**MARKING GRID FOR SOFTWARE DEVELOPMENT**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Classification** | **Range** | **Analysis of Requirements (25%)** | **Knowledge of Programming Logic (30%)** | **Program Compilation & Execution (25%)** | **Syntax & Coding Standards (10%)** | **System Testing (10%)** |
| Distinction | 80 – 100 | Comprehensive understanding of requirements. Program fully implements all required features, including creation of exams, calculating scores and displaying results, demonstrating comprehensive exception handling through use of custom exceptions for validation, providing informative error messages. Exceptions are correctly thrown, caught and handled. Effective implementation of the Printable interface in Student class, allowing for appropriately formatted output in tabular format of student results to text file for both print methods. | Outstanding understanding of how to create objects and specify conditions. Comprehensive use of control flow and data structures, appropriate to problem specified. Program is decomposed into reusable, coherent classes/modules demonstrating comprehensive evidence of modularity and OOP concepts. | Program compiles with no errors, fully implements all required features, including the creation of exams, calculating scores, handling exceptions, and displaying results and contains no logic errors when executed. User prompts/feedback are understandable, clearly laid out and appropriately worded. | Outstanding use of comments, indentation and whitespace.  Suitable variable and object names with variable initialisation. Code is very readable. | Comprehensive testing showing clear understanding of how the implemented system meets the requirements and optimisation implemented. |
| 70 – 79 | Excellent understanding of requirements. Program fully implements all required features, including creation of exams, calculating scores and displaying results, demonstrating suitable exception handling through use of custom exceptions for validation, providing informative error messages. Exceptions are correctly thrown, caught and handled. Suitable implementation of the Printable interface in Student class, allowing for formatted output of student results to text file for both print methods. | Excellent understanding of how to create objects and specify conditions. Very good use of control flow and data structures, appropriate to problem specified. Excellent evidence of modularity and OOP concepts. | Program compiles with no errors, fully implements all required features, including the creation of exams, calculating scores, handling exceptions, and displaying results and contains no logic errors when executed. User prompts/feedback are understandable, but layout and wording are poor. | Effective use of comments, indentation and whitespace.  Suitable naming conventions for variable and object names with variable initialisation. Code is very readable. | Effective testing showing clear understanding of how the implemented system meets the requirements with some evidence of optimisation. |
| Commendation | 60 – 69 | Good understanding of requirements. Classes coded demonstrating some awareness of the logic required including creation of exams, calculating scores and displaying results. Some awareness of exception handling though the creation of custom exceptions for validation. Some printing facilities implemented with limited formatting. | Good understanding of how to create objects and specify conditions. Good use of control flow and data structures, appropriate to problem specified. Good evidence of modularity and OOP concepts. | Program compiles with no errors, fully implements all required features, including the creation of exams, calculating scores, handling exceptions, and displaying results but contains some logic errors when executed. User prompts/feedback are understandable, but layout and wording are poor. | Good use of comments, indentation and whitespace.  Suitable variable and object names. Code is readable. | Very good testing showing good understanding of how the implemented system meets the requirements with limited evidence of optimisation. |
| Pass | 40 – 59 | Reasonable understanding of requirements. Classes coded demonstrating limited awareness of the logic required including basic class creation with basic functionality. Basic awareness of exception handling though the creation of exceptions for validation. Basic printing facilities implemented with limited formatting. | Limited understanding of how to create objects and specify conditions. Limited use of control flow and data structures. Basic evidence of modularity and OOP concepts. | Program compiles with no errors, implements most of the required features, including the creation of exams, calculating scores, handling exceptions, and displaying results but contains several logic errors when executed. User prompts/feedback are understandable, but layout and wording are poor. | Poor use of comments, indentation and whitespace.  Unsuitable variable and object names. | Good testing showing basic understanding of how the implemented system meets the requirements. |
| Fail | 0 – 39 | Insufficient understanding of requirements demonstrated. Unable to code required classes. | Lack of evidence of understanding of objects, conditions, control flow and data structures. No modularity attempted. | Program fails to compile and execute due to errors. User prompts are misleading or non-existent. | No coding standards followed. | Insufficient testing. |
| Marks | |  |  |  |  |  |
| **Tutor comments**  **Signature & Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | | | | |

**FINAL MARK**  / **100%**