provided.
3. The use of calculators is *not* permitted.
4. Write *cleanly* and *legibly*.
5. This paper contains 10 questions.
6. This paper consists of 3 pages.

## QUESTION 1

### Java Overview

(a) **Discuss** the differences between *low-level* and *high-level* programming language categories. [04]

(b) **Name** two *design goals* of the `Java` language. [02]

(c) **Discuss** the differences between `C++` and `Java` with regards to *code domains* and *visibility modifiers*. [04]

Total: 10

---

## QUESTION 2

### Elementary Java Programming

(a) **Provide** the storage size of the following *primitive data types*: **byte**, **int**, **char**. [03]

(b) **How** can the *input* and *output* streams be referenced in a Java application? [02]

(c) **Classify** the *Quick Sort* algorithm in terms of: [05]

- Computational Complexity
- Memory usage
- Approach used
- Stability
- Method employed

Total: 10

---

## QUESTION 3

### Text Processing and Persistence

(a) **Provide** a *regular expression* that matches a list of names with body weights. [05]

For example:

```
Mark 51.2 kg
Thabo 60.5 kg
Sue 45.7 kg
```

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

(c) **Provide** the canonical (full) name of the *interface* which a class must implement in order for that class to be written to file with an **ObjectOutputStream**. [02]

Total: 10

## QUESTION 4

### Object Orientation

(a) **How** does a Java programmer prevent a method from being overridden? [02]

(b) **List** 3 *requirements* a class must conform to in order for it to be an *immutable class*. [04]

(c) Provide a **definition** of a *marker interface*. [02]

(d) **Identify** the *problem* with the following code. [02]

```java
class Animal{}
class Mammal extends Animal{}
public class Test {
  public static void main(String[] args) {
    Object instance = new Animal();
    Mammal dog = (Mammal) instance;
  }
}
```

Total: 10

---

## QUESTION 5

### Graphical User Interfaces

(a) **Name** and **describe** any two *Java GUI frameworks*. [06]

(b) **Name** and **discuss** any two *layout managers*. [04]

Total: 10

---

## QUESTION 6

### Advanced Java Programming

(a) With regards to Java multi-threaded programming:

  i. **Define** a *task*. [02]

  ii. **Define** a *thread*. [02]

  iii. **Discuss** how these two concepts are related. [01]

(b) **Name** the three *wild card formats* supported for generics in Java. [03]

(c) **Name** two **states** a Java applet can transition in to. [02]

Total: 10

---

## QUESTION 7

### Design Patterns

(a) **Name** three of the *golden rules of design patterns*. [03]

(b) **Discuss** the limitations of the *Visitor design pattern*. [04]

(c) **Name** three *structural design patterns*. [03]

Total: 10

## QUESTION 8

### UML

**Provide** a UML class diagram that illustrates the *Abstract Factory design pattern*.

Total: 10

---

## QUESTION 9

### Cold Code

**Provide** Java source code for a ***paintComponent*** method which will draw a filled cyan square of size 50 pixels at the center of the component.

Total: 10

---

## QUESTION 10

### Fill-in code

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

```java
/* Imports omitted */
public class BinaryIO {
  public void saveApp(String path, String ID, int mark, double rate) {
    File binFile = new File(__(A (1 marks))__); // Get file handle
    DataOutputStream binout = null;
    try {
      // Create required streams
      FileOutputStream fos = new FileOutputStream(binFile)
      binout = new DataOutputStream(new __(B (2 marks))__(fos));
      // Write required data
      binout.writeUTF(ID);
      binout.__(C (2 marks))__(mark);
      binout.writeDouble(rate);
      binout.__(D (1 marks))__(); // Finish writing, save changes.
    }
    catch (FileNotFoundException ex) { ex.printStackTrace(); }
    catch (__(E (2 marks))__ e) {
      e.printStackTrace();
    }
    __(F (1 marks))__ {
      if (binout != null) {
        try {
          binout.__(G (1 marks))__();
        }
        catch (/*code omitted*/) { ex.printStackTrace(); }
      }
    }
  }
}
```

Total: 10

---

*The End!*