Name of Project: Virtual Game of Life
Team Members: Ailish Skinner and Stephen Lasko

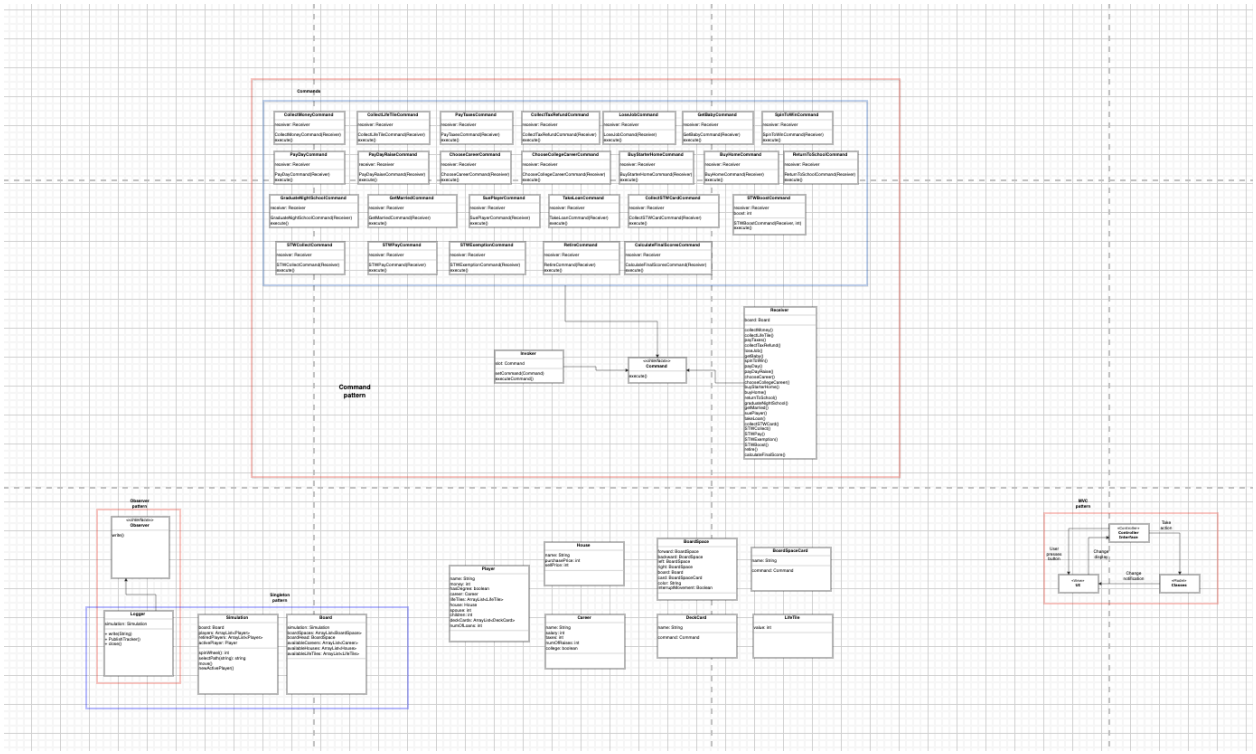
State of System Statement:

This game is heavily developed on the Command Pattern. The commands are used by the board space cards, deck cards, and others in order to trigger events. The game runs through the console in a text based system. The user is able to see the game play out and input their selections this way. Finally, the game's events are observed and stored in a text file for each round.

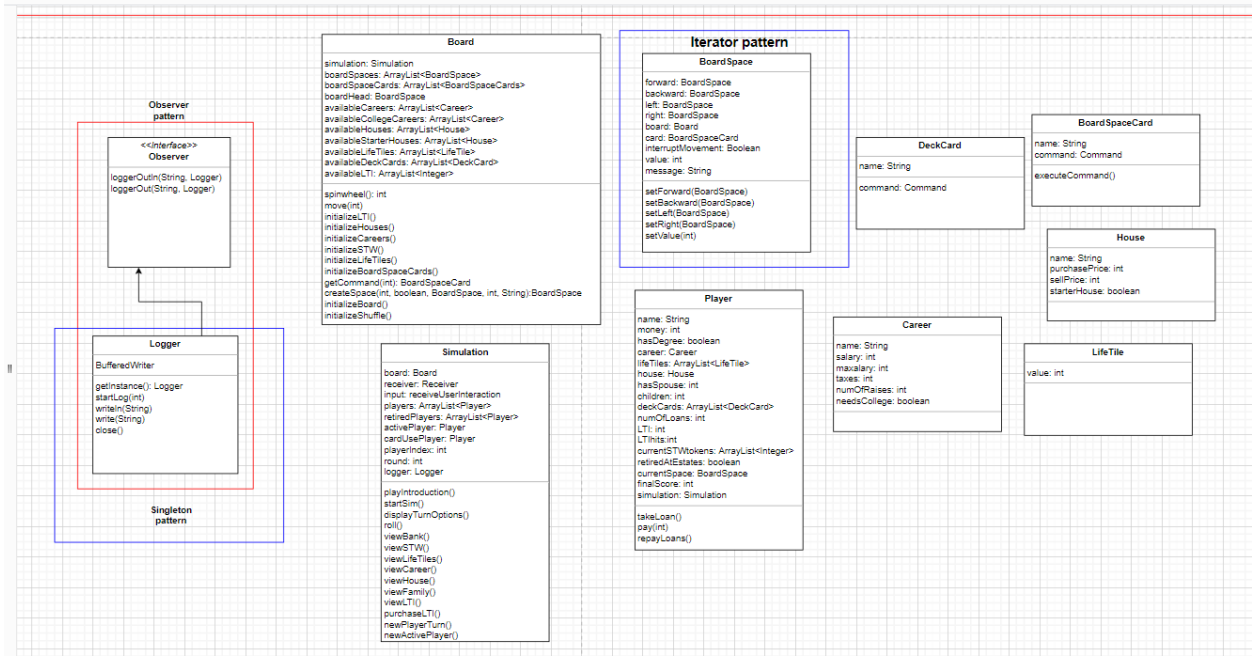
Final Class Diagram and Comparison Sheet:

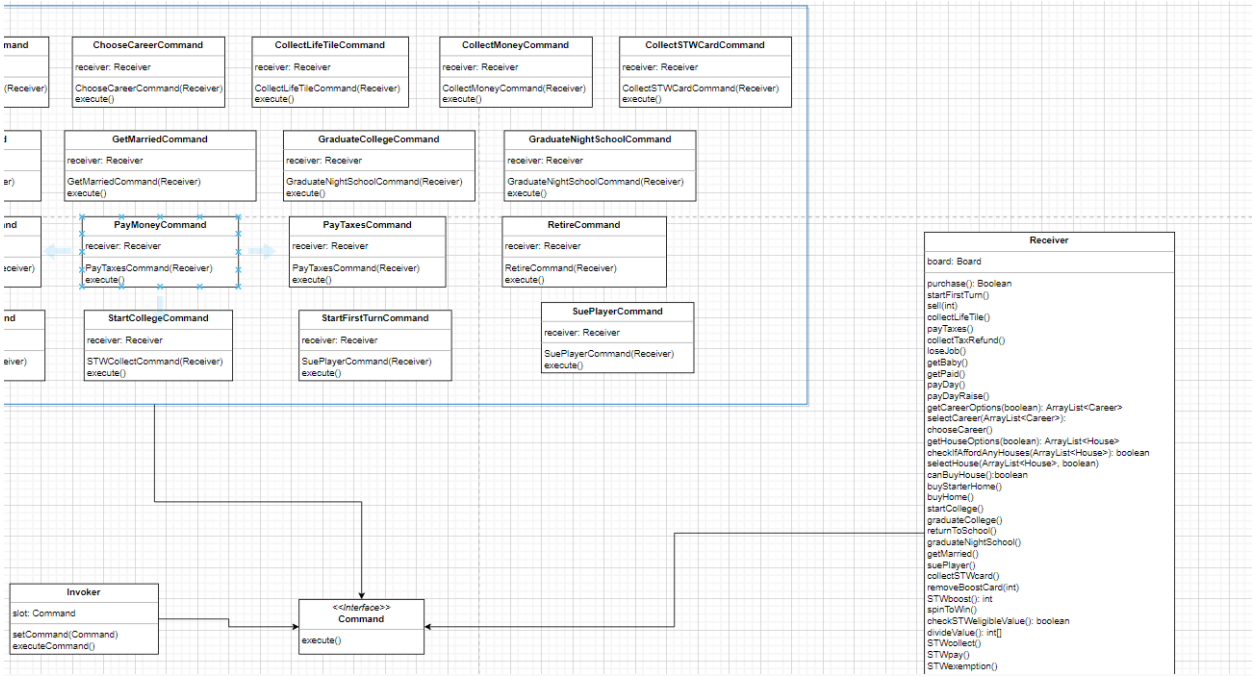
The only big change that happened from our presentation of Project 6 to now is that we implemented a console version of the MVC. The UML Diagram has been updated to note this change. The other three patterns: Observer, Singleton, and Command pattern have been successfully implemented from project 6. Additionally, the iterator pattern is used for the board as it is a linked list that is iterated through. Otherwise, most changes are functional additions.

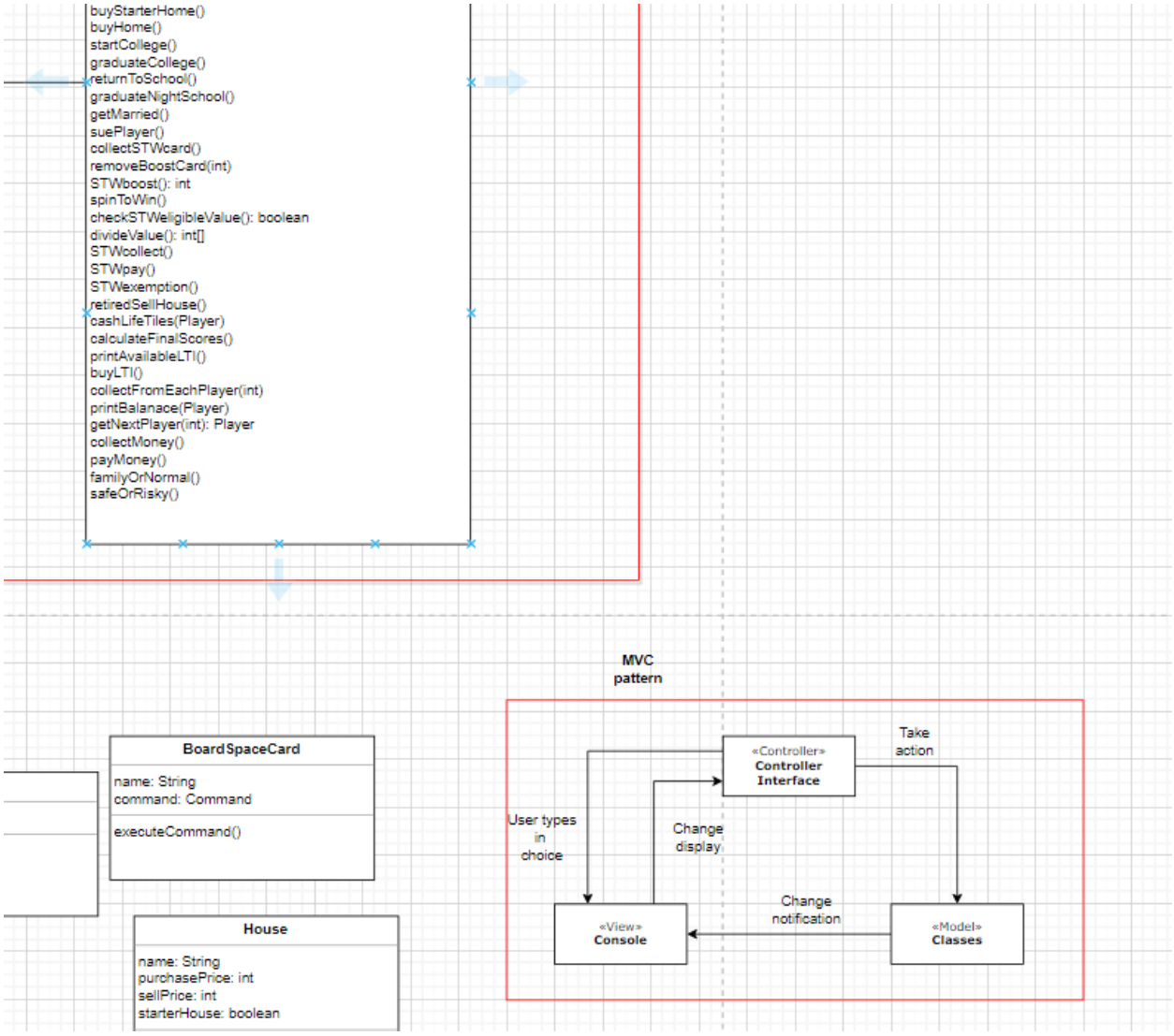
Project 5 UML:

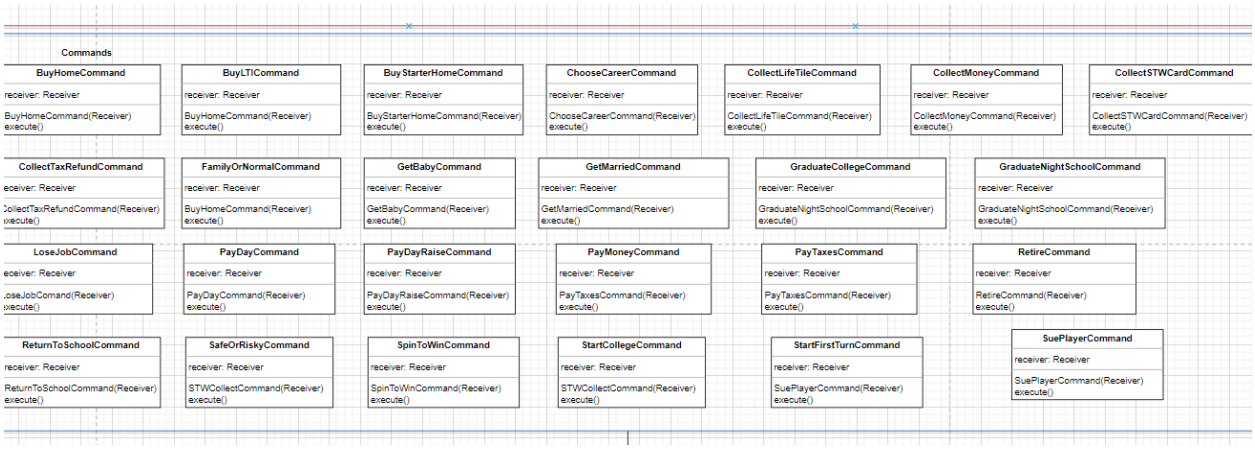


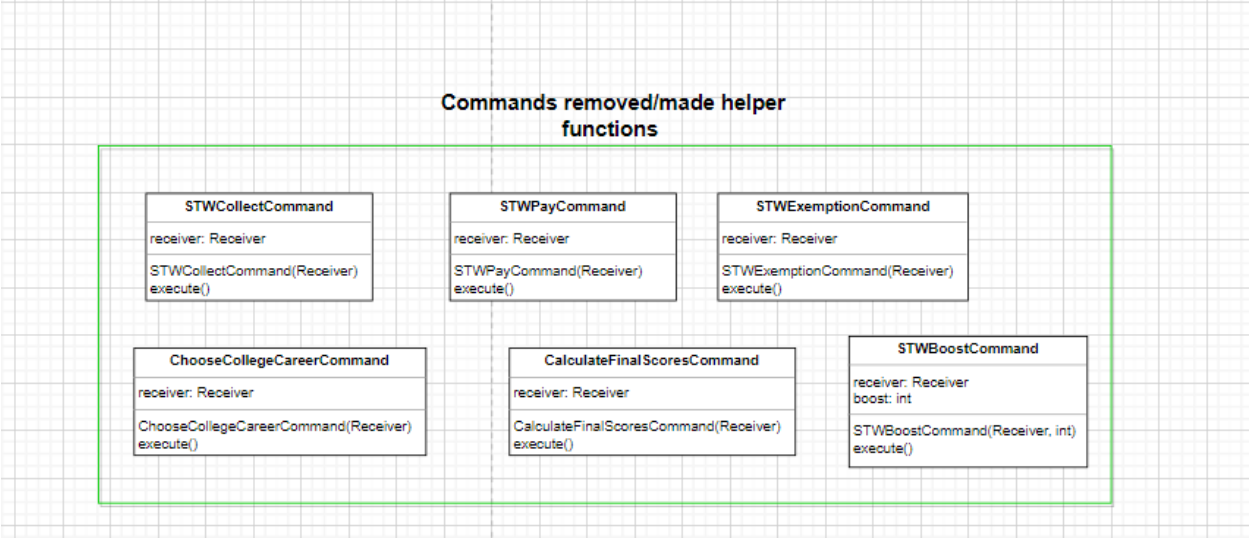
Project 7 Final UML: A lot has been changed…there were more functions added to the command pattern, the MVC pattern has been implemented. The classes are updated and new functions in the command pattern…as well as in the classes that are used for the game such as House and Player.






buyStarterHome()
buyHome()
startCollege()
graduateCollege()
returnToSchool()
graduateNightSchool()
getMarried()
suePlayer()
collectSTWcard()
removeBoostCard(int)
STWboost(): int
spinToWin()
checkSTWeligibleValue(): boolean
divideValue(): int[]
STWcollect()
STWpay()
STWexemption()
retiredSellHouse()
cashLifeTiles(Player)
calculateFinalScores()
printAvailableLTI()
buyLTI()
collectFromEachPlayer(int)
printBalanace(Player)
getNextPlayer(int): Player
collectMoney()
payMoney()
familyOrNormal()
safeOrRisky()
MVC pattern
«Controller»
Controller Interface
Take action
User types in choice
Change display
Change notification
«View»
Console
«Model»
Classes
BoardSpaceCard
name: String
command: Command
executeCommand()
House
name: String
purchasePrice: int
sellPrice: int
starterHouse: boolean


Commands
BuyHomeCommand
receiver: Receiver
BuyHomeCommand(Receiver)
execute()
BuyLTICommand
receiver: Receiver
BuyHomeCommand(Receiver)
execute()
BuyStarterHomeCommand
receiver: Receiver
BuyStarterHomeCommand(Receiver)
execute()
ChooseCareerCommand
receiver: Receiver
ChooseCareerCommand(Receiver)
execute()
CollectLifeTileCommand
receiver: Receiver
CollectLifeTileCommand(Receiver)
execute()
CollectMoneyCommand
receiver: Receiver
CollectMoneyCommand(Receiver)
execute()
CollectSTWCardCommand
receiver: Receiver
CollectSTWCardCommand(Receiver)
execute()
CollectTaxRefundCommand
FamilyOrNormalCommand
receiver: Receiver
BuyHomeCommand(Receiver)
execute()
GetBabyCommand
receiver: Receiver
GetBabyCommand(Receiver)
execute()
GetMarriedCommand
receiver: Receiver
GetMarriedCommand(Receiver)
execute()
GraduateCollegeCommand
receiver: Receiver
GraduateNightSchoolCommand(Receiver)
execute()
GraduateNightSchoolCommand
receiver: Receiver
GraduateNightSchoolCommand(Receiver)
execute()
LoseJobCommand
PayDayCommand
receiver: Receiver
PayDayCommand(Receiver)
execute()
PayDayRaiseCommand
receiver: Receiver
PayDayRaiseCommand(Receiver)
execute()
PayMoneyCommand
receiver: Receiver
PayTaxesCommand(Receiver)
execute()
PayTaxesCommand
receiver: Receiver
PayTaxesCommand(Receiver)
execute()
RetireCommand
receiver: Receiver
RetireCommand(Receiver)
execute()
ReturnToSchoolCommand
receiver: Receiver
ReturnToSchoolCommand(Receiver)
execute()
SafeOrRiskyCommand
receiver: Receiver
STWCollectCommand(Receiver)
execute()
SpinToWinCommand
receiver: Receiver
SpinToWinCommand(Receiver)
execute()
StartCollegeCommand
receiver: Receiver
STWCollectCommand(Receiver)
execute()
StartFirstTurnCommand
receiver: Receiver
SuePlayerCommand(Receiver)
execute()
SuePlayerCommand
receiver: Receiver
SuePlayerCommand(Receiver)
execute()



Third-Party code vs. Original code Statement:

There was no third party code used in this project. The code was based of what we learned how to do in class throughout the semester.

Statement on the OOAD process for your overall Semester Project:

1. Design around Controls: positive
   Designing around the controller pattern allowed this project to have virtually infinite executable functions with minimal integration work. This allowed for extreme robustness and allowed us to easily program the events of the board.

2. An interactable game: positive
   The game is controllable and viewable in the terminal. This allows the user to actually play the game.

3. Trackable results: positive
   The results of the game are able to be tracked and stored in a text file. This helps not only with testing but keeping track of the results of the games.