H171 35 Software Development: Object Oriented Programming 2020/21

# Assessment Task 1 – Outcomes Covered 1-3

# Project instructions

There is one assessment for this Unit. This is an open book project covering all Outcomes. The project is broken down into two stages. Stage 1 is the program implementation and Stage 2 is the testing of the completed program.

You are required to implement a computer game based on the design documentation produced by you for the Unit H172 35 Systems Development: Object Oriented Analysis and Design.

All the Evidence Requirements which you must achieve are detailed after the client brief.

This project will be carried out under supervised and unsupervised conditions, i.e. you may work on this in your own time. The assessor will check the authenticity of any work you have done unsupervised. This may involve methods such as interviews, demonstrations, checking files, etc. and may be carried out at random and pre-arranged times.

The assessor will specify the various deadline periods for the project. It is up to you to determine your own deadlines within these. You may decide to work on multiple tasks at the same time but you should try to fully complete and achieve one stage before completing the next. **Applying this method of working is good preparation for the HND Graded Unit.**

You should read all the Evidence Requirements for each stage and clarify any points with the assessor before you commence the project.

You are required to read the following brief and then complete the stages detailed below:

# Client brief

Weasleys’ Wizard Wheezes have commissioned you to design and develop a digital version of Quidditch. Quidditch is a popular sport played in the Wizarding World; you can read all of the objectives and rules here - <https://en.wikipedia.org/wiki/Quidditch>

The objective of Quidditch, as with most sports, is to be the team that has gained the most points by the end of the match. Matches are played between two opposing teams of seven players riding flying broomsticks; using four balls: a Quaffle, two Bludgers, and a Golden Snitch. Centred around the use of each ball, there are four positions: the Chasers and Keeper (who play with the Quaffle), the Beaters (who play with the Bludgers), and the Seekers (who play with the Golden Snitch). Each team has three Chasers, one Keeper, two Beaters, and one Seeker. Matches are played on a large oval pitch with three ring-shaped goals of different heights on each side. It is an extremely rough but very popular [semi-contact](https://en.wikipedia.org/wiki/Semi-contact) sport, and has a fervent fan following in the Wizarding World.

Your game will be a prototype consisting of a Quidditch field surrounded by an invisible magical barrier (you can have a square pitch to start off, to make it simpler). Two human players will play as Seekers of opposing teams, who vie to be the first to collect the Golden Snitch. This immediately awards the catcher’s team 150 points and proceeds to the next level, or ends the game if the players are already on the last level.

The game will require at least two levels. Each level should be harder to solve than the previous one, either by having obstacles or a more complex starting arrangement for the pitch, or a combination of both. The program should record how many moves the players take to solve a level (hint: give each player a starting number of points, and deduct one for each move). This information should be output visually.

You are advised to start level 1 very simply, with two Seekers and a non-moving Snitch. For level 2, you could maybe make the pitch a more complex shape, and the Snitch should be move too.

# Stage 1 — Implementation

You are required to implement the program in an object oriented manner. The solution must demonstrate each of the following:

**Design requirements:**

♦Abstraction, encapsulation and information hiding used where appropriate

♦Inheritance used

♦Polymorphism used

♦All class-wide variables are private to prevent content coupling

♦Class-wide variables are kept to a minimum to ensure a minimum of common coupling

♦Data coupling is used (parameter passing) in preference to content or common coupling

♦Program does not contain a lot of unnecessary data coupling

♦Classes are highly cohesive

**Implementation requirements:**

♦A working solution which meets the requirements of the given brief

♦Variables are correctly declared and initialised

♦Arithmetic and/or logical operators are used correctly

♦A range of control structures are implemented correctly

♦At least two data structures are implemented correctly

♦The program contains a minimum of four classes, which contain attributes, methods and a constructor method

♦A minimum of three objects are created from the classes, with appropriate initial attribute values set through the constructor methods

♦The program contains at least one overloaded method (this may be a constructor method)

♦Classes are linked appropriately through association, aggregation or inheritance relationships

♦Parameters are passed correctly both within and between objects

♦Appropriate access types are defined for methods, attributes and classes

♦Use of pre-defined classes and/or methods from the standard object library

♦The program appropriately handles errors with exceptions or pre-validation

♦The program code is commented appropriately throughout

# Stage 2 —Testing

After completing Stage 1 you are required to develop a test plan and test the completed program. The test log should identify any areas where the program fails, and detail any fixes and retests required. I will supply a sample Test Plan and Log to show you the standard expected.

# Stage 3 – Complete Package

You should hand in –

* Complete Python code, complete with any image, font or sound files that you have used (you do not have to generate an executable)
* Amended Class Diagram that matches your implementation
* Test plan and log
* A paragraph on the test strategies and methodologies used