Project Two

Muhammad Qazi

Dr. Rubey

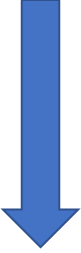
EN.605.201.81 Fall

10/30/2023

Abstract

This assignment focused on designing a game of Blackjack, with a certain set of instructions. For the code, I followed the instructions while opting for certain assumptions that followed the rules of the assignment. I detail the information within the analysis, while noting any potential alternatives that I considered for the process of the assignment.

|  |
| --- |
| BlackjackGameSimulator (main class and method) |
| +x: int  +x2: int  +y: int  +z: int  +a2: int  +xx: int  +z2: int |
| +getPlayerValue() |



|  |
| --- |
| Player2 |
| +x: int  +randomNumber1: int  +N: int |
| +getPlayerValue() |

Shape

|  |
| --- |
| Player3 |
| +x: int  +randomNumber1: int  +N: int |
| +getPlayerValue() |

Analysis

What exists as one of the most popular and famous games of all time, particularly in the world of luxury and the “high” life? The game of Blackjack, popularized centuries ago, features in various casinos and luxury destinations around the world, generally popularized for gambling. For this project, the instructions required students to design a game of Blackjack. This analysis here details the process of designing the game, and the designs implemented.

To start with, I considered the basic rules of the game. For one, the game of Blackjack features various variants and versions of rules. Not a single “standard” exists for it. Compared to more objective games and sports, Blackjack allows for variety. Due to this, I ended up setting a list of basic assumptions to reference and follow in the design process. I, of course, followed basic rules that determined the values of the cards and determined the project constraints based on the instructions. Apart from that, the instructions allowed for a certain degree of ambiguity for the other, non-specified and non-basic rules. An example of this relates to the decks utilized within the game. Some casinos utilize one deck of 52 cards in Blackjack. Some opt for two. Other sites tend to opt for six decks. All of these situations lead to different probabilities for the cards drawn. The chance of drawing a Queen exists as different in a deck of 52 cards, compared to six decks. Thus, I then set a basic assumption for the assignment, that an open-ended amount of cards exists in the casino or location playing the game, and the chance for drawing a specific card equates to 1/13, due to the fact that thirteen different cards exist. Based on this, a chance of 1/13 existed for drawing a specific card within the code, regardless of whether or not the system drew two cards in chronological order, or one card. In terms of another assumption, different locations utilize different rules for the Ace card. Generally, some locations utilize the rule that an Ace equates to 11, or 1 in the case where it forces the total-card value over 21. Other locations allow for the player to determine the value between 11 and 1 themselves. Due to this, I adopted another assumption of setting a unique rule for the game, where an Ace always equated to 11.

In terms of the code itself, I utilized the main class with the main method to run the majority of the code. It set the variables, and ran a “do” cycle that only ended if the player asked to quit the game and not play again. The “do” cycle divided into two halves, for the player cards, and the dealer cards. An “if” statement ran over both halves, where the code only allows for one side to play at a time. Once that side stopped playing, through “staying,” victory, or defeat, the other “if” statement enabled while disabling the current one; handing the game off to the other player to continue the cycle. I built two sub-classes for the main class. The first sub-class of Player2 generated the random combined-value of two cards, while the second one of Player3 generated the random value of one card. The “do” statement in the main class referenced objects to pull in the random values of these cards, through methods. In regards to different approaches considered, and the justifications for rejecting them, I originally built two different “do” statements in the main class. This led to an issue where the game never repeated if the game ended at the stage of the dealer, at least without unnecessary amounts of code. A single “do” statement continued the game at the request of the player, due to the fact that the “if” statement inside allowed for the game to continue indefinitely. I also originally allowed the player to continue playing after running out of the pool of money, to then enter debt and play at a loss, though I then changed that to end the game if the player ran out of money; to follow the instructions and not punish the user.

I appreciated the opportunity to work on this assignment and design a game of Blackjack for the user.

References

(2023, October 30). *Blackjack*. Bicycle.

<https://bicyclecards.com/how-to-play/blackjack>