THE AUSTRALIAN NATIONAL UNIVERSITY

First Semester 2016 — Sample Midsemester Examination

COMP6700
Introductory Programming

Writing Period: 90 minutes

Study Period: 10 minutes

Permitted Materials: Java API documentation,
one A4 sheet of paper with notes on both sides

1. Marks for each question are indicated on the exam paper.
2. All questions shall be answered electronically by creating or modifying files on the computer system as instructed. Do not move or rename files.

   If you have special considerations which prevent you from typing on computer keyboard, you will be provided with an exam paper to write your answers.
3. You may not communicate about this examination with anyone, in any way. Any detected attempt to do so will be dealt with under the rules for Misconduct in Examinations.
4. Leave all smart/mobile phones, tablets, smart watches and other electronic devices in your bag at the door.
5. You may not use the printers, CD-ROM drive or USB flash drives.
6. You must sit at the machine allocated to you.

Advice

1. You will have no internet access except for Java API documentation.

   - A full set of *Java API* documentations is available at
     `http://docs.oracle.com/javase/8/docs/api/index.html`

   A tab is made available on the toolbar of your web browser for the above URL.

2. Each question has an approximate timing estimate. These estimates are a guide only. Be careful not get stuck on particular questions which may yield a small mark. It is probably a good idea to do the easiest parts first.

3. Answers to all questions, including Java source files, shall be saved in files with prescribed names, located in subdirectories of the directory **`mid-semester-exam`** in your home directory. The subdirectories **`q1`**, **`q2`** and **`q3`** are located inside **`mid-semester-exam`**.

4. Fractional marks are possible for all questions. Just because the requirements say "must" does not mean you've failed if your program doesn't do precisely that. Do the best you can, and don't panic.

5. To type your answers in `q?.txt` files, use a *plain text editor* like *Atom, Gedit, Emacs, vi, joe* or any other you may prefer. **Do not use the LibreOffice** program as it does not save files in plain text format.

6. You can use a scrap paper if necessary; use your own or ask us to provide it.

7. Good Luck!

QUESTION 1 [35 marks]

[25–30 min]

(a) Indicate which of the following statements are True or False. (Each correct answer is worth **1.5** marks.)

1. A class name must match the name of a file which contains the class definition.
2. *Java* has four (4) different statements which define a type.
3. *Java* is a homoiconic programming language.
4. *Java* is a prototype based object-oriented programming language.
5. Modern *Java* has elements of a functional programming language.
6. One class cannot contain multiple methods with an identical name.
7. One cannot define more than one external (not inner) Java class in one source file.
8. Objects of `java.lang.String` class are immutable.

Place your answers in section `q1.a` of the file `q1.txt` in the subdirectory **`q1`**.

[12 marks]

(b) Four aspects of object-oriented programming can be described as the acronym "A PIE" — *abstraction, polymorphism, inheritance* and *encapsulation*. Using your own words, explain the meaning of each of them. Every correct answer is worth **2.5** points.

Place your answers in section `q1.b` of the file `q1.txt` in the subdirectory **`q1`**. The order and numbering of the problems in your answer can be arbitrary.

[10 marks]

(c) *Reserved words* are names which cannot be used as identifiers. In the list of words below, select those which are ***not***, Java's reserved words:

```
do,  abstract,  Integer,  long,  case,  main,  println,  until,
                end,  native,   default,   args
```

(You will get **13** marks if you remove *all* non-reserved words, **8** points if you retain one non-reserved word, **3** — if you retain two non-reserved words, **0** if you retain three or more non-reserved words.)

Place your answers in section `q1.c` of the file `q1.txt` in the subdirectory **`q1`**.

[13 marks]

QUESTION 2 [35 marks]

[30 min]

Coming up with a password that meets specific requirements is something a computer program can quite possibly do. Writing such program will give you a chance to use random number generators together with a list of known values.

Create a program that generates a secure password. Prompt the user for the minimum length, the number of special characters, and the number of numbers. Then generate a password for the user using those inputs.

An example of running this program would be:

```
What’s the minimum length? 8
How many special characters? 2
How many numbers? 2
Your password is: aurn2$1s#
```

Hint One You can assume that the following set of special characters can use used in generated passwords:

```
! " # $ % & ’ ( ) * + , - . / : ; < = > ? @ [ \ ] ˆ _ ‘ { | } ˜
```

A simplest way to provide special characters to the program is to incorporate them into a (`private final`) string and a `char`-array which consists of all the above characters.

Hint Two The random selection requirement can be met by employing the class `java.util.Random`, and its `nextInt(int)` method. You will need to read the *Random* class API documentation in case if this class is somewhat of a novelty for you.

[35 marks]

Important: Your answers to Question 2 must be saved in a Java source file `PasswordGenerator.java` in the subdirectory **`q2`**. If you want to add explanations (in case if your program is not complete, has compilation and/or correctness problems), include them in the file `q2.txt` which is already present in the same subdirectory.

QUESTION 3 [30 marks]

(a) [30 min]

This question contains eight parts each requiring a simple answer True or False; each is worth **1.5** point.

- **(a)** An abstract class can extend a concrete (non-abstract) class.
- **(b)** An interface in *Java* can extend multiple other interfaces.
- **(c)** An enum type can extend a class.
- **(d)** An instance method can be overridden in a subclass, but a static method can be hidden in a subclass.
- **(e)** A method access modifier can be changed to more restrictive after overriding.
- **(f)** Private members on an object can be accessed by methods of another object if the two objects have "Is-A" relationship.
- **(g)** When overriding a method whose return value has the type of the ambient class, one can change the return type to match the subclass type.
- **(h)** A class constructor never executes its parent class constructor.

Place your answers in section `q3.a` of the file `q3.txt` in the subdirectory **`q3`**.

[12 marks]

(b) In your own words, explain why you think fully implemented (so called `default`) methods were allowed in interfaces in the latest enhancement of Java (SE 8). How does this affect the *Java* principle of disallowing multiple inheritance by implementation?

Place your answers in section `q3.b` of the file `q3.txt` in the subdirectory **`q3`**.

[8 marks]

(c) This is a harder question.

In a Hollywood film script, a hero called *Citizen* is defined by a class as follows:

Citizen.java

```java
public class Citizen {

   protected String name;

   public Citizen(String s) { name = s; }

   protected String getName() { return name; }

   public void report() { System.out.println("I am " + name); }
}
```

In the Hollywood plot, after doing something dangerous, a citizen must change his/her identity and become *Undercover*, which is defined like this:

Undercover.java

```
public class Undercover extends Citizen {

   public String name;

   public Undercover(String alias, Citizen c) {
      super(c.name);
      this.name = alias;
   }

   public static void main(String[] args) {
      Citizen oldme = new Citizen("Jack Average");
      Citizen newme = new Undercover("John Doe", oldme);
      newme.report();
   }
}
```

When run, the program prints the following on the standard output:

```
I am Jack Average
```

— his original name, and his cover is blown.

To earn the following marks,

(4 marks) Explain why this is happening

(6 marks) Modify the code (but not the `main` method!) to make it print the alias name "John Doe". An elegant solution will get the full marks, (and workable one will get you 4 out of 6).

Provide your answers by modifying the source files `Citizen.java` and `Undercover.java` in the subdirectory **`q3`**. You can provide additional considerations in the section `q3.c` of the file `q3.txt` in the subdirectory **`q3`**.

[10 marks]