Practical Skills Assessment 2 (60% of module mark)

Title: The Enrolment Register

Due: Monday 19th April 2021 at 3:00pm
Feedback provided on or before: Monday 17th May 2021 (4 weeks)

For this assignment you are normally expected to work in pairs.

As a pair, when submitting your assignment you are agreeing to the following statement:

I declare that this is all my own work and does not contain unreferenced material copied from any other source. I have read the University's policy on plagiarism and understand the definition of plagiarism. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions in accordance with the policies and procedures of the University. A mark of zero may be awarded and the reason for that mark will be recorded on my file.

The University policy on **plagiarism** is available at
http://www.ulster.ac.uk/academicservices/student/plagiarism.pdf

Assessment criteria:

- Structured, commented and readable code (20%)
- Program executes according to specification (20%)
- Appropriate use of variables (20%)
- Implementation of suitable program control structures, classes and methods to solve problem (20%)
- Structured approach to testing (20%)

Learning Outcomes:

- Illustrate a knowledge of the programming constructs and data types of the selected language
- Analyze a problem using an object-oriented approach
- Design and develop code in a professional manner that facilitates readability and maintainability
- Solve problems involving a logical component

Overview

This assignment is concerned with the use of arrays to store and manage objects. Write a Java program to store and manage a set of students enrolled on a course according to the following requirements:

Functional Requirements

1. The application needs only to involve one course. There is a need to record details about each student enrolled on the course.
2. The application must allow for up to a maximum of 20 students to be enrolled on the course.
3. For each student, their name, date of birth (dd/mm/yyyy), address and gender should be stored. [*Hint: consider creating an array of type Student, where Student is a class you create containing the name, date of birth, address and gender as instance variables. You **do not** need to perform any validation on the entered name, DOB or address, however you do need to check that the gender has been entered correctly in whatever format you specify. You might also want to inform the user that if any of the data fields for a given student are left blank, that student's full details will NOT be saved beyond the current session*].
4. The program needs to enable students to be **added** to the course or **deleted** from the course. There is **no** requirement to have any facility to modify student details after they have been created. [*Hint: deleting data from an array may leave a blank space in the array, so keep a count of the total number of students and search for the next free array cell when adding another student*].
5. It must be possible to report (i.e. print to the screen) on the course. This report should indicate the course name, lecturer involved, the total number of students enrolled and the percentages of these students who are male and female.
6. At the end of a session, when the program is being terminated, the course details along with the details for each enrolled student should be written to disk. These details should be recorded in two standard text files: one containing course details (called "CourseDetails.txt") and the other holding all the student records, to be called "StudentDetails.txt". [*Hint: don't save blank records to disk, check that there is a valid "student" object in each array cell before writing its contents to disk. If any of the fields within a student's details have been left blank (string length of zero) then that student is invalid and their details should **not** be written to disk*].
7. When the program is started, it should read from these standard text files to re-populate the application with any previously stored data as a starting point. [*Hint: each record requires a new instance of the Student class to be created and linked to the next available free cell in your array*].
8. The program user must also be able to search for a student by name, causing the sought student's details to be displayed if present. Note that you can assume each student's name will be unique.

9. The program must employ a **console** interface only.

You must develop a set of test cases and record the results of applying these tests to your software.

Other assignment requirements:

The adoption of object-oriented principles should be evident in your implementation of the above requirements.

A structured approach to testing should be evidenced by submission of a test plan and outcomes in accordance with the above requirements.

Submission

1. The assignment is to be submitted electronically to Blackboard Learn on or before **Monday 19th April 2021 at 3:00pm** (Week 13). The work should be submitted as a zip file containing the NetBeans project folder and all associated sub-folders for all source code (.java), compiled files (.class) and IDE project files, plus a Word document file containing the test plan and results. If the work is completed in pairs, then only **ONE** copy of the work should be submitted, with **both names and student IDs clearly marked on all the submitted work**.
2. If the work has been completed in pairs, then a team declaration form (available with this assignment specification on Blackboard) must be completed to indicate each team member's agreed percentage contribution to the submitted solution. This completed form should be included in the top-level folder of the NetBeans project, so that it gets zipped with the other files and uploaded to blackboard at the same time. Note that the marks awarded will reflect the percentage contribution declared.
3. A pdf file describing the approach to testing and test cases used should be included with the NetBeans zipped folder in the submission.
4. In addition to submitting the full NetBeans project, you MUST also copy the Java source code for all your classes into a separate pdf file – this is for use by moderators. Include this .pdf in your submission as a separate file.
5. During the laboratory classes in week 13, 14 and 15 (if necessary), at least one person in the pair-submission will be required to give a brief demonstration of their working software to one of the demonstrators. This demonstration will be used to mark the "program executes according to specification" marking criterion.
6. Please ensure that you keep a secure electronic backup copy of your assignment. In the event that your submission cannot be opened or read, you may be asked to provide another copy.

***** IMPORTANT *** Marks Weighting for Pair Contributions:**

This assignment should normally be completed in pairs. If done in pairs, then **each** of the pair must sign the 'team declaration' form (available from the 'Administration' folder on Blackboard), to indicate what percentage contribution they made to the overall effort. Where both in the team contribute equally in overall effort to the team solution, then there is **no**

penalty for working in a team of two as opposed to working individually. Each person will get the full mark awarded to the submitted work.

For any pair submission deviating from an equal division of labour, marks for this assignment will be scaled down according to the declarations made. The scaling will happen according to the following example of the scheme: Consider a pair-submitted piece of work that is deemed worthy of a mark of 50%. Consider also that one of the pair has declared their contribution to be 60% and the other declared at 40% (i.e. 10/50 or 20% less effort than an equitable contribution). The person whose contribution is the lesser will have their mark reduced by 10/50 or 20%, i.e. to 40% in this example. The person whose contribution was set at 60% will receive the full awarded mark of 50%.

In the case of pair-submissions, only ONE copy of the solution is required with submitted source code files and test plans clearly indicating the names and student IDs of both team members.