Technical Manual Brozingos Poker

Introduction

The Brozingos Poker has three forms, startScreen, which is the first form seen by the user, there he can choose how many players he wants to play against, and start the game.

Our game happens in the GameTable, a form containing all the cards and players. Together with the GameTable, the PokerHands form is opened to help the user to identify which combination he has in case he is not an experienced poker player. All the images in the software are carried by labels, except to the two cards the user has, which are buttons.

Inside the Model, the players are stored in an ArrayList, and everywhere else the players in the table are used, they are extracted from this ArrayList. The CardDealer is the responsible for taking the card randomly and giving it to the players or turning in the table.

The player's money is stored in chips, each player has an array of chips where each index of the array is a different tip, and the number stored in the index is the amount of this specific chip.

All the action of the game happens inside the Controller, where all the variables of the software are brought together and where the user stories happen.

All the animations are done using Timers, and they're all stored inside the Animation package.

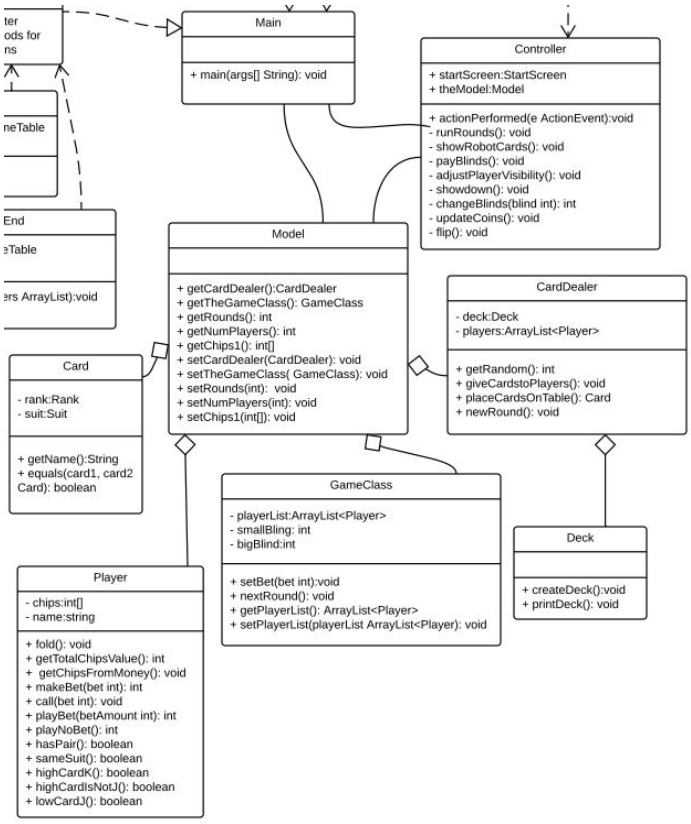
The AI is very primitive, it is based on if else statements to define what the player will do. If a bet happens, the AI will decide whether he folds ,call or bet based on how much money he has and what combination he has in pre defined limits and specifications.

User Stories

User Stories		Priority (1-5)	Tasks	Product Backlog
As a user I should be able to	start a game	1	code the main class	y
			instantiate players and dealer	y
			code deck of cards	y
			code chips	y
			bring all other classes together	y
	fold	2		y
	show	2		y
	pay small/ big blind	2	create token to see who has small or large blind	y
			shift who pays what blind to complete round	y
			code choice: pay or fold	y
			switch who pays what blind in new round	y
			if player cannot pay, allow to check or fold	
	check	3		y
	call	3		y
	cash out	3		
	increase my bet	4	create slider to choose amount to bet	y
			do not allow to bet beyond what player has	y
	end a game	4		y
	decide the number of players	5		
	restart	5		
	see how many chips I have	1	create way for player to view number of chips	y
As a dealer I should be able to	give cards	1	use deck of card to give out new cards	y
			code random card selection: no repeat cards	y
	show the cards	1	show cards for (3) flop, (1) turn, (1) then river using swing	y
			flip over all cards left for the showdown	
	check the best hand	2	check hand function	y
	know who's turn it is	2		
	raise the minimum	3		
	give out the money to the winner	4	add pooled money to player's amount	y
	shuffle cards	5	code card shuffling	y
	keep track of the player's funds	2	update player's chips array when values change	y

The system accomplishes the user stories mainly in the CardDealer class and Playler class, and these two classes use several other classes to help them to achieve the objective of each user story.

OOD- Object Oriented Design



The OOD approach helped us to Divide & Conquer the final product, where each integrant could work on a different part of the software and the OOD made possible the future connection of these parts.

The software follows the Model View Controller approach, and also have other classes that assist the MVC.

Main	
- Runs the Starts the game	- Model - StartScreen - Controller

Player	
- Makes Bet - Calls - Folds - Knows how much it has - Sets the name - Knows if it has big or small blind - For computer player, Knows to play when bet is made - For computer player, knows when to play when no bet is made	- Cards - Chips

Our players have a specific class Player, which contains methods that help the controller and the dealer to manage the game, like how much money it has, and how many chips, and what cards they're holding,also methods to help the AI interface.

CardDealer	
- Distribute random cards to the players - Place cards on the table - Shuffle cards and start new round	- Player - Deck

The dealer has methods to give cards to players, and also carries the deck, the players in the game, and the cards that have already been taken off from the deck.

Deck	
- Store all 52 cards	- CardDealer

The deck is an array with 52 cards, four cards of each kind with four differents suits, the deck is always fixed, the randomness happens when the dealer draws a card from the deck.

Card	
- Showing suit - Showing rank	- Deck - CheckHands

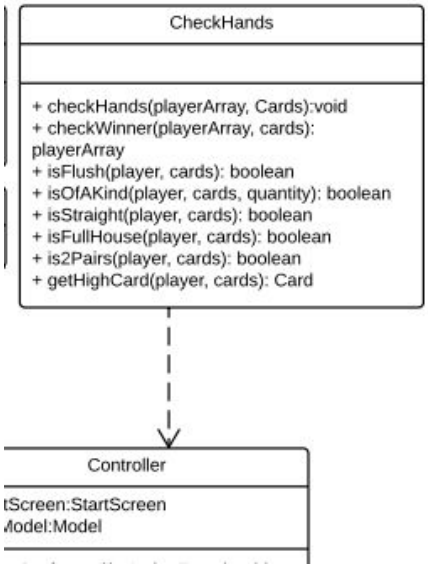
There is a Card class,each card has a Suit enum and a Rank enum, the rank goes from two to aces.

Model	
- Returns the players, deck of cards, card dealer, game class, chips - Sets the players, deck of cards, card dealer, game class, chips	- Player - Deck - Chips - GameClass - CardDealer

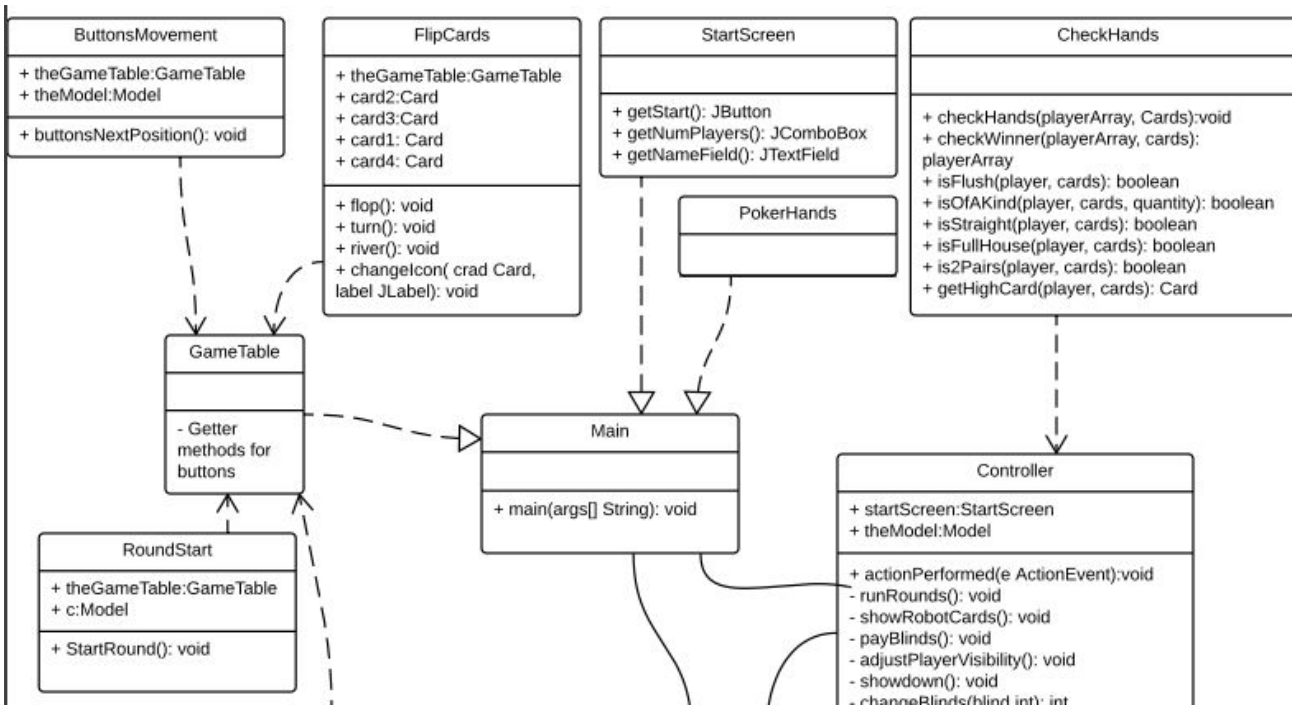
The Model stores all the relevant data to the game, from where the controller gather and gives information and from the view gather the data to show up.

Controller	
- Controls all the actions that are performed by the click of the mouse button - Runs different rounds of game - Shows the card of computer - Updates the visuals from model	- StartScreen - Model - GameTable - RoundEnd - FlipCards - RoundStart

All the actions happen inside the controller, there is where all the listeners to the buttons are located. The controller uses the other classes to maintain the flow of the game, and also makes use of the Animation classes that modify the view.



The CheckHands class is used to define which player has the highest hand in the table, and is used by the Controller to define the winner.



This part of the UML class diagram is responsible for controlling the view part of the software, like all the animations and forms.

StartScreen	
- Creates and displays the start screen for the game - Gets input from user on number of players and name	

RoundStart	
- Controls the movement of the cards from the dealer to the players	- GameTable - Model

RoundEnd	
- Controls the movement of the cards from the players to the dealer	- GameTable - Model - ButtonsMovement

FlipCards	
- Controls the flip of the cards on the table - Controls *flop, turn* and *river*	- GameTable - Card

ButtonMovement	
- Controls the movement of the buttons - Implements certain actions with the click of every button	- GameTable - Model