Section 1 – Introduction

Project Summary:

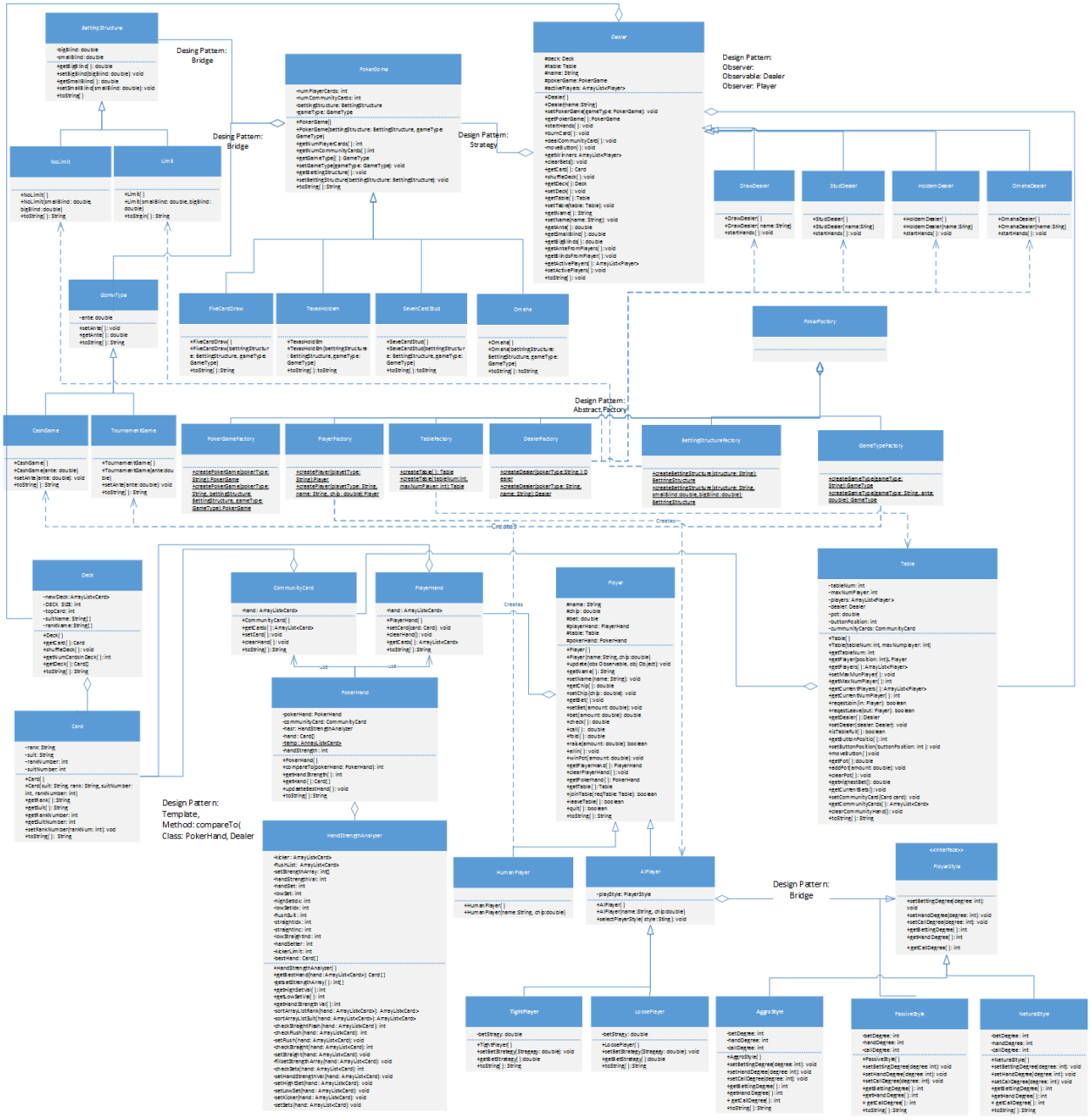
We have designed a poker game. We developed a program to compute the relative strength of poker hands based on a set number of players and following the rules of various poker game variations. This program will generate a complete deck of cards by suit and face-value as well as track the hands of each "player" in the game. We wanted to ensure this program is extensible so that additional game variations can be added and scalable by allowing more players to be added to the game. We also wanted to make sure that betting and AI players could eventually be programmed into the program.

Team Members:

Member Name	Email	Responsibilities
Johnny Hsu	johnnyhsu1106@email.arizona.edu	-updating diagram -writing code (related to dealer and ab. factory)
Justin LeBreck	jlebreck@email.arizona.edu	-writing code (related to players and table) -debugging
Joshua Ziegler	joshuaz@email.arizona.edu	-writing code (relating to card/deck, hands, and game types bridge) -debugging
Michelle Ziegler	mvziegler22@email.arizona.edu	-contact person -writing code (misc and help on hand strength) -compile report

Section 2 – System Design

Class Diagram:



Design Patterns:

1. Bridge (2): We used the bridge pattern when to structure poker tables as either tournament or cash tables and to structure the betting pattern for each table as either no limit or limit poker. We also used a bridge to ensure that AI players can be set up with betting tolerance of loose or tight and with a style of aggressive, neutral, or passive.

2. Strategy (several): We use multiple strategies throughout the design. We have a poker game strategy that chooses between Texas Hold 'Em, 7 Card Stud, 5 Card Draw, and Omaha. We employ strategy to create players as either Human or AI. We use strategy for a list of dealers that corresponds to the list of poker games.

3. Observer (1): The implementation of an observer pattern will notify the Player class, after they join a Table, that the Dealer has dealt a card to either the player's hand or the community hand. The Player class' notify method will call PokerHand's updateBestHand() method to create the player's best hand based on their hand and the community's hand.

4. Abstract Factory (1): The abstract factory will bundle Table, Dealer, Player, BettingStructure, and GameType objects when a poker game instance is created.

Key Features:

The program will deal out cards, compare player hands, and designate a winner for a selected poker game variation. We have set up some basic rules relating to the number of cards given to and shared by players for a select variety of poker games variations. This can be extended to include other game types.

Currently, betting and AI players are not programmed into the code, but we have the structure to ensured that there is space for the required algorithms to be easily inserted.

Section 3 – System Implementation

Main Use Case:

To start a game, the game variation (i.e. Texas Holdem or Omaha), game type (i.e. cash or tournament), betting structure (i.e. 2/4 Limit or 3/6 No Limit), and number of players (1-10) must be set. (See the figure below)

```
// ===================================== Initailize the Game ======================================
// ================================================================================================

//  ================================== Create the all objects ====================================

 // Parameters for Texas Holdem, cash Game(no ante), No Limit
 String pokerType = "Holdem";
 String gameType = "Cash";
 String structure = "NoLimit";
 double ante = 0; // cash game has no ante.
 double smallBlind = 1;
 double bigBlind = 2;
 double chip = 200;

 // Parameters for table
 int tableNum = 1;
 int maxNumPlayer = 9;
 double amount = 0;

 // Parameters for dealer
 String dealerName = "Daniel";

 // Parameters for players
 String playerType = "Human";
 String playerName1 = "Johnny";
 String playerName2 = "Josh";
 String playerName3 = "Justin";
 String playerName4 = "Michelle";
```

In the current program, game variation is the only part of this game set-up that will influence the way the program runs. This selection determines how many personal cards each player will receive (player hand) and how many cards will be shared by all players (community cards). (See the figure below from game printout)

```
This is a Texas Hold 'Em game.
Each player gets 2 card(s) and there are 5 community card(s).
Game Type: Cash Game with $0.0 ante.
Betting Structure: "No-Limit" Table with Big Blind: 2.0 - Small Blind: 1.0
```

A deck is created, shuffled, and then dealt to players and to the table in accordance with the selected game variation rules. Bets are also placed and tracked. Below is an example of a Texas Hold ‘Em game. Each round is played in sequence with the initial 2 cards dealt to each player and additional cards being dealt and displayed according to the rules of the game. (See the figures below for each of the 4 dealing phases)

```
Start Round 1!

Betting small and big blinds:
Josh: Betting - $1.0
Justin: Betting - $2.0

Initial Player Standings:
Player: Johnny
Chip Amount: 200.0
Cards: 3C, KH

Player: Josh
Chip Amount: 199.0
Cards: AH, 4D

Player: Justin
Chip Amount: 198.0
Cards: 7D, 8C

Player: Michelle
Chip Amount: 200.0
Cards: KC, 4C
```

```
============= At Flop Stage =============
The first 3 cards on table are:
2D        4H        9C

player 2's action: bet
Josh: Betting - $10.0
player 3's action: call
player 4's action: call
player 1's action: fold

Current Bets_____________________________
Johnny: Bet - $0.0
Josh: Bet - $11.0
Justin: Bet - $11.0
Michelle: Bet - $11.0

The pot size is: $35.0
```

```
============= At Turn Stage =============
The first 4 cards on table are:
2D        4H        9C        10H

player 2's action: check
player 3's action: bet
Justin: Betting - $30.0
player 4's action: call
player 2's action: fold

Current Bets_____________________________
Johnny: Bet - $0.0
Josh: Bet - $0.0
Justin: Bet - $41.0
Michelle: Bet - $41.0

The pot size is: $95.0
```

```
============= At River Stage =============
The total 5 cards on table are:
2D        4H        9C        10H       5H

player 3's action: bet
Justin: Betting - $60.0
player 4's action: call

Current Bets_____________________________
Johnny: Bet - $0.0
Josh: Bet - $0.0
Justin: Bet - $101.0
Michelle: Bet - $101.0

The pot size is: $215.0
```

For each player, their personal cards and the community card are combined and the best 5-card hand is created. The final hands are then compared for all active players to determine the final winner. The pot is then paid to the winner(s). (See the figure below)

```
Justin's best hand is 10H, 9C, 8C, 7D, 5H.
Michelle's best hand is 4H, 4C, KC, 10H, 9C.

Our winner is: Michelle!

Final Player Standings:

Player: Johnny
Chip Amount: 198.0
Cards: No Cards

Player: Josh
Chip Amount: 189.0
Cards: No Cards

Player: Justin
Chip Amount: 99.0
Cards: No Cards

Player: Michelle
Chip Amount: 314.0
Cards: No Cards
```

Section 4 – Lessons Learned

Key Technical Challenges:

We had a lot of classes and they all had to work together. We divided up the classes so that we could all work on the code independently. In order to make sure that the code could work together, we used Github. Some of us were new to this website, so we ran into some technical difficulties with overwriting each other's code and making sure people always uploaded the most current versions of their independent pieces. However, the most difficult part was making sure that all the pieces fit together into a cohesive working program at the end. We tended to approach the coding in slightly different ways, so that made it difficult to make all the pieces fit together.

Helpful Concepts:

In order to mitigate the problems with combining pieces of the code, we used our original design diagram to break the code into pieces that were most closely related. This allowed each of us to work independently on class groups. The use of design patterns helped us in this process by creating convenient "packages" of code that could be distributed. We also found it helpful to have frequent (weekly) group meetings to discuss ways that the classes would ultimately work together.

Main Take-Aways:

Our main take-aways from this project are:

- Scope Control – We found that we had to keep a close eye on project scope to prevent it from getting out of hand. In the end, we still ended up with a larger project then we had intended to take on in the beginning.
- Iteration – We found that it was important to be flexible and to make sure that we could make changes to sections of our design at every step of the process whenever we realized that there could be a better way to approach problem areas as they became evident.
- Communication – We found that one of the most important aspects of the project was to physically be in the same room to discuss issues and code structure. We found it helpful to apply extreme programming. The use of Github was also a vital part of the cooperative process.
- Design Patterns – We found that having a shared understanding of specific design pattern structure often helped us to understand the way that our codes would ultimately work together. On the other hand, we also found that, as the

project developed, some of the design patterns we had originally planned to use needed to be modified.