Leeds Trinity University

Further Software

Development

Reflective Report

**A logo of a university

Description automatically generated**

Student: Joshua White (2314105)

Course: Further Software Development COM 5003

Module Leader: Dr Antesar Shabut

Submission Date: Friday 13th December 2024

### 

### **GitHub:** [**https://github.com/2314105/COM-5003**](https://github.com/2314105/COM-5003)

### **Degree Classification Calculator: Reflective Report**

#### **Introduction**

The Grade Calculator is designed to allow students to input either custom credits or their module codes, automatically populating the credits field. Students can then add their marks for each module and calculate their predicted grade based on four different methods, ultimately determining the best classification. The program heavily focuses on object-oriented principles (OOP) to ensure the calculator is as modular as possible for long-term resilience. This report reflects on the design, implementation, and future prospects of the project.

### **Design and Implementation**

#### **Object-Oriented Design Principles**

The project underwent multiple iterations, refining the same version through code review, refactoring, and testing. Changes were evaluated for their impact on the overall codebase, and issues were resolved when testing. Core classes like Module, AverageCalculator, Validator, and DirectLevel6Calculator encapsulated functionality, breaking down the program into smaller, manageable units. OOP principles like inheritance and polymorphism were implemented, enabling shared functionality across different classes, such as the AverageCalculator and MarkClassifier.

#### **Class Design**

A Unified Modeling Language (UML) diagram has been generated and added to the project resources to illustrate the relationships and dependencies among the classes. Key classes and their responsibilities are as follows:

* **Module**: Encapsulates module details, including the module code, allocated credits, and marks.
* **Validator and ValidationManager**: Perform data validation tasks, such as ensuring pass marks, verifying marks are categorical, and confirming total credits meet the required threshold.
* **DirectLevel6Calculator and Level5Level6Calculator**: Handle classification and calculation processes using advanced algorithms for weighted averages and profile classifications.
* **DatabaseManager**: Creates a database connections and retrieves module-specific data. The design is expandable to include student login functionalities in future iterations.
* **UI Components**: Provide an intuitive and user-friendly graphical interface, including navigation panels, input fields, and result displays.

These classes were designed with clear separations of responsibility, adhering to the Single Responsibility Principle (SRP).

### **Data Structures and Algorithms**

#### **Data Structures**

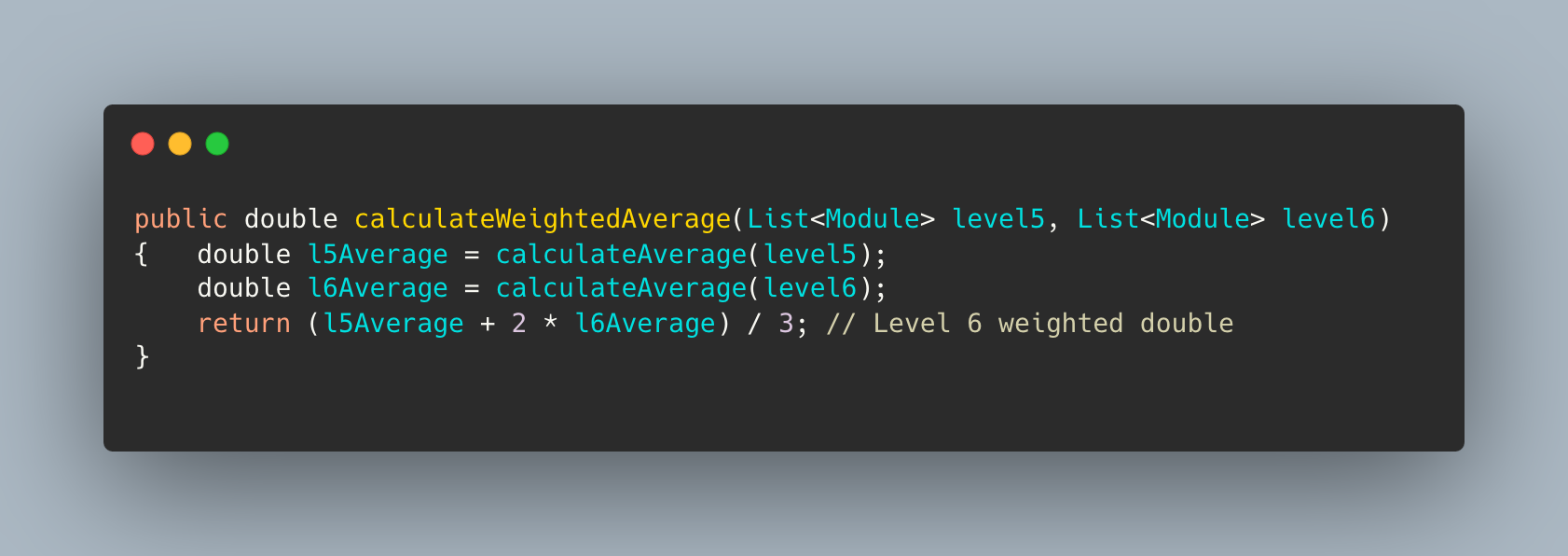
The project heavily relies on the following data structures:

* **List<Module>**: Used throughout the project as the core data structure for storing and processing module-related information. This consistent approach simplifies calculations and allows for future extensions.
* **Streams and Maps**: Introduced/learned during the development process to simplify filtering, mapping, and processing data. For instance, Java Streams were used to filter modules based on grade ranges, while Maps were utilized for mapping navigation buttons to their respective actions.

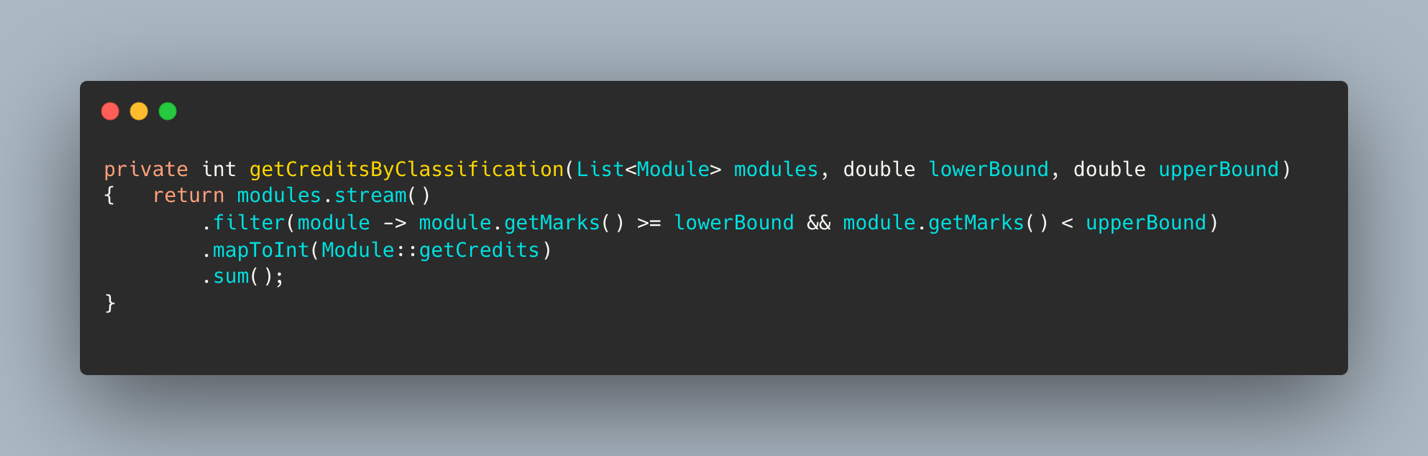
#### **Algorithms**

The application uses algorithms tailored to the problem:

* **Weighted Average Calculation**: Combines Level 5 and Level 6 module marks with predefined weightings to compute overall averages.



* **Profile Mark Classification**: Determines the student's classification based on the distribution of credits across grade thresholds. Sorting and minimum selection algorithms identify the optimal classification.



### **Robustness, Usability, and Efficiency**

#### **Robustness**

Robustness was achieved through reusable validation methods, ensuring data integrity before calculations were performed. Comprehensive error handling in the DatabaseManager class provided meaningful feedback when database connections failed, enhancing reliability.

#### **Usability**

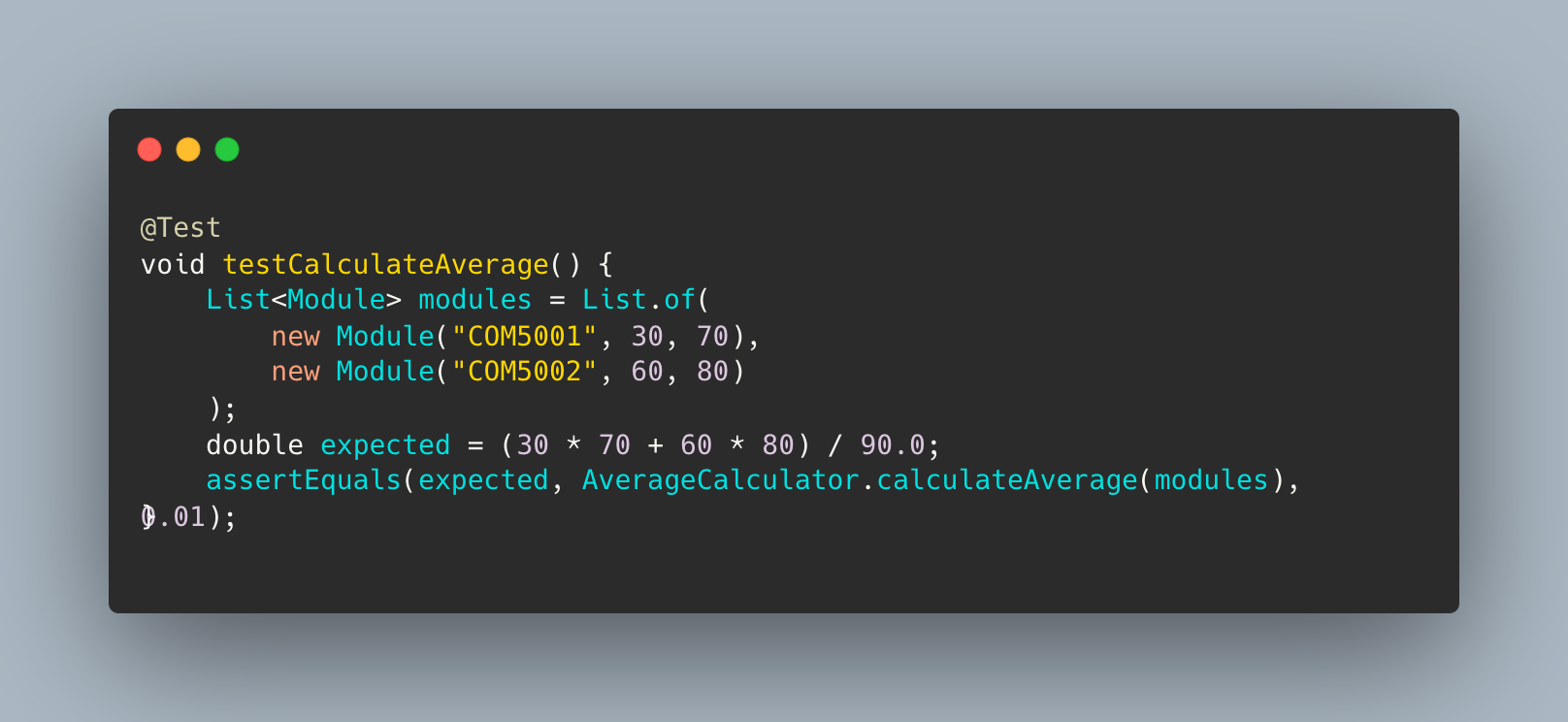
The graphical user interface (GUI) was designed to be simple and intuitive. Buttons and input fields are clearly labelled, and module codes automatically populate the credits field if they exist in the database. This feature significantly reduces user effort and enhances the overall user experience.

#### **Efficiency**

Optimised algorithms ensure scalability for larger datasets, making the application suitable for real-world usage. Prepared SQL statements in database queries reduce processing overhead and protect against injection attacks.

### **Test Driven Development (TDD)**

Test-driven development (TDD) was followed throughout the project. Unit tests were written for key functionalities, such as average calculations, validations, and classification algorithms. Testing ensured that the codebase remained robust and functional during refactoring and iterative development.



### **Reflection and Future Resilience**

#### **Strengths**

**Modularity**: Classes have well-defined responsibilities, making the code easy to test, debug, and extend.

**Reusability**: Core components like validation methods and data structures are highly reusable across the application.

**Extensibility**: Placeholder buttons and modular design allow new features, such as additional calculations, to be integrated seamlessly.

#### **Limitations and Future Improvements**

**Database Dependency**: The current reliance on a local MySQL database limits accessibility. Hosting the database in a cloud environment would address scalability issues.

**Enhanced Error Handling**: Providing more detailed logs would improve debugging and error resolution.

**Expanded Features**: Adding the remaining calculations from the original calculator would make the application fully functional for a broader audience.

### **Conclusion**

The Degree Classification Calculator demonstrates my understanding of OOP principles and showcases my ability to apply them to solve real-world problems. By leveraging robust validation methods, efficient algorithms, and consistent refactoring, I created a modular and scalable application. The project provides a solid foundation for further enhancements, aligning with modern software development practices and user needs.