University of Lincoln
School of Engineering and Physical Sciences
UG Criterion Reference Grid 2024-2025

CMP1903M Object Oriented Programming A02 2024-2025

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%) [Video, Report]	Simple description of the code; fail to mention some OO features which are used in the code. There is some discussion, though, of OO features such as object instantiations and method calls for example.	Clear evidence of the OO features in the code. Inheritance is referred to and shown where it is evident in the code.	Thorough description of the OO features in the code; Inheritance is described in the video and report and examples shown in the code.	Extensive description of the OO features in the code; Inheritance and polymorphism are described and exemplary examples shown in the code and referred to in the report. An explanation of why they benefit your code is also made. Examples of static OR dynamic polymorphism may be present
[LO3] Apply object-oriented principles to the implementation of software programs	Develop an object-oriented solution to a problem (60%) [Code, Video, Report]	A limited implementation is presented. The application works, however, its functionality is incomplete. For example, the specific requirements (as described in the brief document) are not implemented correctly. Erroneous input is handled 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 and methods/method calls are present, but they may not be implemented well. The checklist and video are completed.	An implementation is presented which works. The player can select from a menu and view statistics. The functionality allows to navigate through multiple rooms or to battle monsters with varying difficulty or to manage an inventory with multiple items. 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. C# features such as class constructors are present. The checklist and video are completed.	An implementation is presented which works. The player can select from a menu, and additionally can perform tests The functionality allows at least two of the following: 1) navigate through multiple rooms 2) battle monsters with varying difficulty 3) manage an inventory with multiple items. Erroneous input is handled 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 use is evident. Clear use of public/private access modifiers. C# features such as collections (Lists<>, etc) are used. The checklist and video are completed.	An implementation is presented which is complete. The functionality allows all of the following: 1) navigate through multiple rooms 2) battle monsters with varying difficulty 3) manage an inventory with multiple items. Erroneous input is handled either by error and exception handling. All possible errors are handled. Evidence of additional OO/C# features such as (but not limited to) LINQ, interfaces, virtual/abstract methods are implemented. Protected access control is used. Static(method overloading) and/or Dynamic(method overriding) polymorphism is present The checklist and video are completed.
[LO4] Use testing principles in the testing and debugging of object-oriented applications	Testing practices used to verify/validate a software application (20%) [Code, Video, Report]	A Testing class is used to verify aspects of the game operation. This verification may not be correct though and may not be useful.	A Testing class is used to instantiate a Game object and verify methods are operating correctly.	A testing class is used to instantiate a Game object and verify methods are operating correctly using debug.assert()/trace.assert() methods	A Testing class is used and individual methods are tested. A robust reporting mechanism is used to document the test outcomes (a test log file for example).
Weighting is 70% of the module					

1: Navigate through multiple rooms
2: Battle Monsters with varied difficulty
3: Inventory with multiple items
4: LINQ → Searching inventory items (eg get all spells)
5: Virtual methods → override method (eg Monster.DoDamage)
6: Abstract methods → Can be used in interfaces
7: Interfaces → IHoldsWeapon
8: protected key word - accessible within its class and derived classes (only accessible from current class or derived class)
9: Method overloading
10: Method overwriting
11: A testing class is used
12: Use a test log to document outcomes
13: Error handling → eg all user input