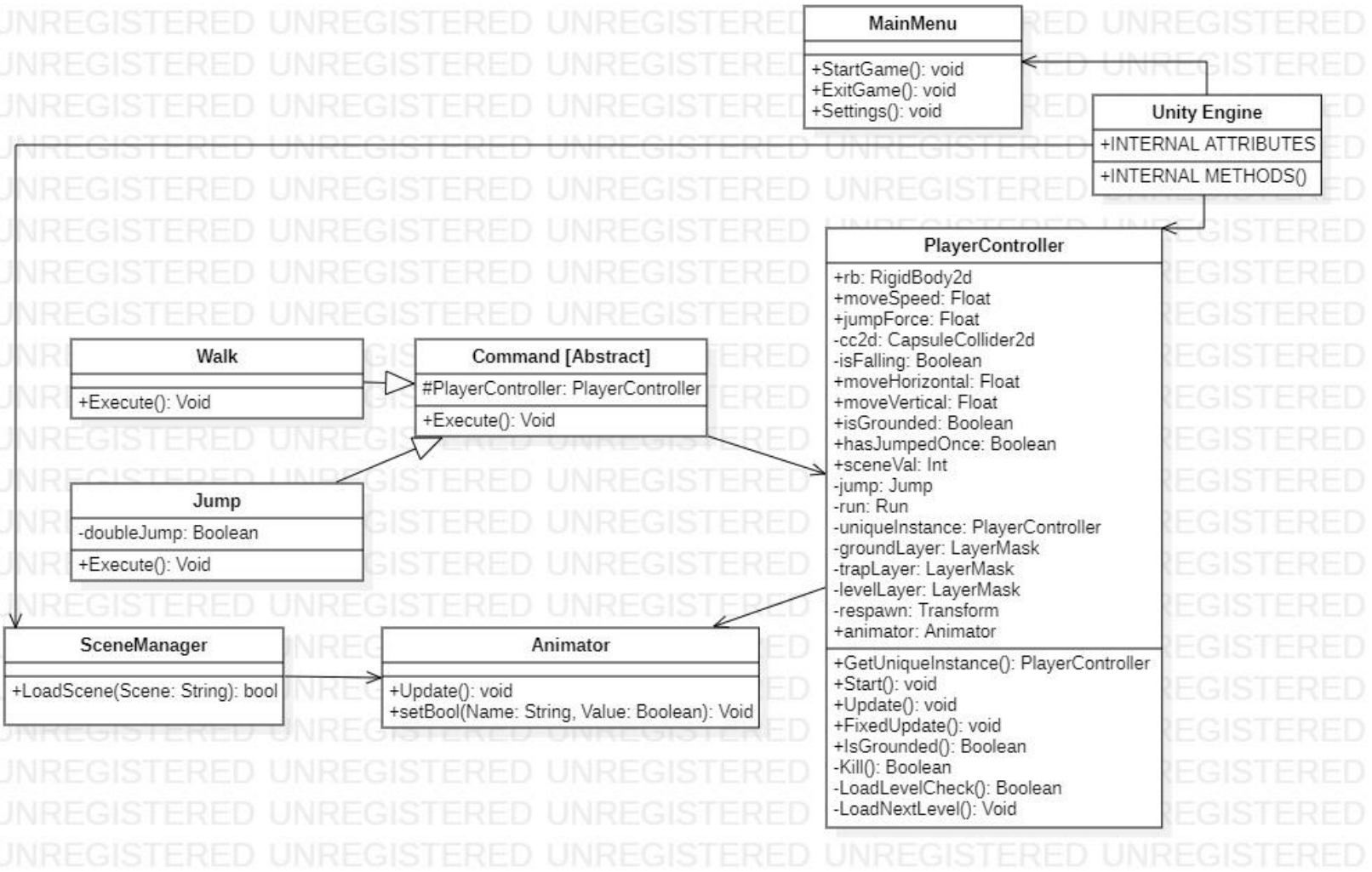
Project 6 Update

Status Summary:

- **Title:** Finding Felix
- **Names:** Owen Smith, Kyle Ma
- **Work Done:** We have prototypes of most of the systems we'll be using. We have prototypes for menus, animation, movement, level design, sprite design, level changes. The player is greeted with a rudimentary menu, and interacting with it appropriately loads the first scene/level. The player can move, traverse platforms, be killed and sent back to the beginning of the level, and move to the next level. Though most of these things need refinement, they are all functional for the demo, and plans for future changes have been drafted. As for how this work was distributed, Owen and Kyle have shared responsibility on most of these components. As far as individual work goes, Kyle has done a lot of level functionality, and Owen has done most of the pixel art.
- **Changes or Issues Encountered**: There have been quite a few big changes to our design as we move through the project. When we were designing our initial UML, we still had a lot to learn about Unity (we still do), and we now see that a lot of the classes we had, particularly a good chunk of the various managers we were going to code from scratch already exist in Unity. There are certain times where creating proprietary managers makes sense (animation, audio), but a lot of these will ultimately be unnecessary.
- **Patterns:** Frankly, the way that Unity's class structure is laid out, particularly Monobehavior, makes it challenging to adhere strictly to some of the design patterns we have seen in Java throughout the semester. That being said, we are still employing these patterns as best we can. Instilling some class structure and execution logic into a pattern like Command helps decouple some of the code that we would likely otherwise keep in the same place. Similarly, our PlayerController employs a Singleton pattern, which is great because we only ever have one instance of one in a given scene, and we can access data regarding location or motion of the character to trigger other events in the scene. All in all, the patterns are helping our design stay encapsulated, abstracted, and decoupled.

Class Diagram:



Plan for Next Iteration:

For the next iteration of our project, we need to implement a pause function, a save and load system for the game, and a quit. We also need to add in art work for the background and clean up how the game looks. The movement also needs to be fixed, as we would like to add more features, such as a dash. There is also low friction currently, which makes the game feel slippery. We also need to add more levels, and clean up the gameplay for each. We also currently have very few animations. Especially during state transitions, such as after dying and moving between levels, we would like to have some animations. We also need to add in our observer pattern.

In terms of design changes, we have decided to discard the strategy pattern, as the jump behaviors aren't complicated enough to necessitate the design pattern. We instead will simply add the jump styles in as another subclass of our command pattern. We will instead switch in the memento pattern for our save system. We have also decided not to make our own collision manager, gravity manager. We will instead just use Unity's built in physics engine. We have decided that our project will not be complicated enough to justify creating our own managers and controllers.

By 12/7, we plan to flesh out the game much more. For completely new functionality, we need to add in the save/load system, pause menu, and quit option. Other than that, most of our goals deal with improvements on elements we already have. For example, we must add in a tileset to improve the appearance of levels and game. We need to add in additional levels, including the final level where Gary will finally find Felix. We also need to improve the movement functionality. We also need to complete an observer pattern for bug fixing issues and console logging.