**ASSIGNMENT COVER SHEET 20220/2021**

|  |  |  |
| --- | --- | --- |
| **LECTURER NAME:** | | |
| **STUDENT NAME: Benoît Gras** | | **STUDENT ID: 24571** |
| **PROGRAMME: Semester Abroad** | **STAGE/YEAR: Semester Abroad ESILV** | |
| **MODULE NAME: Data Structure & Algorithm** | | |
| **ASSIGNMENT NO. & TITLE: CA Project** | | |
| **GROUP (Names & Student IDs): 24571** | | |
| * All assignments must be submitted through Moodle by the agreed submission date. * Only submit a signed hardcopy to your lecturer if requested and you must arrange to hand it directly to the Lecturer and must include this cover sheet. * No submissions will be accepted at Reception. * Late assignments will incur a penalty unless a learner has documented personal mitigating circumstances or if they have been granted an extension. All such documentation must be submitted at least 7 days prior to the submission date. For late submission the assessment grade will be reduced by 10% for each day that the assessment is late. After day five, the assignment will not be accepted. | Date Received: (Lecturer use only) | |
| **MODE OF SUBMISSION: SOFTCOPY  HARDCOPY** | | |
| **COMPONENTS OF SUBMISSION:**  **(e.g. no and type of pieces submitted, no of pages in a report, disk included?)** | | |
| DECLARATION: I declare that:   * *By uploading this work to Moodle I automatically declare that this work is entirely my own and that I have acknowledged all materials and sources used in its preparation;* * *I have not copied in part or whole or otherwise plagiarised the work of anyone else and have not knowingly allowed others to plagiarise my work in this way;* * *I understan* *d that plagiarism is a serious offence and that I am bound by Dorset College policy on Academic Integrity. I understand that I may be penalised if I have violated the policy in any way;* * *This assignment has not been submitted for any other course or module at Dorset College or any other institution, without authorisation by the relevant lecturer(s);* * *I have read and abided by all of the requirements set down for this assignment.*   SIGNATURE\* ………………………………………………………… DATE …11/12/2021……………………  (\* if this is a group assignment, each member of the group must sign). | | |

**ASSIGNMENT SOLUTION**

* Introduction

My first Idea for this project was to go with an application that can provide you the best hostel for your needs. I realized that this kind of application already existed in large numbers. For example, Booking.com or Trivago. As I couldn't find an idea to distinguish my app from the others, I started looking for a new idea. And being a fan of Casinos, I went interested in Blackjack.

This game is very interesting. It mixes gambling, mathematical concepts, luck, and probability. I knew then, that going in this direction could be very interesting.

* Part 1 : General Settings

The first part of this project was to do a playing able game of BlackJack. Basically, this game only need a game set of Cards. Those decks contain a lot of cards. Each different card is in an array that contain: its access, its value, its display and the number of cards of its type. For example a Queen will have as access: 12, value: 10, display: “queen”, and the number of cards changes with how many set of cards there is in the game. For it, I create a class Card which contain this array.

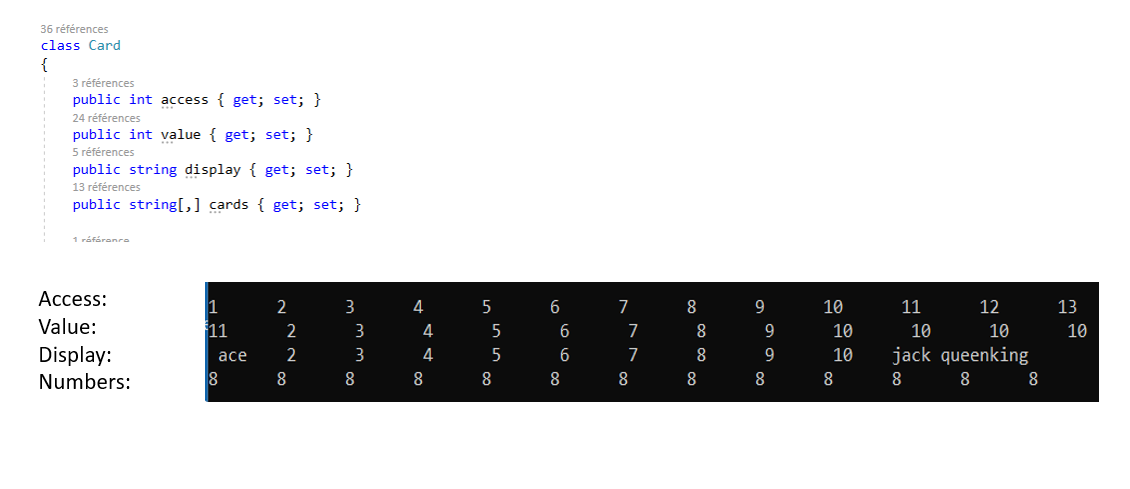


Figure 1: array presnting the differents cards

To create this deck, the user can choose the number of decks he wants. For example for 2 decks, the game will have 104 cards. In order to make it realistic and if the player wants to count the cards (see appendix fig 1), the cards are counted in a stack. So, if a card has already appeared, it can no longer appear. And so, card counting is possible. When the user enter the number of playing decks, there is 52\*thisnumber cards created and put in a list. This list is not shuffled. For this, a method will draw a random card and put it in a stack. Like this, the number of cards in the list decreases as the number of cards in the stack increases.

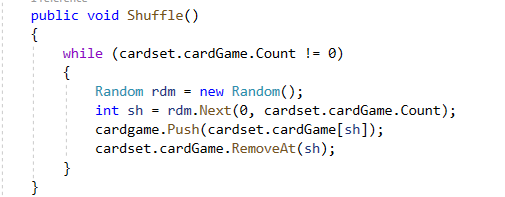


Figure 2:Shuffle Method (the cardgame is the shuffled game)

* Part 2: The Game

Basically, the blackjack is a game, where the player is against the croupier. Each of them have a two cards at the beginning and can ask for other cards or other possible actions. (For more informations check the rules fig 2 in appendix) The player is the first to play and according to his cards he can choose different actions. One of the two cards pick by the croupier is hidden (figure 3).



Figure 3

The game is presented like that (figure 3), and the choices are coded to be only able if your cards allow it (figure 4). For example, to split a game you need a very specific game.

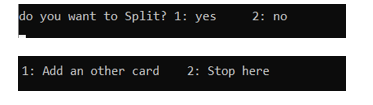
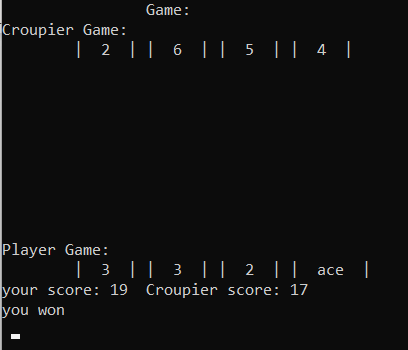


Figure 4



Once the player has played, it’s the croupier time. The croupier follows very specific rules. Before a score of 17 he pick a card, after it he stop. At the end of the croupier turn, the code shows the final hands and tell if you won or lose. In case of a BlackJack you directly won even if the croupier reach 21 too.

Figure 5

Part 3: Distinctive features

The first point is that my app ensures that the player can count cards. They can even choose the number of set of cards they want. So, this app can be used as a training for card counting.

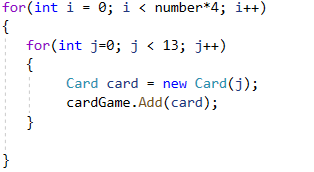
This double loop (figure 7) allows to create each card of the card game. The variable “number” is the number chosen by player.

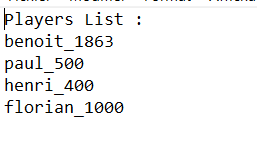
Figure 6

Figure 7

The figure 6 show how the player game is built and how the card game decrease gradually through the game

The second point is that each player has an account, and a balance on the app. The class Player can create those. Those data are saved on a txt File. As so, they can add and withdraw money from their account. And they have to bet at each game they do. (Look the win amount fig 3 in appendix)

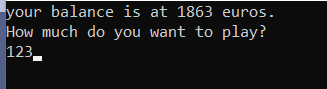


Figure 8

The data are saved like on the fig 10.

To see the code that allow to change the list

Figure 9



The player can bet as much as he wants as long as it does not exceed half of his balance. Some of the game's options allow you to double your bet during the game.

Figure 10

The third point is that there is a table showing the best thing to do for each game. So, the app advises the player on the best action to take. This table depends on the player's game and the dealer's visible card. (You can find it in the appendix fig 4).

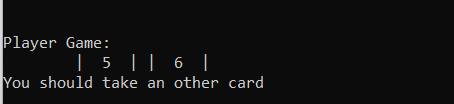
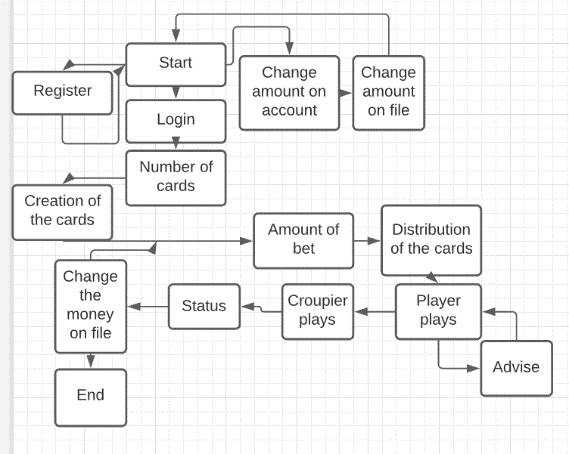


Figure 11

An advice can be displayed like that.

The difficult part of this advices was to analyse all the possible games in order to find out from the different tables which is the best option. To do this, the csv tables were read, then once a game was analysed, the app looked in the right table for the advice to give to the player.

* Conclusion:

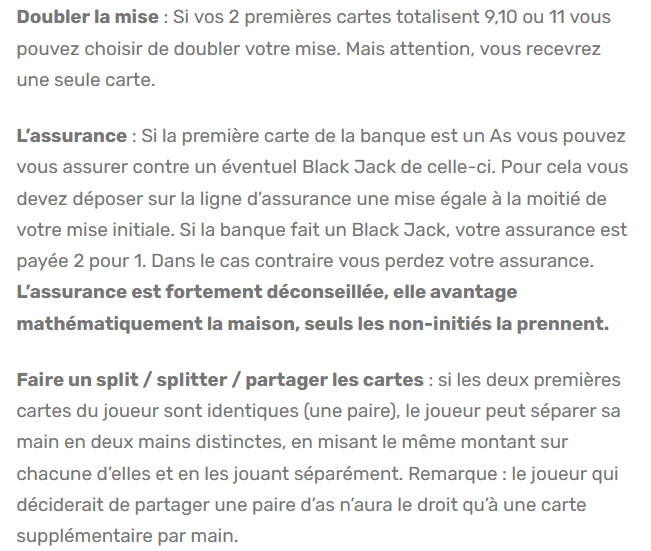
Here is the pseudo code of my algorithm. It gives a rough idea of how it works.

A more detailed UML of the algorithm can be fin in the appendix.(fig 5)

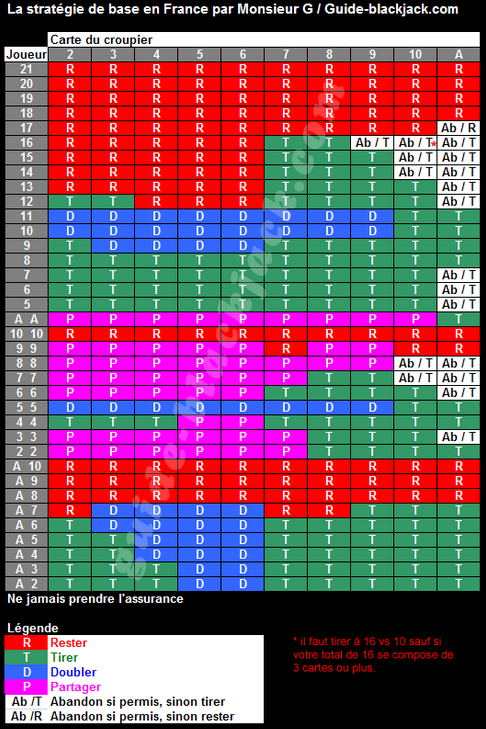
References

- *https://www.casino-zen.com/regles-du-black-jack/*

The first step was to reminder all the rules and specificities of BlackJack. So, I went to this casino website to be sure of what I was working on. And to add all the rules to my app. Particularly, thos were the rules I was not familiar with:



- *https://www.guide-blackjack.com/compter-cartes-hi-lo-2.html*

Here is the website that allowed me to find the BlackJack probability table to give the best advice to the player. I have of course checked on other sites that this table was good. It shows the 4 principal possible actions with the one that has the best chance to win.

On the left it’s the player game.

On the top it’s the first card of the croupier

Questions

1. The part chosen, is where the algorithm find out wich is the advise to give to the player. This part can be found on the appendix(app fig 6) The difficulty with this part was to compare the player game with the croupier game. And first go to the good table.(app fig 6) secondly, find the perfect match to find out the perfect advise.

a.

This algorithm is iterative. And it’s like that cause it’s a searching functions.

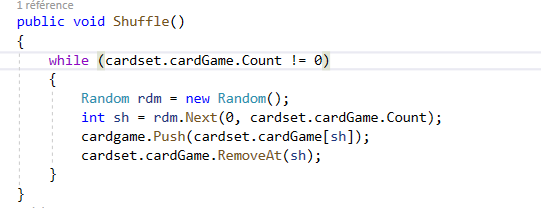
b.

The big(O) of this method is O(1). Indeed, the foreach loop will be executed maximum 21 times.

And each actions in this loop are only executed once by loop.

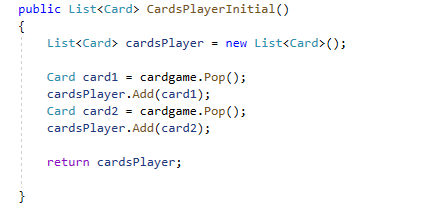
It’s the same for the while loop.

And there is no other loop in this part.



There are many parts of my algorithm in which I encountered problems that took a long time to solve but this part was the most interesting. Indeed, I have been looking for an efficient way to shuffle a large number of cards (if you choose to play with 10 decks, you end up with 520 cards). This version comes after a long time of research. The cards arrive sorted in a list and come out in a shuffled stack.

If I couldn’t use the stack , for a deck of cards, I probably have used an array. And when you pick a card the value at this placement become null and ther is a searching loop that find an emplacement that is not null. This method would have deteriorated the efficiency of the algorithm. And I would have to change a lot of lines. Like these one:



Appendix

* Fig1

Card counting consists of counting in your head, starting from 0, the cards that have already passed in the game. Thus, for strong cards, you can for example make -1 and for weak cards make +1. So when you get to -20, you know that statistically fewer high cards can fall and vice versa. So you can adapt your game to this. (for further informations: https://en.wikipedia.org/wiki/Card\_counting)

* Fig 2

If you want informations about all the blackjack rules, you can go to this website:

[https://bicyclecards.com/how-to-play/blackjack/\*](https://bicyclecards.com/how-to-play/blackjack/*)

* Fig 3

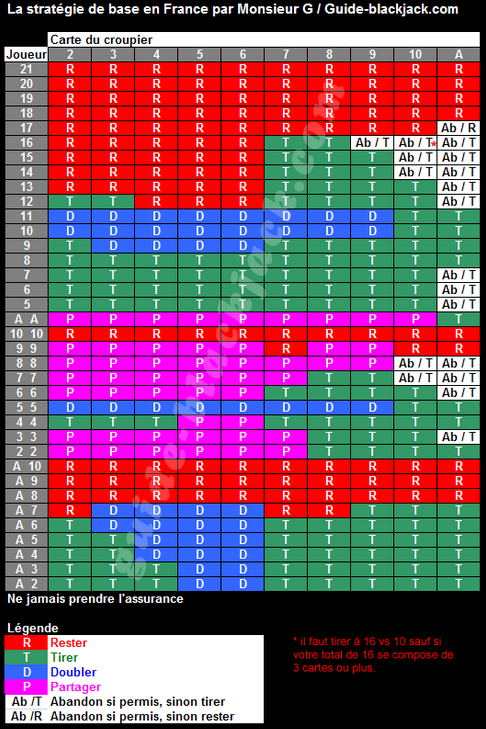
This part of code calculate who won the game. And how much money the player won or lose.

If the playerdoes a BlackJack, he win 2.5\*his bet.

If he win, it’s twice is initial bet.

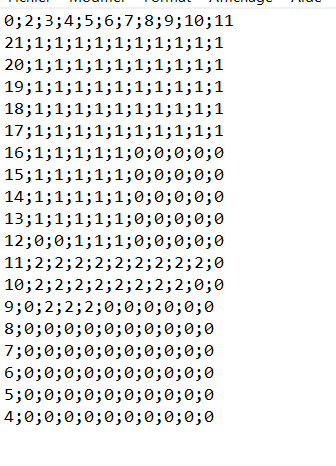
If he lose, it’s half is initial bet.

In case of a split, the bet is doubled and the game too. So, the alogithm pass trough 2 times this part.

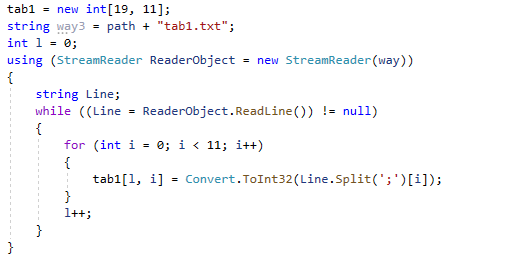
* Fig 4

The advices tables are from this picture. I divide it in three parts and put them in csv files for the can is able to read them.

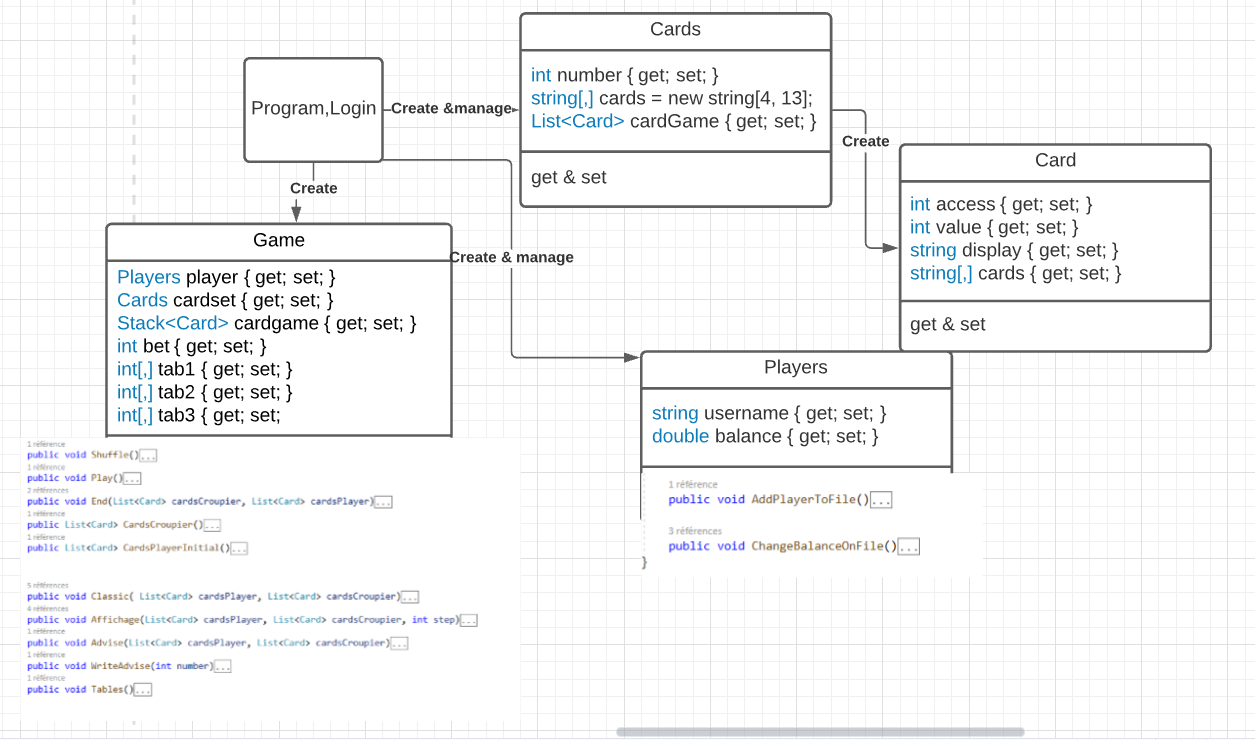
For example the first tab(tab1 in the code) is like that.



The code read them, by copying in three different arrays like that:



* Fig 5



UML of the algorithm that shows the different classes

* Fig 6

Equation 1

