Australian National University

2017 Semester 1 - Final Examination

Software Construction

(COMP6442/COMP2100)

Writing Period: 3 hour duration
Study Period: 15 minutes duration
Permitted Materials: One A4 page with notes on both sides.
Note also the standard lab tools are available including: Java, eclipse, gedit, vim, emacs, git, umbrello, dia, gcc, man, the calculator on the computer, ...
The Java API is available at file:///usr/share/doc/openjdk-7-jre-headless/api/index.html

NO calculator permitted (physical electronic device).

Please Read The Following Instructions Carefully.

This exam will be marked out of 100 and consists of 4 questions. Questions are of unequal value. The value of each question is shown in square brackets. Questions that are partitioned into parts show the number of marks given to each part within square brackets.

Students should attempt all questions. Answers must be saved into the question's directory (Q1p1, Q1p2, Q2, Q3, Q4) using the file(s) described in the question statement. Marks may be lost for giving information that is irrelevant.

Network traffic may be monitored for inappropriate communications between students, or attempts to gain access to the Internet.

The marking scheme will put a high value on clarity so, as a general guide, it is better to give fewer answers in a clear manner than to outline a greater number of less clear answers.

Question 1

Part I – Multiple Choice [30 marks]

The multiple choice questions are available via a program in the 'Q1p1' directory. This program is also used for entering your answers. To run the multiple choice program either start it by *double clicking the icon on the desktop* or by starting it in a terminal. This is done by:

- opening a Terminal,
- from the command line change directory (cd) into the Q1p1 directory, and
- run the java MultipleChoice program.

```
% cd ~/Q1p1
% java -jar MultiChoice.jar
```

Your answers are automatically saved every time you press a button. Hence once you have completed your answers you can simply exit from the program (either by pressing the window closing “x” or by selecting the “exit” menu item). Note at any time you can restart the program and change your answers. **Do not edit files in this directory.** Also you can only have one instance of the MultiChoice program running at any time.

There are 15 questions. Each answer you get correct **gains you 2 marks**, each incorrect answer **loses 0.5 marks**. Questions left unanswered neither lose nor gain marks.

If the question statement contains an underline, that is “_____”, then the selected answer should correctly complete the question statement. Some question statements may contain two underlined sections, in which case the answer has two parts separated by a comma. The first part corresponds to the first underlined section and the second part corresponds to the second underlined section.

Part II - Short Answer [10 marks]

Highest marks are gained by providing clear, concise, and short answers. Save your answers in the text file 'Q1p2answers.txt' in the directory Q1p2 (this file is already created with places to add your answers, so edit the file 'Q1p2answers.txt' and save your answers directly into it). Use your favourite editor e.g. vim, gedit etc. Please make sure that this file is saved both as you progress through the exam and before the exam ends.

i. [4 marks] Consider the statement “Both **Java** and **bash** are complete powerful programming languages”. What facts help back up this statement? What are some limitations of **bash** as a general purpose programming language?

ii. [4 marks] Explain the “observer” design pattern. Provide an example that illustrates the use of the observer design pattern.

iii. [3 marks] What is a non-deterministic finite state machine? Explain the relationship between non-deterministic finite state machines and deterministic finite state machines.

Question 2 [30 marks]

The aim of this question is to complete a mini software development project. This involves:

i) Understanding the requirements for a simple application,

ii) Creating a software design in UML,

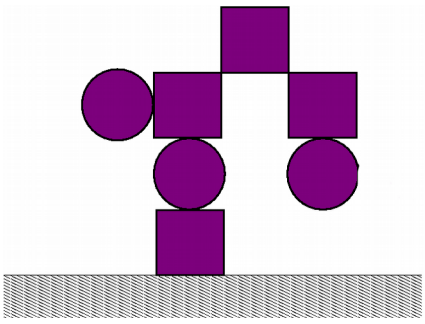
iii) Implementing the application using Java,

iv) Demonstrating the use of some software engineering tools such as git, Makefiles

v) Writing JUnit test cases and testing the implementation,

vi) Writing appropriate documentation. Your code and answers must all be included in the Q2 directory.

Suppose you are given the task of writing a program that helps an artist design a sculpture. The sculpture is made out of flat steel plates. These plates are either square or circular. The square plates have sides that are 1 meter in length. The circular plates have a diameter of 1 meter. The sculpture will be attached to a wall, hence it will be flat and can be modeled as a 2-dimensional object. The centers of the steel plates must be placed on a grid that is 1 meter by 1 meter. Your program will need to be able to store the design of a particular sculpture. Also the artist wishes to be able work out the total amount of steel that is used, this requires a method *"area"* which returns the total area of the current design. Also the artist wishes to be able to determine if the sculpture *"hasBalance"*. The sculpture *"hasBalance"* if and only if all the circular plates are touching at least one square plate. You are not required to either parse a sculpture or provide a GUI for displaying or designing sculptures. The following is an example of a sculpture that *"hasBalance"*:



In answering this question you may constrain the problem making some reasonable assumptions about the problem. Please state any assumption you make in answering this question in the DesignSummary.txt file.

Your tasks are:

a) [10 marks] Write a basic working version of the sculpture program using Java. Put you code in the Q2 directory.

b) [3 marks] Creating a software design for the program in UML. Export the diagram to a file called UML.png in the Q2 directory. Provide a short summary

explaining your design in the file named DesignSummary.txt provided in the Q2 directory. You can also state assumptions you make regarding the problem in this DesignSummary.txt file.

c) [2 marks] Setup and use git within the Q2 directory for revision control and make at least 3 commits.

d) [3 marks] Write JUnit unit tests to unit test the methods “hasBalance” and “area”

e) [2 marks] Write a Makefile for building the program with 2 targets: make, and make clean.

f) [10 marks] Overall quality of submission. This includes aspects such as properly formatted code and UML diagram, enough detailed comments within code, appropriately named variables/fields/methods/classes, meaningful comments for git commits and use of appropriate design.

Hints:

- keep the design simple,
- Check that your code/makefile works from the command line within the Q2 directory.

Question 3 [20 marks]

A simple functional programing language has been created. The representation and execution of programs within this language has been implemented. What is yet to be implemented is the parser that uses a simple tokenizer to take the source code and turn it into Java objects that represents the programs. Your task for this exam question is to implement the parser.

The grammar for this language is:

```
<exp> ::= 0 | inc ( <exp> ) | dec ( <exp> ) | <variable> |
          <function name> ( <exps> ) | ( <exp> ? <exp> : <exp> )
<exps> ::= <exp> | <exp> , <exps>
<function> ::= <function name> ( <vars> ) = <exp>
<functions> ::= <function> | <function> <functions>
<vars> ::= <variable> | <variable> , <vars>
```

The Q3 directory contains the code for this project. Put your code for completing the parser in the Exp.java file in the Q3 directory. You only need to add code to complete the methods *parseExp, parseExps, parseFunction, parseFunctions*, and *parseVars*.

Question 4 [10 marks]

An AVL binary search tree for representing a set of integers has been implemented. Originally a simple binary search tree approach was used, however, its performance was poor as there would often be a sequence of ascending integers added to the set making the tree unbalanced, hence the developers thought an AVL approach would work better. Their implementation has a significant problem which meant that the original simple binary search tree approach worked better then their AVL implementation. The code for the AVL implementation is given in the Q4 directory, your task is to investigate and solve this problem with their implementation.

[3 marks] What is this significant problem? Describe and analyze the problem. Put your answer in a comment in "NETree.java" in the Q4 directory.

[7 marks] Modify the implementation fixing this problem. (Modify the files in the Q4 directory)