MURDOCH UNIVERSITY

ICT167 Principles of Computer Science

TJA, 2019

Assignment 2 (worth 20% for the unit)

Due Date: midnight 7 April (Sunday)

All Students: Submit the Assignment via LMS by the due date.

Late penalty: 10% per day penalty for delayed submissions unless prior extension of deadline is obtained from the unit coordinator.

You should keep a copy of your work. Your submission must include a completed assignment cover sheet. An electronic copy of the assignment cover sheet is available at the unit LMS site.

Note: **The question requires you to use an array (NOT ArrayList) and NOT any Java library class to store the data.**

Read and understand the information at http://our.murdoch.edu.au/Educational-technologies/What-you-need-to-know/

Question

First, you need to design, code in Java, test and document a **base** class, Student. The Student class will have the following information:

A. Title of the student (eg Mr, Miss, Ms, Mrs etc)

B. A first name (given name)

C. A last name (family name/surname)

D. Student number (ID) – an integer number (of type **long**)

E. A date of birth (in day/month/year format – three ints) - **(Do NOT use the Date class from JAVA)**

The student class will have **at least** the following constructors and methods:

(i) two constructors - one without any parameters (the **default constructor**), and one with parameters to give initial values to all the instance variables.

(ii) a reasonable number of set and get methods.

(iii) methods to compute the final overall mark and the final grade (which will be overridden in the respective child classes). These two methods will be `void` methods that set the appropriate instance variables. Remember one method can call another method. If you prefer, you can define a single method that sets both the overall mark and the final string grade, but if you do this, use a helper method.

(iv) an equals method which compares two student objects and returns true if they have the same student names, the same date of birth and the same student number, otherwise it returns false.

(v) method to sort the array into ascending order of students' numbers (IDs)

You may add other methods in the student class as you see appropriate.

Design, code in Java, test and document (at least) three classes – an UndergraduateStudent class and PostGraduateStudent class (which both derive from the Student class) and a client program.

For undergraduate students:

(a) There are three assignments, each marked out of a maximum of 100 marks and equally weighted. The marks for each assignment are recorded separately.

(b) There is weekly practical work. The marks for this component are recorded as a total mark obtained (out of a maximum of 10 marks) for all practical work demonstrated during the semester.

(c) There is one final examination that is marked out of a maximum of 100 marks and recorded separately.

(d) An overall mark (to be calculated within the program)

(e) A final grade, which is a string (to be calculated within the program)

The final grade, for undergraduate students, is to be awarded on the basis of an overall mark, which is a number in the range 0 to 100 and is obtained by calculating the weighted average of the student's performance in the assessment components. The criteria for calculating the weighted average is as defined below:

> The three assignments together count for a total of 45% (15% each) of the final grade, the practical work is worth 10%, and the final exam is worth 45% of the final grade.

For postgraduate students:

(a) There is one group assignment.

(b) There is weekly tutorial work. The marks for this component are recorded as a total mark obtained (out of a maximum of 10 marks) for all tutorial work demonstrated during the semester.

(c) There is one final examination that is marked out of a maximum of 100 marks and recorded separately.

(d) An overall mark (to be calculated within the program)

(e) A final grade, which is a string (to be calculated within the program)

The final grade, for postgraduate students, is to be awarded on the basis of an overall mark, which is a number in the range 0 to 100 and is obtained by calculating the weighted average of the student's performance in the assessment components. The criteria for calculating the weighted average is as defined below:

> The group assignment is worth 60% of the final grade, the tutorial work is worth 10%, and the final exam is worth 30% of the final grade

A grade is to be awarded for undergraduate and postgraduate students as follows: An overall mark of 80 or higher is an HD, an overall mark of 70 or higher (but less than 80) is a D, an overall mark of 60 or higher (but less than 70) is a C, an overall mark of 50 or higher (but less than 60) is a P, and an overall mark below 50 is an N.

The client program will allow entry of these data for several different student into an **array** and then perform some analysis and queries. In this assignment, your client program is only dealing with undergraduate students only.

Your client class (program) will provide the user with a menu to perform the following operations:

1. Quit (exit the program)

2. Add (to the array) all information about an undergraduate student (except the overall mark and the grade) by reading it from the keyboard or from a file and determine the undergraduate student's overall mark and grade.
3. Output from the array the details (all information including the overall mark and the grade) of all undergraduate students currently held in the array
4. Compute and output the average overall mark for undergraduate students only
5. Determine and display how many undergraduate students obtained an overall mark equal to or above the average overall mark and how many obtained an overall mark below the average overall mark
6. Given an undergraduate student number (ID), view all details of the undergraduate student with that number. If the undergraduate student is not found in the array, an appropriate error message is to be displayed
7. Given an undergraduate student's name (both surname and given name – ignoring case), view all details of that undergraduate student. If the undergraduate student is not found in the array, an appropriate error message is to be displayed
8. Sort the array of undergraduate student objects into ascending order of the undergraduate students' numbers (IDs), and output the sorted array
9. Sort the array of undergraduate student objects into ascending (alphabetical/dictionary) order of undergraduate students' surnames using a different sorting algorithm to the one used in (8) above, and output the sorted array.

Note that the program will loop around until the user selects the first option (Quit).

Set up a student array of N student objects (undergraduate students only), and test it with N = 6 (at least). You have to provide the test data with your program. You can store your test data in a file or hard coded in your program.

The client class should be well-structured and should have a reasonable number of methods in addition to the main method. Devise suitable test data to test all sections of program code.

There is no requirement for the program to store data when the program quits. The interaction with the user can be via the command line (i.e., no graphical user interface is expected).

Devise suitable test data to test all sections of program code. You will need to provide all the test data used.

Your program should also include a method (e.g., StudentInfo( )) to output your student details (name, student number, mode of enrolment, tutor name, tutorial attendance day and time) at the start of program results.

Note: The question requires you to use an array (not ArrayList) and not any Java library class to store students' details. Also, the sorting algorithms used must be coded within your program and not called from any Java libraries.

Distribution of marks for assessment

An approximate distribution of marks for assessment is given below. The question will be marked out of 100 as follows:

Correct solution design and implementation: 75 marks
(which includes the design and implementation of classes as specified in the question above; programming style, use of comments, use of methods, parameters, input validation, readability, presentation of output etc.)

External Documentation (problem specification, algorithm, program limitations, program testing and test results, program listings, etc.): 25 marks

Required internal documentation (i.e. in the source code):

- a beginning comment clearly stating title, author, date, file name, purpose and any assumptions or conditions on the form of input and expected output;
- other comments giving useful low-level documentation and describing each component;
- well-formatted readable code with meaningful identifier names and blank lines between components (like methods and classes).

Required External Documentation:

1. **Title:** a paragraph clearly stating title, author, date, file names, and one-line statement of purpose.
2. **Requirements/Specification**: a paragraph giving a brief account of what the program is supposed to do. State any assumptions or conditions on the form of input and expected output.
3. **User Guide**: include clear instructions on how to access, compile and run the program.
4. **Structure/Design/Algorithm**: Outline the design of your program. Give a written description, use diagrams (eg, **UML**) and use pseudocode for algorithms.
5. **Limitations:** Describe program shortfalls (if any), eg, the features asked for but not implemented, the situations it can not handle etc.
6. **Testing**: describe your testing strategy (the more systematic, the better) and any errors noticed. Provide a copy of your results of testing in a document saved in Word format. **Note that a copy of the sample test data and results from a program run is required (copy from the program output window and paste to a Word file).** Your submitted test results should demonstrate a thorough testing of the program.
7. **Source program listings**: save all your Java source code in a document in MS Word format.

All of the above external documentation must be included in that order in a file saved in MS Word format.

It is also necessary to submit all Java source code of your programs (i.e., all classes that you have designed and implemented). You should develop the programs using the NetBeans IDE. It will make it easy to collect sample output. The whole NetBeans project should be submitted.

The external documentation together with the NetBeans project folder containing all classes for the program must be compressed in a .zip file before submitting. Make sure that all necessary files are submitted so that the programs can be viewed, compiled and run successfully.

The final version of the program should compile and run under Java SE 8 and NetBeans IDE 8.0 (or later version).

Associate Professor Dr Kevin Wong
ICT167 Unit Coordinator