Jessica Ryan, Kate Pendavinji,
Ria Thakkar, Aanvi Guliani

# Project 7: Semester Project – Final submission

## Final Project Report

**Poker Game** by Jessica Ryan, Kate Pendavinji, Aanvi Guliani, and Ria Thakkar

### Final State of System Statement

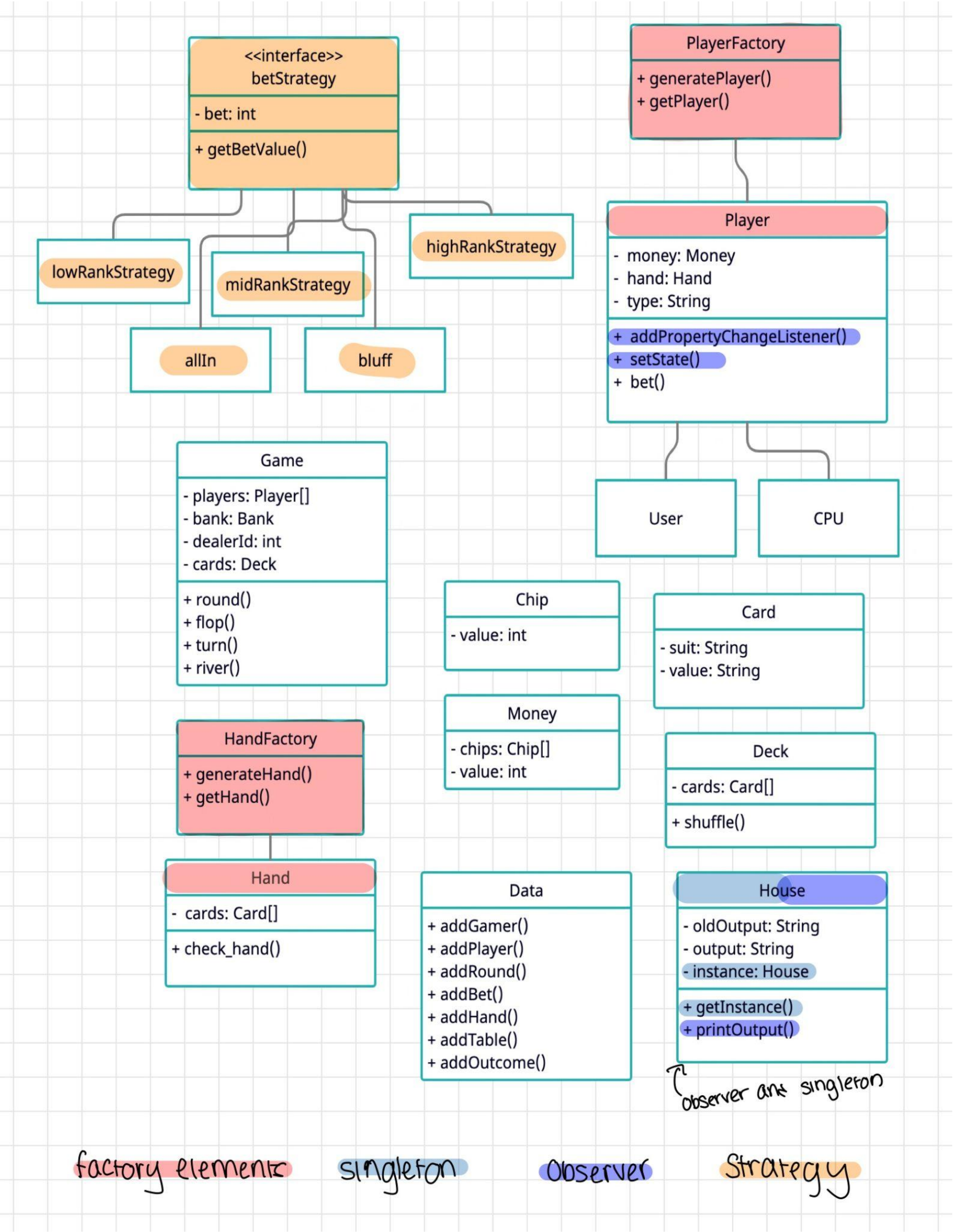
In the final state of our system we were able to implement all object oriented programming features: *strategy*, *factory*, *singleton*, and *observer* patterns. Our *strategy pattern* is probably our most prominent feature which handles all of the hand ranking strategies. These can be seen in our highRankStrategy, lowRankStrategy, and midRankStrategy files on our repository. It controls the different levels of CPU player strategies. Our *factory pattern* handles player settings and creates CPU players featured in our playerFactory file on our repository. Our *singleton pattern* is utilized in our Table file on our repository handling states in the table such as game phase, card dealing, and etc. *Observer pattern* can also be found in our Table file and functions as an announcer for game state updates. In addition to all of our patterns which we initially claimed in our statement document for Project 5, we were able to use the *model controller pattern*. Using Swift in xcode allows us to have instant modeling of what our application's user interface would look like and how it responds to our code/changes.

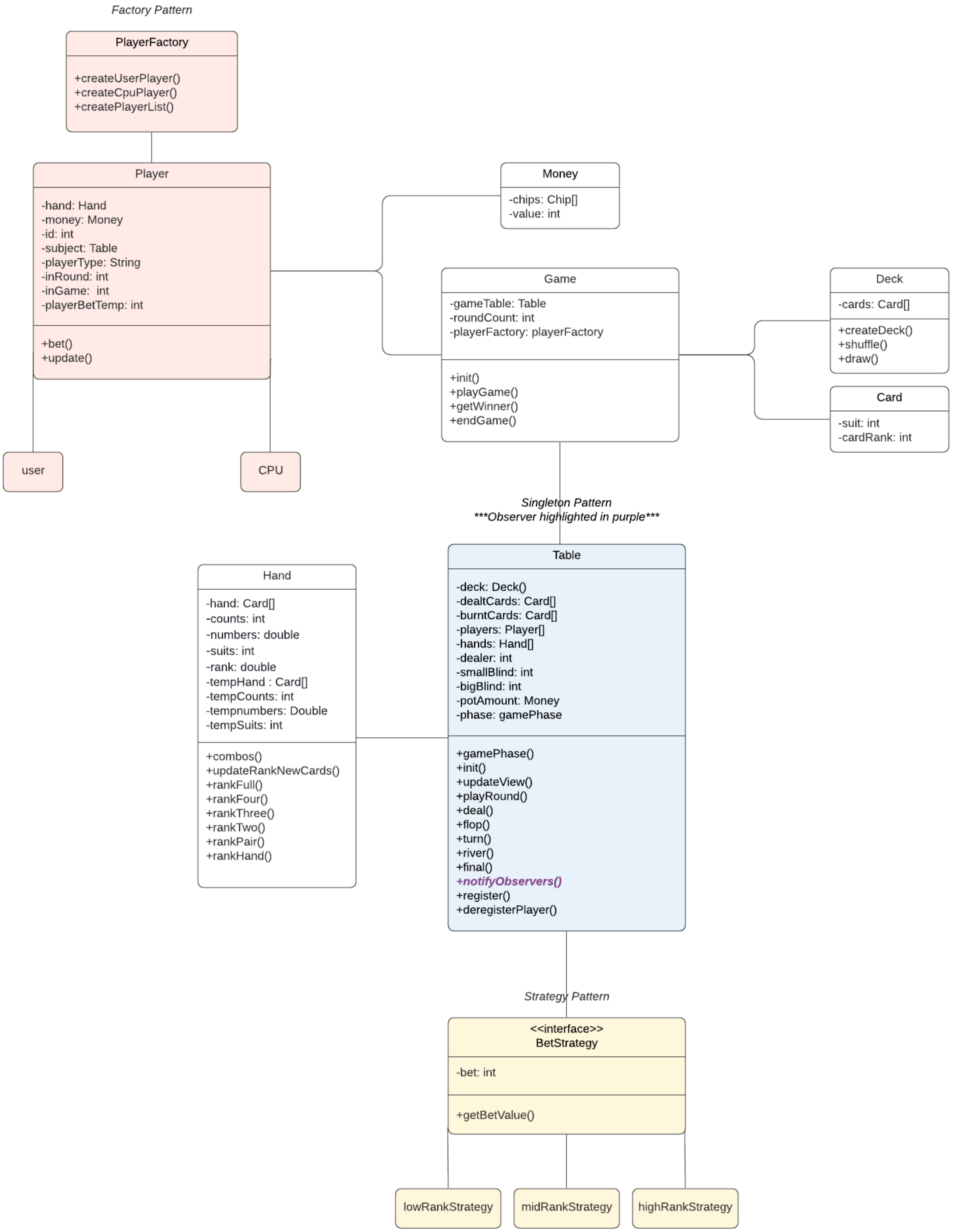
We were also able to implement a database using Sqlite using table creation, insertion, and reading to store items such as login information and game status. In addition to this we implemented a user interface and graphics in general which allowed us to see the functionality of our code. As the user plays the game and uses the application, their actions and data get stored for further use.

This is a contrast to Projects 5 and 6. In Project 5 we stated we were only going to be using four object oriented design features. Instead we ended up being able to add another due to the nature of our coding environment. In Project 6 we were only really able to fully implement about 1-2 design patterns. We were able to implement the full 5 by the final product of our poker application. We also beefed up the user interface *significantly.* For our demonstration of Project 6 we had an extremely bare bones model view of our application page consisting of a login page with no functionality as well as a gameplay screen which featured 2 cards and 3 functionless buttons. We now have access to real poker gameplay with sufficient cards per person and the ability to play against CPUs. You can now also log in and your data will be stored. The buttons are no longer functionless and tie into our code to support gameplay. Overall we went from a completely functionless application with object oriented design components to a functioning game utilizing said components.

### Final Class Diagram and Comparison Statement

Shown below is first our class diagram from Project 5 followed by our current class diagram (labeled in footers for further clarification). As you can see we updated a lot of our classes and their respective functions and variables. We removed the Data and House classes and replaced them with a Table class which holds similar functionality. Table maintains the singleton pattern as well as the observer pattern similar to the previous House class. It is also evident that we have added connections to our classes. Without functional code it was a little bit difficult to conceptualize how all of our classes would tie together. Overall hopefully you can see a more cohesive structure for our poker game.


<<interface>>
betStrategy
- bet: int
+ getBetValue()
lowRankStrategy
midRankStrategy
highRankStrategy
allIn
bluff
PlayerFactory
+ generatePlayer()
+ getPlayer()
Player
- money: Money
- hand: Hand
- type: String
+ addPropertyChangeListener()
+ setState()
+ bet()
User
CPU
Game
- players: Player[]
- bank: Bank
- dealerId: int
- cards: Deck
+ round()
+ flop()
+ turn()
+ river()
Chip
- value: int
Card
- suit: String
- value: String
Money
- chips: Chip[]
- value: int
Deck
- cards: Card[]
+ shuffle()
HandFactory
+ generateHand()
+ getHand()
Hand
- cards: Card[]
+ check_hand()
Data
+ addGamer()
+ addPlayer()
+ addRound()
+ addBet()
+ addHand()
+ addTable()
+ addOutcome()
House
- oldOutput: String
- output: String
- instance: House
+ getInstance()
+ printOutput()
observer and singleton
factory elements
singleton
Observer
Strategy



***Current Class Diagram***

## Third-Party Code vs. Original Code Statement

```
  /********************************************************************/
 / ****************Description of source listed above of source link***************/
/********************************************************************/
```

One of our third-party-sources was the following website in order to figure out the logic of how to rank our poker hands. This is extensively used in our Table file featured on our repository which basically sorts out game flow logic for poker.

https://towardsdatascience.com/poker-with-python-how-to-score-all-hands-in-texas-holdem-6fd750ef73d

Another third-party-source utilized is listed below. This was used in our Hand file in order to aid best hand calculations.

https://newbedev.com/apple-swift-generate-combinations-with-repetition

This third-party source utilized is integrating Sqlite with swift. It was used to figure out how to get our login page to actually be able to read and store the data entered, although the example has a slightly different utilization as shown in DBHelper in our repository.

http://www.wepstech.com/sqlite-in-ios-with-swift-5/

We utilized this source in order to get our user interface working. It helped us figure out how to resize imaging in order to get our asset cards to fit a phone screen properly. Can be seen in the ContentView file in our repository.

//https://www.advancedswift.com/resize-uiimage-no-stretching-swift/

## Statement on the OOAD Process for your Overall Semester Project

- List three key design process elements or issues (positive or negative) that your team experienced in your analysis and design of the OO semester project

1. A key design process that our team experienced is integrating object oriented concepts which we mainly had practice in Java, to an entirely different code environment and language. It was incredibly challenging maintaining functionality and meaningfulness in our code while also trying to maintain a functional user interface.
2. An element of strength in working together was that everyone did as much as they were able to do. We all had very different backgrounds and lives outside of this class, so it was very important that everyone took on only what was necessary in order to maintain balance.
3. A final element we maintained throughout the semester project was technical difficulties and overall remote work. We were able to troubleshoot all technical issues remotely quickly and efficiently.

## Code Submission – GitHub Repository URL with Complete Semester Project System

https://github.com/jess-ryan/poker.git

Demonstration – Recorded demonstration video – 30 Points

https://youtu.be/Qw-VIfh0ExI