Java Coursework (20 marks)

Student score management program design

In the following time you will be required to write an educational management system in JAVA to your individual ability. This task will count for 20% of the final grade for this course and a higher code check rate will be considered as the task not being completed.

Design requirements:

The system may include Person, Student, Teacher, Course and Score classes. The properties and functions of the classes may be designed as required and reflect the relationships of the classes, such as inheritance, association, etc., in order to achieve the functionality of student grade management. There are at least 5 students and 1 teacher per course, and each student takes at least 3 courses.

Functional requirements:

Write menu options that allow the user to select the following functional requirements:

1) Query information related to a course, such as number of students, teacher's name, list of all students' names, etc.;
2) Query the marks of a student for all courses;
3) Query the marks of all students in a course and calculating the average mark for that course;
4) Query the ranking of marks in a course
5) Query the percentage of all students in a course in different mark bands: e.g., Excellent (90-100), Good (80-89), Medium (70-79), Pass (60-69), Failed students (0-59);
6) Modify a student's mark in a course through the keyboard;
7) Add or remove students from a course;
8) Log in or out of the system.

You can also implement other functions as required, which will give you extra marks depending on the level of difficulty (no more than 20), such as implementing menu effects in a GUI, importing or deleting students, teachers, courses with one click, etc. It is important that the various types of programs are interlinked, more loopholes or BUG are still not allowed.

Submission requirements:

1) Program design document: This includes an analysis of the functional requirements of the system, a diagram of the UML classes, whether each function of the program is implemented properly, and whether BUG occurred during the design process and how they were resolved (or not resolved yet). The document is submitted in the form of a WORD document, which can be interspersed with code content as appropriate, and requires clear and informative typography to ensure that the functional requirements have been correctly analyzed and that the specific implementation of each function has been successfully presented.

2) Source code: Original code that implements the functions mentioned above, with certain comments to be kept between the code.

Marking criteria:

[Excellent: 16-20]. 1) The code will achieve the above-mentioned functions perfectly and run without bugs. there are additional functions implemented. 2) The code is well designed, well structured, with clear and necessary comments. 3) Standardization of programming files and reasonable code structure. 4) Programming documentation meets the requirements and reflects the full results of one's work and is beautifully laid out and formatted.
[Good: 10-15]. 1) The above functions will be achieved basically, and there are no bugs or a small number of acceptable bugs after running. 2) The code is well designed and structured, with a small number of comments retained. 3) Standardization of programming documentation and reasonable code structure. 4) Programming documentation reflects some of the individual's work.
[Poor: 0-9] 1) The above functions are not fully implemented and bugs appear after running. 2) The code design is unreasonable, basically without logic, or is identified as plagiarism. 3) The programming documentation is not standardized and the code structure is unreasonable. 4) Missing programming documentation, or not reflecting personal work product.