**ТЕХНИЧЕСКА ДОКУМЕНТАЦИЯ НА ПРОЕКТ JOKER POKER**

******

***Изготвили:***

*Иван Иванов*

*Росица Ташева*

*Любомир Велев*

*Боряна Родопска*

Съдържание

[Задание: Joker Poker 2](#_Toc488834437)

[Обща струткура 3](#_Toc488834438)

[Диаграма 3](#_Toc488834439)

[Функционално описание на приложението 4](#_Toc488834440)

[Клас Game 4](#_Toc488834441)

[Клас Texture 4](#_Toc488834442)

[Клас ButtonObject 5](#_Toc488834443)

[Клас PaytableObject 5](#_Toc488834444)

[Клас Card 5](#_Toc488834445)

[Клас Deck 6](#_Toc488834446)

[Базов Клас Screen 6](#_Toc488834447)

[Клас Evaluation 8](#_Toc488834448)

[Клас Recovery 12](#_Toc488834449)

[Клас Music 12](#_Toc488834450)

[Изпълнение на функционалностите 13](#_Toc488834451)

[Клас Game 13](#_Toc488834452)

[Клас Texture 15](#_Toc488834453)

[Клас ButtonObject 16](#_Toc488834454)

[Клас PaytableObject 17](#_Toc488834455)

[Клас Card 18](#_Toc488834456)

[Клас Deck 18](#_Toc488834457)

[Клас Screen 20](#_Toc488834458)

[Клас Recovery 25](#_Toc488834459)

[Клас EvalFlush 26](#_Toc488834460)

[Клас EvalFourOfAKind 26](#_Toc488834461)

[Клас EvalFourToFlush 26](#_Toc488834462)

[Клас EvalFourToRoyalFlush 26](#_Toc488834463)

[Клас EvalFullHouse 27](#_Toc488834464)

[Клас EvalJokerOnly 27](#_Toc488834465)

[Клас EvalKingsOrBetter 27](#_Toc488834466)

[Клас EvalNaturalRoyalFlush 27](#_Toc488834467)

[Клас EvalStraight 27](#_Toc488834468)

[Клас EvalStraightFlush 28](#_Toc488834469)

[Клас EvalThreeOfKind 28](#_Toc488834470)

[Клас EvalThreeToRoyaiFlush 28](#_Toc488834471)

[Клас EvalTwoPair 28](#_Toc488834472)

[Клас EvalWildRoyalFlush 29](#_Toc488834473)

[Клас FourToStraightFlush 29](#_Toc488834474)

[Клас Music 29](#_Toc488834475)

# Обща струткура

## Диаграма

## C:\Users\Vaniugata\AppData\Local\Microsoft\Windows\INetCache\Content.Word\JokerPokerSctructure.png

# Задание: Joker Poker

Да се създаде игра от тип видео покер по стратегията Kings or Better със следните изисквания:

**Основни състояния:** Играта трябва да има пет основни състояния:

**Intro** – начален екран с няколко основни бутона за управление и/или настройка ).

**Game** – основна логика, специфична за конкретната игра. Това състояние може да се характеризира с други подсъстояния, математика, таблици на печалбите, анимации, бутони за управление и/или настройка и др.

**Win** – това е състоянието, в което се намира играта при печалба (може да бъде различно за различните типове игри).

**Bonus** – представлява състояние на допълнителна печалба, под формата на друга

игра и/или мистериозно натрупана сума.

**Outro** – в това състояние се влиза по инициатива на играча, след натискане на бутон CashOut. По този начин се изплащат текущите пари на играча.

**Recovery**: Способност за възстановяване – играта трябва да запази всички текущи

състояния и настройки в XML файл, за да може при евентуално аварийно събитие ( отпадане на захранването или каквото и да е събитие водещо до изключване на приложението ) играта да се възстанови там, от където е прекъсната.

Intro – всяка игра трябва в това си състояние да има следните бутони:

\* Insert Credit – с този бутон се добавят кредити в играта ( стойността, която ще се добави

може да бъде фиксирана или произволна )

\* Cashout – бутон за изплащане на парите натрупани в играта.

\* Volume – бутон за регулиране звука на играта

\* Info – бутон с кратка информация за правилата на играта и валутата плюс единствената

деноминация на която ще се играе. Ако играта има някакви специални фигури или бонуси, то те също трябва да бъдат описани в info страница. Бутонът е активен по всяко време. Всеки екип

трябва да избере една от следните деноминации: 0.01, 0.25, **5.00**, 2000.00 ( пример: ако избера

деноминация 0.25 BGN и вкарам 300 кредита от бутона Insert Credit и след това веднага

натисна бутона Cashout, сумата, която ще се изпише на поздравителния екран е 75 BGN ).

\* Start new game – чрез този бутон се нулират всички текущо запазени състояние и/или

настройки в XML файла, стартира се нова ( чиста ) игра и записването започва отново.

\* Resume game – с този бутон се зареждат всички текущо запазени настройки в XML

файла, като играта по този начин се възстановява там от където е спряна.

Game – {...специфично състояние за всяка игра...}

Win – {...специфично състояние за всяка игра...}

Bonus – {...специфично състояние за всяка игра...}

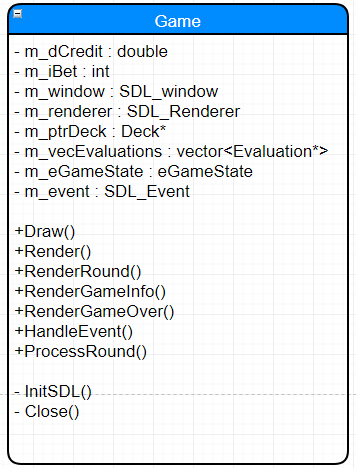
Outro – трябва да се покаже поздравителен екран със спечелената сума, конвертирана от

кредити в пари, според избраната деноминация, като този екран трябва да стои 10 секунди и

след това да се премине автоматично към екран Intro. Към това състояние се преминава с бутон CashOut от играта.

## Функционално описание на приложението

### Клас Game

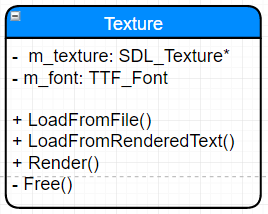


Приложението започва със създаване на инстанция от класа **Game**. Неговата роля е да бъде платформата, която да обединява всички отделни компоненти на проекта. Класът притежава и отговаря за прозореца на приложението и прилежащият му renderer. В **Game** се реализират всички функционалности свързани със самото игране на играта, което включва гейм логика и интерфейс.

### Клас Texture

Класът **Texture** е помощен клас. Той има общи функционалности за изрисуването на дадена текстура върху екрана. Може да изрисува изображения и текст заредени от файлове със съответните формати. **Тexture** се използва в във всички останали класове с изключение на класовете **Recovery**, **Music**, **Evaluation** и наследниците му.

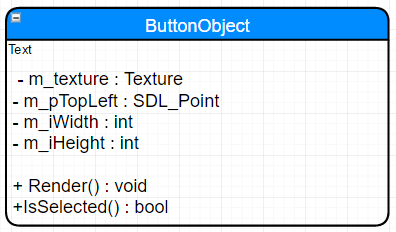
Забележка: В **Texture** няма логика за това кога едно изображение да се рисува на екарана, всеки клас който я използва съдържа в себе си тази логика и в даден момент извиква функционалността на **Texture** да нарисува изображението на екрана.



### Клас ButtonObject

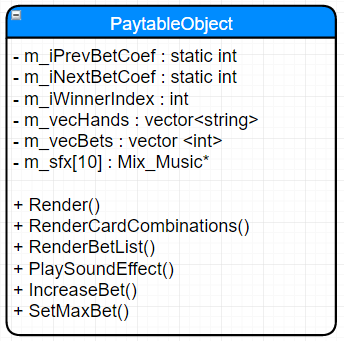
Класът **ButtonObject** е помощен клас. Той има общи функционалности за изрисуването на бутон от дадена текстура на екрана и информация за това дали бутонът е активен или не (курсорът на мишката да се намира в интервала на лицето му). **ButtonObject** използва функционалностите от **Texture** за да показва текстури на екрана.

Забележка: В **ButtonObject** няма логика за това каква функционалност да се изпълнява когато даден бутон е натиснат. Всеки клас който го използва съдържа в себе си тази логика и в момент на натиснат бутон тя бива извиквана.



### Клас PaytableObject

Класът **PaytableObject** се грижи за таблицата на залозите в играта. Той изобразява текстурата на самата таблица и данните в нея под формата на текст. В таблицата на залозите се съдържат всички възможни комбинации за печалба придружени от самата стойност на печалбата. В момент на печалба съответстващият ред се маркира с цвят различен от останалите и спечелената сума се прибавя към общия кредит за играта. В този клас има 2 бутона за промяна на текущия залог. Първият бутон увеличава коефициента на печалба с 1 и в таблицата се визуализират новите стойности на печалбите. Вторият бутон задава максималния коефициент за печалбите. За реализация на логиката се използват помощните класове **Texture** и **ButtonObject**.

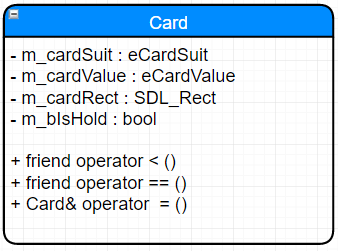


### Клас Card

Клас Card е част от системата за обработка на картите в играта. Той съдържа данни за идентификация на карта: боя, стойност, дали картата е задържана и дали е добра за **AutoHold** стратегията.

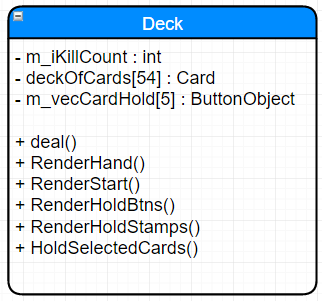
В Card има и функционалност за сравнение на 2 карти и за присвояване на една карта на друга.

Забележка: Повече информация за **AutoHold** в информацията за клас **Deck**.



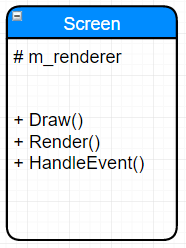
### Клас Deck

Клас **Deck** е част от системата за обработка на картите в играта. Той съдържа в себе си множество от всички карти и множество от 5 карти използвани при всеки етап на раздаване на ръка в играта. В класа има функционалности за раздаване на 5 карти на случаен принцип, визуализиране на картите, задържане и визуализиране на печеливши карти и затъмняване на непечелившите карти в ръката при край на играта. **Deck** използва обекти от тип **Card** за съхранение на цялото тесте карти и за ръката от 5 карти, обекти от тип **ButtonObject** и **Texture** за реализиране на логиката за операциите над картитe.



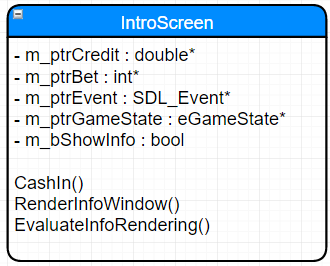
### Базов Клас Screen

Клас **Screen** е базов клас който представлява шаблон за представянето на различните състояния на играта. В него се задават функционалности за рисуване по екрана и обработка на събития, като тези функционалности биват имплементирани в наследниците на класа.



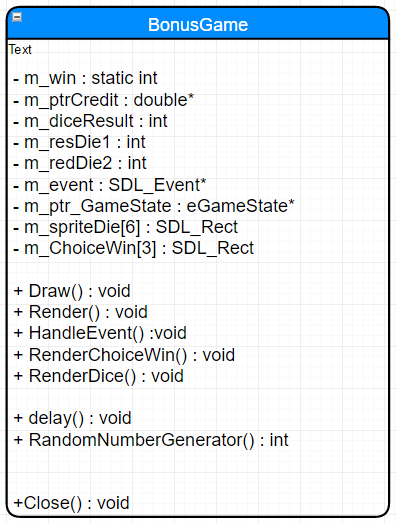
#### Клас IntroScreen

Клас **IntroScreen** е наследник на класа **Screen**. Той се грижи за началния екран появяващ се при стартиране на приложението. В него има бутони за започване на нова игра, възвръщане на започната игра, добавяне на кредити, и визуализиране на информационнен екран с правилата на играта. В него има инстанции на помощните класове ButtonObject и Texture.



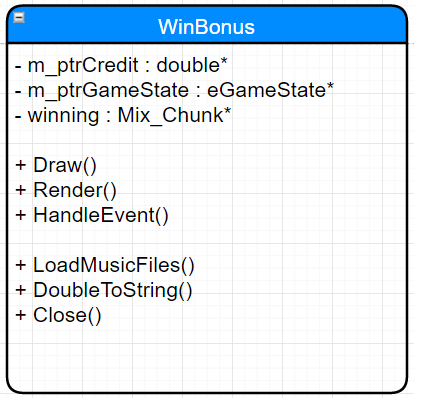
#### Клас BonusGame

Клас **BonusGame** наследник на класа **Screen**. Той се грижи зa екранa появяващ се при стартиране на бонъс играта в приложението , както и логиката на бонус играта. В него има три бутона за избор на печалба, мигащ надпис “Choice win” , анимация на въртящи се зарове, и логика за печалба или не. В него има инстанции на помощните класове ButtonObject и Texture.



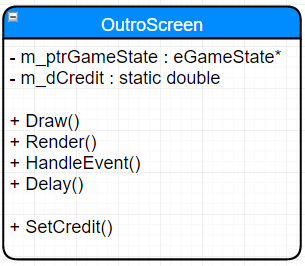
#### Клас WinB**onus**

Клас **WinBonus** наследник на класа **Screen**. Той се грижи зa екранa появяващ се при печалба от бонъс играта в прилогението . В него има поздравителен екран и надпис , информиращ потребителя за новото състояние на кредита. В него има инстанции на помощния клас Texture.



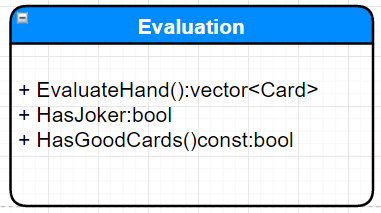
#### Клас OutroScreen

Клас **IntroScreen** е наследник на класа **Screen**. Той се грижи за поздравителния екран при край на играта. В него има функционалнос за визуализиране на надпис със спечелената сума конвертирана в предварително зададената валута на играта и препращане към началният екран след 10 секундно изчкване.



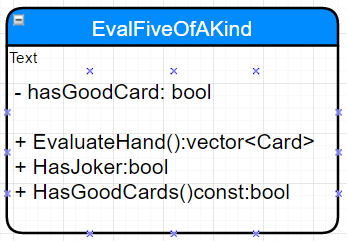
### Клас Evaluation

Клас **Evaluation** е базов клас който представлява шаблон за определянето на петте раздадени карти „EvaluateHand“ и дали има жокер “HasJoker”.



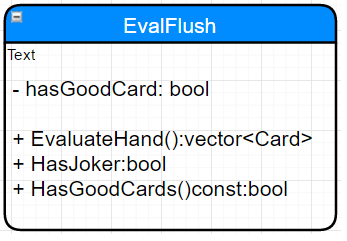
#### Клас EvalFiveOfAKind

Клас **EvalFiveOfAKind** е наследник на класа Evaluation. Той проверява дали текущата ръка носи печалба от тип "Five of a kind" и има ли добри карти за “AutoHold“ режима.



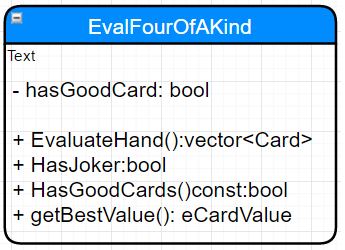
#### Клас EvalFlush

Клас **EvalFlush** е наследник на класа Evaluation. Той проверява дали текущата ръка носи печалба от тип "Flush" и има ли добри карти за “AutoHold“ режима.



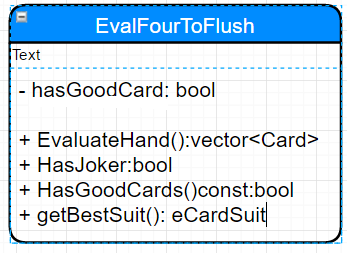
#### Клас EvalFourOfAKind

Клас **EvalFourOfAKind** е наследник на класа Evaluation.Той проверява дали текущата ръка носи печалба от тип "Four of a kind" и има ли добри карти за “AutoHold“ режима.



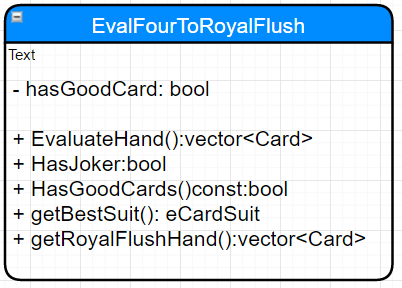
#### Клас EvalFourToFlush

Клас **EvalFourToFlush** е наследник на класа Evaluation. Той е помощен за “AutoHold“ режима.



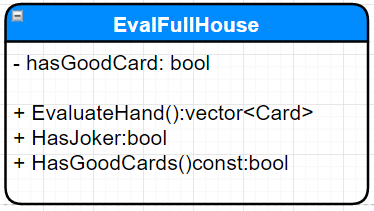
#### Клас EvalFourToRoyalFlush

Клас **EvalFourToRoyalFlush** е наследник на класа Evaluation. Той е помощен за “AutoHold“ режима.



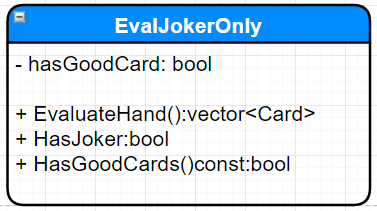
#### Клас EvalFullHouse

Клас **EvalFullHouse** е наследник на класoвете EvalThreeOfKind и EvalTwoPair Той проверява дали текущата ръка носи печалба от тип "Full house" и има ли добри карти за “AutoHold“ режима.



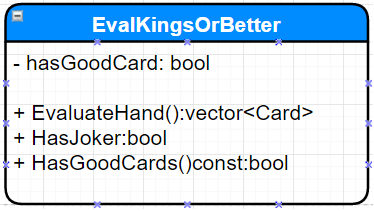
#### Клас EvalJokerOnly

Клас **EvalJokerOnly** е наследник на класа Evaluation.Той е помощен за “AutoHold“ режима.



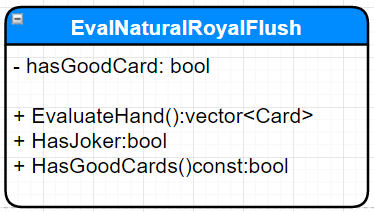
#### Клас EvalKingsOrBetter

Клас **EvalKingsOrBetter** е наследник на класа Evaluation. Той проверява дали текущата ръка носи печалба от тип "Kings or better" и има ли добри карти за “AutoHold“ режима.



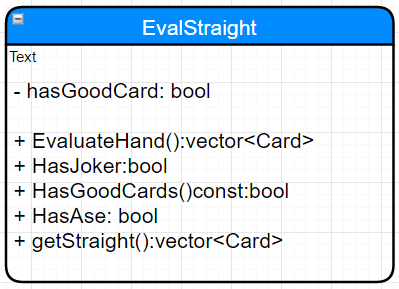
#### Клас EvalNaturalRoyalFlush

Клас **EvalNaturalRoyalFlush** е наследник на класа EvalStraightFlush. Той проверява дали текущата ръка носи печалба от тип “ Natural royal flush” и има ли добри карти за “AutoHold“ режима.



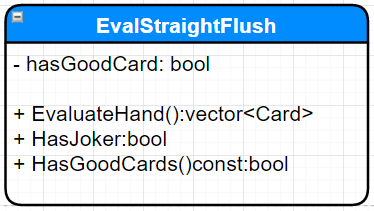
#### Клас EvalStraight

Клас **EvalStraight** е наследник на класа Evaluation. Той проверява дали текущата ръка носи печалба от тип "Straight" и има ли добри карти за “AutoHold“ режима.



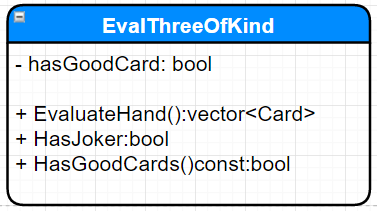
#### Клас EvalStraightFlush

Клас **EvalStraightFlush** е наследник на класовете EvalStraight и EvalFlush. Той проверява дали текущата ръка носи печалба от тип "Straight flush" и има ли добри карти за “AutoHold“ режима.



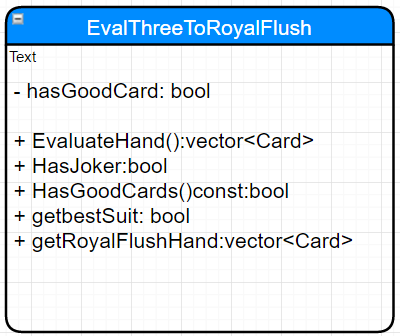
#### Клас EvalThreeOfKind

Клас **EvalThreeOfKin**d наследник на класа Evaluation. Той проверява дали текущата ръка носи печалба от тип "Three of a kind" и има ли добри карти за “AutoHold“ режима.



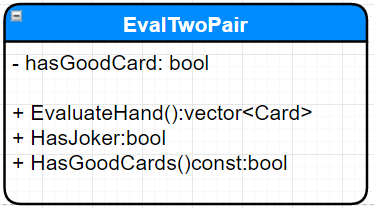
#### Клас EvalThreeToRoyaiFlush

Клас **EvalThreeToRoyalFlush** е наследник на класа Evaluation.Той е помощен за “AutoHold“ режима.



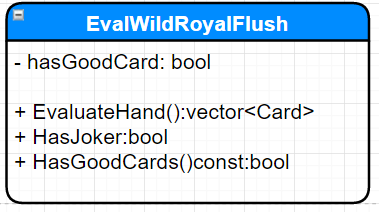
#### Клас EvalTwoPair

Клас **EvalTwoPair** е наследник на класа Evaluation. Той проверява дали текущата ръка носи печалба от тип "Two pair" и има ли добри карти за “AutoHold“ режима.



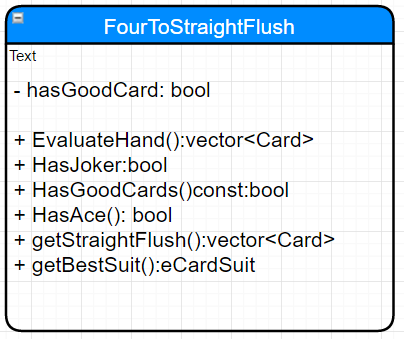
#### Клас EvalWildRoyalFlush

Клас **EvalWildRoyalFlush** е наследник на класа EvalStraightFlush. Той проверява дали текущата ръка носи печалба от тип "Joker royal flush" и има ли добри карти за “AutoHold“ режима.



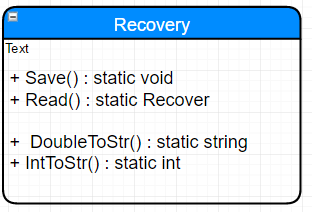
#### Клас FourToStraightFlush

Клас **FourToStraightFlush** е наследник на класа Evaluation.Той е помощен за “AutoHold“ режима.



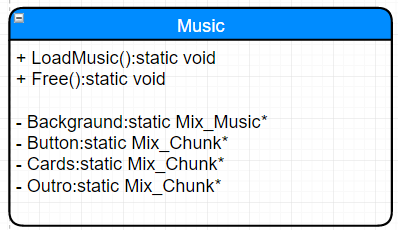
### Клас Recovery

Класът **Recovery** е клас, който има списобността да запазва и възстановява текущото състояние на играта в и от XML файл. При евентуално аварийно положение играта да се възстанови там, от където е прекъсната.



### Клас Music

Клас Music е класа, който отговаря за инициализацията и изпълнението на музикалните файлове,звукове и ефекти.В него са създадени функций,който се имплементират в други класове,в който е необходимо да присъстват музикалните файловевте от този клас.



## Изпълнение на функционалностите

### Клас Game

**Член променливи:**

- m\_dCredit : double

Съдържа стойността на кредитите по време на игра.

- m\_iBtet : int

Съдържа текущия залог по време на игра.

- m\_window : SDL\_Window\*

Структура от библиотеката SDL, която идентифицира прозореца на приложението.

- m\_renderer : SDL\_Renderer\*

Структура от библиотеката SDL, която представя текущото състояние на рендърингa.

- m\_tBackground : Texture

Обект от клас текстура – съдържа текстурата на фона.

- m\_tCredit : Texture

Обект от клас текстура – съдържа текстурата на кредитите.

- m\_tGameOver : Texture

Обект от клас текстура – съдържа текстурата на надписа “GAME OVER”.

- m\_paytable : PaytableObject\*

Сочи към инстанция на класа PaytableObject.

- m\_ptrDeck : Deck\*

Сочи към инстанция на класа Deck.

- m\_vecEvaluations : vector<Evaluations\*>

Вектор от съдържащ указатели към инстанциите на наследниците на класа Evaluation.

Използва се в алгоритъма за изчисляване на текущата ръка карти. При итерация през вектора се извиква метода за изчисляване на ръката “EvaluateHand()”.

- m\_btnCashOut : ButtonObject\*

Обект от класа ButtonObject. Използва се в член-функцията “ProcessMouseInput()”. Променя текущото състояние на „OUTRO“.

-m\_btnDealDraw : ButtonObject\*

Обект от класа ButtonObject. Използва се в член-функцията “ProcessMouseInput()”.

Извиква функцията “ProcessRound()”.

- m\_btnMusic : ButtonObject\*

Обект от класа ButtonObject. Използва се в член-функцията “ProcessMouseInput()”.

Извиква функцията „PlayMusic()“.

- m\_btnMusicPlus : ButtonObject\*

Обект от класа ButtonObject. Използва се в член-функцията “ProcessMouseInput()”.

Извиква функцията „// ф-ция за увеличаване на силата на звука“.

- m\_btnMusicMinus : ButtonObject\*

Обект от класа ButtonObject. Използва се в член-функцията “ProcessMouseInput()”.

Извиква функцията „ф-ция за намаляване на силата на звука“.

- m\_btnMusicPause : ButtonObject\*

Обект от класа ButtonObject. Използва се в член-функцията “ProcessMouseInput()”.

Извиква функцията „ф-ция за спиране на звука“.

- m\_bIsGameOver : bool

Флаг използван във фунцията „Render()“. С него се проверява дали е необходимо да се извика функцията “RenderGameOver()”.

- m\_bIsBonus : bool

Флаг използван във функциятa „ProcessRound()“. С него се проверява дали трябва да се премине в състояние „BONUS“.

+ m\_eGameState : eGameState

Енумератор използван за превключване на състоянията в играта.

+ m\_event : SDL\_Event

Oбща структура предствляваща събитията от библиотеката SDL.

**Член-функции:**

+ Game() : void

Конструктор инициялизиращ всички полета на класа и изпълва на вектора m\_vecEvaluations със съставящите го обекти.

+ Draw() : void

Презентира текущия кадър в играта.

Извиква функциите „SDL\_RenderPresent(SDL\_Renderer\* )“ и „SDL\_RenderClear(SDL\_Renderer\* )“.

+ Render() : void

Oсновната функция за рендериране по екрана. В нея се рендерират всички обекти на игралното поле и се извикват допълнителни функции за рендериране.

+ RenderRound() : void

Помощна функция извиква се във функцията „Render()“, тя от своя страна извиква функциите „RenderHand()“, “RenderHoldBtns()”, “RenderHoldStamps()“. С тази функция се капсулира визуализацията на един рунд в играта.

+ RenderGameInfo() : void

Помощна функция извиква се във функцията „Render()“. Тя се грижи за визуализирането на текущите кредити и залог.

+ RenderGameOver() : void

Помощна функция извиква се във функцията „Render()“. Тя се грижи за визуализирането на текста за край на играта.

+ HandleEvent() : void

Основна функция за улавяне на събитията в играта(интеракция с потребителя - натискане на бутони, операции с мишката).

+ ProcessKeyInput() : void

Функция долавяща събитията подавани от клавиатурата. Извиква се във функцията „HandleEvent()“.

+ ProcessMouseInput() : void

Функция долавяща събитията подавани от мишката. Извиква се във функцията „HandleEvent()“.

В тази функция се извикат функцията на бутоните „IsSelected()“.

+m\_iCounterVolumeMusic():int

Функция,с помощта на която се регулира звука(увеличава,намалява със стъпка 10,до 100),посредством бутоните,създадени във необходимите класове.

+ ProcessRound() : void

Основна функция капсулираща цикъла на една игра. В нея се извикват функциите за раздаване на карти, след това се итерира през вектора “m\_vecEvaluations” за да се извикат функциите на съставящите го обекти за оценавяне на ръката. Тук също се изпращат сигнали до таблицата на залозите за визуализиране на печалата и се орпеделя дали е необходимо стартирането на бонус играта или прекратяване на рунда. В нея се определя и живота на обекта „m\_ptrDeck“.

- InitSDL() : void

Функция която инициализира необходимите настройки на SDL библиотеката.

- Close() : void

Освобождава ресурсите инициaлизирани от SDL.

### Клас Texture

**Член-променливи:**

- m\_texture : SDL\_Texture\*

Съдържа в себе си текстура на заредено в паметта изображение.

- m\_font : TTF\_Font\*

Съдържа в себе си фонт.

**Член-функции:**

+ Texture(SDL\_Renderer\*, string) :

Конструктор инициализиращ текстура, като се подава рендерер и път към файл с изображение.

+ LoadFromFile(SDL\_Renderer\*, string) : void

Зарежда изображение в текстурата. Подаваме като 2-ри параметър пътят до изображението.

+ LoadFromRendererdText(SDL\_Renderer, string, string, SDL\_Color, int) : void

Преобразува текст в текстура, като му подаваме следните параметри: SDL\_Rendererer, пътят на .ttf файла, текста, който искаме да визуализираме на екрана, цвят на текста и размер на шрифта.

+ Render(SDL\_Renderer, int, int, int, int, SDL\_Rect) : void

Функция която рендерира дадена текстура на екрана по подадени параметри: SDL\_Renderer, координати по “x” и “y”, широчина и височина с който да се визуализира изображението на екарана и структура от тип правоъгълник с която да се определи каква част от изображението да се рендерира на екрана.

+ InitFont(string, int) : void

Функция инициализираща фон по зададен път до .ttf файла и размер на шрифта.

- Free() : void

Освобождава ресурсите на текстурата и фонта.

### Клас ButtonObject

**Член-променливи:**

- m\_texture : Texture

Текстурата за изображението на бутона.

- m\_pTopLeft : SDL\_Point

Точка определяща координатите на бутона върху екрана в позиционирана в горния ляв ъгъл на бутона.

- m\_iWidth : int

Широчина на бутона в пиксели.

- m\_iHeight : int

Височина на бутона в пиксели.

**Член-функции:**

+ ButtonObject(SDL\_Renderer = NULL, string = “”, int = 0, int = 0, int = 100, int = 100) :

Конструктор по подразбиране. Подава рендерер и път към изображението на конструтора не текстурата, здава координати по “x” и „y“, широчина и височина на бутона.

+ Render(SDL\_Renderer, SDL\_Rect, int = 0, int = 0, int = 100, int = 100) : void

Функция която рендира изображението на бутона със даден път към изображението, зададени координати по “x” и „y“, със зададени широчина и височина.

Забележка: Бутоните имат 2 състояния – селектиран и неселектиран затова изображенията изполвани за бутони тябва да бъдат в определен формат за да функционират правилно а и именно 2 те състояния да бъдат позиционирани на изображението едно под друго на еднакво разстояние.

+ IsSelected() : bool

Функция индикираща дали мишката се намира в или извън бутона.

### Клас PaytableObject

**Член-променливи:**

- m\_texture : Texture

Текстура за изображението на таблицата.

- m\_tText : Texture

Текстура за текста в таблицата.

- m\_font : TTF\_Font

Шрифт за текста на таблицата.

+ m\_btnBetOne : ButtonObject

Бутон за увеличаване на залога с коефициент 1. При натискане извиква функцията „IncreaseBet()“.

+ m\_btnBetMax : ButtonObject

Бутон за задаване на максимален залог. При натискане извиква функцията „SetMaxBet()“.

- m\_iPrevBetCoef : static int

- m\_iNextBetCoef: static int

Идентификаторите се използват в алгоритъма за промяна на залога имплементиран във функциите „IncreaseBet()“ и „SetMaxBet()“.

- m\_vecHands : vector<string>

Вектор съдържащ имената на печелившите комбинации.

-m\_vecBets : vector<int>

Вектор съдържащ стойностите на печелившите комбинации.

-m\_sfx : array Mix\_Music\*

Масив от 11 елемента съдържа 11 уникални звукови ефекта съответстващи на печалбите в таблицата на залозите.

**Член-функции:**

+ PaytableObject(SDL\_Renderer\*) :

Конструктор приема параметър SDL\_Renderer за да инициализира текстурите, зарежда музикалните файлове в масива m\_sfx, и инициализира стойности на останалите членове на класа.

+ LoadWinSounds() : void

Функцията се извиква в конструктора, ролята и е да зареди в масива музикалните файлове.

+ Render(SDL\_Renderer\*) : void

Основна функция капсулираща визуализацията на елементите на таблицата. В нея е имплементирана функционалността за рендерирането на бутоните за залог.

+ RenderCardCombinations(SDL\_Renderer\*) : void

Функцията визуализира наименованията и стойностите на печелившите комбинации.

+ RenderBetList(SDL\_Renderer\*, int) : void

Функцията визуализира с различен от останалите редове цвят реда на печелившата комбинация по подаден индекс съответстващ на позиция във вектора на печелившите комбинации.

+ PlaySoundEffect(int) : void

Функцията изпълнява определен звуков ефект при подаден индекс съответстващ на елемент от масива с музикалните ефекти m\_sfx.

+ IncreaseBet() : void

Функцията увеличава коефициента на залога с 1.

+ SetMaxBet() : void

Функцията задава максимална стойност на залога.

### Клас Card

**Член-променливи:**

- m\_cardSuit : eCardSuit

Енумератор за инициализация на боята на картата.

- m\_cardValue : eCardValue

Енумератор за инициализация на стойността на картата.

- m\_cardRect : SDL\_Rect

Структура от библиотеката SDL. Служи за да определи позицията и размерите на картата в изображението съдържащото всички карти от тестето.

- m\_bIsHold : bool

Булева променлива участваща в логиката за определяне на задържаните карти от потребителя при първото раздаване на цикъла.

**Функции и член-функции:**

+ operator > (const Card&, const Card&) : friend bool

Предефиниране на оператора на сравнение “по-голямо” . Сравнява картите само по стойност.

+ operator == (const Card&, const Card&) :friend bool

Предефиниране на оператора за сравнение „равно“. Сравнява картите по боя и по стойност.

+ operator = (const Card&) :

Предефиниране на оператор за присвояване. Присвоява боята, стойността, SDL\_Rect структурата, булевата променлива за задържане на картата.

### Клас Deck

**Член-променливи:**

**-** m\_iKillcount : int

Променлива от целочислен тип. Изпълнява ролята на брояч за етапите на играта. Използва се във функцията ProcessRound() и се увеличава във функцията deal().

- m\_texture : Texture

Използва се да съхранява текстурата на изображението съдържащо всички карти.

- m\_tDim : Texture

Текстурата съхранява изображението за затъмняване на непечелившите карти в момент на край на текущият рунд.

- deckOfCards : array Card

Масивът съдържа всички карти използвани в играта. Изпълнява важна роля в процесите на раздаване на картите.

- hand : vector <Card>

Векторът съхранява в себе си 5 карти. Те представляват една ръка. Използва се във функционалностите за логика на играта: раздаване на карти, задържане, определяне на подходящи карти за AutoHold стратегията.

+ m\_arrCardHold : array ButtonObject

Масив съдържащ 5 бутона които да подават информация на функцията HoldSelectedCards() за задържане на картите.

**Член-функции:**

+ Deck() :

Конструктор. Инициализира принадлежащите на класа текстури. Зарежда картите в масива decOfCards, подава стойност на генератора на числа на случаен принцип.

+ deal() : void

Основна функция в съставянето на гейм логиката. Изпълнява функцията да напълни вектора hand с 5 неповтарящи се карти избрани на случаен принцип.

+ isCardInHand(Card& card) : bool

Функцията проверява дали подадената в листа с параметри карта вече присъства в текущата ръка.

+ RenderCard(SDL\_Renderer, SDL\_Rect\*, SDL\_Rect\*) : void

Функцията рендерира конкретна карта от главната текстура с всички карти.

+ RenderHand(SDL\_Renderer\*) : void

Функцията рендерира 5-те карти от раздаването на екрана с помощта на функцията RenderCard().

+ RenderStart(SDL\_Renderer\*) : void

Функцията рендерерира 5 карти на екрана с лице надолу.

+RenderHoldBtns(SDL\_Renderer\*) : void

Рендерира 5 невидими бутона на екрана.

+ HoldSelectedCards() void

Използва информацията от RenderHoldBtns(SDL\_Renderer\*) за да задържи избраните от потребителя карти

+ RenderHoldStamps(SDL\_Renderer\*) : void

Използва информацията от функцията HoldSelectedCards() и визуализира етикети върху задържаните от потребителя карти.

+ DimCards(SDL\_Renderer\*)

Предвидено е функцията да придава визуален ефект на картите неучастващи в изчисляването на печалбата.

### Клас Screen

**Член-променливи:**

# m\_renderer : SDL\_Renderer

Този указател сочи към SDL\_Render от клас Game.

**Член-функции:**

+ Screen(SDL\_Renderer\*)

Конструктор. Инициализира m\_renderer към SDL\_Renderer\* от класа Game.

+ Draw() : void

Презентира текущия кадър в играта.

Извиква функциите „SDL\_RenderPresent(SDL\_Renderer\* )“ и „SDL\_RenderClear(SDL\_Renderer\* )“.

+ Render() : void

Това е чисто виртуален метод. Бива наследяван и инициализиран от наследниците на класа Screen.

+ HandleEvent() : void

Това е чисто виртуален метод. Бива наследяван и инициализиран от наследниците на класа Screen.

#### Клас BonusGame

**Член променливи:**

- m\_ptrCredit : double\*

Съдържа стойността на кредитите по време на игра.

-m\_ win : static int

Съдържа печалбата необходима за бонус играта .

-m\_diseResult : int

Съдържа общия резултат от двата зара .

-m\_resDie1 : int

Съдържа резултатa от зар 1 .

-m\_resDie2 : int

Съдържа резултатa от зар 2 .

- m\_tBackground : Texture

Обект от клас текстура – съдържа текстурата на фона.

- m\_tChoiceWin : Texture

Обект от клас текстура – съдържа текстурата на мигащия надпис „Choice win“ .

- m\_spriteDie[6] : SDL\_Rect

Масив от обекти от библиотека SDL2 тип SDL\_Rect

- m\_ChoiceWin[3] : SDL\_Rect

Масив от обекти от библиотека SDL2 тип SDL\_Rect

- m\_spriteDieTexture: Texture

Обект от клас текстура – съдържа текстурата на зар.

- m\_buttonX1 : ButtonObject\*

Указател към класа ButtonObject. Използва се в член-функцията “HandleEvent()”.

С този бутон се избира печалба \* 2 и са забъртат заровете.

- m\_buttonX2 : ButtonObject\*

Указател към класа ButtonObject. Използва се в член-функцията “HandleEvent()”.

С този бутон се избира печалба \* 5 и са забъртат заровете.

- m\_buttonX2 : ButtonObject\*

Указател към класа ButtonObject. Използва се в член-функцията “HandleEvent()”.

С този бутон се избира печалба \* 10 и са забъртат заровете.

- m\_btnMusic : ButtonObject\*

Указател към класа ButtonObject. Извиква функцията „PlayMusic()“.

С този бутон се пуска фоновата музика.

- m\_btnMusicPlus : ButtonObject\*

Указател към класа ButtonObject. Извиква функцията „// ф-ция за увеличаване на силата на звука“. С този бутон се увеличава силата на фоновата музика.

- m\_btnMusicMinus : ButtonObject\*

Указател към класа ButtonObject. Извиква функцията „ф-ция за намаляване на силата на звука“. С този бутон се намалява силата на фоновата музика.

- m\_btnMusicPause : ButtonObject\*

Указател към класа ButtonObject. Извиква функцията „ф-ция за спиране на звука“.

С този бутон се спира фоновата музика.

- m\_ptrGameState : eGameState\*

Този указател сочи към m\_eGameState в класа Game.

- m\_event : SDL\_Event\*

Oбща структура предствляваща събитията от библиотеката SDL.

- Buttonpress : Mix\_Chunк\*

Указател към Mix\_Chunк от библиотека SDL2. Представлява звук, зареден в паметта.

- RollDice : Mix\_Chunk\*

Указател към Mix\_Chunк от библиотека SDL2. Представлява звук, зареден в паметта.

**Член-функции:**

+ BonusGame() : void

Конструктор инициялизиращ всички полета на класа .

+ LoadDieFiles() : void

Зарежда изображението на зара в текстурата.

+ LoadChoiceWineFiles() : void

Зарежда изображението на надписа в текстурата.

+ LoadMusicFiles() : void

Зарежда музикалния файл.

+ Draw() : void

Презентира текущия кадър в играта.

Извиква функциите „SDL\_RenderPresent(SDL\_Renderer\* )“ и „SDL\_RenderClear(SDL\_Renderer\* )“.

+ Render() : void

Oсновната функция за рендериране по екрана. В нея се рендерират всички обекти на игралното поле и се извикват допълнителни функции за рендериране.

+ RenderChoiceWin() : void

Помощна функция извиква се във функцията „Render()“. Тя се грижи за визуализирането на мигащия надпис “Choice win” .

+ RenderDice() : void

Помощна функция извиква се във функцията „delay()“. Тя се грижи за визуализирането на заровета в играта .

+ delay(Uint32) : void

Помощна функция извиква се във функцията „HandleEvent()“. Тя се грижи за визуализирането на въртенето заровета в играта с помощта на текущото време .

+ RandomNumberGenerator() : int

Помощна функция извиква се във функцията „RenderDice()“. Тя се грижи за генериране на случаини числа .

+m\_iCounterVolumeMusic():int

Функция,с помощта на която се регулира звука(увеличава,намалява със стъпка 10,до 100),посредством бутоните,създадени във необходимите класове.

+ HandleEvent() : void

Основна функция за улавяне на събитията в играта(интеракция с потребителя - натискане на бутони, операции с мишката) и се грижи за поредността на събитията .

- Close() : void

Освобождава ресурсите инициaлизирани от SDL.

#### Клас WinBonus

**Член променливи:**

- m\_ptrCredit : double\*

Съдържа стойността на кредитите по време на игра.

- m\_tBackground : Texture

Обект от клас текстура – съдържа текстурата на фона.

- m\_tText : Texture

Обект от клас текстура – съдържа текстурата на променения кредит при печалба от бонус играта“ .

- m\_ptrGameState : eGameState\*

Енумератор използван за превключване на състоянията в играта.

- m\_event : SDL\_Event\*

Oбща структура предствляваща събитията от библиотеката SDL.

- winning : Mix\_Chunк\*

Указател към Mix\_Chunк от библиотека SDL2. Представлява звук, зареден в паметта.

**Член-функции:**

+ WinBonus() : void

Конструктор инициялизиращ всички полета на класа .

+ LoadMusicFiles() : void

Зарежда музикалния файл.

+ Draw() : void

Презентира текущия кадър в играта.

Извиква функциите „SDL\_RenderPresent(SDL\_Renderer\* )“ и „SDL\_RenderClear(SDL\_Renderer\* )“.

+ Render() : void

Oсновната функция за рендериране по екрана. В нея се рендерират всички обекти на игралното поле.

+ DoubleToString(double) : string

Помощна функция извиква се във функцията „Render()“. Тя се грижи за преминаване от тип double към string .Необходимо е за визуализацията на кредита чрез SDL.

+ HandleEvent() : void

Основна функция за улавяне на събитията в играта(интеракция с потребителя , операции с мишката) .

- Close() : void

Освобождава ресурсите инициaлизирани от SDL.

#### Клас IntroScreen

- m\_ptrCredit : double\*

Този указател сочи към m\_dCredit в класа Game. Използва се за добавяне на кредити при започване на нова игра.

- m\_ptrBet : int\*

Този указател сочи към m\_iBet в класа Game. Използва се при възстановяване на игра.

- m\_tBackground : Texture

Текстура съдържаща изображението за фона на екрана.

- m\_tCredit : Texture

Текстурата се използва да се визуализира текущия кредит.

- m\_tInfo : Texture

Текстурата съдържа изображението на инфо екрана.

- m\_ptrEvent : SDL\_Event\*

Този указател сочи към m\_event в класа Game.

- m\_ptrGameState : eGameState

Този указател сочи към m\_eGameState в класа Game.

- m\_btnNewGame : ButtonObject

С този бутон се преминава в състояние на игра, като се започва нова игра.

- m\_btnResumeGame : ButtonObject

С този бутон се преминава в състояние на игра, като се възстановяват игра от ХМL файл.

- m\_btnCashIn : ButtonObject

С този бутон се добавят кредити със стъпка 10.

- m\_btnInfo : ButtonObject

С този бутон се визуализира и скрива информационният екран.

- m\_btnMusic : ButtonObject

С този бутон се пуска фоновата музика.

- m\_btnMusicPlus : ButtonObject

С този бутон се увеличава силата на фоновата музика.

- m\_btnMusicMinus : ButtonObject

С този бутон се намалява силата на фоновата музика.

- m\_btnMusicPause : ButtonObject

С този бутон се спира фоновата музика.

- m\_bShowInfo : bool

Тази променлива участва в логиката за показване на информационния екран във функцията Render().

-m\_bShowPlayButton : bool

Тази променлива участва в логиката за редуване на пауза и плей бутоните във функцията Render().

**Член-функции:**

+ Draw() : void

Тази функция извиква функцията Draw() от класа Screen.

+ Render() : void

Тази функция се грижи за цялостната визуализация в интро екрана. Визуализира бутоните, фона и текущия кредит.

+ HandleEvent() : void

Основна функция за улавяне на събитията в интро екрана(интеракция с потребителя , операции с мишката) .

+ CashIn(double) : void

Тази функция прибавя към текущия кредит подадена по параметър сума.

+ RenderInfoWindow() : void

Тази функция визуализира информациионния екран.

+m\_iCounterVolumeMusic():int

Функция,с помощта на която се регулира звука(увеличава,намалява със стъпка 10,до 100),посредством бутоните,създадени във необходимите класове.

+ EvaluateInfoRendering() : void

Тази функция определя показването и скриване на информационния екран при натискания на бутона Game Info.

#### Клас OutroScreen

**Член-прменливи:**

- m\_tBackground : Texture

Текстура, съхранява изображението за фон на екрана.

- m\_tCredit : Texture

Текстура, съхранява в себе си стойността на текущия кредит под формата на текстура.

- m\_ptrEvent : SDL\_Event\*

Указател, сочи към m\_event от класа Game.

- m\_ptrGameState : eGameState

Указател, сочи към m\_eGameState от класа Game.

**Член-функции:**

+ Draw() : void

Функцията презентира текущото състояние на рендерирането.

+ Render() : void

Функцията рендерира изображение и текст на екрана.

+m\_iCounterVolumeMusic():int

Функция,с помощта на която се регулира звука(увеличава,намалява със стъпка 10,до 100),посредством бутоните,създадени във необходимите класове.

+ Delay() : void

Функцията изпълнява задържане на текущото състояние на екрана за 5 секунди, след което променя състоянието на играта на INTRO.

### Клас Recovery

**Член-функции:**

+ Recovery() : void

Конструктор .

+ Save() : static void

Основна функция за запис в XML файл.

- Read() : stetic Recover

Основна функция за четене от XML файл и връща структура Recover.

Поле на структура Recover:

- credit : double

-bet : int

- win : double

+ DoubleToString(double) : string

Помощна функция извиква се във функцията „Save()“. Тя се грижи за преминаване от тип

int към string .Необходимо за осъществяване на запис в XML.

+ intToString(double) : string

Помощна функция извиква се във функцията „Save()“. Тя се грижи за преминаване от тип

int към string .Необходимо за осъществяване на запис в XML.

#### Клас EvalFiveOfAKind

**Член-функции:**

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има ли четири карти с еднаква стойност и жокер.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията .

### Клас EvalFlush

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има ли четири карти от едина боя и жокер.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията

### Клас EvalFourOfAKind

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има ли четири карти от един вид и жокер.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

### Клас EvalFourToFlush

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Основна за “AutoHold“ режима.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

getBestSuit(std::vector<Card>):eCardSuit

### Клас EvalFourToRoyalFlush

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Основна за “AutoHold“ режима.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

+getBestSuit(std::vector<Card>):eCardSuit

Връща най-добрата боя за ръката.

+getRoyalFlushHand(eCardSuit):std::vector<Card>

Връща роял флъш ръка.

### Клас EvalFullHouse

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има ли три карти от един вид или две карти от един вид и жокер .Използва основните функции на класовете ЕvalThreeOfKind и EvalTwoPair.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

### Клас EvalJokerOnly

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Основна за “AutoHold“ режима.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията

### Клас EvalKingsOrBetter

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има ли два попа или две аса, или поп и жокер, или асо и жокер.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

### Клас EvalNaturalRoyalFlush

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има "Straight flush" и десетка, и няма жокер.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

### Клас EvalStraight

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има поредни четири карти или три и жокер.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

+HasAse (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли асо или няма и връща булев отговор.

### Клас EvalStraightFlush

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има "Straight” и “flush". Връща 3 или -1.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

### Клас EvalThreeOfKind

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има ли три карти от един вид или две карти от един вид и жокер.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

### Клас EvalThreeToRoyaiFlush

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Основна за “AutoHold“ режима.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията

getBestSuit(std::vector<Card>):eCardSuit

Връща най-добрата боя за ръката.

getRoyalFlushHand(ecardSuit): std::vactor<Card>

### Клас EvalTwoPair

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има ли две двоики еднакви карти от вид.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

### Клас EvalWildRoyalFlush

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Оснивна функцията, която проверява ръката има ли "Straight flush" и жокер, и десетка или вале.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

### Клас FourToStraightFlush

+ EvaluateHand(std::vactor<Card>& hand) : std::vactor<Card>

Основна за “AutoHold“ режима.

+HasJoker (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли жокер или няма .

+HasGoodCards()const:bool

Помощна функцията проверява има ли карти за “AutoHold“ стратегията.

+HasAse (std::vactor<Card>& hand):bool

Помощна функцията проверява има ли асо или няма и връща булев отговор.

getBestSuit(std::vector<Card>):eCardSuit

Връща най-добрата боя за ръката.

### Клас Music

**Член-променливи:**

-Mix\_Music\* Background:static

Поинтер от Mix\_Music,който държи фоновата музика на бекграунда.

-Mix\_Chunk\* Button:static

Поинтер към звуковите ефекти при натискане на бутоните.

-Mix\_Chunk\* Cards:static

Пойнтер към звуковият ефект при зареждането на картите.

-Mix\_Chunk\* Outro:static

Пойнтер,който държи фоновата музика,при стартиране на аутро екрана.

**Член-функций**:

+LoadMusic():static void

Функция,която зарежда музиката.