# Design Overview for Simple Card Game

Name: Marco Giacoppo

Student ID: 104071453

# Summary of Program

The program is a simple card game that allows players to compete against each other. At the start of the game, a deck of cards is shuffled and dealt to the players. The players take turns playing cards, and the player with the highest card wins the round. The game continues until all the cards have been played, and the player with the most points at the end of the game is declared the winner.

The game can be extended to include different types of card games, such as Blackjack or Poker, by inheriting from the Game class and adding additional properties and methods specific to each game type.

**Include a sketch of sample output to illustrate your idea.**

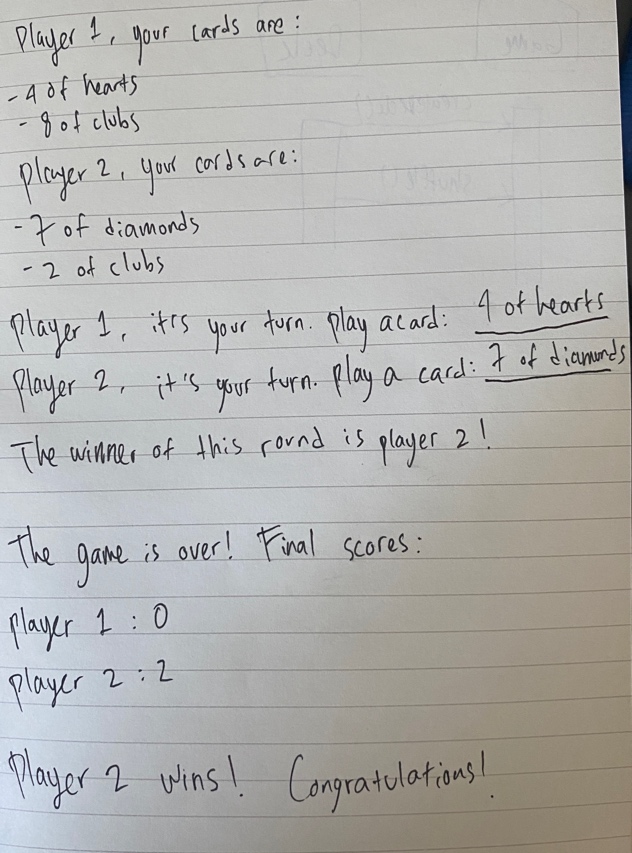


Figure 2: SecondPart

Text, letter

Description automatically generated

Figure 1: FirstPart

# Required Roles

Describe each of the classes, interfaces, and any enumerations you will create. Use a different table to describe each role you will have, using the following table templates.

Table 1: Player Details

|  |  |  |
| --- | --- | --- |
| **Responsibility** | **Type Details** | **Notes** |
| Hold cards | Field,  “List<Card>” | A list of “Card” objects representing the player’s hand |
| Play a card | Method,  “PlayCard(Card card)” | Takes a “Card” object as a parameter and removes it from the player’s hand |
| Score | Field,  “int” | The player’s current score |
| Name | Field,  “string” | The player’s name |

Table 2: Deck Details

|  |  |  |
| --- | --- | --- |
| **Responsibility** | **Type Details** | **Notes** |
| Hold cards | Field,  “List<Card>” | A list of “Card” objects representing the deck of cards |
| Shuffle | Method,  “void” | Shuffles the deck of cards |
| Deal | Method,  “List<Card>” | Deals a specified number of cards from the top of the deck |

Table 3: Card Details

|  |  |  |
| --- | --- | --- |
| **Responsibility** | **Type Details** | **Notes** |
| Value | Field,  “int” | The value of the card (2-10, Jack, Queen, King, Ace) |
| Suit | Field,  “string” | The suit of the card (Hearts, Diamonds, Clubs, Spades) |
| ToString | Method,  “string” | Returns a string representation of the card (“10 of hearts”, “Ace of spades” etc.) |

Table 4: CardValue enumeration details

|  |  |
| --- | --- |
| **Value** | **Notes** |
| 2-10 | The numerical value of the card |
| Jack | The face card with a value of 11 |
| Queen | The face card with a value of 12 |
| King | The face card with a value of 13 |
| Ace | The face card with a value of 14 |

Table 5: CardSuit enumeration details

|  |  |
| --- | --- |
| **Value** | **Notes** |
| Hearts | The suit of hearts |
| Diamonds | The suit of diamonds |
| Clubs | The suit of clubs |
| Spades | The suit of spades |

# Class Diagram

Provide an initial design for your program in the form of a class diagram.

Diagram

Description automatically generated

There are four classes in this illustration: Game, Deck, Card, and Player. The Game class is a representation of the entire game and includes functions for the game's beginning, play, and conclusion. A deck of cards is represented by the Deck class, which also provides dealing and shuffling methods. The Card class, which represents a single playing card, includes a ToString() method for displaying the card as a string as well as methods for determining the value and suit of the card. The Player class, which represents a player in the game and provides methods for adding cards to hands, playing cards from hands, and determining the number of hands a player has, is the last class mentioned.

# Sequence Diagram

Chart

Description automatically generated

Since the main functionality of the game is to deal cards to the players, shuffle the deck when necessary, and determine the winner of the game, I created a sequence diagram that shows the process of dealing cards to the player. This involves the ‘Game’ class and the ‘Deck’ class, as well as the ‘Player’ and ‘Hand’ classes.