***BLACKJACK***

Peter Martic

Inhalt

[*Spielablauf* 4](#_Toc58932676)

[Karte verlangen 4](#_Toc58932677)

[Karte verzichten 5](#_Toc58932678)

[Allgemeine Regel 5](#_Toc58932679)

[Klassendiagramm 6](#_Toc58932680)

[GUI 7](#_Toc58932681)

[Game 8](#_Toc58932682)

[Was macht die Klasse? 8](#_Toc58932683)

[Hit 8](#_Toc58932684)

[Stay 8](#_Toc58932685)

[WinLossDraw 8](#_Toc58932686)

[Game 8](#_Toc58932687)

[StartNewGame 8](#_Toc58932688)

[NewStack 9](#_Toc58932689)

[GetPlayerCards 9](#_Toc58932690)

[GetPlayerCards 9](#_Toc58932691)

[GetPlayerPoints 9](#_Toc58932692)

[GetDealerPoints 9](#_Toc58932693)

[BustPlayer 10](#_Toc58932694)

[BustDealer 10](#_Toc58932695)

[Balance 10](#_Toc58932696)

[GetMoney 10](#_Toc58932697)

[SaveGame 10](#_Toc58932698)

[Player 11](#_Toc58932699)

[Was macht die Klasse? 11](#_Toc58932700)

[Bet 11](#_Toc58932701)

[TakeMoney 11](#_Toc58932702)

[GetBalance 11](#_Toc58932703)

[Player 11](#_Toc58932704)

[Wallet 12](#_Toc58932705)

[Was macht die Klasse? 12](#_Toc58932706)

[Wallet 12](#_Toc58932707)

[Deposit 12](#_Toc58932708)

[WithDraw 12](#_Toc58932709)

[Dealer 13](#_Toc58932710)

[Was macht die Klasse? 13](#_Toc58932711)

[Dealer 13](#_Toc58932712)

[TakeCardStack 13](#_Toc58932713)

[CreateNewStack 13](#_Toc58932714)

[Person 14](#_Toc58932715)

[Was macht die Klasse? 14](#_Toc58932716)

[TakeCard 14](#_Toc58932717)

[GetPoints 14](#_Toc58932718)

[ClearHand 14](#_Toc58932719)

[GetCards 14](#_Toc58932720)

[Person 15](#_Toc58932721)

[Stack 16](#_Toc58932722)

[Was macht die Klasse? 16](#_Toc58932723)

[GetCard 16](#_Toc58932724)

[Shuffle 16](#_Toc58932725)

[AddDeck 16](#_Toc58932726)

[Stack 16](#_Toc58932727)

[Hand 17](#_Toc58932728)

[Was macht die Klasse? 17](#_Toc58932729)

[TakeCard 17](#_Toc58932730)

[ClearHand 17](#_Toc58932731)

[Hand 17](#_Toc58932732)

[BlackjackCard 18](#_Toc58932733)

[Was macht die Klasse? 18](#_Toc58932734)

[GetValue 18](#_Toc58932735)

[BlackjackCard 18](#_Toc58932736)

[Card 19](#_Toc58932737)

[Was macht die Klasse? 19](#_Toc58932738)

[Card 19](#_Toc58932739)

[ToString 19](#_Toc58932740)

[CardValue 20](#_Toc58932741)

[Was macht die Klasse? 20](#_Toc58932742)

[Werte 20](#_Toc58932743)

[CardColor 21](#_Toc58932744)

[Was macht die Klasse? 21](#_Toc58932745)

[Deck 22](#_Toc58932746)

[Was macht die Klasse? 22](#_Toc58932747)

[GetCards 22](#_Toc58932748)

[Deck 22](#_Toc58932749)

[BlackJack(GUI) 23](#_Toc58932750)

[Was macht das GUI? 23](#_Toc58932751)

[btnHit\_Click() 23](#_Toc58932752)

[neuesSpielToolStripMenuItem\_Click() 23](#_Toc58932753)

[beendenToolStripMenuItem\_Click() 23](#_Toc58932754)

[ControlStructures() 24](#_Toc58932755)

[btnStay\_Click() 24](#_Toc58932756)

**BlackJack – Anleitung**

# ***Spielablauf***

Vor jedem Spiel beginnt der Dealer **5 Decks** zu nehmen, um diese dann zu einem **Stack** zu **vermischen**. Jetzt **platziert** der Spieler einen **Einsatz, der von seiner Geldtasche geholt wird,** auf das **bezeichnete Feld**, das auf dem **Spieltisch** **vor ihm** liegt. **Ist** der **Einsatz** **getätigt**, **beginnt** der **Dealer** die Karten **auszuteilen**, die er sich vom **Stack** holt. Der **Dealer** **nimmt** sich eine **1. offene Karte** **vom Stack**, um sie danach **auf seine Hand** zu **geben**. Als nächstes holt er sich eine **2. Karte vom Stack**, die er dann dem **Spieler** **gibt**. Der **Spieler** **gibt** die **Karte** dann in seine **Hand**. **Zum Schluss** **nimmt** sich der **Dealer** eine **2.Karte**, die er sich wieder vom **Stack** holt**, auf seine Hand**. Und diesen **Vorgang** **wiederholt** er **für** den **Spieler** noch einmal, damit er **dem Spieler** die **Karte** **wieder geben** kann und der **Spieler** die **Karte** danach auf seine **Hand**. Der **Dealer** **enthüllt** **nach dem Austeilen** **kurz für sich**, ob die **verdeckte Karte** von ihm **eine Blackjack-Kombination** **mit** seiner **ersten Karte** **bildet**, **falls ja** **deckt** er die **Karte auf**, **damit** der **Spieler** sie auch **sehen kann \*(hat der Spieler auch ein Blackjack so ist es ein Gleichstand bzw. *PUSH*)\***. Ist der **Kartenwert** vom **Spieler** **kein Blackjack \*(wenn der Dealer Blackjack hat)\*** 🡪 **verliert sofort alles bzw. Karten** und **Einsatz werden vom Dealer eingezogen, auch die von ihm selbst 🡪 Alle Karten werden wieder mit dem Stack zusammengemischt**. Das **gleiche** **gilt** für den **Dealer**, also wenn Der **Spieler** **diese Kombination** **hat**, **aber** der **Dealer nicht**, dann bekommt der **Spieler** einen **Gewinn** von **(3:2)**. Der **Dealer** **sammelt** **danach alle** **Karten von ihm** und **dem Spieler ein**, **um** sie danach **wieder** **mit** dem **Stack** **zu vermischen**. **Ansonsten**, wenn das alles nicht in Kraft tritt, ist der **Spieler** **gezwungen** **sich** **zu entscheiden**, **ob** er noch eine **Karte möchte** **oder** **nicht**.

## ***Karte verlangen***

**Möchte** der **Spieler** eine **weitere Karte**, so holt **der Dealer** sich eine **Karte vom Stack** und **übergibt** diese **Karte** dem **Spieler**. Der **Spieler** **nimmt** die **Karte** und **gibt** sie **in** **seine** **Hand**. Hat der **Spieler** mit seinem **Kartenwert** die **21 noch nicht überschritten**, so kann er noch eine **Karte verlangen** 🡪 der **Vorgang** **wiederholt sich** **(ob er Karte möchte oder nicht)**. Falls der **Spieler über** den **Wert 21** kommt, **verliert** er **alles**, das heißt der **Einsatz**, den er **getätigt** **hat** und die **Karten**, die er hatte, werden **vom Dealer eingezogen (auch die von ihm selbst) 🡪 Alle Karten werden wieder mit dem Stack zusammengemischt.**\*

### ***Karte verzichten***

Hat sich der **Spieler** aber **gegen** eine **Karte entschieden**, so **enthüllt** der **Dealer** seine **2. Karte**. Hat der **Dealer** mit seiner **Hand** **17 oder mehr Punkte**, **muss** er stehen **bleiben 🡪** falls der **Dealer** zu dem Zeitpunkt gleich **mit** dem **Spieler** **gleich steht**, so **bekommt** **der** **Spieler** seinen **Einsatz** **zurück** und der **Dealer** **nimmt** wieder **alle Karten** und **vermischt** sie **mit dem Stack**. Ist der **Kartenwert** **höher** **als** vom **Spieler**, **aber** **nicht** **über 21**, so **gewinnt** der **Dealer** **und** **entzieht** den **Einsatz des Spielers sowie alle Karten vom Spiel** 🡪 **Karten** **werden** wieder **mit dem Stack vermischt**. Ist der **Wert vom Spieler höher als** der **vom Dealer**, so **bekommt** der **Spieler** einen **Gewinn (1:1)** 🡪 **Dealer** **nimmt** danach **alle Karten** **vom** **Spiel** und **vermischt** sie **mit dem Stack**. Wenn der **Dealer** **über den Kartenwert 21** ist, **bekommt** der **Spieler wieder** einen **Gewinn (1:1)** 🡪 **Dealer** **nimmt** danach **alle Karten** **vom** **Spiel** und **vermischt** sie **mit dem Stack**. Ansonsten, **bekommt** der **Spieler keinen Gewinn** 🡪 **Dealer** **nimmt** danach den **Einsatz des Spielers** und **alle Karten** **vom** **Spiel** 🡪 **vermischt** sie **mit dem Stack**. Hat der **Dealer** **16 oder weniger Punkte**, **muss** er eine **weitere** **Karte** **vom Stack ziehen, solange** **bis** er **mindestens** **auf** oder **über** **17** ist. Wenn er **keine Karte mehr nehmen darf**, werden die **verschiedenen Ausgangsmöglichkeiten geprüft**, die oben schon aufgeschrieben wurden.

**\*Nachdem die Karten eingesammelt wurden und zum Stack vermischt wurden, kann der Spieler entscheiden, ob er noch einmal spielen möchte\***

### ***Allgemeine Regel***

##### Regel 🡪 Der Dealer/Spieler muss ein Ass stets mit elf Punkten zählen, es sei denn, er würde auf diese Weise den Wert 21 überschreiten, nur dann zählt er das Ass mit einem Punkt 🡪 Zum Beispiel, wenn der Spieler/Dealer ein Ass und eine 9 hat, muss er das Ass mit elf und die Hand mit 20 Punkten bewerten und darf keine weitere Karte ziehen

### ***Klassendiagramm***

**Abbildung 1(Klassendiagramm 07.10.20)**

###### **Klassen**

* + **Stack**
* **AddDeck()**
* **Shuffle()**
* **GetCard()**
* **CardCast()**
  + **Deck**
* **GetCards()**
  + **Card**
* **+ String Filename**
  + **BlackJackCard**
* **GetValue()**
  + **Manager**
* **Hit()**
* **Stand()**
* **Bust()**
* **WinLossDraw()**
  + **Person**
* **Dealer**
* **Player**
* **Wallet**
* **Enums**
  + **CardColor**
  + **CardValue**

### GUI

Bei den **FlowLayoutPanels** wurde der **Anchor** **(Bottom,Left)** eingestellt und **AutoSize** **eingeschaltet**.

Bei den **Buttons** **(HIT,STAY)** wurde der **Anchor** **(Bottom, Left)** eingestellt und **AutoSize** **ausgestellt**. Beim **Button** **(BET)** wurde der **Anchor** **(Right)** **eingestellt**. Die **Einstellung** vom Button **„BET“** **gelten** auch für die **TextBox**, wo der **Einsatz** getätigt wird.

Bei allen **Labels** wurde der **Anchor** **(Bottom, Left)** **eingestellt** und **AutoSize** **ausgestellt**, wobei beim Label **„Einsatz“** der **Anchor** auf **(Right) gesetzt** ist.

Abbildung 2(Maßstabgetreuer Entwurf)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung .1(Kleiner Entwurf)

# **Game**

## **Was macht die Klasse?**

* Dient zur Verwaltung des Spiels, die dazu mehrere Klassen beinhaltet.

### **Hit**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert eine Karte zurück. Die Hit-Methode bekommt eine Karte vom Dealer, damit diese Karte im letzten Schritt dem Spieler übergeben werden kann,( falls der Benutzer diese Methode mit einem Button auslöst).

### **Stay**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert keinen Wert zurück. Die Stay-Methode startet den Zug des Dealers. Der Dealer nimmt solange eine Karte, bis er nicht auf den Kartenwert 17 oder höher kommt.

### **WinLossDraw**

#### **Was macht die Methode und was wird von ihr erwartet?**

Die Methode liefert eine ganze Zahl zurück. In dieser Methode wird geprüft, ob der Dealer oder Spieler gewonnen hat, indem die Kartenwerte verglichen werden. Also, wenn der Spieler einen höheren Kartenwert als der Dealer hat aber nicht über 21 liegt, dann hat der Spieler gewonnen und es wird der **int-Wert 1** zurückgeliefert. Falls es umgekehrt ist, wird der **Wert 2** zurückgeliefert. Wenn die beiden den gleichen Kartenwert haben, dann wird der **Wert 0** zurückgegeben.

### **Game**

Dies ist keine Methode, sondern der Konstruktor der Klasse und besitzt deswegen an keinen Rückgabewert. Der Konstruktor erstellt einen neuen Spieler beziehungsweise Dealer und jeweils die Liste für den Dealer und Player.

### **StartNewGame**

#### **Was macht die Methode und was wird von ihr erwartet?**

StartNewGame liefert nichts zurück. In der Methode werden alle Karten gelöscht, die sich in den jeweiligen Listen befinden, das heißt die Liste des Dealers und Spielers, damit sie im Anschluss wenn das neue Stack erstellt und vermischt wurde, mit jeweils 2 Karten befüllt werden.

### **NewStack**

#### **Was macht die Methode und was wird von ihr erwartet?**

Die Methode liefert nichts zurück. Bei dieser Methode erstellt der Dealer ein neues Stack für das neue Spiel.

**SetBet**

#### **Was macht die Methode und was wird von ihr erwartet?**

Auch diese Methode liefert nichts zurück, hat aber einen Parameter mit einem int-Wert. In dieser Methode wird der Einsatz des Spielers getätigt. Dazu wird die Methode des Spielers verwendet, die später näher erklärt wird.

### **GetPlayerCards**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert eine Liste von Karten zurück. Hier werden die ganzen Karten, die sich auf der Hand des Spielers befinden als Liste zurückgeliefert. Dazu wird die Methode der Klasse Person verwendet, die später näher erklärt wird.

### **GetPlayerCards**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert eine Liste von Karten zurück. Hier werden die ganzen Karten, die sich auf der Hand des Dealers befinden als Liste zurückgeliefert. Dazu wird die Methode der Klasse Person verwendet, die später näher erklärt wird.

### **GetPlayerPoints**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert den Datentyp int zurück. In dieser Methode wird der Wert der Hand des Spielers zurückgeliefert, dabei wird eine Methode verwendet, die sich in der Klasse Person befindet, benutzt. Auf diese Methode wird in einem späteren Zeitpunkt Bezug genommen.

### **GetDealerPoints**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert den Datentyp int zurück. In dieser Methode wird der Wert der Hand des Dealers zurückgeliefert, dabei wird eine Methode verwendet, die sich in der Klasse Person befindet, benutzt. Auf diese Methode wird in einem späteren Zeitpunkt Bezug genommen.

### **BustPlayer**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert den Datentyp bool zurück. Die Methode prüft, ob der Spieler über den Wert 21 gekommen ist, falls dies passiert, wird der Wert true zurückgegeben, ansonsten false. Der Wert der Hand vom Spieler wird mit der Methode, die sich in der Game-Klasse befindet, geholt(GetPlayerPoints).

### **BustDealer**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert den Datentyp bool zurück. Die Methode prüft, ob der Dealer über den Wert 21 gekommen ist, falls dies passiert, wird der Wert true zurückgegeben, ansonsten false. Der Wert der Hand vom Dealer wird mit der Methode, die sich in der Game-Klasse befindet, geholt(GetDealerPoints).

### **Balance**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert den Datentyp int zurück. Die Methode liefert den vorhandenen Geldbetrag, der sich im Wallet des Spielers befindet, zurück .Hier wird eine Methode benutzt, die sich in der Klasse Player befindet, auf die aber später näher eingegangen wird.

### **GetMoney**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Bei GetMoney wird dafür gesorgt, dass der Spieler seinen Gewinn bekommt. Dies geschieht mit einer Methode, die sich in der Player-Klasse befindet. Auf diese wird, wie bei den anderen Methoden, später Bezug genommen.

### **SaveGame**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Hier wird der Spielstand des aktuellen Spiels mit dem mitgegebenen File-Namen gespeichert, indem zuerst die Karten, der Wert des Wallets und der Kartenwert des Spielers gespeichert werden. Um danach die Karten und den Kartenwert des Dealers zu speichern. Diese Daten werden in einer Datei hinzugefügt.

# **Player**

## **Was macht die Klasse?**

* Diese Klasse wird dazu verwendet, um einen neuen Spieler zu erstellen, damit man diesen in der Game-Klasse bzw. im GUI verwenden kann.

### **Bet**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Sie wird verwendet damit der Spieler einen Einsatz, der im Parameter mitgegeben wird, tätigen kann. Zudem basiert diese Methode auf die Methode in der Wallet-Klasse, die sich WithDraw nennt. Die Methode Bet selbst wird in der Game-Klasse(SetBet) verwendet.

### **TakeMoney**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Sie wird verwendet damit der Spieler seinen Gewinn entnimmt. Zudem basiert diese Methode auf die Methode in der Wallet-Klasse, die sich Deposit nennt. Die Methode TakeMoney-Methode wird in der Game-Klasse(GetMoney) verwendet.

### **GetBalance**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert einen int-Wert zurück. Sie wird verwendet damit der Spieler seinen aktuellen Wallet-Wert bekommt. Diese Methode verwendet dafür die Eigenschaft „Balance“, die sich in der Wallet-Klasse befindet. Die Methode selbst wird in der Game-Klasse (Balance) verwendet.

### **Player**

#### **Was macht die Methode und was wird von ihr erwartet?**

Dies ist ein Konstruktor. Im Konstruktor wird ein Wallet erstellt und mit einem Starbudget ausgestattet, welches in der Wallet-Klasse definiert wird.

# **Wallet**

## **Was macht die Klasse?**

* Die Wallet-Klasse ist dazu zuständig, damit der Spieler seine Wette setzen kann, bzw. seinen Gewinn auszahlen kann.

### **Wallet**

#### **Was macht die Methode und was wird von ihr erwartet?**

Dies ist keine Methode, sondern der Konstruktor. Im Programm sind 2 Konstruktoren enthalten, wobei ein Konstruktor einen Parameter mit dem Datentyp int übergeben bekommt. Wird kein Wert mitgegeben, so bekommt das Wallet einen fixen Startwert von 100$, ansonsten wird der mitgegebene Wert eingetragen.

### **Deposit**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Jedoch besitzt diese Klasse einen Parameter -im Parameter wird ein int-Wert mitgegeben-. Diese Methode dient dazu, um Geld in die Wallet einzuzahlen, d.h, wenn der Spieler seinen Gewinn einzahlen möchte. Zudem wird diese Methode in der Klasse Player benutzt(TakeMoney).

### **WithDraw**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Die Methode beinhaltet einen Parameter mit einem int-Wert. Sie wird benutzt, um Geld abzuheben(vom Wallet), damit der Spieler seine Wette setzen kann. Die Methode wird in der Player-Klasse verwendet.

# **Dealer**

## **Was macht die Klasse?**

* Diese Klasse wird dazu verwendet, um einen neuen Spieler zu erstellen, damit man diesen in der Game-Klasse bzw. im GUI verwenden kann. Zusätzlich ist der Dealer dazu verantwortlich das Stack zu erstellen und verwalten, damit gespielt werden kann.

### **Dealer**

#### **Was macht die Methode und was wird von ihr erwartet?**

Dies ist keine Methode, sondern der Konstruktor. Im Konstruktor wird ein neues Stack erstellt, damit es dann mit den Methoden verbunden werden kann z.B Shuffle().

### **TakeCardStack**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert eine Karte zurück. In dieser Methode nimmt der Dealer die oberste Karte vom Stack (mit der GetCard-Methode, die sich in der Stack-Klasse befindet), um sie später dem Spieler geben zu können.

### **CreateNewStack**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. In dieser Methode wird zuerst ein neues Deck erstellt, damit dann ein neues Stack mit dem Deck erstellt werden kann. Am Ende wird das Stack nochmal gemischt (mit der Methode Shuffle, die sich in der Stack-Klasse befindet).

# **Person**

## **Was macht die Klasse?**

* Das ist die Superklasse für die Player- und Dealer-Klasse. Die Klasse selbst, ist dazu da um die vorgegeben Methoden zu verwenden(die beide brauchen) und um so wenig wie möglich Programmcode zu schreiben.

### **TakeCard**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück, hat aber eine Karte als Parameter. In dieser Methode bekommt die jeweilige Person(Dealer, Player) eine Karte in seine Hand (mit der Methode, die sich in der Hand- Klasse befindet „TakeCard“). Also der Dealer nimmt sich eine Karte vom Stack(diese befindet sich zuerst im Parameter) und gibt sie auf seine Hand. Der Player bekommt vom Dealer die Karte(befindet sich zuerst im Parameter) und gibt sie in seine Hand.

### **GetPoints**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert den Datentyp int zurück. Mit dieser Methode bekommt man den Hand-Wert der jeweiligen Person(mit der Eigenschaft, die sich in der Hand-Klasse „Points“) befindet.

### **ClearHand**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert den Datentyp void zurück. Mit dieser Methode wird der Wert der Hand, der jeweiligen Person, zurückgesetzt. Dies geschieht mit der Methode “ClearHand“, die sich in der Hand-Klasse befindet.

### **GetCards**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert eine Liste von Karten zurück. Bei dieser Methode wird die Eigenschaft „CompleteHand“(befindet sich in der Hand-Klasse) verwendet, damit man alle Karten, die sich in der jeweiligen Hand befinden, bekommen kann.

### **Person**

#### **Was macht die Methode und was wird von ihr erwartet?**

Dies ist ein Konstruktor. In diesem wird nur eine neue Hand erstellt.

# **Stack**

## **Was macht die Klasse?**

* Diese Klasse wird dazu verwendet, damit eine neues Stack erstellt werden kann.

### **GetCard**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert eine Karte zurück. Bei dieser Methode wird eine Karte von der obersten Stelle genommen. Diese Methode wird in der Methode „TakeCardStack“ verwendet, damit der Dealer eine Karte nehmen kann.

### **Shuffle**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. In dieser Methode werden die Karten, die sich im Stack befinden, vermischt. Dies passiert mithilfe der Random-Klasse. Shuffle wird in der Dealer- Klasse verwendet(„CreateNewStack“).

### **AddDeck**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück und hat einen Parameter(dort befindet sich eine Instanz, der Klasse Deck). Bei dieser Methode wird mithilfe der Methode der Klasse Deck („GetCards“) ein neues Deck in das Stack gegeben.

### **Stack**

#### **Was macht die Methode und was wird von ihr erwartet?**

Das ist der Konstruktor. Im Konstruktor wird eine neue Liste von Karten erstellt für das Stack.

# **Hand**

## **Was macht die Klasse?**

Diese Klasse wird dazu verwendet, damit der Kartenwert ausgegeben werden kann(Dealer, Player).

### **TakeCard**

#### **Was macht die Methode und was wird von ihr erwartet?**

Die Methode liefert nichts zurück hat aber wohl einen Parameter. Im Parameter ist eine Karte der Klasse BlackjackCard. In dieser Methode wird eine übergebene BlackjackCard in die Liste bjcs hinzugefügt.

### **ClearHand**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert void zurück. In dieser Methode werden die Karten, die sich auf der Hand gelöscht, besser gesagt die Liste bjcs wird geleert.

### **Hand**

#### **Was macht die Methode und was wird von ihr erwartet?**

Das ist der Konstruktor der Klasse. Hier wird die Liste bjcs erstellt.

# **BlackjackCard**

## **Was macht die Klasse?**

Diese Klasse wird dazu verwendet, um die Kartenwerte so umzuwandeln, dass es für das spätere Spiel richtig gezählt werden kann. Dazu erbt diese Klasse von der Super-Klasse Card.

### **Value**

#### **Was macht die Methode und was wird von ihr erwartet?**

Das ist eine Eigenschaft und liefert den Datentyp int zurück. Hier werden die Karten so umgewandelt, dass die Karten im späteren Spiel richtig gezählt werden können. Wenn die Karte z.B ein Ass, dann wird der Wert 11 für die Karte gespeichert. Und alle Bildkarten werden mit dem Wert 10 gezählt.

### **BlackjackCard**

#### **Was macht die Methode und was wird von ihr erwartet?**

Das ist der Konstruktor der Klasse, der sämtliche Parameter, aufgrund der zur erbenden Klasse Card, enthält. Hier wird nur die Instanz „value“(befindet sich eigentlich in der Klasse Card) ein gerechter Startwert zugeordnet.

# **Card**

## **Was macht die Klasse?**

Diese Klasse wird dazu verwendet, um eine Karte zu erstellen. Auch werden hier der richtige Wert bzw. das richtige Bild für die Karte zugewiesen. Dazu ist diese Klasse die Super-Klasse für die Klasse BlackjackCard.

### **Card**

#### **Was macht die Methode und was wird von ihr erwartet?**

Das ist der Konstruktor der Klasse, der die folgenden Parameter besitzt: Instanz des Enums CardColor und CardValue. Im Konstruktor werden, die im Parameter vorhandenen Werten, den Instanzen, die sich in der Klasse selbst befinden, zugewiesen.

### **ToString**

#### **Was macht die Methode und was wird von ihr erwartet?**

Die Methode liefert einen string zurück. In dieser Methode wird mithilfe der Enums die Art der Karte als auch der Wert durch Trennung eines “\_“ zusammengefügt und zurückgeliefert.

***Bsp:*** “Karo\_Zwei“

# **CardValue**

## **Was macht die Klasse?**

Diese Klasse(Enum) wird dazu verwendet, um der Karte eine von 13 Werten zuzuweisen.

### **Werte**

* Ass = 1,
* Zwei,
* Drei,
* Vier,
* Fuenf,
* Sechs,
* Sieben,
* Acht,
* Neun,
* Zehn,
* Bube,
* Dame,
* Koenig

Dem Ass wird erst in der Klasse BlackjackCard der Wert 11, bzw. den Bildkarten der Wert 10 zugewiesen.

# **CardColor**

## **Was macht die Klasse?**

Diese Klasse(Enum) wird dazu verwendet, um der Karte die passende Art zuzuweisen, also(Herz, Pik, Karo, Kreuz)

# **Deck**

## **Was macht die Klasse?**

Diese Klasse wird verwendet, um eines neues mit der Klasse Card zu erstellen, damit danach ein neues Stack für das Spiel erstellen kann.

### **GetCards**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert eine Liste von Karten zurück. In dieser Methode wird das Deck zurückgeliefert, besser gesagt, die Liste deck mit den vorhandenen Karten.

### **Deck**

#### **Was macht die Methode und was wird von ihr erwartet?**

Dies ist der Konstruktor. Hier werden die ganzen Karten erstellt für das Deck. Das heißt, dass 2 for-Schleifen verwendet werden, eine ist für die Erstellung der Art der Karten und die andere für die Werte. Die erste ist für die Art und muss deswegen 4 durchlaufen, wegen (Karo, Pik, Kreuz, Herz). Die 2. muss 13 durchlaufen werden.

# **BlackJack(GUI)**

## **Was macht das GUI?**

Das GUI verwendet die Game-Klasse als Logik, und ist Benutzeroberfläche für das eigentliche Spiel.

### **btnHit\_Click()**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Diese Methode gibt dem Spieler eine Karte in die Hand. Zuerst wird eine PictureBox für die neue Karte erstellt, um danach die Karte in dieser PictureBox anzuzeigen und in das FlowLayoutPanel des Spielers hinzuzufügen. Am Ende der Methode wird die Methode BustPlayer der Game-Klasse und die Methode ControlStructures() aufgerufen, die sich im GUI befindet.

### **neuesSpielToolStripMenuItem\_Click()**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. In dieser Methode wird ein neues Spiel gestartet, falls darauf gedrückt wird. Im ersten Schritt werden die Labels, wo sich die Werte der Hände befinden, auf 0 und die vorhandenen Buttons auf false gesetzt. Danach kommt ein neues Fenster auf, wo angefordert wird, wie viel der Spieler setzen will (FrmEinsatz). Hat der Spieler seine Wette getätigt, wird seine Wallet-Balance und sein Einsatz angezeigt, wobei das Wallet entsprechend der Wette angezeigt werden muss. Im folgenden Schritt wird ein neues Spiel gestartet(mit entsprechender Methode aus der Game-Klasse). Das FlowLayoutPanel des Dealers bzw. Players wird mit jeweils 2 Pictureboxen befüllt und Karten, die sie bekommen haben, wobei die 1. Karte des Dealers verdeckt ist. Am Ende wird geprüft, ob einer der beiden Personen BlackJack hat, ist dies der Fall wird der entsprechende Verlust/Gewinn dem Spiel gutgeschrieben. Nach dem das passiert ist, wird mit der vorhandenen Methode ControlStructures weiter abgefragt.

### **beendenToolStripMenuItem\_Click()**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Diese Methode beendet das Spiel mithilfe „Appliaction.Exit()“.

### **ControlStructures()**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Diese Methode ist dazu zuständig sämtliche abfragen auszuführen, also ob der Spieler kein Geld mehr hat, noch spielen möchte oder auch nicht, nachdem er verloren/gewonnen hat.

### **btnStay\_Click()**

#### **Was macht die Methode und was wird von ihr erwartet?**

Diese Methode liefert nichts zurück. Diese Methode startet den Zug des Dealers. Falls der Spieler zufrieden mit seinem Stand ist, kann er den jeweiligen Button drücken, um diese Methode zu starten. Zuerst wird die erste Karte des Dealer aufgedeckt, damit im Anschluss mit der entsprechenden Methode(„Stay“), die sich in der Game-Klasse befindet, der Zug gestartet werden kann. Wenn der Dealer fertig mit seinem Zug ist werden seine Karten mit der Methode „GetDealerCards“ aktualisiert, um sie dann durch eine for-Schleife aufzudecken und im FlowLayoutPanel anzuzeigen. Ist dies erledigt, wird abgefragt, welche Person verloren, gewonnen oder über den Kartenwert 21 gekommen ist. Dies alles passiert, wieder durch die entsprechenden Methoden aus der Game-Klasse. Wenn der Player gewonnen hat, bekommt er seinen Gewinn, ansonsten geht er leer aus. Es kann aber auch sein, dass ein Gleichstand der Kartenwerte besteht, also die Werte des Dealers/Players. In diesem Fall bekommt der Spieler seinen Einsatz zurück. Am Ende wird wieder Methode ControlStructures() ausgeführt.