## Lab Week 1 – Polymorphism.

# Objectives

The objectives for this week are:

* Setting up the Github repository and cloning the required code.
* Defining polymorphic classes.
* Implementing safe downcasting.
* **(Additional material)** The use of destructors in polymorphism.

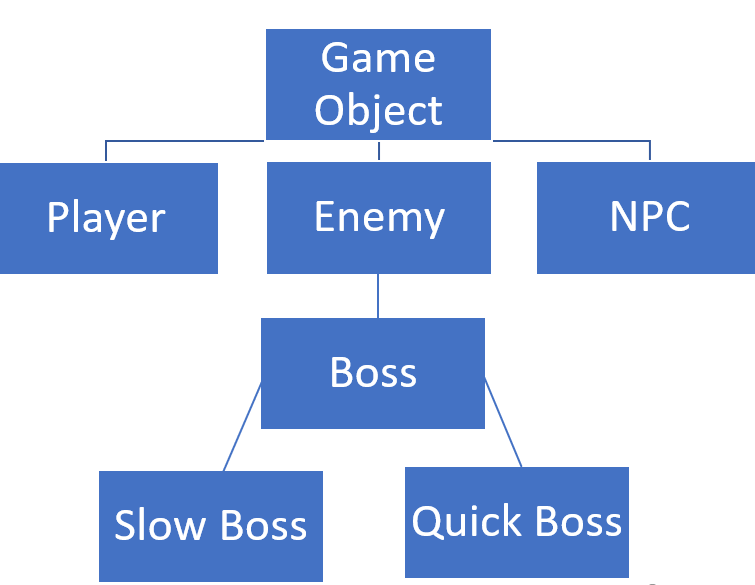
# Tasks

# Task 1

* Set up a new folder for this module calling it IMAT2906 \_C++\_AdvancedOO.
* Clone the week 1 lab code repo into the sub folder lab code/week 1 Polymorphism.
* Using CMake, create the solution.
* Simple CMake commands will be covered in the first labs and lecture.

**Task 2**

* Look over the code in the solution for polymorphism.
* Take some time to understand what is happening and the structure.
* With the code that you have been provided with:
* The structure you want to replicate is:



* + Split it reflecting a more defined OO approach.
  + Create a Player class that is derived from the GameObject.
  + The Player should overload the GameObject virtual functions and also supply another function that can be overridden.
  + Have another object that is derived from game object. This will be a boss.
  + The boss will allow for the creation of both a quick and slow boss by adding attributes to them in the form of functions. These should be:
    - Slow boss: heavy weapon setup and fire
    - Quick boss: change in wheels / tracks
  + NB: These don’t need to be shown on the vehicles themselves
* Your Main.cpp should:
  + Construct a player, a boss and an NPC.
  + In this first instance, they only need to be shown to be created in the console. **Use cout or another method to do this.**

# Task 4

* Create functions so that each of the game objects can be drawn. You may want to think about a single set of functions that can be used for this that exist in the Game Object class.

# Task 3

* Using the code, demonstrate the use of upcasting, focussing on one the enemy classes.
* Comment this, demonstrate how it works in the console.

**Task 4**

* Create destructors for the GameObject and newly created enemy classes so that they align with being used as polymorphic classes and would not produce possible memory leaks.
* If you are unsure of the process for this, take another look over the lecture notes and videos / ask your tutor.