**COM1027 Coursework (80%)**
**Grade Descriptor/ Rubric**

| Category | Explanation | 0 | 25 | 50 | 75 | 100 |
|---|---|---|---|---|---|---|
| **Clases and Class encapsulation** | Did you provide all the fields and constructors for the classes as per the UML diagram? | The majority of the fields were defined with the wrong visibility and the initialisation of fields in constructors was largely incorrect. | Some of the fields were defined with the wrong visibility and several of the initialisation of fields were not correct in the constructors. | The majority of the fields had appropriate visibility and were initialised appropriately in the constructors. The majority of the parameters used within constructors were sensible but some were superfluous. | All the fields had appropriate visibility and were initialised appropriately in constructors. The majority of the parameters used within constructors were sensible but there may have been some redundancy. | All the fields had appropriate visibility and were initialised appropriately in the constructors. There was clear thought about what parameters were needed in the constructors and they were efficiently used. |
| **Accessors/Modifiers** | Did you use the appropriate setters/getters? Did you expose any data structures? | The code was not systematic in its use of getters and setters and there may also have been leakage of encapsulation principles. | The code was limited in its use of getters and setters and there may also have been leakage of encapsulation principles. | The code was systematic in its use of getters and setters but encapsulation principles were not adhered to. | Only setters and getters that were required were included and/or therefore the API was tightly defined, and no leakage of encapsulation was present anywhere in the code. However there were some inconsistencies in either the setters/getters or in the encapsulation. | Only setters and getters that were required were included, therefore the API was tightly defined, and no leakage of encapsulation was present anywhere in the code. |
| **Methods (excluding getters)** | Did you define all the required methods with the appropriate visibility? | No methods are defined or methods do not compile/function as expected. | Methods are defined, but signatures do not match the class diagram or description | Methods are defined, but do not behave as expected. | Methods are defined and perform the expected behaviour. | Methods are defined and perform the expected behaviour; The code is efficinet and uses local variables where necessary; Code is written defensively to validate parameters. |
| **Algorithms** | Did you provide good algorithms, nice loops, no redundant local variables? Did you use appropriate data structures to support your algorithms, were they efficient for what you needed? | The complex methods in the code were not documented clearly and the algorithms were not clear. | The complex methods in the code were not documented clearly but there was some structure to the code. | The complex methods in the code were documented and there was some structure to the code. | The complex methods in the code were documented clearly and there was good structure to the code. | The complex methods in the code were documented clearly and there was excellent efficient structure to the code with efficient lookup of data structures. |
| **Abstraction** | Did you implement correctly the modularisation of the code? | There was no attempt to use abstraction in the code. | There was limited attempt to use design abstraction in the code. | There was an attempt to use abstraction and modularisation in the code but this was not implemented correctly. | Excellent and appropriate use of the required keywords to perform abstraction in the code. | Excellent and appropriate use of the required keywords to perform abstraction in the code and excellent use of the abstract and sub classes in making the code extensible. |
| **Validation** | Did you check for validity of parameters? Was exception handling implemented? | There was no validation or exception handling included in the code. | There was a limited attempt at validation or exception handling included in the code. | There was a reasonable attempt at validation and exception handling included in the code but this was used inconsistently. | There was good use of validation of parameter input and use of exceptions throughout the majority of the code. | There was excellent validation of parameter input and use of exceptions throughout the code. |
| **Style** | Did you follow the style guidelines when commenting and formatting your code? | Little or no use of appropriate formatting, comments or variable names as defined by the style guidelines | Some use of formatting, comments and variable names but this is not used consistently. | Some use of formatting, field and method comments, variable names and code style, but this is in need of improvement to follow the guidelines. | Appropriate use of formatting, field and method comments, variable names and overall code style that follows the guidelines. | Appropriate use of formatting, field and method comments, variable names and overall code styl, which follows a consistent set of guidelines to provide readable and maintainable code. |
| **Tests** | Did you define all the required test methods to check the functionality of your code? | Little or no tests defined | Some tests are defined but some or all may fail to run. | Some tests are defined which may run and pass, but not the entire code functionality is tested. | Essential tests are defined. All tests run and pass. | An appropriate set of tests are defined (such as white box, black box and requirements tests). All tests run and pass. |