Table of Contents

Problem Statement

Introduction

Console Card Game is just that, a card game that can be played on the console. The point of creating this game is to learn about OOD, more specifically, the use of Polymorphism as well as test-driven development. At the end of this exercise, I should have a better understanding of software design.

The Game

There are three different games that can be played in Console Card Game they are:

- Ten uses a 13-card board. Pairs of cards whose point values add to 10 are selected and removed, as are quartets of kings, queens, jacks, and tens, all of the same rank.
- Eleven uses a 11-card board. Pairs of cards whose point values add to 11 are selected and removed, as are face card of kings, queens, and jacks.
- Thirteen uses a 10-card board. Ace, 2, …, 10, jack, queen correspond to the point values of 1, 2, …, 10, 11, 12. Pairs of cards whose point values add up to 13 are selected and removed. Kings are selected and removed singly.

The only unique feature that each game has is their special case to remove one or more cards.

Implemtation Design

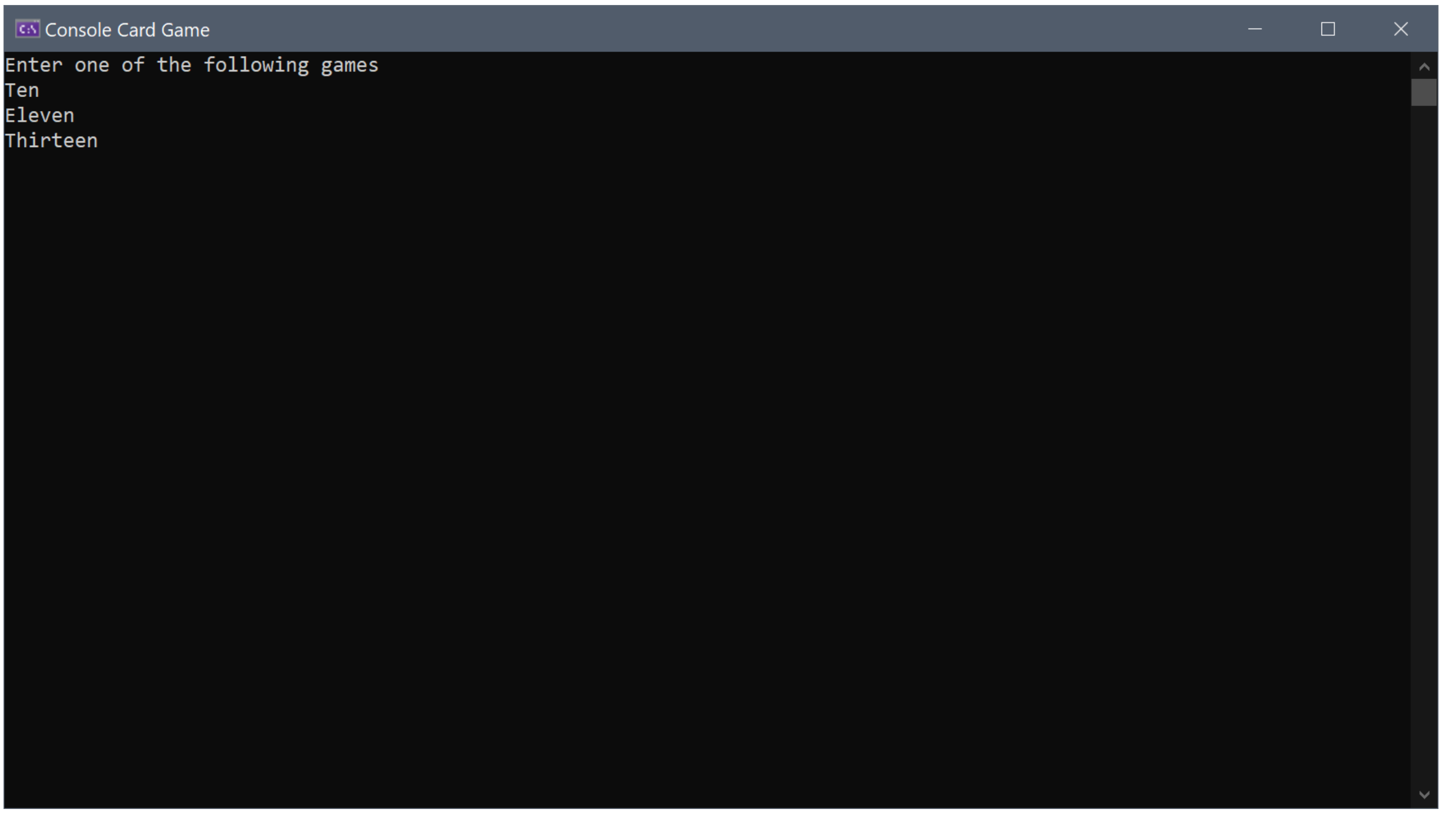
1. The game will create a deck and shuffle it
2. The game will deal from the deck several cards (depending on the type of game) to the board and allow the player to select any number of cards.
3. The player will select two cards from the board, the sum of which should depend on the type of game; for example, if the game is thirteen, then the sum of two cards will be thirteen.
4. The game will validate the value of the sum. If it's less than or greater than the game type, then no cards will be removed. If it is exactly the game type, the game will remove the two cards.
5. The game will identify the presence of a unique case to remove cards. For example, in eleven King, Queen, and Jack are all selected together on the board, the game will allow the player to remove three cards instead of two.
6. At any point in the game, the player can replace all cards up to the maximum number of cards that can be delt on the board. For example, in the game Thirteen, the maximum amount cards that can be delt is ten.
7. The game is won if there are no more cards in the deck or on the board.
8. If there is no possible way to remove cards on the board, the player can forfeit.
9. If the player has won, then the game will add the player name to the leaderboard. At this point, the player will have the option to change their name. This is not the only time the player can change their name at any point; they will be allowed to change their name.
10. The game will include a HighScore to keep track of the number of wins, losses.

How To Play

Start Game

To start the game type one of the three games.

*note: the game is not case sensitive.

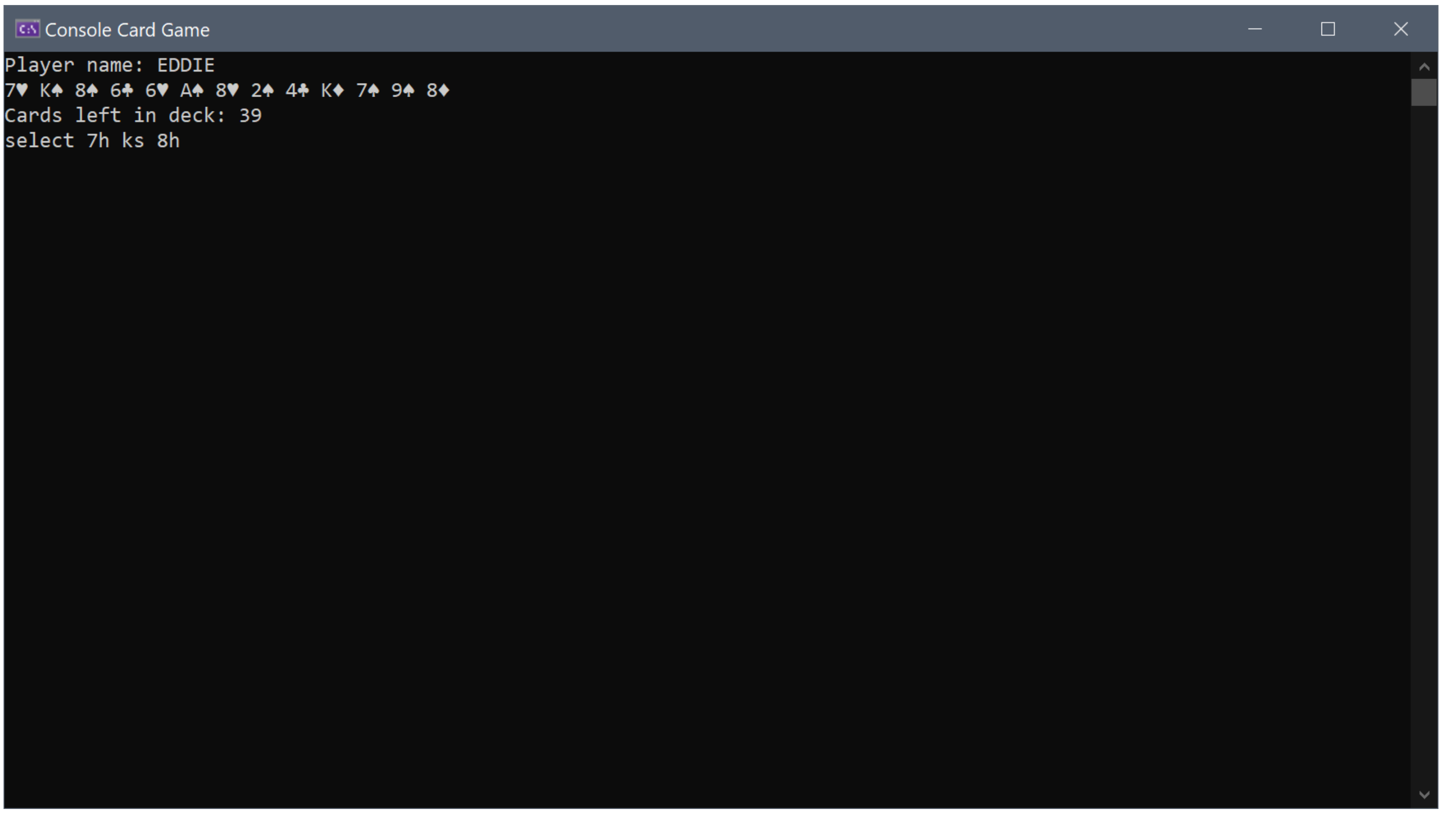


```
Enter one of the following games
Ten
Eleven
Thirteen
```

Select Cards / Unselect Cards

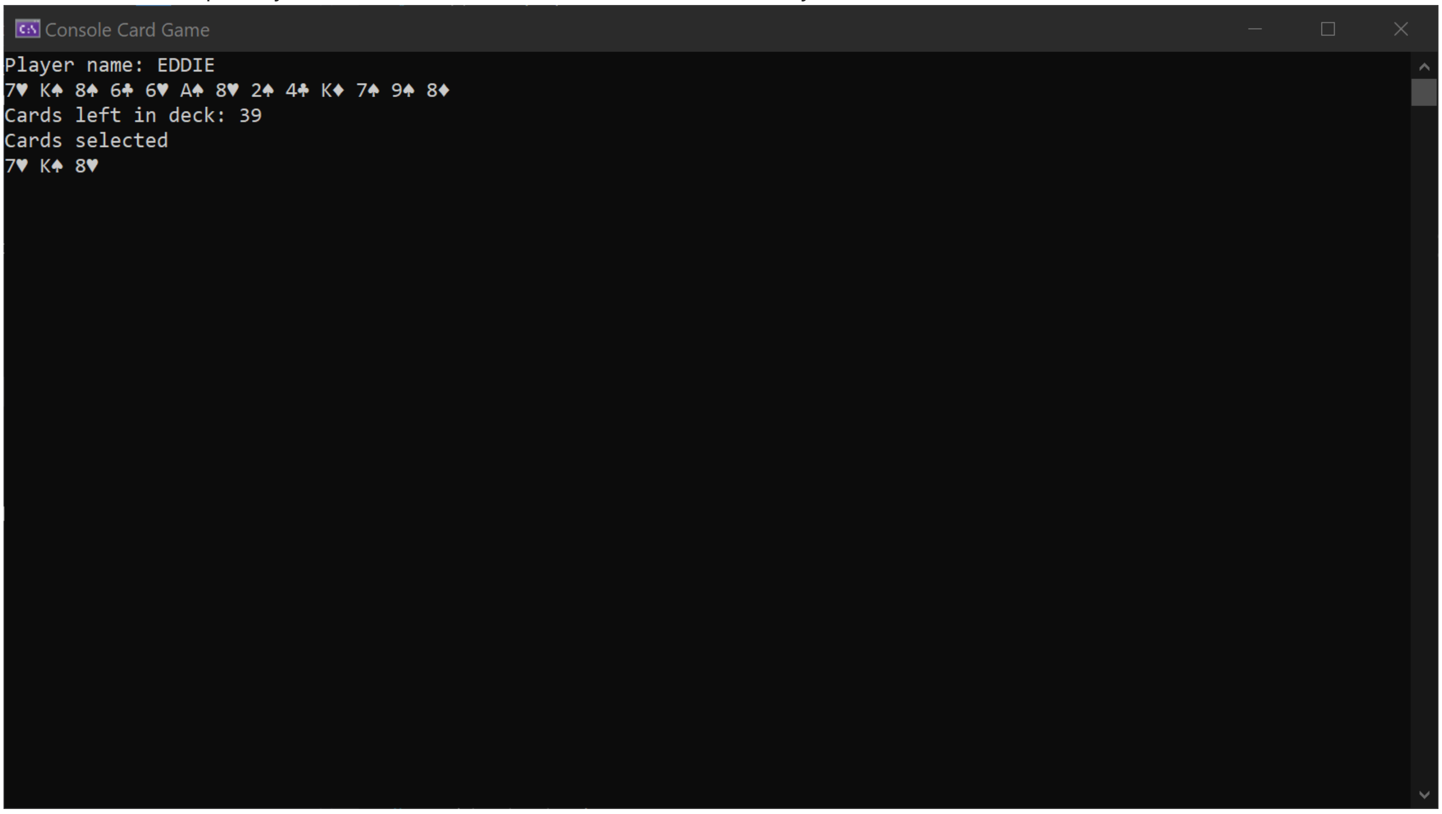
To select a card type select with two charchter that represent a card. To select a card type, select two characters that represent a card. For example, if I needed 7 of hearts and 4 of diamonds, it would be **select 7h 4d**

Face cards are A for ace, T for Ten, J for Jack, Q for Queen, K for King



```
Player name: EDDIE
7♥ K♠ 8♠ 6♣ 6♥ A♠ 8♥ 2♠ 4♣ K♦ 7♠ 9♠ 8♦
Cards left in deck: 39
select 7h ks 8h
```

After you have selected a card, it will display on the console. To unselect a card, you have to type the card once again using two characters. At this point, you can also select a card and unselect a card if you want.



```
Player name: EDDIE
7♥ K♠ 8♠ 6♣ 6♥ A♠ 8♥ 2♠ 4♣ K♦ 7♠ 9♠ 8♦
Cards left in deck: 39
Cards selected
7♥ K♠ 8♥
```

Remove Cards

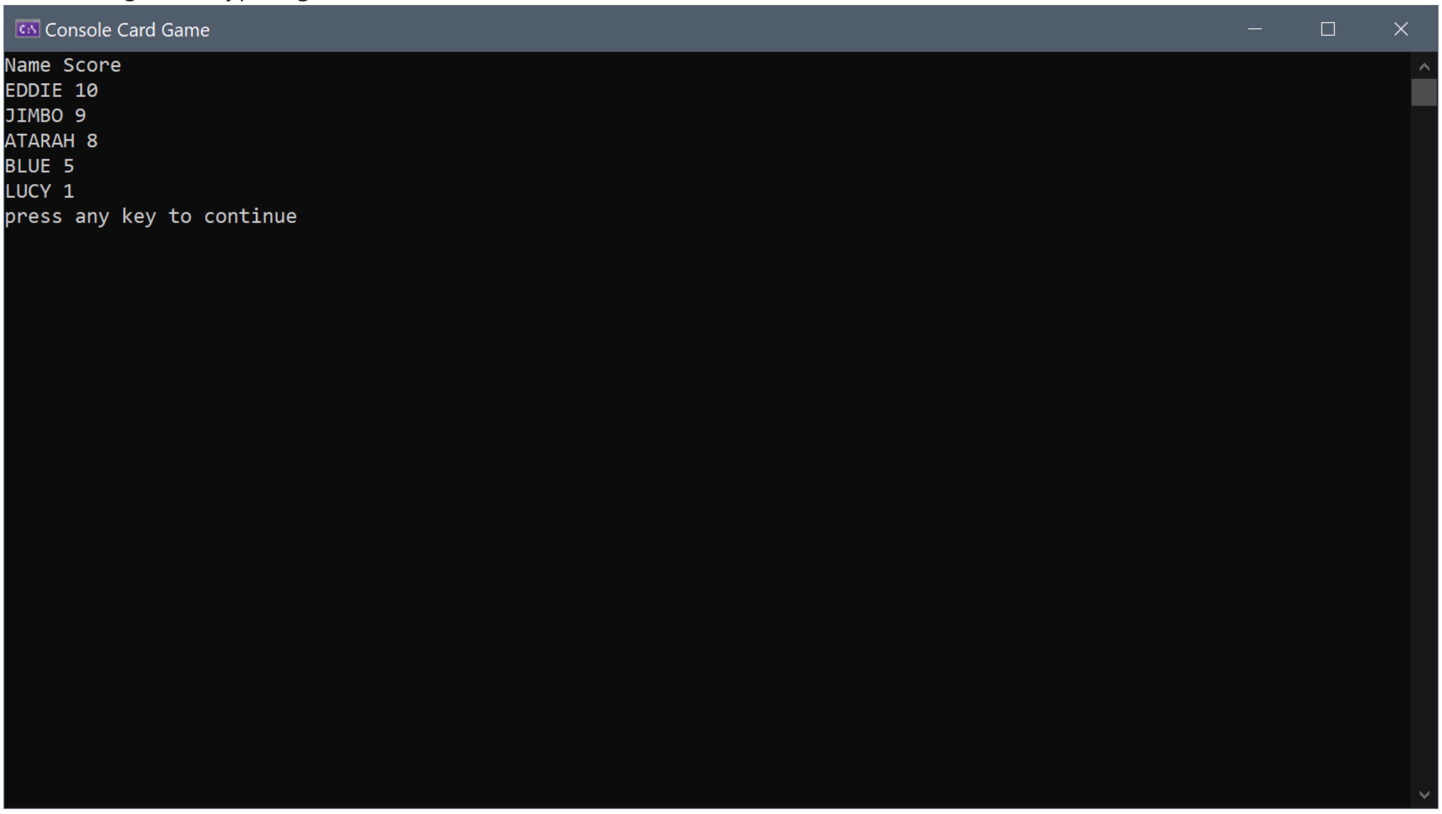
To remove a card type **remove**

Replace Cards

To Replace a card type **replace**

HighScore

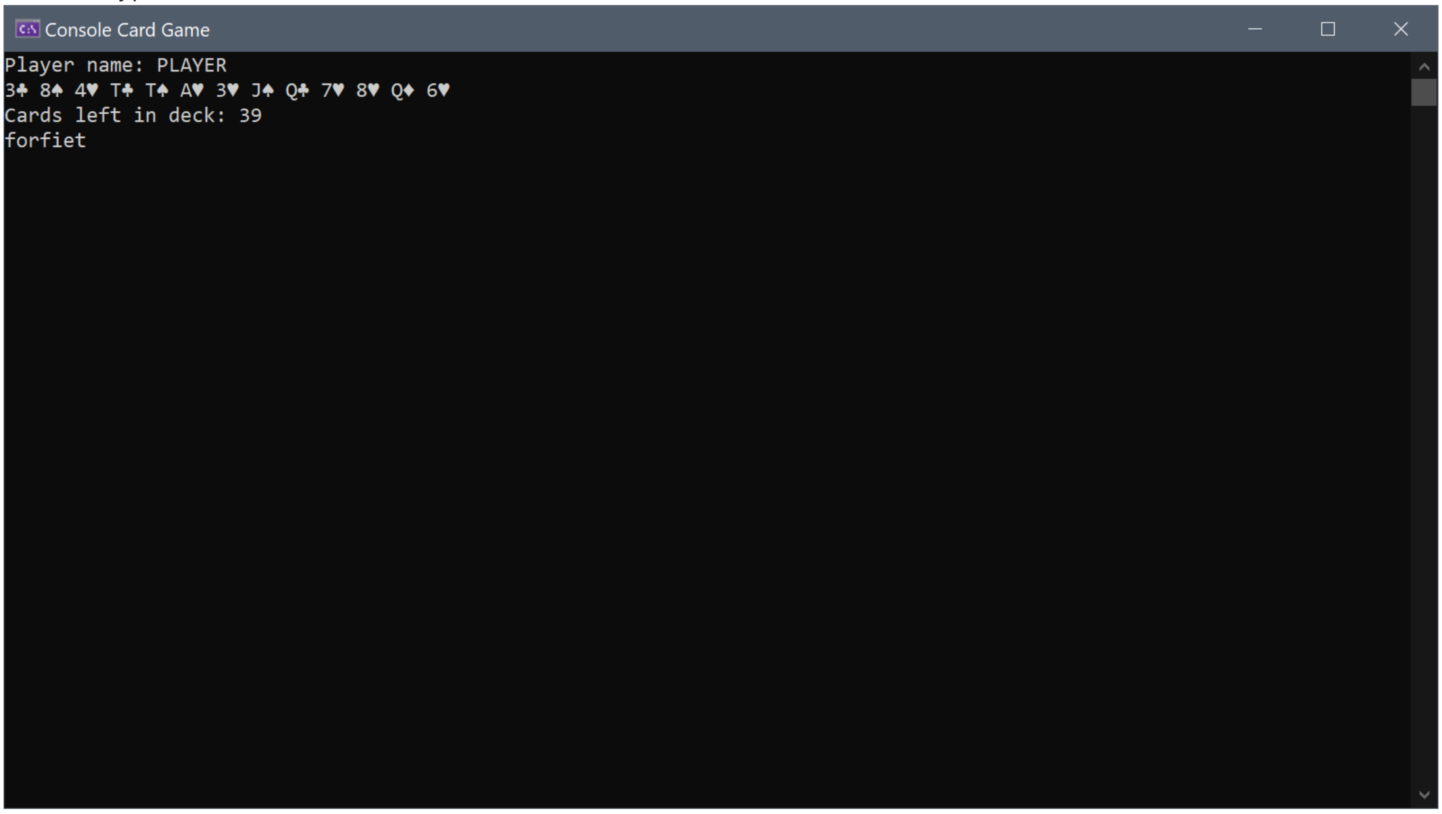
To show highscore type **highscore**



```
Name Score
EDDIE 10
JIMBO 9
ATARAH 8
BLUE 5
LUCY 1
press any key to continue
```

Forfiet

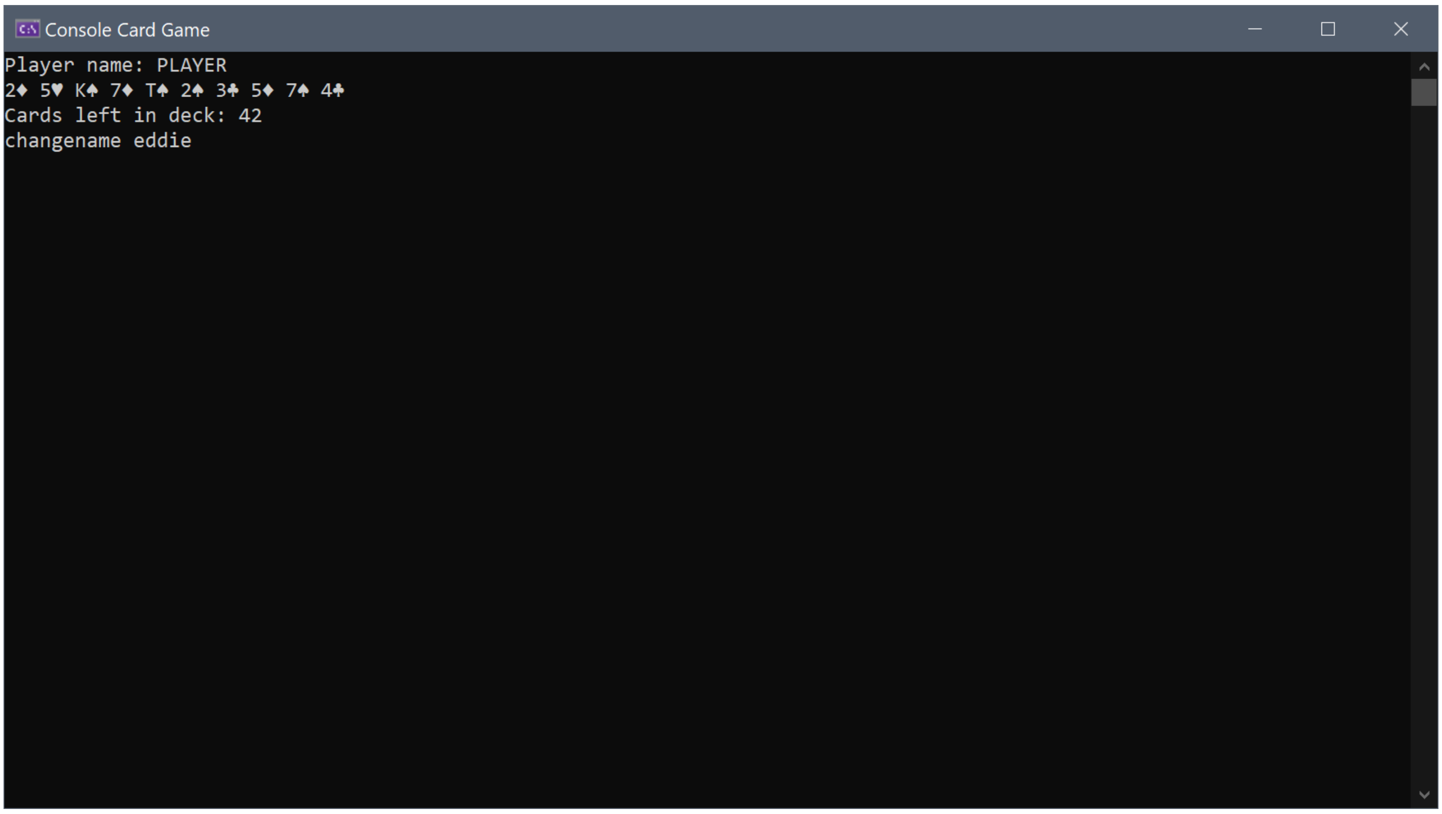
To forfiet type **forfiet**



```
Player name: PLAYER
3♣ 8♠ 4♥ T♣ T♠ A♥ 3♥ J♠ Q♣ 7♥ 8♥ Q♦ 6♥
Cards left in deck: 39
forfiet
```

Change Name

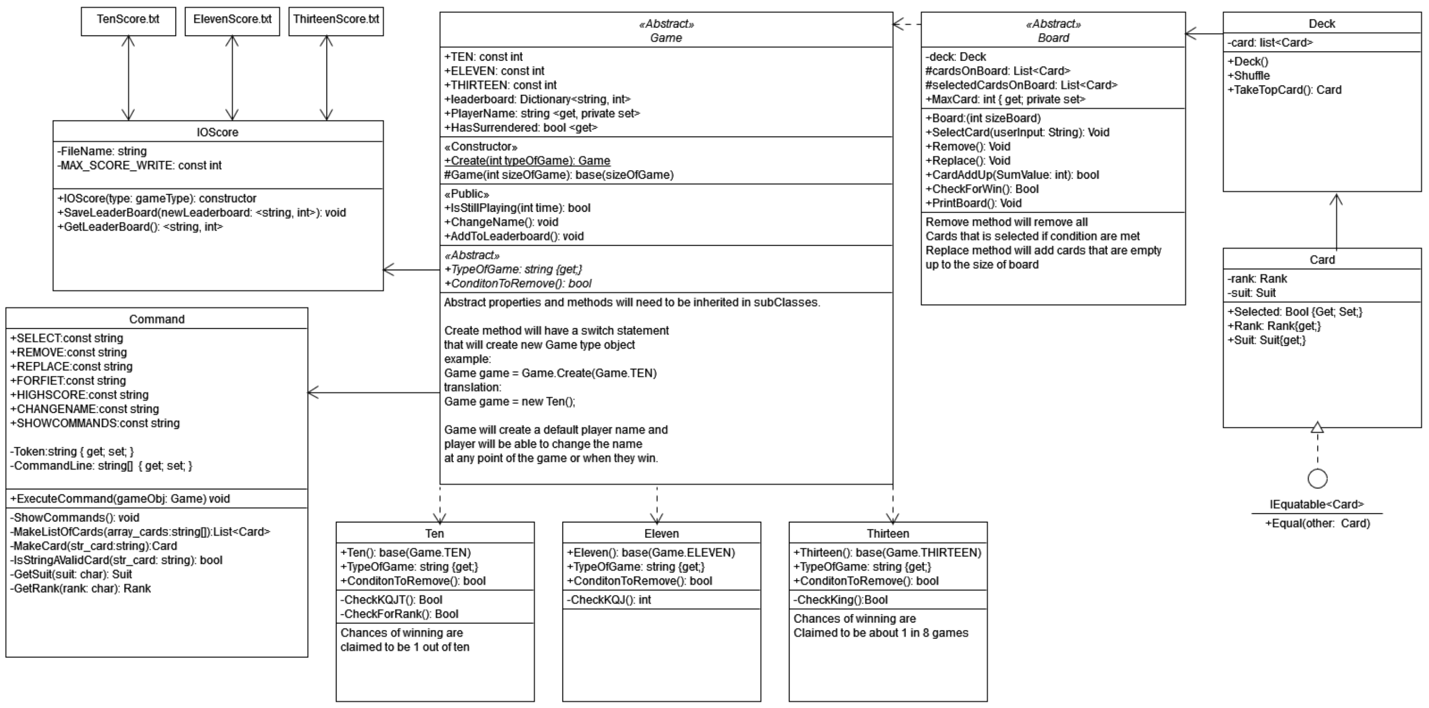
to change your name type changename with your new name Ex: **changename eddie**



Console Card Game

```
Player name: PLAYER
2♦ 5♥ K♠ 7♦ T♠ 2♠ 3♣ 5♦ 7♠ 4♣
Cards left in deck: 42
changename eddie
```

UML Design


TenScore.txt
ElevenScore.txt
ThirteenScore.txt
IOScore
-FileName: string
-MAX_SCORE_WRITE: const int
+IOScore(type: gameType): constructor
+SaveLeaderBoard(newLeaderboard: <string, int>): void
+GetLeaderBoard(): <string, int>
Command
+SELECT:const string
+REMOVE:const string
+REPLACE:const string
+FORFIET:const string
+HIGHSCORE:const string
+CHANGENAME:const string
+SHOWCOMMANDS:const string
-Token:string { get; set; }
-CommandLine: string[] { get; set; }
+ExecuteCommand(gameObj: Game) void
-ShowCommands(): void
-MakeListOfCards(array_cards:string[]):List<Card>
-MakeCard(str_card:string):Card
-IsStringAValidCard(str_card: string): bool
-GetSuit(suit: char): Suit
-GetRank(rank: char): Rank
«Abstract»
Game
+TEN: const int
+ELEVEN: const int
+THIRTEEN: const int
+leaderboard: Dictionary<string, int>
+PlayerName: string <get, private set>
+HasSurrendered: bool <get>
«Constructor»
+Create(int typeOfGame): Game
#Game(int sizeOfGame): base(sizeOfGame)
«Public»
+IsStillPlaying(int time): bool
+ChangeName(): void
+AddToLeaderboard(): void
«Abstract»
+TypeOfGame: string {get;}
+ConditonToRemove(): bool
Abstract properties and methods will need to be inherited in subClasses.
Create method will have a switch statement
that will create new Game type object
example:
Game game = Game.Create(Game.TEN)
translation:
Game game = new Ten();
Game will create a default player name and
player will be able to change the name
at any point of the game or when they win.
«Abstract»
Board
-deck: Deck
#cardsOnBoard: List<Card>
#selectedCardsOnBoard: List<Card>
+MaxCard: int { get; private set>
+Board:(int sizeBoard)
+SelectCard(userInput: String): Void
+Remove(): Void
+Replace(): Void
+CardAddUp(SumValue: int): bool
+CheckForWin(): Bool
+PrintBoard(): Void
Remove method will remove all
Cards that is selected if condition are met
Replace method will add cards that are empty
up to the size of board
Deck
-card: list<Card>
+Deck()
+Shuffle
+TakeTopCard(): Card
Card
-rank: Rank
-suit: Suit
+Selected: Bool {Get; Set;}
+Rank: Rank{get;}
+Suit: Suit{get;}
IEquatable<Card>
+Equal(other: Card)
Ten
+Ten(): base(Game.TEN)
+TypeOfGame: string {get;}
+ConditonToRemove(): bool
-CheckKQJT(): Bool
-CheckForRank(): Bool
Chances of winning are
claimed to be 1 out of ten
Eleven
+Eleven(): base(Game.ELEVEN)
+TypeOfGame: string {get;}
+ConditonToRemove(): bool
-CheckKQJ(): int
Thirteen
+Thirteen(): base(Game.THIRTEEN)
+TypeOfGame: string {get;}
+ConditonToRemove(): bool
-CheckKing():Bool
Chances of winning are
Claimed to be about 1 in 8 games

Namespace CardGame

Classes

Board

Creates a abstract board that reprsents a board of cards

Card

A class that represents a playing card

Command

Command class that contains a list of commands for user

Deck

A class that represents a deck of playing cards

Game

A Abstract class that can represent one of the three game Ten, Eleven, Thirteen

IOScore

Enums

Rank

Enums that represent ran

Suit

Enums that represents suits

Class Board

Creates a abstract board that reprsents a board of cards

Inheritance

System.Object
Board
Game

Namespace: CardGame

Assembly: Console Card Game.dll

Syntax

```
public abstract class Board
```

Constructors

Board(Int32)

Declaration

```
protected Board(int sizeOfBoard)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	sizeOfBoard	

Properties

CardsOnBoard

Declaration

```
protected List<Card> CardsOnBoard { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<Card>	

MaxCard

Declaration

```
protected int MaxCard { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

SelectedCardsOnBoard

Declaration

```
protected List<Card> SelectedCardsOnBoard { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<Card>	

Methods

CardAddUp(Int32)

Declaration

```
protected bool CardAddUp(int sumOfCards)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	sumOfCards	

Returns

TYPE	DESCRIPTION
System.Boolean	

HasWon()

Checks if user has won the game

Declaration

```
public bool HasWon()
```

Returns

TYPE	DESCRIPTION
System.Boolean	True if the deck is empty and there are no cards on the board

printBoard()

Prints to the console all cards on the board and remaining cards in the deck.

Declaration

```
public void printBoard()
```

Remove()

Removes all cards that are selected

Declaration

```
public void Remove()
```

Replace()

Replace cards on the board till it shows the maximum cards that should be on the board

Declaration

```
public void Replace()
```

Reset()

Resets the board

Declaration

```
public void Reset()
```

SelectCard(List<Card>)

Cards that are on the board that match the list 'cardsToBeSelected' will be selected

Declaration

```
public void SelectCard(List<Card> cardsToBeSelected)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.List<Card>	cardsToBeSelected	Card that the user wishes to select

Class Card

A class that represents a playing card

Inheritance

System.Object
Card

Implements

System.IEquatable<Card>

Namespace: CardGame

Assembly: Console Card Game.dll

Syntax

```
public class Card : IEquatable<Card>
```

Constructors

Card(Rank, Suit)

Declaration

```
public Card(Rank rank, Suit suit)
```

Parameters

TYPE	NAME	DESCRIPTION
Rank	rank	
Suit	suit	

Properties

Rank

Declaration

```
public Rank Rank { get; }
```

Property Value

TYPE	DESCRIPTION
Rank	

Selected

Declaration

```
public bool Selected { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Suit

Declaration

```
public Suit Suit { get; }
```

Property Value

TYPE	DESCRIPTION
Suit	

Methods

Equals(Card)

checks if both cards are equal

Declaration

```
public bool Equals(Card other)
```

Parameters

TYPE	NAME	DESCRIPTION
Card	other	Other card

Returns

TYPE	DESCRIPTION
System.Boolean	true if rank and suit are equal to each other

SelectCard()

Selects this card

Declaration

```
public void SelectCard()
```

ToString()

Returns two string of the card

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
System.String	

Overrides

System.Object.ToString()

Implements

System.IEquatable<T>

Class Command

Command class that contains a list of commands for user

Inheritance

System.Object

Command

Namespace: CardGame

Assembly: Console Card Game.dll

Syntax

```
public class Command
```

Fields

CHANGENAME

Declaration

```
public const string CHANGENAME = "CHANGENAME"
```

Field Value

TYPE	DESCRIPTION
System.String	

FORFIET

Declaration

```
public const string FORFIET = "FORFIET"
```

Field Value

TYPE	DESCRIPTION
System.String	

HIGHSCORE

Declaration

```
public const string HIGHSCORE = "HIGHSCORE"
```

Field Value

TYPE	DESCRIPTION
System.String	

REMOVE

Declaration

```
public const string REMOVE = "REMOVE"
```

Field Value

TYPE	DESCRIPTION
System.String	

REPLACE

Declaration

```
public const string REPLACE = "REPLACE"
```

Field Value

TYPE	DESCRIPTION
System.String	

SELECT

Declaration

```
public const string SELECT = "SELECT"
```

Field Value

TYPE	DESCRIPTION
System.String	

SHOWCOMMANDS

Declaration

```
public const string SHOWCOMMANDS = "SHOWCOMMANDS"
```

Field Value

TYPE	DESCRIPTION
System.String	

Methods

ExecuteCommand(Game)

Excutes the command issued by the player

Declaration

```
public void ExecuteCommand(Game gameObj)
```

Parameters

TYPE	NAME	DESCRIPTION
Game	gameObj	The current game

GetCommand(String)

converts a string input into a token and commands

Declaration

```
public void GetCommand(string input)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	input	

Class Deck

A class that represents a deck of playing cards

Inheritance

System.Object
Deck

Namespace: CardGame

Assembly: Console Card Game.dll

Syntax

```
public class Deck
```

Constructors

Deck()

Create a new deck

Declaration

```
public Deck()
```

Properties

Count

Gets the count of the remaining cards in the deck.

Declaration

```
public int Count { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

IsEmpty

Returns true or false if deck is empty

Declaration

```
public bool IsEmpty { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

Shuffle()

Shuffle the deck

Declaration

```
public void Shuffle()
```

TakeTopCard()

Takes the top card of the deck and removes it from the list

Declaration

```
public Card TakeTopCard()
```

Returns

TYPE	DESCRIPTION
Card	Returns a Card object of the top card

Class Game

A Abstract class that can represent one of the three game Ten, Eleven, Thirteen

Inheritance

System.Object
Board
Game

Inherited Members

Board.CardsOnBoard
Board.SelectedCardsOnBoard
Board.MaxCard
Board.Reset()
Board.Remove()
Board.Replace()
Board.SelectCard(List<Card>)
Board.CardAddUp(Int32)
Board.HasWon()
Board.printBoard()

Namespace: CardGame

Assembly: Console Card Game.dll

Syntax

```
public abstract class Game : Board
```

Constructors

Game(Int32)

Declaration

```
protected Game(int sizeOfBoard)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	sizeOfBoard	

Fields

ELEVEN

Game Eleven board size is 11

Declaration

```
public const int ELEVEN = 11
```

Field Value

TYPE	DESCRIPTION
System.Int32	

TEN

Game Ten board size is 13

Declaration

```
public const int TEN = 10
```

Field Value

TYPE	DESCRIPTION
System.Int32	

THIRTEEN

Game Thirteenis board size is 10

Declaration

```
public const int THIRTEEN = 13
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

GameType

gets the type of game

Declaration

```
public abstract int GameType { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

HasSurrendered

Gets the boolean value if the player has surrenered

Declaration

```
public bool HasSurrendered { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Leaderboard

Declaration

```
public Dictionary<string, int> Leaderboard { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.Int32>	

PlayerName

Declaration

```
public string PlayerName { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Methods

AddToLeadrBoard()

Adds player to the leaderboard

Declaration

```
public void AddToLeadrBoard()
```

ChangeName(String)

Changes player name

Declaration

```
public void ChangeName(string newName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	newName	player new name

ConditionToRemove()

Collects all the selected cards anc check if the conditions of the game allows for player to remove the cards

Declaration

```
public abstract bool ConditionToRemove()
```

Returns

TYPE	DESCRIPTION
System.Boolean	Returns true if cards can be removed

Create(String)

Factory method that creates an instancei of a subclass of game

Declaration

```
public static Game Create(string gameType)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	gameType	The type of game the player wants to play

Returns

TYPE	DESCRIPTION
Game	Subclass Game Object

Forfiet()

Player can choose to surrrender

Declaration

```
public void Forfiet()
```

IsStillPlaying(Int32)

Checks if player is still playing after 10s the player is checked as surrenered

Declaration

```
public void IsStillPlaying(int time)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	time	

PrintHighScore()

Declaration

```
public void PrintHighScore()
```

Class IOScore

Inheritance

System.Object
IOScore

Namespace: CardGame

Assembly: Console Card Game.dll

Syntax

```
public class IOScore
```

Constructors

IOScore(Int32)

Declaration

```
public IOScore(int gameType)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	gameType	

Properties

filePath

Declaration

```
public string filePath { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Methods

GetLeaderBoard()

Declaration

```
public Dictionary<string, int> GetLeaderBoard()
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.Int32>	

SaveLeaderBoard(Dictionary<String, Int32>)

Declaration

```
public void SaveLeaderBoard(Dictionary<string, int> newLeaderboard)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.Int32>	newLeaderboard	

Enum Rank

Enums that represent ran

Namespace: CardGame

Assembly: Console Card Game.dll

Syntax

```
public enum Rank
```

Fields

NAME	DESCRIPTION
Ace	
Eight	
Five	
Four	
Jack	
King	
Nine	
Queen	
Seven	
Six	
Ten	
Three	
Two	

Enum Suit

Enums that represents suits

Namespace: CardGame

Assembly: Console Card Game.dll

Syntax

```
public enum Suit
```

Fields

NAME	DESCRIPTION
Clubs	
Diamonds	
Hearts	
Spade	