CSCI 4448 - Object Oriented Analysis and Design
Project 6- Final Report

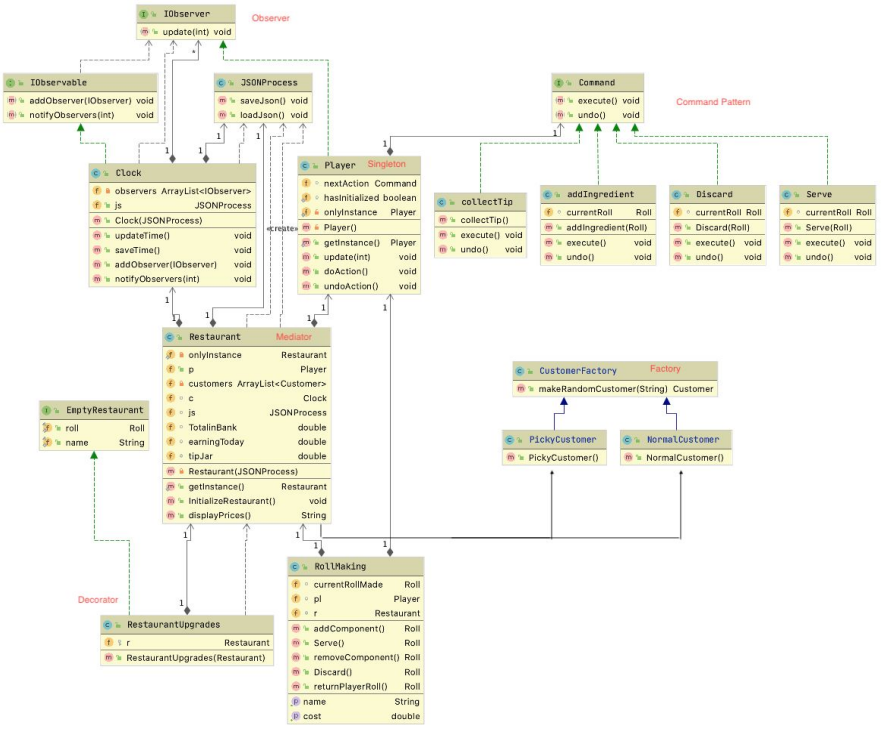
Restaurant Tycoon - A Unity Game

Team Members:
Andrea Chamorro
Luke Joyce
Richard Ortecho

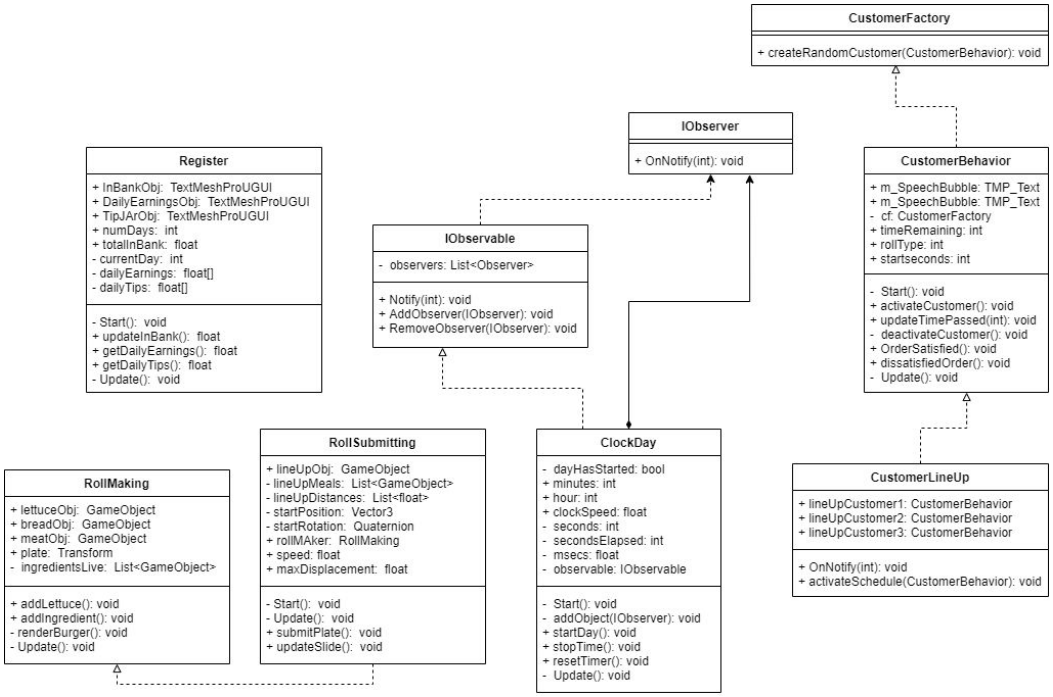
Final State of System:

Our final product includes the main functionalities of generating customers who visit our restaurant, make orders, and react to the completed order, as well as a cash register that tracks sales data. Our game-play interface allows for users to respond to customer's orders by clicking ingredients to create meals, clear a plate if they need to restart, and serve the order to the customer. Each customer arrives with a predetermined wait time. Once they appear with their order presented, a timer runs down and you must finish creating their order before the time runs out, or else they will be dissatisfied with your service and leave the restaurant. As stated in our demo video, if we had more time we would apply this feature of customers being dissatisfied to a restaurant "ratings & reviews" system that would give the game more of an objective.

Initial Class Diagram:



Final Class Diagram:



Changes between Initial vs Final OO Design:

We did not get around to implementing all parts of the system that were intended, and for that reason did not get to each of the design patterns we wanted to use. We still implemented the observer pattern, as we made the customer wait-time functionality a listener to the observable clock object. The behavior of two different kinds of customer was almost fully implemented using a strategy pattern. Customers were to have two main types - picky and normal. These types would be randomly generated using our Customer Factory (factory method). Overall, although we did not get to implementing every design pattern we had intended to have, our game was still pretty functional. Unity allows users to create objects that interact with each other without much background knowledge of object oriented design as it is extremely versatile and easy to use when you want to bring your idea to life (although there is a steep learning curve).

Third party Code vs Original Code Statement:

We stuck with using Unity Game Engine to bring our idea to life. Unity connects its "UnityGameObjects" to C# scripts. This is where all of our object oriented practices were applied. We did not get around to implementing the games ability to save and load games, which would have required us to keep data in a JSON file. If we were to add that functionality in, we imagine that saving the game would be pretty easy since, while the game itself has a lot of complex graphics features, the main features that define a user's current game state are just the daily earnings, daily sales, total earnings, and a few other things like customer lineup.

OOAD Process Statement:

1. **One key design process element we encountered was**
   The observer pattern proved to be the most useful design pattern in our game design, as it acted as a driver for customers to arrive, order and interact in the gameplay. Based on this interpretation, we can assume that the observer pattern is one of the more important patterns to know when using OOAD practices in the real world.

2. **One key design process challenge we encountered was**
   Specific to our project, it was difficult to figure out how to associate our design patterns with the GameObjects in Unity. GameObjects are their own entity in unity and do not necessarily relate to a c# script, so we had to reconsider our choices for patterns to use in certain cases.

3. **Another key design process challenge we encountered was**
   Not resorting to just making things work. It was definitely difficult to follow the object oriented design practices we were taught when we knew we could take the easy route and just make things work, or make "spaghetti code" without consideration for code weight, coupling and cohesion.