CSCI - 4448 OOAD

# Project 6

*KYRAN BUTLER | CONNOR ELY*

*What is the Project Called?*

- The Project is Called “Definitely Not Minecraft”

*Who is on your team?*

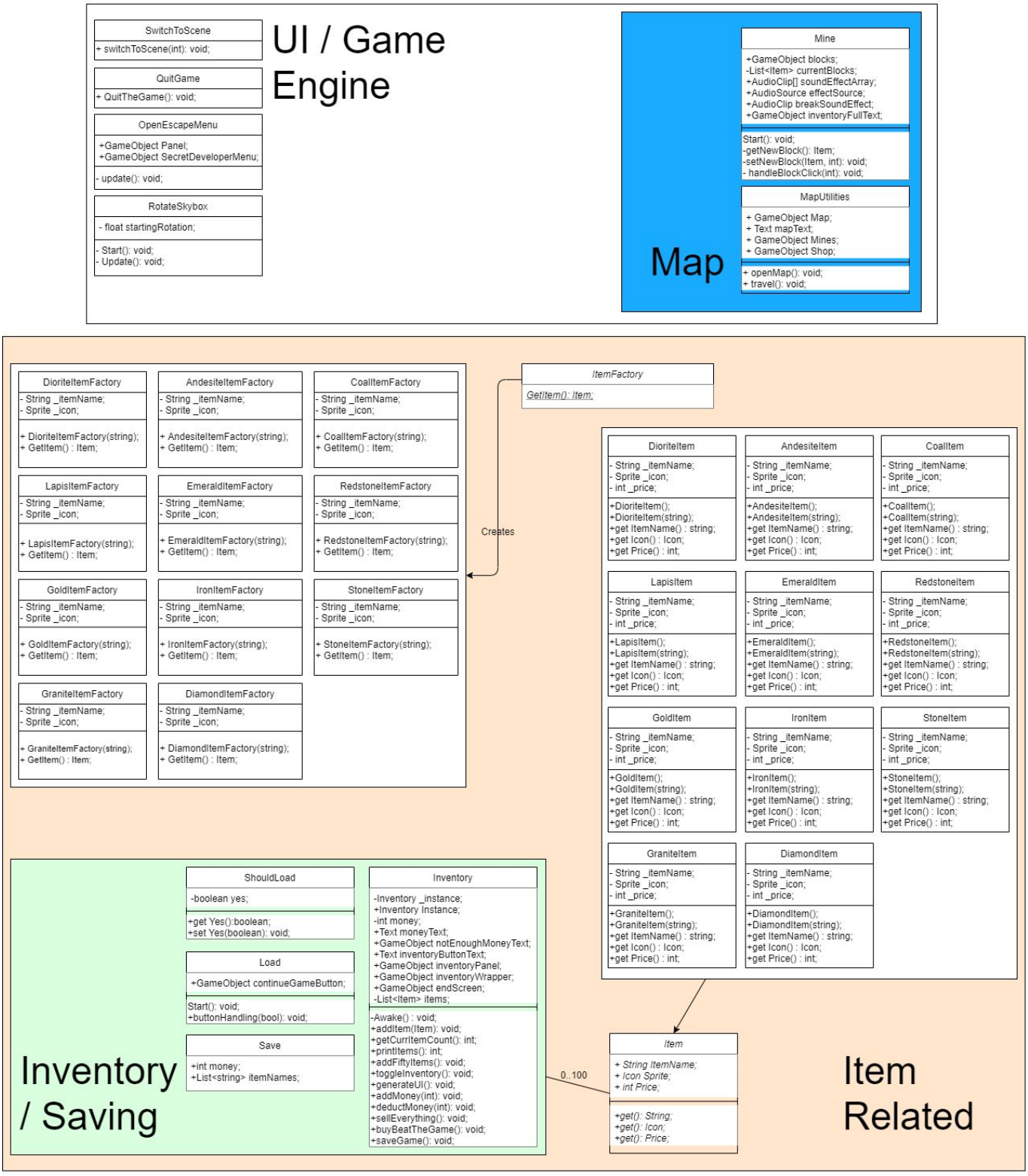
- Kyran Butler, Connor Ely, ~~and Lily the Foxgirl~~

*Final State of System Statement*

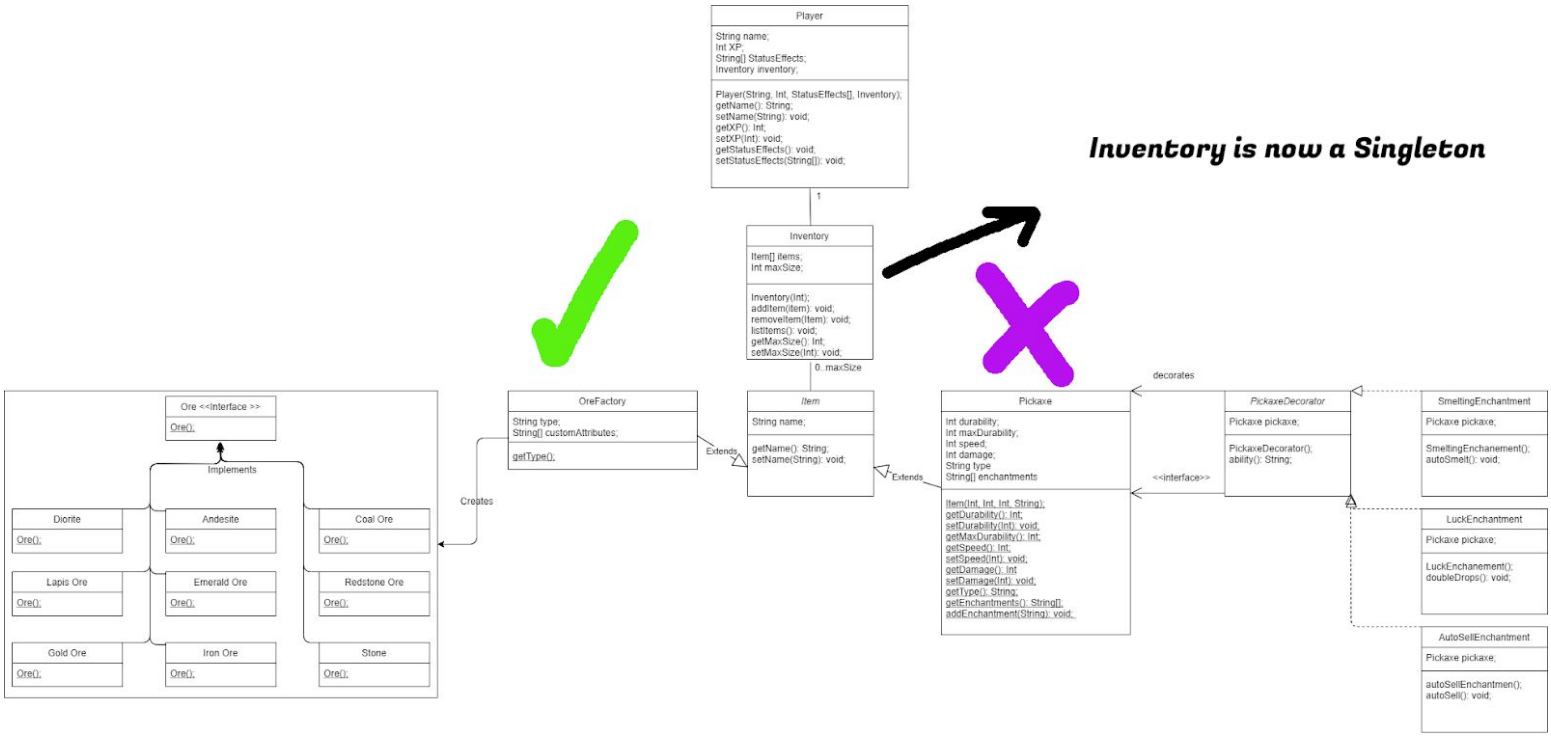
- First off, we had a ton of fun making this. Giving us the freedom to explore infinitely different ways to accomplish tasks and add features really helped more than other classes to open our eyes to what the world of computer science has to offer. We both got to learn how to use a game engine to put together a cohesive game, even if it’s a little rough around the edges!
- Now, feature wise, we have most everything in that we wanted to - only barring a few features that we deemed were needlessly complicated.
    - We implemented / created:
        - A standalone game that can be run on any Windows machine (and, theoretically mobile / macOS / other systems!)
        - A main menu with a rotating skybox and options to create a new game or work off an existing save
        - The ability to save and load your progress!
        - A multiple scene game where the player navigates by using a map
        - A randomly generated set of blocks (Factory Pattern) that the player can collect into their inventory (Singleton Pattern)
        - A simple shop, where the player can buy and sell
    - We *did not* implement
        - Multiple different types of pickaxes, with different mining speeds (the base speed is applied to all blocks)
        - Experience Points (XP), money is a measure of the progress, and XP wouldn’t have a value for the features currently available
        - Statuses, so that the user would mine faster or get double drops for a specified amount of time. Again, more complicated than needed.
- Essentially, anything *not* implemented was thought to be implemented up to this point, therefore those are the differences from the last Project check-in (5).

*Final Class Diagram and Comparison Statement*

(newest diagram)



(before diagram)




(explanation)

- We used a Singleton pattern for the Inventory - allowing us to easily have the inventory persist between scenes in Unity and to confirm we only have one inventory.
- We used a Factory pattern for the block generation, allowing us to easily have different attributes per item and allow us to dynamically create them at runtime.
- We used Unity as a framework, which explains some of the weird classes like GameObject or Icon.
- Key changes between this submission and the last project submission include the removal of the Pickaxe (which would have served as a decorator), and the swap of an Inventory from no real OOP to a Singleton.

*Third-Party code vs. Original code Statement*

- We used Unity 2018.4.20f1. This can be downloaded from Unity's website - https://unity.com/
- We used this for reference for the Factory - https://www.c-sharpcorner.com/article/factory-method-design-pattern-in-c-sharp/
- All of the other code is self made, referenced from the Unity documents - https://docs.unity3d.com/Manual/index.html
- Assets, such as sounds, icons, and fonts are almost *entirely* ripped from Minecraft - https://www.minecraft.net/en-us

*Statement on the OOAD process for your overall Semester Project*

- One of the most useful elements of the process was deciding to use a Singleton. This was a great idea, as we had been able to successfully debug that there were multiple Inventories being created, something that now cannot be done with a Singleton.
- A problem that we encountered was the creation and maintenance of the Factory classes. We had developed (quickly) a method of creating different Items for our user to attain, however we wanted to use another OO method. So, we decided to use a Factory pattern - which resulted in a *lot* of code duplication, and any time we wanted to create a new Item, it took way longer than our previous faster period. Whether or not this means we did something wrong in the creation of the ItemFactory, it was not a pleasant experience (although the factory does work correctly!).
- Lastly was the file structure. We had not worked with Unity before, and to be honest we had not even written in C#, the language you use with Unity! So, we had no idea what a proper file structure was for this project. We began doing our best, grouping like items into different folders. However, as time progressed we constantly refactored our file structure, and in the end we believe it's clear and makes sense where every class is in relation to each other.

*Github Repo*

- Unsure if this is supposed to be here but, here it is! - https://github.com/KyButler/totally-not-minecraft

Thank you for the semester.