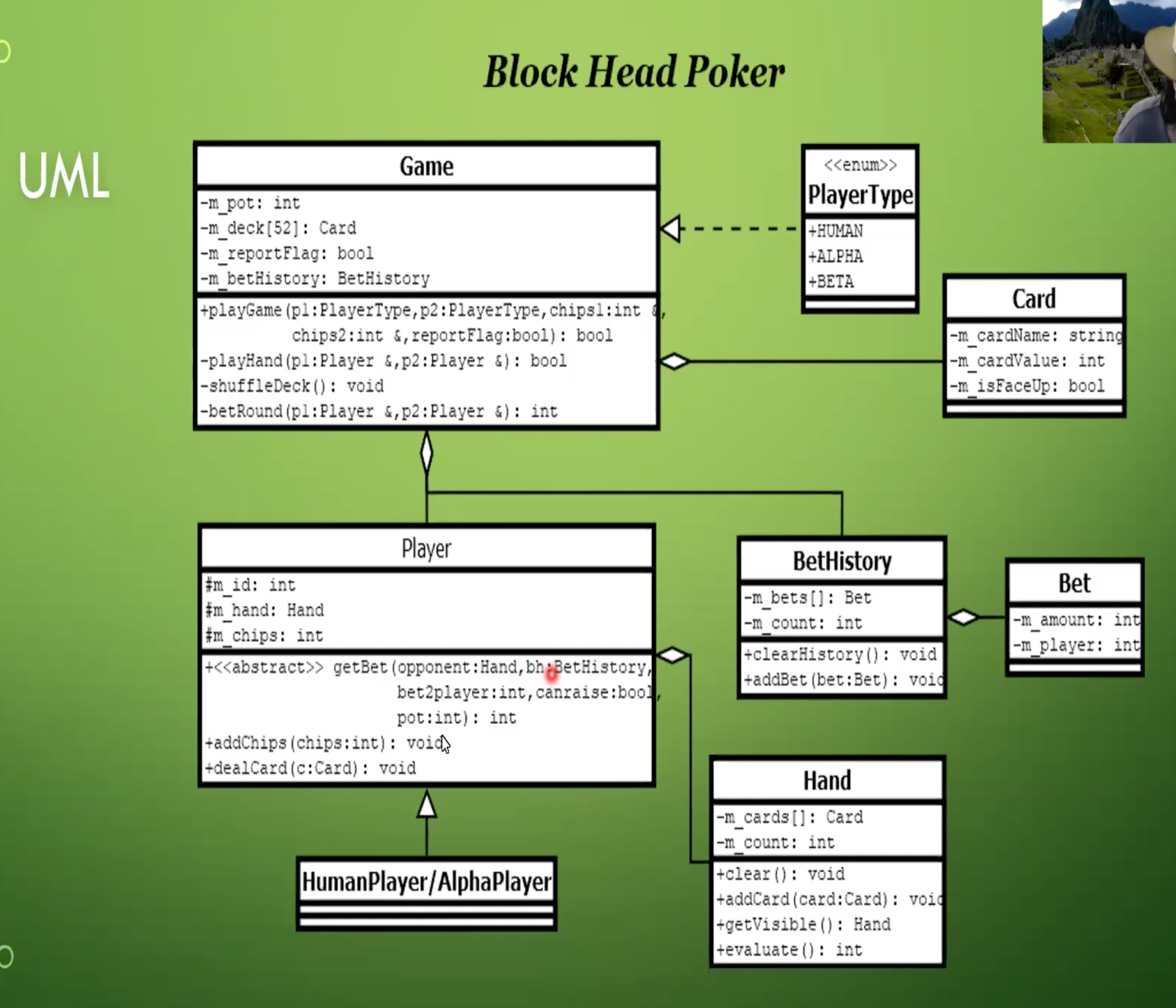
**Blockhead Poker**

**Phase 1**

**Implementation:**

****

**Differences from the UML presented in the lecture:**

While implementing the game, I have used the same UML structure that is presented to us in the lecture because the UML successfully divides the tasks into different classes and methods. I needed to make some changes to ease the work I need to do. For example, it is cumbersome to implement dynamically growing arrays by myself therefore I used STL vectors. Also, I have added a class called Util which has all helper functions and constants inside.

**Flow of the program:**

I used a command line interface instead of a graphical interface. The program has a main method as a driver class that has a Game object to be able to play the game and a Util object to be able to initialize the chips with the constant values presented in the project description. Then, player types are asked to the users. Currently, since Beta player is not implemented, the user only has these choices: HH (Human vs Human), HA (Human vs Alpha), AA (Alpha vs Alpha), QUIT. With respect to the input value, game.playGame(PlayerType p0, PlayerType p1, int &chips0, int &chips1, bool reportFlag) is called.

This function creates 2 players with respect to the PlayerType values. Then it makes the initial setup and calls playHand(Player &p0, Player&p1) for 10 times and calls playHand(Player &p1, Player&p0) for 10 times since a game consists of 20 hands. The change in the order is important since it changes the player who makes the first bet. The playHand method creates and shuffles the deck. It deals 3 cards to each player, and gets the initial buy-in chips from each player. There are at most 3 betting rounds in each hand.

Betting rounds are handled inside betRound(Player &p0, Player &p1) which returns an integer value with respect to the betting round result. betRound(Player &p0, Player &p1) returns -1 to mean both players are still in the game, 0 or 1 indicates player that won (opponent folded), -2 to indicate human quit game.

**Test Plan:**

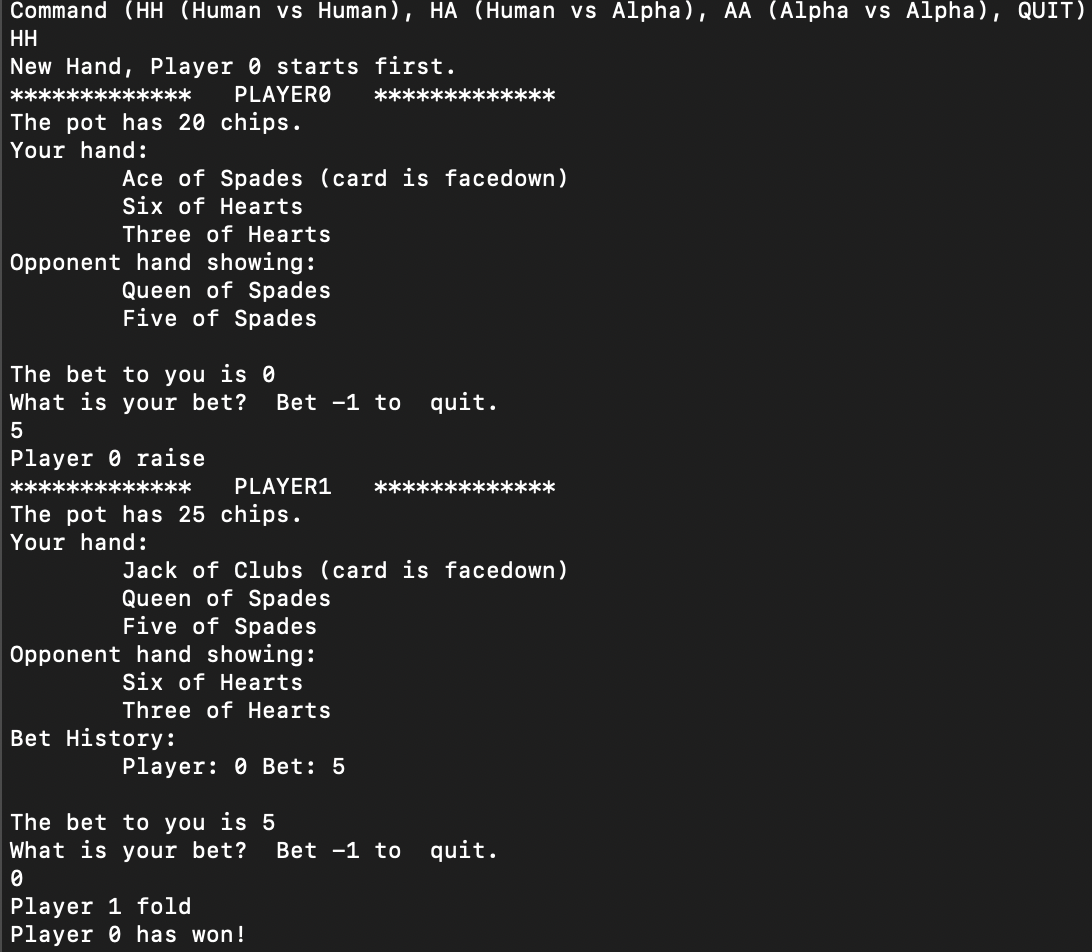
**Test 1: Human – Human**

**Input:**

* Human1 bets 5
* Human2 bets 0

**Output:**

* Human2 folds.



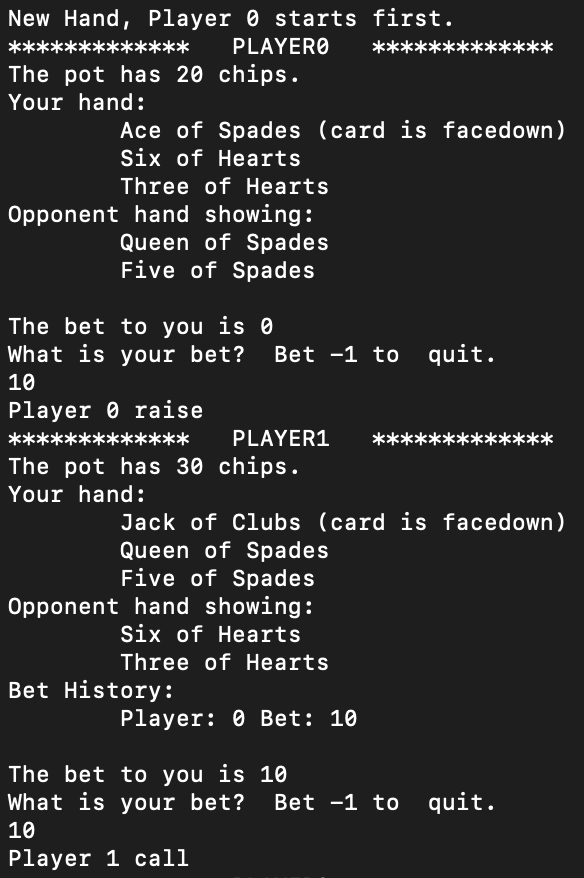
**Test 2: Human – Human**

**Input:**

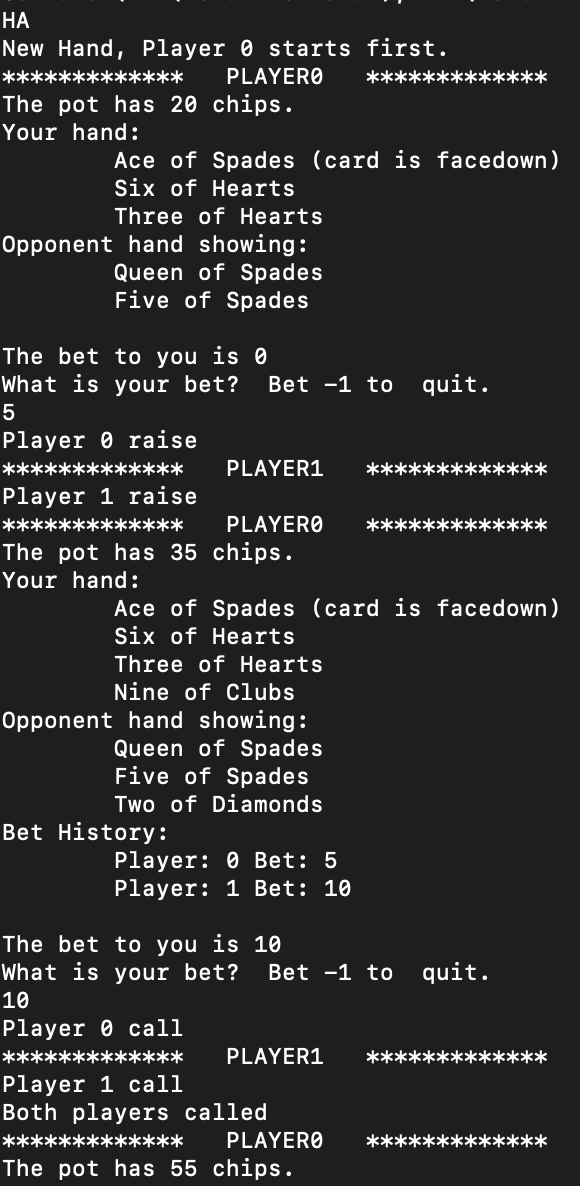
* Human1 bets 10
* Human2 bets 10
* Human1 bets 10

**Output:**

* Both players called



**Test 3: Human – Alpha**

****