Cassino Card Game

Riku Tikkanen, 736507, TIK, first year student
27.04.2020

1. General Description

A Cassino card game with a graphical user interface for 2-N players. The goal of the game is to collect as many points as possible by fishing cards from the table with cards that the player currently carries in his hand. At the beginning of every rounds the deck is shuffled and 4 cards are dealt to every player with the face down. After this 4 cards are placed on the table faced up. The game is played clockwise and the first player is next to the dealer. In this case the first player is randomly chosen, since there is no physical dealer.
On his/her turn every player will try to take as many cards as possible from the table by matching their sum to one card in their hand. For example, if the player has ♠9 he/she may take cards ♣3 and ♥6 from the table or any other combination which adds to a sum of 9. The value of the cards picked up are stored for scoring. If the player can not pick up any cards he/she must put one of his/her cards on the table. The number of cards on the table varies and every player must always draw a new card from the deck after using a card so that he/she has 4 cards in his/her hand. After the deck runs out, everyone plays until there are no cards left in the deck. If a player gets all the cards from the table at the same time, he/she gets a so called sweep which is written down.

Card values:
Aces = 14 in hand and 1 on table
♦10 = 16 in hand and 10 on table
♠2 = 15 in hand and 2 on table
For all other cards the values are the same in the hand and on the table.

Scoring:
For every sweep a player gets 1 point
Every Ace grants 1 point
The player with the most cards gets 1 point
The player with the most spades gets 2 points
The player with ♦10 gets 2 points
The player with ♠2 gets 1 point

After all cards have been played the points for each player are calculated and if no player has 16 points a new round is initiated. After a player reaches 16 points the winner announced when the round ends.
I am planning to complete the project as Demanding with graphical interface and the possibility to play against 0-N computer controlled opponents. I will restrict the total number of players N to 4 to make the game more enjoyable to the player.

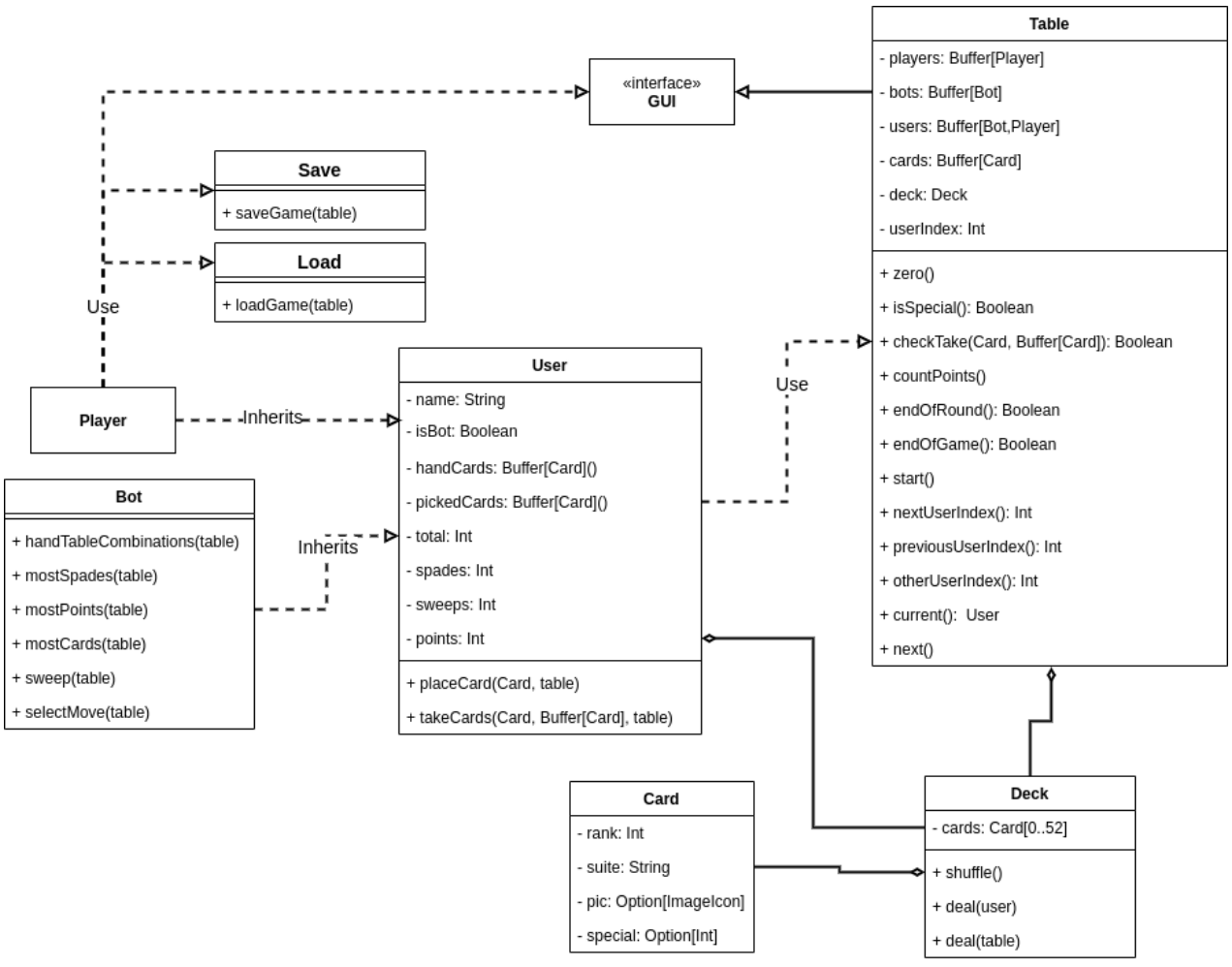
2. User Interface

The program has a graphical user interface(GUI) which starts with an example game with 2 players and 2 bots. The user may play along with the example game if he/she wants to. The GUI consists of cards and two buttons for placing and taking cards. In the right corner a console displays the number of cards left in the deck and error messages. The name of the player in turn is displayed below the console. Each table card in the middle of the GUI and each user card on the bottom of the screen represents a button as well. The cards on the sides only represent labels for the other players cards. The player on turn may activate only one of his/her cards on the bottom of the screen and as many of the table cards as he/she wants to. To attempt to take the selected cards from the table the player pushes the Place card button. To place a card on the table the player pushes the Take cards button.

Instead of playing the example game, the user may initiate a new game by pressing the Game button in the Menu bar. Two sliders will appear for selecting the number of players and the number of bots. After pressing the new Game button a new game is started.
By pressing the File button in the Menu bar the user may choose to load a previously saved game, save the current game to a file or exit the game by pressing the Exit button.

3. Program Structure

UML



The game consits of a graphical user interface which inherits items through the table class. The graphical user interface is operated by the player. The main component of the game are the Cards which are initiated in the Deck class. The Deck class shuffles the cards and deals them to the table and the users using shuffle(), deal(user) and deal(table). The User main class inherits the subclasses Player and Bot. Users have a name, two lists for “handcards” and “tablecards” and variables for counting points. The Player class uses the GUI to communicate with the program. The Bot class has additional functions to calculate the best move.
The Bot class creates all possible handcard-tablecard combinations at the beginning of the turn. From those combinations the bot class tries to choose one in the following order by using the selectMove(table) method: Try to empty the table by taking a sweep with sweep(table), choose the combination which gives the most points with mostPoints(table), choose the combination which gives the most spades with the mostSpades(table) function and choose the combination which gives the most cards with the mostCards(table) function. If no such combination exists, the bot will place a card on the table.
The Table class consists of players and bots and a deck of cards which will be used to deal cards to the individual users. It has a method checkTake(Card, Buffer[Card]) which checks if the take attempted by a player is valid. The endOfRound and endOfGame methods are helper methods to check the state of the game. Using the CountPoints method the points for each user are counted at the end of each round. The start() method is used to shuffle the deck and deal the initial cards to the players and the table. By calling the next() method the turn shifts to the next player. The methods nextUserIndex(), previousUserIndex() and otherUserIndex are helper methods for placing the cards in the GUI.
The player may save a game using the Save class and load a game using the load class.

4. Algorithms

The tables CountPoints method calculates the points for each user according to the rules with addition.
To validate a users move a list of combinations is created from all the possible combinations of the selected handcard and all the tablecards available. The combination choosen by the player is compared with the list and if it does not exist in the list the players move is invalid. The Bot class creates a same kind of list of combinations and selects the combination which gives the most points according to the rules.

5. Data Structures

The main object of the program are playing cards which are stored in mutable arrays. In order to save a game a text file is created which can be later read to load a previously saved file.

6. Files and Internet access

The program does not have internet access but it does create a text file when saving the game.

A text file looks like this(with deckCards on one line):

name:example player 1:false
points:0
handCards:D3,H7,C6,H2,
pickedCards:
sweeps:0
name:example player 2:false
points:0
handCards:S13,C11,S4,H1,
pickedCards:
sweeps:0
name:example bot 1:true
points:0
handCards:S2,S10,D5,H12,
pickedCards:
sweeps:0
name:example bot 2:true
points:0
handCards:D12,S5,H5,H3,
pickedCards:
sweeps:0
tableCards:H4,S11,D13,S7,
deckCards:D6,S9,S6,H9,D4,C9,H8,D10,D2,C12,D8,C5,C1,D11,S3,D1,S8,C3,C13,S12,D7,
C8,C2,H6,C10,S1,C7,D9,H10,C4,H11,H13,
END

7. Testing

The program was tested during the creation of the text version.

8. Known bugs and missing features

Handling errors when reading/loading files
The points from loaded files are not always correct.
Random crashes when too many cards are on the table
Blue card labels do not update correctly with too few players
If I had more time I would run through the code to find the index errors and clean in up.

9. 3 best sides and 3 weaknesses

Best sides are the Bot class algorithm, visualisation and user friendliness in the GUI.

Worst sides are the quality of the code(specially in the GUI), error handling in reading a file and the unkown crashing when too many cards are on the table.

10. Deviations from the plan, realized process and schedule

In the first two weeks the classes were created for user, deck and table. In the next two weeks the first test game version was being created to test functionality and errors. After the first month work with the GUI was started and progress was really really slow because scala swing seems to be such an outdated library and no one uses it anymore. It was difficult to ask older students for help because no one remembered using it. Creating the GUI was the most difficult part of the game and many hours went into making it. A lot of time went into frustration with the buttons. The save and load methods were created at the very end of the progress after the GUI was already starting to finish.

11. Final evaluation

A Cassino game with a graphical user interface was created to let a player play against other players and bots. Players may select cards from the interface and choose an action. A game may be saved and a previously saved game may be loaded from the menu. A new game can be initialized at any point.

I am happy with the outcome of the GUI and the main methods of the program but some bugs are still bothering me, for example the random crashing. The GUI could still be improved by using more than one panel for the cards which would make updating easier. The quality if the code in the GUI is not good and it shows the frustration of making it.

If I had to start from the beginning I would spend less time in making the text game format and more time choosing a good GUI library(other than scala.swing, perhaps java.swing with wrappers)

12. References

Stackoverflow.com
youtube.com Mark Lewis channel
http://otfried.org/scala/gui.html
https://www.cs.helsinki.fi/u/wikla/OTS/Sisalto/examples/html/ch32.html