# Reassessment

**Submission Date: 12th July 2024 @ 1200**

**Weighting: 70%**

**WARNING:**

- All submitted work will be electronically scanned for plagiarism and the use of Artificial Intelligence (AI) software. The work that you submit must be your own; any material from other sources must be correctly cited and referenced. If unreferenced material is detected a student will be reported for plagiarism. If AI should not be used in the production of this work. Use of AI which has not been acknowledged is an academic misconduct offence and again will lead to a report of plagiarism. University regulations require that this fact is clearly stated on the report.
- Assessor reserve the right to viva you to explain the code and it's implementation.

---

## Brief

The main aim of this assignment is to develop and test student's understanding and practical skills in mobile apps design.

The goal is to design an intuitive and functional app that allows users to record their academic grades for each module, providing a record of assessment marks. App will be able to tell student their current degree classification. The emphasis is on implementing a class-based structure, ensuring efficient data management, and incorporating persistent storage.

The assessment involves defining classes for years, modules and assessments, with specific attributes tailored to academic data representation.

To enhance the user experience, a user interface must be developed, facilitating the addition, editing, and deletion of modules and assessments. Additionally, you are expected to implement data validation checks, ensuring the accuracy of the recorded information.

## Coursework Submission Link

- Moodle submission link
  - One Android Project compressed with a .zip or .gzip extension
  - One applications, .apk file

## Learning Outcomes

On completion of the assignment, the student will develop:

1. an in-depth understanding of object-oriented programming
2. understanding of the basic principles of mobile apps design
3. skills in designing a mobile app.

## Rubic Summary

Full rubric is at the bottom of the brief.

- Clean Code: 25%
- OOP: 25%
- KDoc: 15%
- App Design: 35%

---

## Deliverables and Instructions

## Module Class

Define a `Module` class with attributes:

- name, code, level, credit
- overall mark as a percentage, and classification.
- The module should be linked to one or more assessments. The sum of assessment weightings should exceed 100%, ensuring that combined marks do not exceed 100%.
- Allow users to add, edit, and delete modules using instances of the `Module` class. Include methods for adding assessments to each module.

### Assessment Class

Define an `Assessment` class with the following attributes:

- Name, Weighting (as a percentage), Marks achieved (out of 100)
- Ensure that assessment types are predefined, and weightings are specified in percentages.
- Create instances of the `Assessment` class to manage assessment details for each module.

### Persistent Storage

Implement a mechanism to save and load data using JSON, CSV, or XML. Choose one based on your preference and ease of implementation:

### Degree Classification Implementation

UoG Academic Council and Governing Body have stated that:

- Level 5 is worth 20%
- Level 6 is worth 80%

$$\text{Overall Mark} = (0.1 \times \text{Average Marks at Level 5}) + (0.8 \times \text{Average Marks at Level 6})$$

### Overall Progress using Classes

- Utilise class instances and stored data to calculate and display overall progress classification.
- Implement validation checks within the class methods to ensure accurate data handling.

### Use KDOC

Update documentation to guide users on the new class-based structure and data storage methods.

## Design

App should be designed to be intutitive, user friendly, accessible, suitable colour palette, font families, font size and weightings. Images where deployed should be royalty free and of good quality. Your report will provide references for these.

---

## Rubric in Full

### Clean Code

| Performance Level | Description |
|---|---|
| 0-29% Fail | Code exhibits a lack of understanding and application of clean code principles. The structure is severely deficient, hindering comprehension. Naming conventions are inconsistent and fail to adhere to best practices. The overall code quality is substantially below the expected standard. |
| 30-39% Fail | Limited understanding of clean code principles is evident. The code structure is basic, but significant improvements are needed for clarity. Naming conventions are inconsistent and do not fully adhere to best practices. The code, while functional, lacks the sophistication expected in clean code. |
| 40-49% Satisfactory | The code demonstrates a basic understanding of clean code principles. The structure meets minimum requirements but lacks sophistication. Naming conventions are basic and partially follow best practices. While the code is acceptable, it falls short of achieving a higher level of cleanliness and clarity. |
| 50-59% Good | There is a good understanding and application of clean code principles. The code structure is above average, contributing to clarity and understanding. Naming conventions are good and mostly follow best practices. The code demonstrates a commendable effort in adhering to clean code standards. |
| 60-69% Very Good | The code exhibits a very good understanding of clean code principles. The structure is impressive, contributing to a high level of clarity and understanding. Naming conventions are very good and consistently follow best practices. The code showcases a level of cleanliness that goes beyond the expected standard. |
| 70-79% Excellent | An excellent understanding and application of clean code principles are evident. The code structure is exceptional, setting a benchmark for clarity and understanding. Naming conventions are excellent and consistently follow best practices. The code exemplifies an exceptional standard of cleanliness and organization. |
| 80-100% Exceptional | The code demonstrates an exceptional understanding and application of clean code principles. The structure is outstanding, demonstrating an unparalleled level of clarity and understanding. Naming conventions are exceptional and consistently follow best practices. The code sets an extraordinary standard for cleanliness and represents an outstanding example of clean and maintainable code. |

**OOP with Kotlin Concepts**

| Performance Level | Description |
|---|---|
| 0-29% Fail | The classes are inadequately defined, with missing attributes or incorrect implementation, reflecting a fundamental lack of understanding and application of object-oriented principles, including constructors and access modifiers. The organization is severely deficient, and naming conventions are inconsistent. The overall structure hinders comprehension, leading to a significant failure in meeting the required standards. |
| 30-39% Fail | The definition of classes is basic, with room for improvement. Some attributes may be missing or poorly organized. The implementation shows a limited understanding, and there are issues in adhering to object-oriented principles, including constructors and access modifiers. Naming conventions are inconsistent, contributing to a less clear structure. Significant improvements are needed to meet the expected standards. |
| 40-49% Satisfactory | The definition of classes is satisfactory, including necessary attributes and basic use of constructors and access modifiers. The implementation meets minimum standards but lacks sophistication, indicating a basic understanding of object-oriented principles. Naming conventions are basic and partially follow standards. The overall organization is satisfactory but requires refinement for a more polished structure. |
| 50-59% Good | The definition of classes is good, demonstrating above-average implementation with a clear organization. The understanding and application of object-oriented principles, including constructors and access modifiers, are evident. Naming conventions are good and mostly follow standards. The overall structure is above average, contributing to clarity and understanding. |
| 60-69% Very Good | The definition of classes is very good, showcasing an impressive implementation with a high level of organization, effective use of constructors, and appropriate access modifiers. The adherence to object-oriented principles, including inheritance, is commendable. Naming conventions are very good and consistently follow standards. The overall structure is highly impressive, contributing to a detailed and comprehensive understanding. |
| 70-79% Excellent | The definition and implementation of classes are excellent, setting a benchmark for excellence in organization and adherence to object-oriented principles, including advanced use of constructors, access modifiers, and inheritance. Naming conventions are excellent and consistently follow standards. The overall structure is exceptional, demonstrating an outstanding level of clarity and understanding. |
| 80-100% Exceptional | The definition and implementation of classes are exceptional, demonstrating outstanding organization and adherence to object-oriented principles, including sophisticated use of constructors, access modifiers, and inheritance. Naming conventions are exceptional and consistently follow standards. The overall structure is outstanding, setting an extraordinary standard for clarity and understanding. |

**KDoc**

| Performance Level | Description |
|---|---|
| 0-29% Fail | The code lacks proper documentation, with either absent or insufficient KDoc comments. There is little to no clarity on the purpose and functionality of the code. KDoc comments, if present, lack consistent and standard naming conventions, hindering understanding. Overall, the documentation is severely deficient, falling far below the expected standard. |
| 30-39% Fail | Basic documentation is provided with minimal clarity in the KDoc comments. The understanding of KDoc principles is limited. While the documentation meets basic requirements, it lacks sophistication, and clarity is minimal. Naming conventions within the KDoc comments are basic and may not fully adhere to standard practices. Improvements are needed to achieve a higher level of documentation quality. |
| 40-49% Satisfactory | The code meets minimum documentation requirements, but sophistication is lacking. KDoc comments provide some clarity on the purpose and functionality of the code. However, there is room for improvement in achieving a more comprehensive understanding through documentation. Naming conventions within the KDoc comments are basic and partially follow standard practices. The documentation is satisfactory but falls short of a higher standard. |
| 50-59% Good | Good documentation is present, offering a clear understanding of the purpose and functionality of the code through KDoc comments. The understanding and application of KDoc principles are commendable. Naming conventions within the KDoc comments are good and mostly follow standard practices. The documentation represents a level of quality that goes beyond the expected standard. |
| 60-69% Very Good | Very good documentation is provided, offering a comprehensive understanding of the purpose and functionality of the code through well-crafted KDoc comments. The understanding and application of KDoc principles are impressive. Naming conventions within the KDoc comments are very good and consistently follow standard practices. The documentation sets a benchmark for excellence. |
| 70-79% Excellent | The documentation is excellent, providing an exceptional understanding of the purpose and functionality of the code through meticulously crafted KDoc comments. The understanding and application of KDoc principles are exceptional, setting a benchmark for excellence. Naming conventions within the KDoc comments are excellent and consistently follow standard practices. The documentation exemplifies an outstanding standard. |

*Continued overleaf*

| Performance Level | Description |
|---|---|
| 80-100% Exceptional | Exceptional documentation is present, offering an outstanding understanding of the purpose and functionality of the code through impeccably crafted KDoc comments. The understanding and application of KDoc principles are unparalleled, setting an extraordinary standard. Naming conventions within the KDoc comments are exceptional and consistently follow standard practices. The documentation represents an outstanding example of thorough and maintainable documentation. |

**App Design/Usability**

| Performance Level | Description |
|---|---|
| 0-29% Fail | The app design falls below the standard, lacking an intuitive interface and presenting significant usability challenges. User interaction is problematic, leading to a poor user experience. Themes, if present, are inconsistent and fail to enhance the overall design. Accessibility features are either absent or insufficient. Naming conventions for UI elements lack consistency and do not follow best practices. Overall, the app usability is inadequate and hinders user engagement. |
| 30-39% Fail | The app design is below the expected standard, lacking essential features and impacting user interaction. Usability improvements are needed to provide a more intuitive user experience. Themes, if present, are basic and inconsistent, contributing to a less appealing design. Accessibility features are limited. Naming conventions for UI elements are basic and inconsistent with best practices. The overall usability requires significant enhancements. |
| 40-49% Satisfactory | The app design meets basic standards but requires improvements for a more intuitive user experience. Essential features are provided, but additional enhancements are needed. Themes, if present, are basic and partially follow best practices. Accessibility features meet minimum requirements but lack sophistication. Naming conventions for UI elements are basic and partially follow best practices. The overall usability is satisfactory but needs refinement. |
| 50-59% Good | The app design and usability are good, including valuable features and providing a reasonably intuitive user experience. Themes, if present, are good and follow best practices, contributing to an appealing design. Accessibility features are good, ensuring a more inclusive experience. Naming conventions for UI elements are good and follow best practices. The overall usability is above average, meeting expectations. |
| 60-69% Very Good | The app design and usability are very good, including advanced features and providing a highly intuitive user experience. Themes, if present, are very good and consistently follow best practices, enhancing the overall visual appeal. Accessibility features are very good, ensuring inclusivity. Naming conventions for UI elements are very good and consistently follow best practices. The overall usability sets a benchmark for excellence. |

*Continued overleaf*

| Performance Level | Description |
|---|---|
| 70-79% Excellent | The app design and usability are excellent, setting a benchmark for advanced features and providing an exceptional user experience. Themes, if present, are excellent and consistently follow best practices, contributing to an outstanding visual appeal. Accessibility features are excellent, ensuring a highly inclusive experience. Naming conventions for UI elements are excellent and consistently follow best practices. The overall usability is exceptional, representing an outstanding example of mobile app design. |
| 80-100% Exceptional | The app design and usability are exceptional, setting a benchmark for advanced features and providing an outstanding user experience. Themes, if present, are exceptional and consistently follow best practices, contributing to an extraordinary visual appeal. Accessibility features are exceptional, ensuring the highest level of inclusivity. Naming conventions for UI elements are exceptional and consistently follow best practices. The overall usability is unparalleled, representing an extraordinary standard in mobile app design. |

*END OF DOCUMENT*