Texas Hold’Em Group 03

79109P Antti Ahonen antti.ahonen@aalto.fi

79874V Mikko Salonen mikko.salonen@aalto.fi

58122B Mikko Närjänen mikko.narjanen@aalto.fi

Specification of requirements

In the project we will implement atleast the following minimum requirements:

- A text-based user-interface for single player games.
- A working implementation of the Texas hold'em game logic.

We will implement the limit version of the game at first.

- A very simple AI that is at least marginally better than a random AI.

The number of players (1 human) is selectable up to six.

- The AI is not allowed to cheat (use the knowledge of the players hidden cards or the composition of the deck).
- Some kind of a wager system using some form of currency.

We will also implement the following optional requirements:

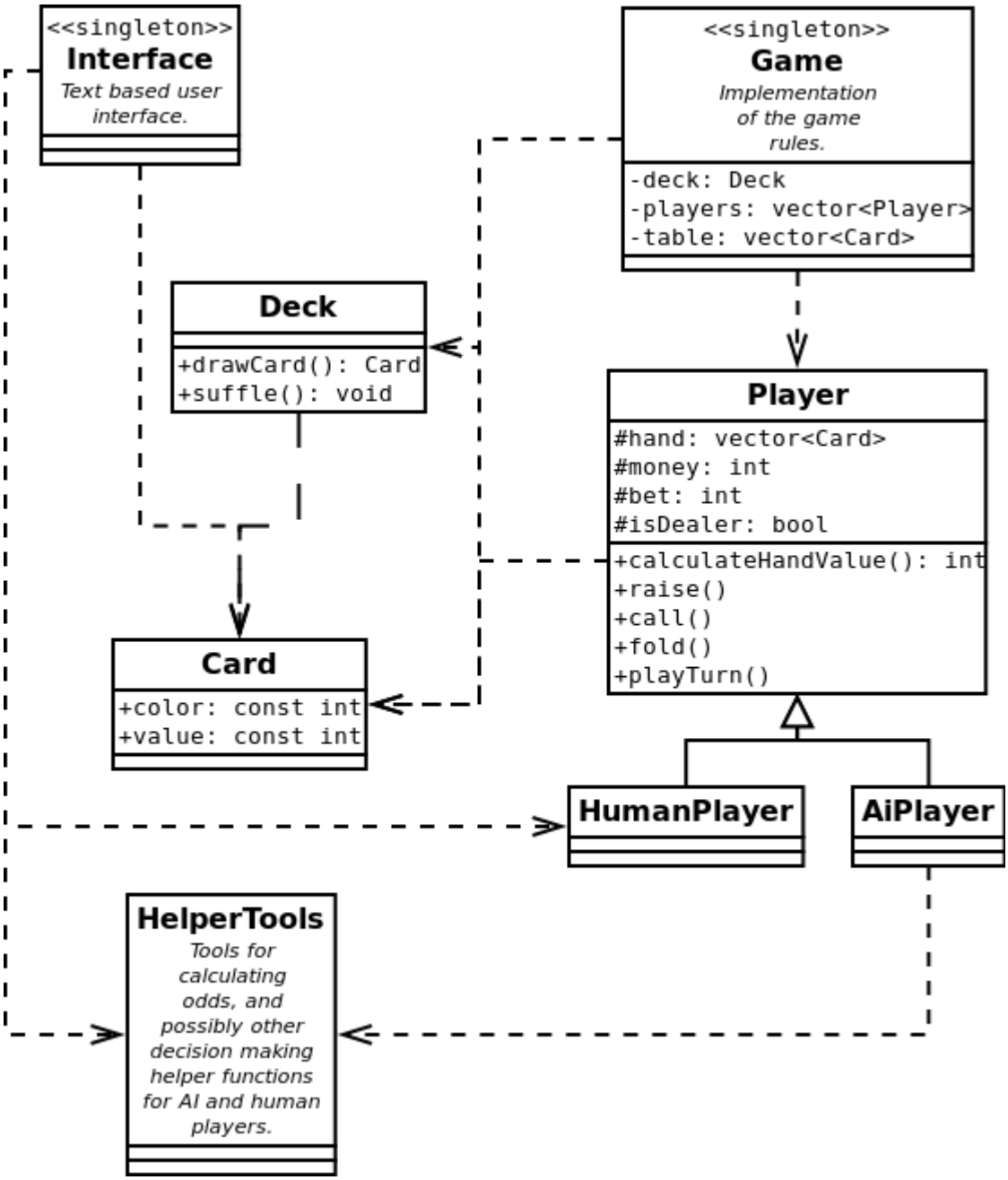
- A more advanced AI.
- Tools for players, such as for calculating the odds of a certain hand.

If there is there is time left after implementing the above features, we will also try to implement:

- A graphical user interface.
- No-limit version of texas hold’em.

Program architecture

The game runs on text-based UI. Here the user will see the implemented game logic and rules and gets to interact with the game. Commands will be given on the command line. For example check, fold or raise(xxx).



UML-diagram of the program

The UML explains the program structure. It should be pretty accurate on the class level, but some methods and variables are missing and the existing ones may change.

The artificial-intelligence of the game will use to its advantage the player hand's odds calculator.

The human player will see the odds to different type of poker hands by typing odds on the command line. The same tool will be used in the AiPlayer class so that the AI knows if it has a good hand or not. It will play the game accordingly to the value and possible odds.

At the moment we think that we'll implement the Player function calculateHandValue() by making a hash of the seven card hand and checking it's value from a look-up-table. We will use an external table from this site: http://www.pst.ifi.lmu.de/~hammer/poker/handeval.html. We also considered an algorithm which

forms all the possible hands from the number of cards at the moment on the table. There the algorithm would check for possible hand options. We chose to use the look-up version algorithm because hopefully it will be faster.

Task sharing

At first we will implement the basics of the game together physically in the same place (Maari). These include for example the classes Card, Deck and the very basic functionality of Game and Player-classes. From there we will see how much have we accomplished coding this way, and see if there is a need to develop the project separately. The main components of the program, which need a lot work are Game, Player (especially the calculateHandValue()) and HelperTools classes. Game-class will be implemented by Mikko S, Player by Mikko N and Interface & HelperTools by Antti A.

Testing

The parts of the project that can be unit-tested are HelperTools for calculating the hand-odds. This will happen by writing a separate test-program. The result correctness can be checked with existing odd-calculators. Players most difficult algorithm, the function calculateHandValue() can be unit tested with a separate test-program. Testing of the whole system will be done step by step. We will test how the different classes work together by implementing the Game class and an UI-class to actually try to run the game. At the stage where the inner logic and the game seems to be working, we can start to implement a proper functioning AI-class. At the last stage when the game and AI seems working, we can use people that are not involved in the project, to play the game and see if they find bugs/errors from the project. Of course we will also play the game a lot by ourselves too.

Schedule

4.11 Deadline for project plan (about 5 hours)

8.11 We will see on Maari and start the implementation of the project, after this weekly meet-up and individual work to get the minimum requirements done (about 30-40 hours of work per member).

22.11 We hope to see the minimum requirements fulfilled at this point. After this we will implement the optional features of the project. This will take about 10-20 hours of work per member.

30.11 At this point the program will hopefully be ready for final testing and writing of the project documentation (about 5 hours of work).

External libraries

External libraries will not be used if the GUI – optional feature is not implemented. If we will use the look-up-table algorithm for calculating the hand value, external resource will be needed (the look-up table itself).

If we have enough time and decide to implement the GUI we will use the QT for that.