Applications are given four marks, one for each of the following categories: program correctness, code elegance, documentation, and readability.

1 Program Correctness

The application meets all of the program specifications, i.e., the student has completed all of the base tasks including following submission instructions (e.g., the student submitted a Jar file of their BlueJ project). A level 5 can only be attained if all base tasks and all of the Challenge Tasks are completed.

Level 5 – The application works as described in the assignment; all base tasks are completed; the implementation demonstrates originality, creativity, and technical excellence in the completion of all challenging tasks the student included documentation describing the challenge tasks that were completed and how; all submission instructions were followed correctly

Level 4 – The application works as described in the assignment; all base tasks are completed; the student has completed some of the challenge tasks; the student included documentation describing the challenge tasks that were completed and how; all submission instructions were followed correctly

Level 3 – The application works as described in the assignment; all the base tasks are completed, however, no challenge task was attempted; all submission instructions were followed correctly

Level 2 – Minor details of a task(s) are violated; the application functions correctly on the majority inputs/actions

Level 1 – Some tasks are incomplete; the application functions incorrectly on some inputs/actions

Level 0 – Significant details of a task are violated, or the program often exhibits incorrect behaviour, e.g., the application crashes on some input or when a particular action is performed; the application does not run because of one or more syntax errors

2 Code Elegance

The application is written in such a way that the code is reusable and efficient (i.e., memory usage and complexity). The application appropriately uses loops and functions to reduce code complexity, reduce repeated code and increase maintainability. The application does not have hard-coded solutions or poorly designed solutions. A poorly designed solution is overly complicated, utilises excessive amounts of memory or utilises a slower approach to a problem. A level 5 can only be attained if all base tasks and all Challenge Tasks are implemented.

Level 5 – Student demonstrates **excellent** use of classes and functions to produce reusable and maintainable code, where possible, in the base and all of the challenge tasks; the code is efficient without sacrificing readability and understanding

Level 4 – All of the base tasks and some of the challenge tasks are implemented in such a way that code can be reused, where possible; the majority of the code is written in such a way that is easy to maintain, i.e., add new or extend features; the code is efficient without sacrificing readability and understanding;

Level 3 – The application is implemented in such a way that only a few code segments could be rewritten to increase code re-usability; the code is fairly efficient without sacrificing readability and understanding; the student made a poor design choice in a single area

Level 2 – The application is implemented in such a way that many code segments (or functions) could be rewritten to increase code re-usability; some of the code is unnecessarily complex or poorly designed

Level 1 – The application is unnecessarily complex and/or uses brute force; the application contains many instances where the code could have been written in an easier, faster or better fashion

Level 0 – The application is not organised for re-usability, e.g., all the code is in a single class (or function); no effort is made to create reusable code; the application is excessively long and poorly organised

3 Documentation

The application is sufficiently documented. Good documentation/comments should explain what the code does and how it does it. Comments can also be used to highlight nuances in your solution, e.g., a segment of code that only works under certain conditions.

Level 5 – The documentation is well written, organised and clearly explains what the code is accomplishing and how; the student documents the contents of each class at the beginning of the file; the student documents the purpose of each function, i.e., the function's parameters and return values at the beginning of each function; the student documents each logical block of code when performing non-obvious operations; Every file includes header information (e.g., student name and k-number)

Level 4 – The documentation consists of comments that are useful in understanding the code and/or structure of the program; each file has header information (e.g., student name and k-number)

Level 3 – The documentation consists of comments that are somewhat useful in understanding the code; some of the comments state the obvious

Level 2 – The documentation is simply comments embedded in the code but does not help the reader understand the code

Level 1 – One or more code segments could benefit from comments or the code is overly commented

Level 0 – The application has no comments, not even the header with the student's name or k-number

4 Readability

The application is easy to understand and uses good programming practices.

Level 5 – The application is exceptionally well organised and very easy to understand; the student used indentation appropriately and consistently to delineate code blocks; each function performs a single well defined operation; the student used meaningful identifier names, i.e., good function and variables names; the student used white space between logical code blocks; the student uses consistent spacing around operators and variables; classes are self-contained with private data hidden and methods are public only when necessary

Level 4 – The code is clean, understandable and organised; the student used meaningful identifier names, i.e., good function and variables names; the student used white space between logical code blocks

Level 3 – The application has minor issues such as inconsistent indentation

Level 2 – The application has one or two issues that makes the program difficult to understand such as poorly named identifiers and disorganised code

Level 1 – The application has more than two issues that makes the program difficult to understand

Level 0 – The code is readable only by someone who knows what it is supposed do; the application has several issues that makes the program difficult to understand