|  |  |  |
| --- | --- | --- |
| COMP7013 OOP | | |
|  |  |  |
| Final Project 40% | | |

Submission details  
This project is due on the **28th of April 2023** at 23:59. Late submissions after that date will incur penalties

* 10% deduction for submissions less than 1 week late
* 20% deduction for submissions less than 2 weeks late
* No submissions will be accepted after the 12th May

Submissions must be uploaded to Canvas via the associated Canvas Assignment. Please create **a single ZIP file** containing

* All code, database, and other files that you developed
* All compiled binaries and the JavaDoc generated documentation
* A report on the Unit testing that you carried out as well as on the evaluation of the memory leak (see below) in a Word or PDF file
* A screen recording showing the GUI functionality of your application

My advice to you is to utilize the code that you created in your Practical Assessments as a starting point for your Project. Please ensure that the code you submit is clean and well documented. If classes, methods, or attributes have become obsolete in your updated project then they should be removed.

Project outline – student records GUI  
I want you to develop an application that stores student records at MTU; the records consist of information about each student and information about the modules they have completed including the grades they achieved. This will allow the university to stay informed about student performance.

All persistent data should be stored in an JDBC compliant Database (MySQL, SQLite (<https://github.com/xerial/sqlite-jdbc>)) (not an ArrayList), and you should use JavaFX to develop a desktop application with a GUI (do not use Scene Builder).

Project requirements  
The desktop GUI should support the following basic functions:

1. Add/remove students to/from the database (each student should have at least name, ID, date of birth, and their current semester as attributes)
2. Add/remove modules to/from the database (each module should at least have name, code, and semester as attributes)
3. Enter/modify/delete student grades (each student should have grades for all modules completed until their current semester; grades can be a percentage, or NP if the module was not completed)
4. Search for students in the database to display their information and the modules they passed (>=40%).

In addition to the basic functionality:

1. Implement an extra Button in your GUI to simulate a memory leak by creating new Student objects (in Heap space) in an infinite loop until the application runs out of memory. Note how long it took and the memory at the point of exception. Set the VM size to half of normal then use the same button and observe what happens/how long it takes to get the out of memory exception. Attach a Word or PDF document which shows your analysis of this memory leak and explain in your own words what is happening.

Your solution should:

* Apply OOP principles and implement a clear package structure
* Implement appropriate design patterns, in particular MVC, where applicable
* Use the JDBC API to access the SQLite database
* Include JavaDoc documentation and Unit test documentation for all classes

Also make sure that:

* appropriate objects are used in your code (not just String) to represent attributes (e.g. dates) and validate user inputs accordingly
* the database is always kept updated and synchronised with the desktop application
* queries to access the underlying database do not retrieve any unnecessary information (e.g. modules not passed for the student search)
* when deleting any record from that database, that all associated information is removed as well

Evaluation criteria

|  |  |
| --- | --- |
| Video included in submission | 1 |
| Overall code quality | 2 |
| Project and package structure | 2 |
| Appropriate GUI layout | 1 |
| OOP principles | 1 |
| Use of design patterns | 2 |
| Student class implementation | 1 |
| Module class implementation | 1 |
| Overall student/module/grade integration | 1 |
| SQL setup and JDBC connection | 2 |
| Appropriate DB schema | 2 |
| Add student to DB | 2 |
| Remove student from DB | 3 |
| Add module to DB | 2 |
| Remove module from DB | 3 |
| Manage grades in DB | 2 |
| Search students in DB | 3 |
| Unit tests | 2 |
| Javadoc | 2 |
| Memory leak implementation | 2 |
| Memory leak evaluation | 3 |