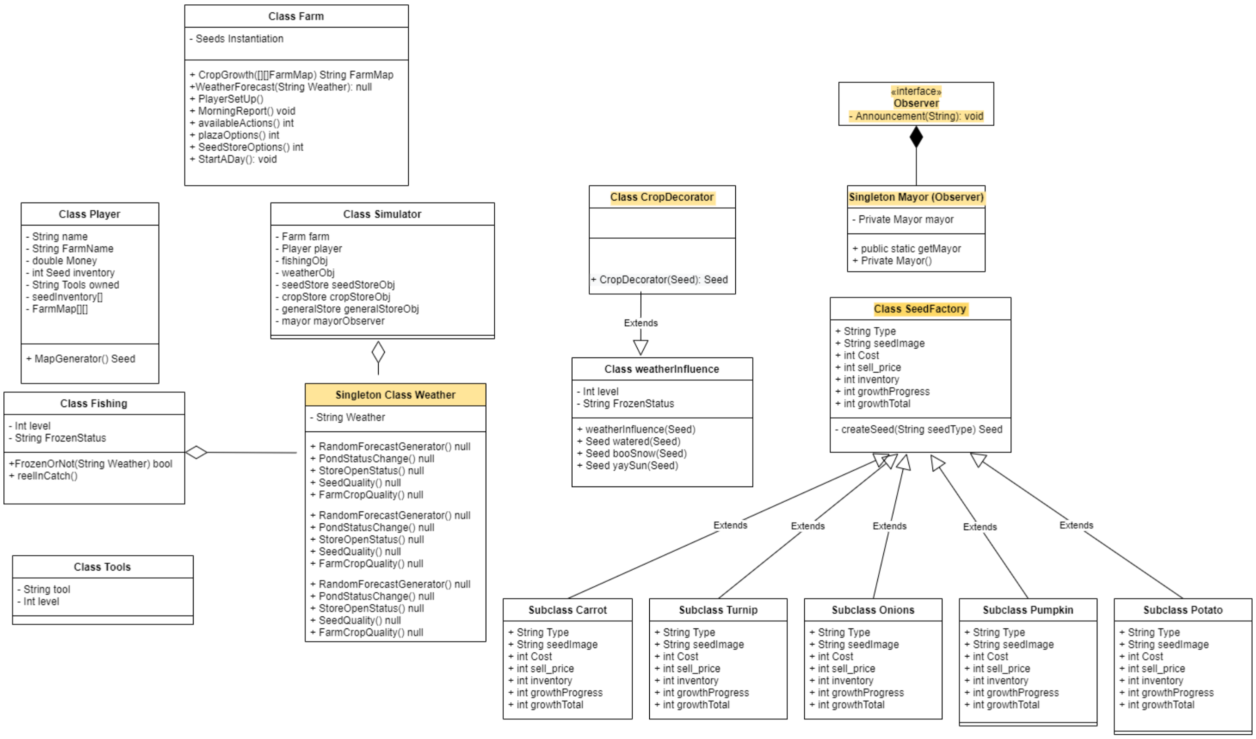
Helen Kim, Shawn Jaffe, & Lakshya Jaishankar
Fall 2021 CSCI 4448
Project 7 Deliverables

Stardew Horizon

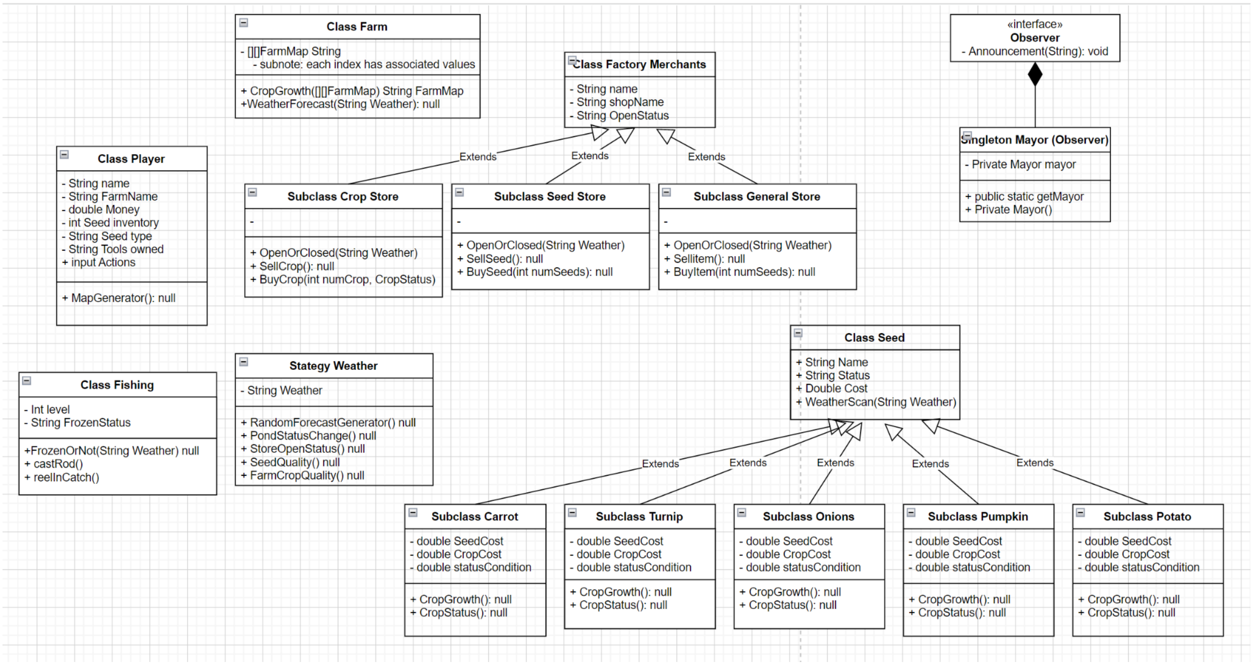
Final State of System Statement

The goal of our project is to simulate a text-based version of the popular Stardew Valley game. Our simulation requires user input to play the game and utilizes various object-oriented analysis and design techniques such as singletons, factory, observer, and decorator patterns. Multiple features were implemented to make this game as complete as possible including a player's farm class which is generated and updated throughout the game based on the player's actions, a seed class for all the different crops a player could plant (factory), a class for keeping track of crop growth (decorator), a fishing class that allows the player to fish for food using different tools and earn money, a weather class that is only instantiated once per day (singleton), and a mayor that makes a morning announcement (observer). One feature that was not implemented was having our weather class set up as a strategy pattern. This was difficult to implement as the methods and subclasses we would have needed to develop within weather were rather different from one another. We could not execute a behavior that only changed a little bit between the classes. Due to lack of time, another feature that ended up not being implemented was a store class that utilized the factory pattern. The player could purchase and/or sell crops and seeds at the store.There were some changes made from project 5 and 6 until now. Two new classes were added, a decorator class for crops and a class called weather influence which extends from the crop decorator. The crop decorator will allow specific crop objects to either continue growing once planted or halt progress based on the weather.

Final UML Diagram



UML Diagram from Project 5



As shown, there were some changes made from project 5's UML diagram to our current version. For instance, some of our patterns changed. Instead of implementing a strategy pattern for the weather class, we utilized a singleton pattern. Two new classes are included in our current diagram: crop decorator and weather influence. We removed a store class and expanded upon our seed factory class. There were also some overall modifications to certain methods belonging to these classes to take into account those class changes.

Third party vs Original code

A lot of the implementations that are basic simulation and formatting of the code is original code. However, we did use Bruce's simulation and main file formatting in order to run our game similarly to how the game store was run. We did use a lot of syntax online for aspects such as the random number generator from sources such as geekforgeeks and javapoint. Much of the decorator code design, however, was largely sourced from the textbook "head first design patterns". Though it may not have been word for word the formatting and syntax was based on how the book implemented specific patterns. In this aspect, the use of how to go about it was original and the basic structure of how the pattern should look was based on the book.

Overall OOAD Process

Overall, our intentions of how the simulator would run went similar to how we originally designed it but how we went about it was different. Our usage of factory changed from project 5, which was a change for the better because rather than for store shops it made much more sense when applied to different kinds of seeds and creating temporary seed objects that were empty rather than null when required. Though when attempting to implement template for all the stores, it didn't turn out well if at all because we ended up abandoning it and not actually using it. It would have been ideal as the stores all have generally the same functionalities, with some methods being the same and others just having a variation of it. We couldn't work template within our project without having large modifications to what we already had when it came to all the loops with user inputs. One design process that was more of a middle ground of good and bad (but definitely something good when it came to our overall project implementation) was the changing of our weather class from strategy to singleton at the suggestion of a professor. We ended up using it more as a check when attributes were affected and it's string value was the only change from day to day. But having only one instance of it served well in restricting our declarations of it so that the weather patterns wouldn't change in one day for the sake of our simulation or constantly having the same weather by accident. It would have served better as a strategy had we implemented our overall simulation the way we had planned but with the changes we made singleton was a great change that was needed.