CMP1903M Object Oriented Programming Assessment 2 2021-2022

Learning Outcome	Criterion	Pass	2:2	2:1	1
[LO2] Identify the values of object-oriented design and programming	Illustrate OO features which were used (20%)	Simple description of the code; fail to mention some OO features which are used in the code; you provide some description of encapsulation/data abstraction, inheritance, interfaces or polymorphism in the code.	Clear evidence of the OO features in the code. Encapsulation/data abstraction referred to and shown where they are evident. Additional descriptions of inheritance and/or interfaces may also present.	Thorough description of the OO features in the code; encapsulation/data abstraction described and examples shown in the code. Additional descriptions of inheritance, interfaces and polymorphism features used in the code are also present.	Extensive description of the OO features in the code; encapsulation/data abstraction completely described and exemplary examples shown in the code. Comprehensive descriptions of inheritance, interfaces and polymorphism features used in the code are also present.
[LO3] Apply object-oriented principles to the implementation of software programs	Develop an object-oriented solution to a problem (60%)	A limited implementation is presented. The application works, however its functionality is incomplete. For example the rules of the game (as described in the brief) are not implemented correctly Erroneous input is handled either by error or exception handling methods, but the errors are not handled completely and/or all possible errors are not handled. Some evidence of object-oriented features such as classes, object instantiation, encapsulation and methods/method calls are present, but they may not be implemented well. The checklist and video is completed.	An implementation is presented which works. The functionality allows the dice game to be played according to the rules of the game. Erroneous input is handled either by error or exception handling methods. All errors may not be addressed. Clear evidence of object-oriented features such as classes, object instantiation, encapsulation and methods/method calls are present. The checklist and video is completed.	An implementation is presented which works. The functionality allows the card game to be played according to the rules of the game. A log file may be used to summarise game statistics. Erroneous input is handled either by error and exception handling methods. i.e. the game does not crash with erroneous input. Thorough evidence of object-oriented features such as classes, object instantiation, encapsulation and methods are present. Inheritance/interface use is evident. Clear use of public/private access modifiers. Static polymorphism, eg. method/operator overloading, is present. The checklist and video is completed.	An implementation is presented which works. The functionality allows the dice game to be played according to the rules of the game. Erroneous input is handled either by error and exception handling. All possible errors are handled. Extensive evidence of OO features such as (but not limited to) classes, object instantiation, encapsulation , methods, data abstraction and inheritance, virtual/abstract methods are implemented. Protected access control. Dynamic polymorphism (method overriding) The checklist and video is completed.

[LO4] Use testing principles in the testing and debugging of object-oriented applications	Testing practices used to verify/validate a software application (20%)	A basic testing strategy is evident.	A clear testing strategy is used. Some of the example features listed in the brief are used to test inputs and observe outputs.	A thorough testing strategy is used. Individual methods may be tested. The example features listed in the brief are used to test inputs and observe outputs. .	An effective testing strategy is used. Individual methods are tested. The example features listed in the brief are used to test inputs and observe outputs.
Weighting is 70% of the module					