T1 Programmierung

Prof. Dr. rer. nat. Alexander Auch

Programmentwurf ON23A Gruppe 1

Roulette

**Gruppenmitglieder**

Tobias Wawak (7806387)

Jonas Bauer (2168965)

Julian Köhnlein (7561542)

Pia Kühnle (9395955)

Elia Küstner (2018795)

Inhaltsangabe

1. **Spielidee**
   1. historischer Kontext
   2. Spielerkonto
   3. Setzmöglichkeiten und Gewinnchancen
   4. erneutes Spielen
   5. optische Drehen des Rads auf der Konsole
2. **Initialisierung der Klasse Roulette**
   1. Variablen
   2. Arrays
   3. Hashmaps
3. **Methoden** 
   1. allgemeine Informationen
   2. main ()
   3. kontoAufladen ()

ss

**1. Spielidee**

**1.1 Einleitung (historischer Kontext)**

Das Roulette-Glücksspiel hatte seinen Ursprung in Frankreich im 18. Jahrhundert in Paris. Der Name “Roulette” ist französisch und bedeutet “kleines Rad”. Das Spiel verbreitete sich schnell in ganz Europa und erlangte anschließend im 19. Jahrhundert auch Popularität in den USA, wodurch viele neue Varianten des Spiels entstanden. Roulette ist bis heute ein sehr populäres Spiel in der Gesellschaft weltweit.

**1.2 Spielprinzip**

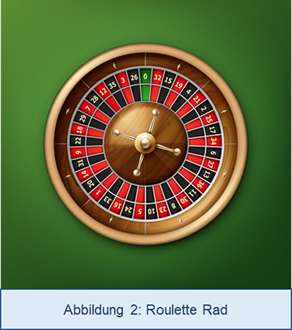
Das Herzstück des Roulettes ist das rotierende Rad, das in nummerierte und farbige Taschen unterteilt ist. Die Nummerierungen reichen von 0 bis 36. Die Felder sind nach einem komplexen System angeordnet, das gewährleisten soll, dass die Chancen, auf die gewettet werden kann, möglichst gleichmäßig angeordnet sind. Auf dem Roulette Board sind die Felder abwechselnd in rot und schwarz gefärbt, eine Ausnahme bietet hier das Feld 0, welches in den meisten Roulette-Varianten grün gefärbt ist.

Das Spiel beginnt mit einer Phase des Wettens, in der der Spieler Geldbeträge auf mögliche Spielausgänge setzt. Im physischen Casino setzt der Spieler bei diesem Vorgang Chips auf ein sogenanntes Roulette Tableau (Abbildung 1). Hier gibt es viele Möglichkeiten, auf die der Spieler setzen kann, die je nach Version des Spiels variieren. Aufgrund der Komplexität von Roulette haben wir uns bei der Spiel-Implementierung auf die beiden Möglichkeiten des Setzens auf einzelne Zahlen und Farben beschränkt. Bei der Umsetzung des Spiels haben wir zudem darauf Wert gelegt, das Drehen des Rads visuell in der Konsole aufzugreifen.



**Tabelle: Setzmöglichkeiten**

|  |  |  |
| --- | --- | --- |
| Setzmöglichkeiten | Gewinnwahrscheinlichkeit | Gewinn |
| **Farben (Rot / Schwarz / Grün)** | | |
| Rot / Schwarz | 48,6% | 1:1 |
| Grün | 2,7% | 35:1 |
| **einzelne Zahlen** | | |
| Zahl | 2,7% | 35:1 |



Nachdem die Spieler ihre Beträge gesetzt haben, wird das Rad (Abbildung 2) vom Croupier in Rotation versetzt und eine kleine gegen die Laufrichtung geworfen. Das Feld, auf dem die Kugel zum Stoppen kommt, bestimmt über die Gewinne und Verluste der einzelnen Spieler. Die Gewinne werden anschließend ausgezahlt.

Durch die vielen Wettmöglichkeiten können die Wetten je nach Risikobereitschaft und Budget angepasst werden. Durch das einfache Spielkonzept eignet sich Roulette gleichermaßen für Anfänger und erfahrene Spieler. Durch sein charakteristisches Rad und die Spannung bei jedem Spin hat es seinen festen Platz in der Welt der Glücksspiele.

**2. Initialisierung der Klasse Roulette**

**2.1 Variable**

Zuallererst wird in der Klasse Roulette eine Boolean Variable mit dem Namen “spielen” initialisiert und mit dem Wert true gefüllt. Diese Variable sagt aus, ob das Spiel gerade läuft oder nicht.

**2.2 Arrays**

In der Klasse werden die drei Arrays rot, Spiel und Farben definiert. Das Array rot wird mit den Zahlen gefüllt, die auf einem Roulette-Board rot gefärbt sind. Das Spiel Array enthält die Werte Zahl und Farbe, was später bei der Auswahl der Wette zum Einsatz kommt. Im Array Farben sind die drei Farben gespeichert, die auf einem Roulette-Board vertreten sind, also grün, schwarz und rot.

**2.3 Hashmaps**

In der Roulette Klasse werden zwei Hashmaps definiert, die Hashmaps Wheel und FarbPunkte. In Farbpunkte wird jeder Farbe ein Farbpunkt zugeordnet, was später bei der Darstellung des Roulette Rads benötigt wird. Die Hashmap Wheel ordnet jeder Zahl, die sich auf dem Roulette Rad befindet, einen Farbpunkt in der richtigen Farbe zu.

**3. Methoden**

**3.1 allgemeine Informationen**

Um Redundanzen im Code möglichst zu vermeiden, haben wir Bestandteile / Elemente des Codes, die sich häufig wiederholen, in Methoden ausgelagert. Das Beheben von Bugs, die Verständlichkeit des Codes für die anderen Teammitgliedern und die Länge des Codes profitieren von dieser ergriffenen Maßnahme.

**3.2 main ()**

Die main-Methode hat die Signatur **public static void main(String[] args) throws InterruptedException.** Diese Methode begrüßt zuallererst den Spieler mit einer Willkommensnachricht auf der Konsole und ruft danach die Methode **kontoAufladen()** auf und lässt sich von dieser einen Wert für die vorher definierte Integer Variable “konto” geben. Im Anschluss wird die Methode **kontostandAusgeben()** aktiviert und der die Variable “konto” samt vorher definierten Kontostand mitgeliefert. Nachdem auch diese abgeschlossen ist, wird die Methode **roulette()** aktiviert. Diese Methoden befinden sich in einem Try-Catch-Block, hiermit wird bewirkt, dass bei Fehlern im Programm folgende Fehlermeldung auf der Konsole ausgegeben wird “Es gab einen Fehler. bitte Spiel neustarten!”.

**3.3 kontoAufladen ()**

Die Methode hat die Signatur **public static int kontoAufladen(Scanner scanner)** und umfasst den Vorgang der Aufladung des Kontos. Der Spieler wird durch den Satz “Wie viel möchten Sie aufladen? (1-1000€ möglich)” dazu aufgefordert, seinen gewünschten Wert in die Konsole einzugeben. Die darauf folgende Eingabe des Spielers wird per Scanner erfasst und in der Integer Variable “eingabe” gespeichert. In der Bedingung einer While Schleife wird die Methode **pruefeEingabeKontostand()** aufgerufen, in der überprüft wird, ob die Eingabe gültig ist. Solange diese Methode “false” zurückliefert, wird dem Spieler folgender Satz auf der Konsole ausgegeben “Bitte einen Betrag zwischen 1€ und 1000€ eingeben:” und der Aufladeprozess startet von neuem. Wenn die Eingabe erfolgreich ist, gibt die Methode pruefeEingabeKontostand() “true” zurück womit die While-Schleifenbedingung nicht mehr erfüllt ist und diese damit beendet wird. Mit dem return der Variable “eingabe” ist diese Methode beendet.