Casino

Generated by Doxygen 1.11.0

Chapter 1

Casino

Simple practising tool meant for fun and as a proof of concept

TODO list:

- Create games
  - ~~Black Jack~~
  - Poker
  - ~~Dices~~
  - Roulette
  - ~~Double or Nothing~~
- ~~Add currence/balance~~
- ~~Add the ability to save your balance for later log in~~
- **GUI :)**

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 BJ Struct Reference

Public Member Functions

- **BJ** (**Player** &player)

  *Constructs a new instance of the **BJ** (p. 4) class.*
- void **startGame** (**Player** &player)

  Starts the BlackJack game.
- void **rndCard** (int[ ], **Hand** &)
- void **giveCards** (int[ ], **Hand** &, **Hand** &)
- void **showCards** (**Hand** &, **Hand** &)
- void **printCard** (int)
- void **calculateScore** (**Hand** &)
- void **dealersMove** (int[ ], **Hand** &, **Hand** &)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 BJ()

```
BJ::BJ (
            Player & player)
```

Constructs a new instance of the **BJ** (p. 4) class.

This constructor initializes a new instance of the **BJ** (p. 4) class and starts the game for the specified player.

Parameters

player	The player object for whom the game is being started.

4.1.2 Member Function Documentation

4.1.2.1 startGame()

```
void BJ::startGame (
            Player & player)
```

Starts the BlackJack game.

This function allows the player to play the BlackJack game. It takes a reference to a **Player** (p. 9) object as a parameter. The function prompts the player to enter their bet, deals the cards, and allows the player to make choices (hit or stand). After the player's turn, the function determines the outcome of the game and updates the player's balance accordingly. The game continues until the player runs out of balance or chooses to quit. At the end of the game, the function displays the total wins and losses of the player.

Parameters

player	The **Player** (p. 9) object representing the player.

The documentation for this struct was generated from the following files:

- BJ.h
- BJ.cpp

4.2 Dice Struct Reference

Public Member Functions

- **Dice** (**Player** &player)

  *Constructs a new instance of the **Dice** (p. 5) class.*
- void **startGame** (**Player** &player)

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Dice()

```
Dice::Dice (
            Player & player)
```

Constructs a new instance of the **Dice** (p. 5) class.

This constructor initializes a new instance of the **Dice** (p. 5) class and starts the game for the specified player.

Parameters

player	The player object for whom the game is being started.

4.2.2 Member Function Documentation

4.2.2.1 startGame()

```
void Dice::startGame (
            Player & player)
```

Starts the dice game.

This function allows the player to play the dice game. It takes a reference to a **Player** (p. 9) object as a parameter. The player is prompted to enter their bet and choose what to bet on (red, blue, or tie). The dice are rolled and the outcome is determined. If the player wins, their balance is increased and the winnings are displayed. If the player loses, their balance is decreased and a message is displayed. The player can choose to play again or quit the game.

Parameters

player	The player object for whom the game is being started.

The documentation for this struct was generated from the following files:

- Dice.h
- Dice.cpp

4.3 DorN Struct Reference

Public Member Functions

- **DorN** (**Player** &player)

  *Constructs a new instance of the **DorN** (p. 6) (Double or nothing) class.*
- void **startGame** (**Player** &player)

4.3.1 Constructor & Destructor Documentation

4.3.1.1 DorN()

```
DorN::DorN (
            Player & player)
```

Constructs a new instance of the **DorN** (p. 6) (Double or nothing) class.

This constructor initializes a new instance of the **DorN** (p. 6) class and starts the game for the specified player.

Parameters

player	The player object for whom the game is being started.

4.3.2 Member Function Documentation

4.3.2.1 startGame()

```
void DorN::startGame (
            Player & player)
```

Starts the Double or Nothing game.

This function allows the player to play the Double or Nothing game. The player is prompted to enter their bet and choose heads or tails. The outcome is determined randomly, and the player's balance is updated accordingly. The game continues until the player's balance reaches zero or the player chooses to exit. At the end of the game, the total number of wins and losses is displayed.

Parameters

player	The player object for whom the game is being started.

The documentation for this struct was generated from the following files:

- DorN.h
- DorN.cpp

4.4 Hand Class Reference

Public Member Functions

- **Hand** ()

  *Constructs a new instance of the **Hand** (p. 8) class. used to keep track of the player's hand in the game of BlackJack.*

Public Attributes

- int **nCards** [10]
- int **aces**
- int **score**
- int **altScore**
- int **n**
- bool **end**

The documentation for this class was generated from the following files:

- Hand.h
- Hand.cpp

4.5 Login Struct Reference

Public Member Functions

- tuple< std::string, double, int, int > **getPlayer** ()

  Reads the player data from the CSV file.
- void **changeCSV** (double bet, int additionW, int additionL)
- void **printList** ()

4.5.1 Member Function Documentation

4.5.1.1 changeCSV()

```
void Login::changeCSV (
            double bet,
            int additionW,
            int additionL)
```

Changes the CSV according to the input of +- bet which reduces or increases the **Player** (p. 9)'s account balance. Additionaly increments wins or loses in the csv file.

4.5.1.2 getPlayer()

```
tuple< std::string, double, int, int > Login::getPlayer ()
```

Reads the player data from the CSV file.

This function reads the player data from the CSV file and stores it in the respective vectors. the **Player** (p. 9) is prompted to enter their username, if the username is not in the names vector the code repeats until a recognised username is entered.

4.5.1.3 printList()

```
void Login::printList ()
```

Replaces the current csv file with an updated one using vectors of names, balance, wins and losses.

The documentation for this struct was generated from the following files:

- Login.h
- Login.cpp

4.6 Player Class Reference

Public Member Functions

- **Player** ()

  Player *(p. 9) class to store player data of balance, wins and losses.*
- **Player** (std::string initialName, int initialBalance, int initialWins, int initialLosses)
- int **getBalance** () const
- int **getWins** () const
- int **getLosses** () const
- const std::string & **getName** () const
- void **setName** (const std::string &newName)
- void **setBalance** (int newBalance)
- void **setWins** (int newWins)
- void **setLosses** (int newLosses)
- void **increaseBalance** (int amount)
- void **decreaseBalance** (int amount)
- void **incrementWins** ()
- void **incrementLosses** ()

The documentation for this class was generated from the following files:

- Player.h
- Player.cpp

Chapter 5

File Documentation

5.1 BJ.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <cstdlib>
00004 #include <ctime>
00005 #include "Player.h"
00006 #include "Hand.h"
00007
00008 struct BJ
00009 {
00010 public:
00011     BJ(Player& player);
00012     ~BJ();
00013     void startGame(Player& player);
00014     void rndCard(int[], Hand&);
00015     void giveCards(int[], Hand&, Hand&);
00016     void showCards(Hand&, Hand&);
00017     void printCard(int);
00018     void calculateScore(Hand&);
00019     void dealersMove(int[],Hand&, Hand&);
00020 };
00021 #pragma once
```

5.2 Dice.h

```
00001 #include "Player.h"
00002 #include <iostream>
00003
00004 struct Dice{
00005     public:
00006         Dice(Player& player);
00007         ~Dice();
00008
00009         void startGame(Player& player);
00010 };
```

5.3 DorN.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <cstdlib>
00004 #include <ctime>
00005 #include "Player.h"
00006
00007 struct DorN
00008 {
00009 public:
00010     DorN(Player& player);
00011     ~DorN();
00012     void startGame(Player& player);
00013 };
```

5.4 Hand.h

```
00001 #ifndef HAND_H
00002 #define HAND_H
00003
00004 #include <string>
00005
00006 class Hand{
00007 public:
00008     Hand();
00009     int nCards[10];
00010     int aces;
00011     int score;
00012     int altScore;
00013     int n;
00014     bool end;
00015 };
00016
00017 #endif
```

5.5 Login.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <cstdlib>
00004 #include <ctime>
00005 #include <vector>
00006 #include "Player.h"
00007
00008 using namespace std;
00009
00010 struct Login
00011 {
00012 private:
00013     int counter = 0;
00014     int playerNum;
00015     vector<string> names;
00016     vector<double> balance;
00017     vector<int> wins;
00018     vector<int> losses;
00019 public:
00020     tuple<std::string, double, int, int> getPlayer();
00021     void changeCSV(double bet, int additionW, int additionL);
00022     void printList();
00023 };
```

5.6 Player.h

```
00001 #ifndef PLAYER_H
00002 #define PLAYER_H
00003
00004 #include <string>
00005
00006 class Player {
00007 public:
00008     Player();
00009     Player(std::string initialName, int initialBalance, int initialWins, int initialLosses);
00010
00011     int getBalance() const;
00012     int getWins() const;
00013     int getLosses() const;
00014     const std::string& getName() const;
00015
00016     void setName(const std::string& newName);
00017     void setBalance(int newBalance);
00018     void setWins(int newWins);
00019     void setLosses(int newLosses);
00020
00021     void increaseBalance(int amount);
00022     void decreaseBalance(int amount);
00023     void incrementWins();
00024     void incrementLosses();
00025
00026 private:
00027     std::string name;
00028     int balance;
00029     int wins;
00030     int losses;
00031 };
00032
00033 #endif
```

Index