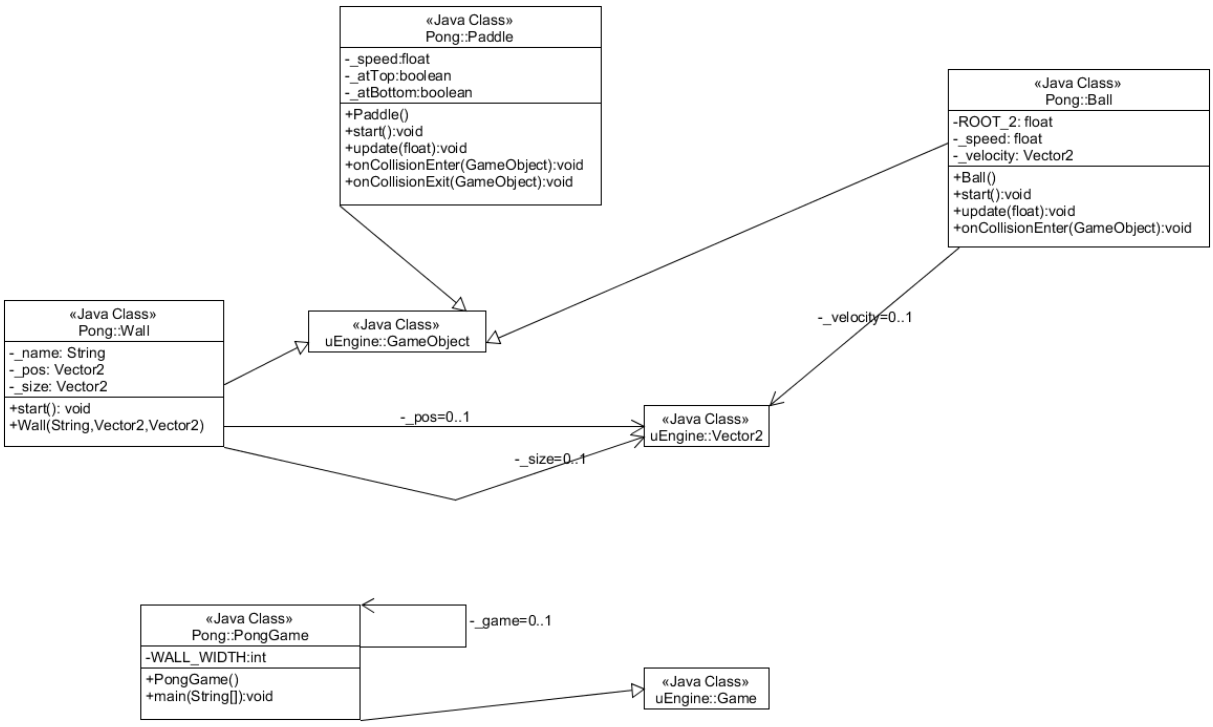
Student Name: Mohammed Hamada Gasmallah Student ID: 10042860

Assignment 1, Part 2 Report

CISC486

1. **Create a class diagram of the Pong game provided with uEngine. Include all classes in the Pong game itself, and all important attributes and public operations in those classes. If any of these classes reference or derive from classes in the uEngine itself, include those classes in your diagram (but you do not need to include their attributes or operations.) Use the [UMLet UML diagram editor] to create your class diagrams.**



Above is my UML diagram made in UMLet for the Pong game provided with the uEngine. Note that the arrows with white centers define an extension of the class. IE: the Pong::PongGame and the white arrow pointing to uEngine::Game defines the relationship as Pong::PongGame class as extending uEngine::Game abstract class.

2. **Which part of the code implements the scene graph? Is this actually a graph?**

Since the Pong::PongGame class extends the uEngine::Game abstract class, it uses the addGameObject function in the uEngine::Game abstract class to add gameObjects to a scene graph contained within. The scene graph is the variable _gameObjects which is a private list of GameObject. This is being used as the scene graph. This is not precisely a scene graph due to the lack of parent/child relationships within the list. This list does not define a parent/child relationship between any of the GameObjects and this is essential for all scene graphs.

3. **What design pattern does the Input engine use? Why is this design pattern used?**

An important design pattern used by the Input engine is the façade pattern (which consequently also uses the Singleton pattern). This design pattern is used to allow for an interface that the user may find contains useful information. This allows us to provide both a simplified interface to more complicated code, as well as allows us to manage a complex or difficult system into an abstracted interface. The singleton pattern ascertains that there is only one instance of the interface instantiated.

4. **The Component class is not used in the provided code. Give a concrete example of how the Component class might be useful.**

An example of how the Component class may be used, is as an object that may be added to GameObjects so that they are run during updates. For a more concrete example, one can look at adding an artificial intelligence component that must be updated with all updates of GameObjects.

5. **Explain why use of Java's paintComponent method is not a true implementation of the frame loop concept. How should this really be done?**

The traditional frame loop concept contains a hard push to the render engine to force it to render the scene. This is not something that can be ignored or pushed to a further time. Unfortunately, if one reads the JFrame documentation, one can find that the JFrame repaint() method, does not in fact guarantee a repainting of the frame, it merely requests one to all the JPanels and their children. The paintComponent/repaint() method of rendering to a window is not the traditional frame loop and is often referred to as Passive Rendering (see link). Using Active Rendering and using a proper rendering loop allows for a smaller overhead on the rendering engine, essential for obtaining the best of framerates. The proper active rendering loop is often done by obtaining all graphics and rendering them for one frame and then disposing the graphics at the end of the screen. This is simplified with a front and back buffer to allow quick swapping between them for faster rendering.