Blackjack simulator / client optimisation

Henrik Eideberg

Introduction

Black jack game to evaluate the 'best game rules' for a black jack player/client.

Why?
Some co-workers are developing a blackjack server in erlang. They have invited players to create blackjack clients/players to compete with other clients/players using their server. To be able to win, I would like to create my own application in which I could simulate and come up with the best client and to work out 'the best set of rules'.

Who will use it?
Will be used to evaluate the best 'set of rules' to win the black jack tournament. The player will be connected to a black jack server (developed by my co-workers and interface not yet available).

What subject from the course will be covered in the project?

- **Inheritance**
  The class HumanPlayer will implement the interface Participant.
  The abstract class ComputerPlayer will implement the interface Participant. Additional levels of ComputerPlayer will inherit the abstract class ComputerPlayer.
- **Polymorphism**
  The class Table will maintain a list of Participants in which it will at run time add either HumanPlayer or ComputerPlayer.
- **Abstraction**
  The class ComputerPlayer is an abstract class. Subclasses will be derived from this, e.g. ComputerPlayerLevel1 (which is the most simple version of a ComputerPlayer).
- **Exception**
  Wherever it is necessary exception handling will be implemented. Exception handling shall be implemented when implementing the serialisation/deserialisation of rules (player-rules, dealer-rules and table-rules).
- **Serialisation**
  It shall be possible to import and export dealer, player and table rules. This will be done with help of binary or XML serialisation/deserialisation.
- **Generics**
  The class Table will maintain a list of Participants to track who is playing.

The class Dealer will maintain a dictionary in which we connect hands with the players. This will be used when Dealer informs the Table and the Players who won, lost or tied the game.

- **Delegates/Events**
  The class HumanPlayer and Dealer will have to send information to Table (on game, hand and player updates). For this events will be used.

Backlog

Below is an attempt to summarize the needed classes, the attributes and methods they need to have.

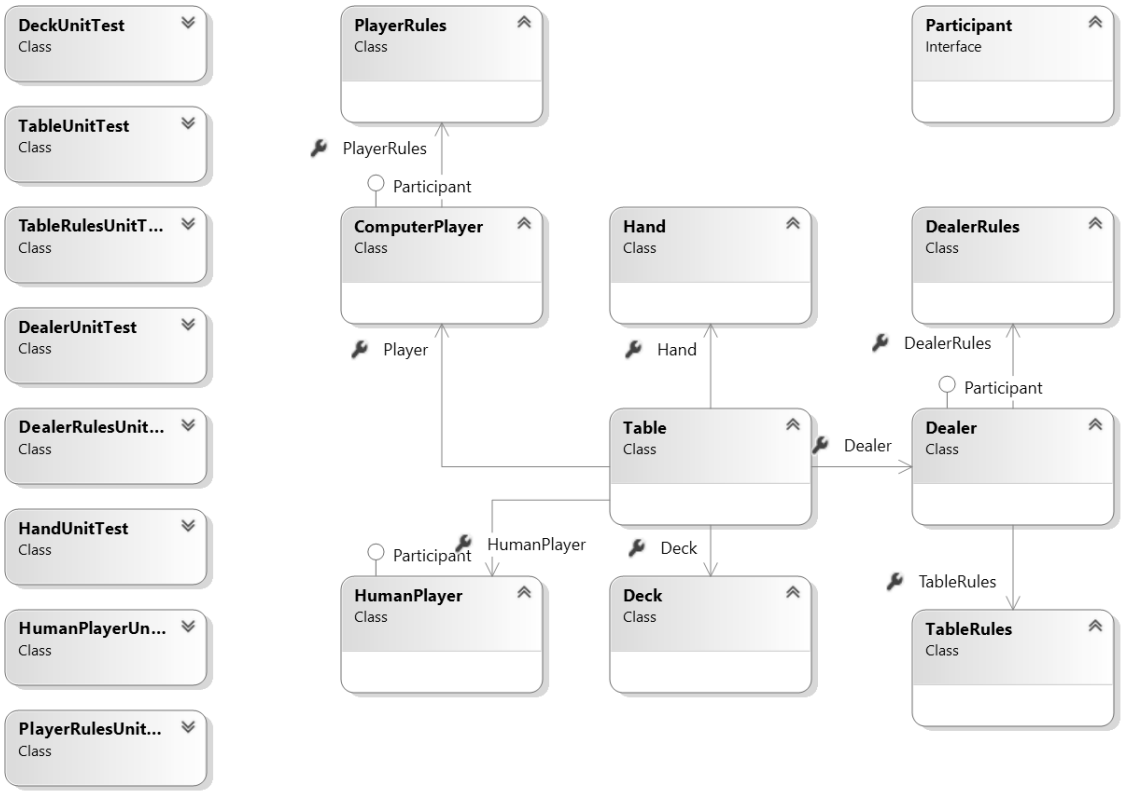
Item	Name	Description
1	Deck <class>	1. Create deck. Number of decks (1, 2, 4, 6, or 8) 2. Draw one card. Externally - give one card to player/dealer. Internally - move one card from deck to trash 3. Shuffle deck. Send event when deck is shuffled. Should be able to shuffle any-size-deck. 4. Attributes a. deck (list of integers) b. trash (list of integers) 5. Methods a. Draw b. Reshuffle deck 6. Events a. Deck has been shuffled
2	Table <class>	Is the 'spider in the web', receives all the events, starts all the other classes and distributes information from and to classes/participants. First version connects and uses the deck.
3	GUI	1. Lobby: Wpf class 2. Table: wpf class
4	Hand <class>	1. Attribute a. Id: binary (0001 0001) b. List of cards c. Number of splits: int d. Split aces: bool e. Is doubled: bool f. BetAmount: int g. BlackJack: bool h. InPlay: bool

		2. Methods a. Get Hand value b. Properties for all above attributes
5	Participant <interface>	1. Attributes a. Name: string b. Hand(s): id(s) of hands it is playing 2. Methods a. Get property for all the attributes 3. GetStatistics(). Method to return the statistics of played hands, e.g. nr of won
6	Table <class>	Second version to use deck and hand. Visible from GUI.
7	GUI	1. 'Dealer setup': wpf class 2. Cards with numbers (2-10 & Ace): textblock
8	Dealer <class>	Implements Interface Participant 1. Methods a. Draw - decides based on the dealer rules whether or not to draw a card to the dealer. If successful verification of rules, use the deck and draw a card. b. Receive player action i. Receiving from table: 1. Action (e.g. draw, split, etc) 2. Player id 3. Hand id c. Verify player action (accepted or rejected) i. Player action is verified against table rules d. Perform player action i. Sends event to table who performs the requested action on the hand. 2. Event to Table to update a certain hand 3. Event to Table to add a hand 4. Additional attributes a. House rules: class instance b. Player and Hand ids: Dictionary in which we connect the players and the hands during a game.
9	HumanPlayer	Human player will only convey the will of the human user 1. Implements Interface Participant 2. Attributes

	<class>	○ Money ○ Set of rules: instance of PlayerRules 3. Event in which one of following information is conveyed to Table. ○ Draw ○ Stop ○ Double ○ Split ○ Bet? Table reroutes this information to the Dealer who evaluates and performs requested action.
10	Table <class>	Third version to include use of Dealer class and HumanPlayer class (used in a list of participants). At this stage Table only does what it is told.
11	GUI	1. 'Player setup': wpf class a. Human player b. Computer player (N/A at this time)
12	TableRules <class>	1. Methods a. EvaluateHand b. ProvideAvailableOptions 2. When this is implemented dealer class shall use it
13	PlayerRules <class>	1. Methods a. EvaluateHand b. EvaluateGame c. ProvideBestAvailableOption
14	Import/export rules	It shall be possible to import and export dealer, player and table rules. This is done with help of serialisation.
15	ComputerPlayer <abstract class>	Implements interface Participant. 1. Attributes ○ Money ○ Set of rules: instance of ComputerRules
16	ComputerPlayerLevel1 <class>	Inherits the abstract class ComputerPlayer 1. The first version of the Computer player will only draw one card. 2. Further levels will expand and explore the 'set of rules' - found in class PlayerRules

17	Game statistics	1. Each game should be saved in a human readable text file in which a. Winning percentage for each player b. Hands played, lost, won, drawn c. Cards drawn d. Number of BJ e. etc...
18	Hand analysis	Analyze current hand and provide percentage of: - Chance of drawing 10, non-10 or Ace - Chance of dealer drawing 10, non-10 or Ace

Class Diagram



Note 1: The class ComputerPlayer is an abstract class and will be ‘replaced’ by instances of classes that inherits ComputerPlayer.

Time schedule with reference to backlog

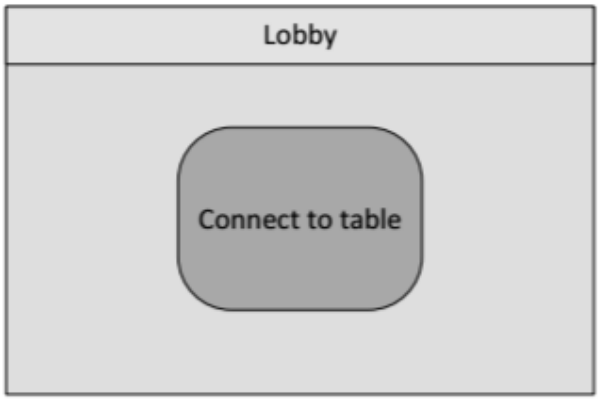
Backlog		May															June
Item	Description	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1

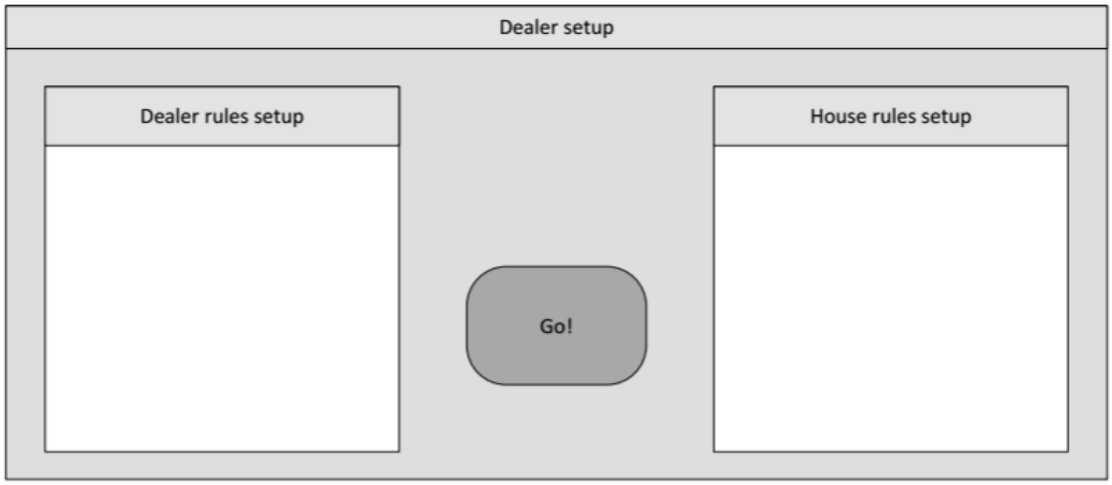
	Project proposal																	
1	Deck																	
2	Table. 1st version																	
3	GUI. Lobby and Table																	
4	Hand																	
5	Participant																	
6	Table. 2nd version																	
7	GUI. Dealer and cards																	
8	Dealer																	
9	HumanPlayer																	
10	Table. 3rd verision																	
11	GUI. Player setup																	
12	TableRules																	
13	PlayerRules																	
14	Import/Export																	
15	ComputerPlayer. 1st version																	
16	Game statistics																	
	Submission																	

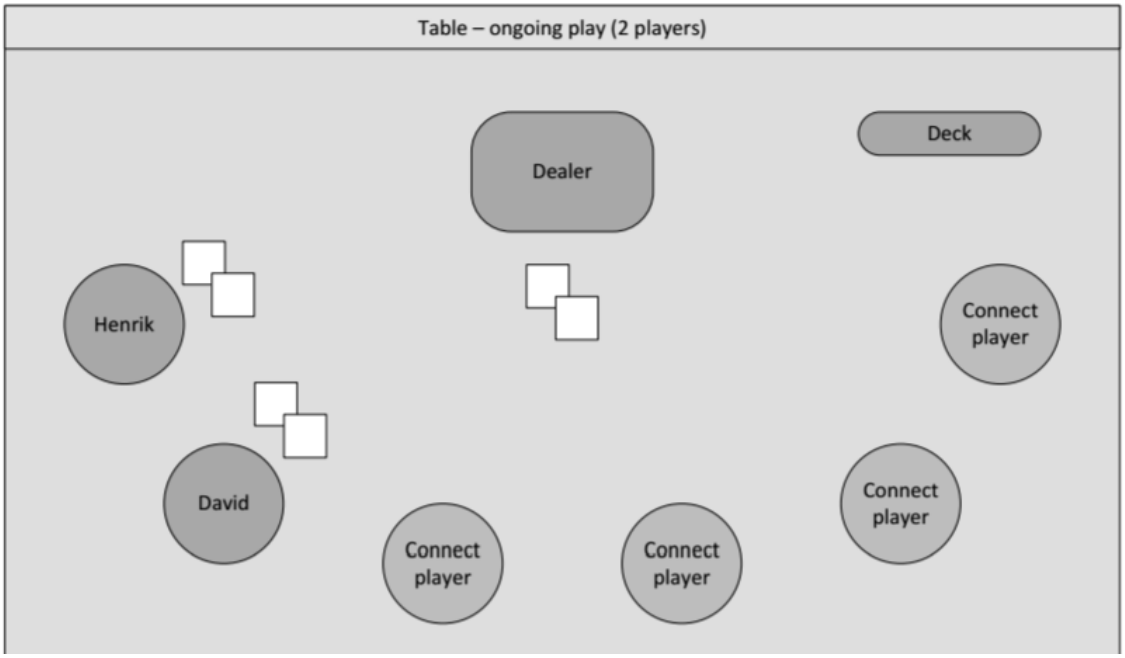
Note: testing is included in the tasks. It is assumed that there will be unit testing for each class. During development, the program will be run and tested.

Misc

GUI example pictures







Dealer rules

- Dealer should hit on soft 17 (ace and 6)?
    - Default: stands on soft 17

Table rules

- Number of resplits?
    - Default: No limit
- Split aces => allowed to hit again? allowed to split again?
    - Default: only one card will be dealt to each ace (no split, double and no more hit on either hand)

- Double after split?
    - Default: a double after split is allowed
- On what cards can a player double down?
    - Default: only allowed on hard (hard = 2 cards) 9, 10 and 11
- Hole card or not?
    - Default: no-hole card
- What happens at tie? Dealer wins or tie?
    - Defaul: Tie