Programmieren II (Java)

4. Praktikum: Vertiefung Objektorientierung

Sommersemester 2022

Christopher Auer, Tobias Lehner

Abgabetermine

Lernziele

- Vertiefung Objektorientierung in Java
- Abstrakte Klasse, Ableitung, `super`-Methodenaufrufe
- Interfaces, implementieren von Interfaces
- Gleichheit und Identität

Hinweise

- Sie dürfen die Aufgaben *alleine* oder zu *zweit* bearbeiten und abgeben
- Sie müssen *4 der 5* Praktika bestehen
- *Kommentieren* Sie Ihren Code
  - Jede *Methode* (wenn nicht vorgegeben)
  - *Wichtige* Anweisungen/Code-Blöcke
  - *Nicht kommentierter* Code führt zu *Nichtbestehen*
- Bestehen Sie eine Abgabe *nicht* haben Sie einen *zweiten Versuch*, in dem Sie Ihre Abgabe *verbessern müssen*
- *Wichtig*
  - Sie sind einer *Praktikumsgruppe* zugewiesen, *nur* in dieser werden Ihre Abgaben *akzeptiert*
  - Beachten Sie dazu die Hinweise auf der Moodle-Kursseite

Aufgabe 1: Spielautomaten ★☆ bis ★★

In dieser Aufgabe werden die Spielautomaten eines Herstellers in Java umgesetzt. Der Hersteller produziert momentan drei unterschiedliche Automaten:

- Auf dem Gerät *Roulette* können die Roulettespiele
  - *Plein* - Hierbei wird auf eine konkrete Zahl zwischen 0 und 36 gesetzt. [36]
  - *Rouge* - Hierbei wird auf rote Zahlen gesetzt (1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30, 32, 34 und 36). [2]
  - *Noir* - Hierbei wird auf schwarze Zahlen gesetzt (2, 4, 6, 8, 10, 11, 13, 15, 17, 20, 22, 24, 26, 28, 29, 31, 33 und 35). [2]
  - *Pair* - Hierbei wird auf gerade Zahlen gesetzt. [2]
  - *Impair* - Hierbei wird auf ungerade Zahlen gesetzt. [2]
  - *Manque* - Hierbei wird auf die Zahlen zwischen 1 und 18 gesetzt. [2]
  - *Passe* - Hierbei wird auf die Zahlen zwischen 19 und 36 gesetzt. [2]
  - *Premiere douzaine* - Hierbei wird auf das erste Duzent Zahlen gewettet, also die Zahlen zwischen 1 und 12. [3]
  - *Moyenne douzaine* - Hierbei wird auf das mittlere Duzent Zahlen gewettet, also die Zahlen zwischen 13 und 24. [3]
  - *Derniere douzaine* - Hierbei wird auf das letzte Duzent Zahlen gewettet, also die Zahlen zwischen 25 und 36. [3]

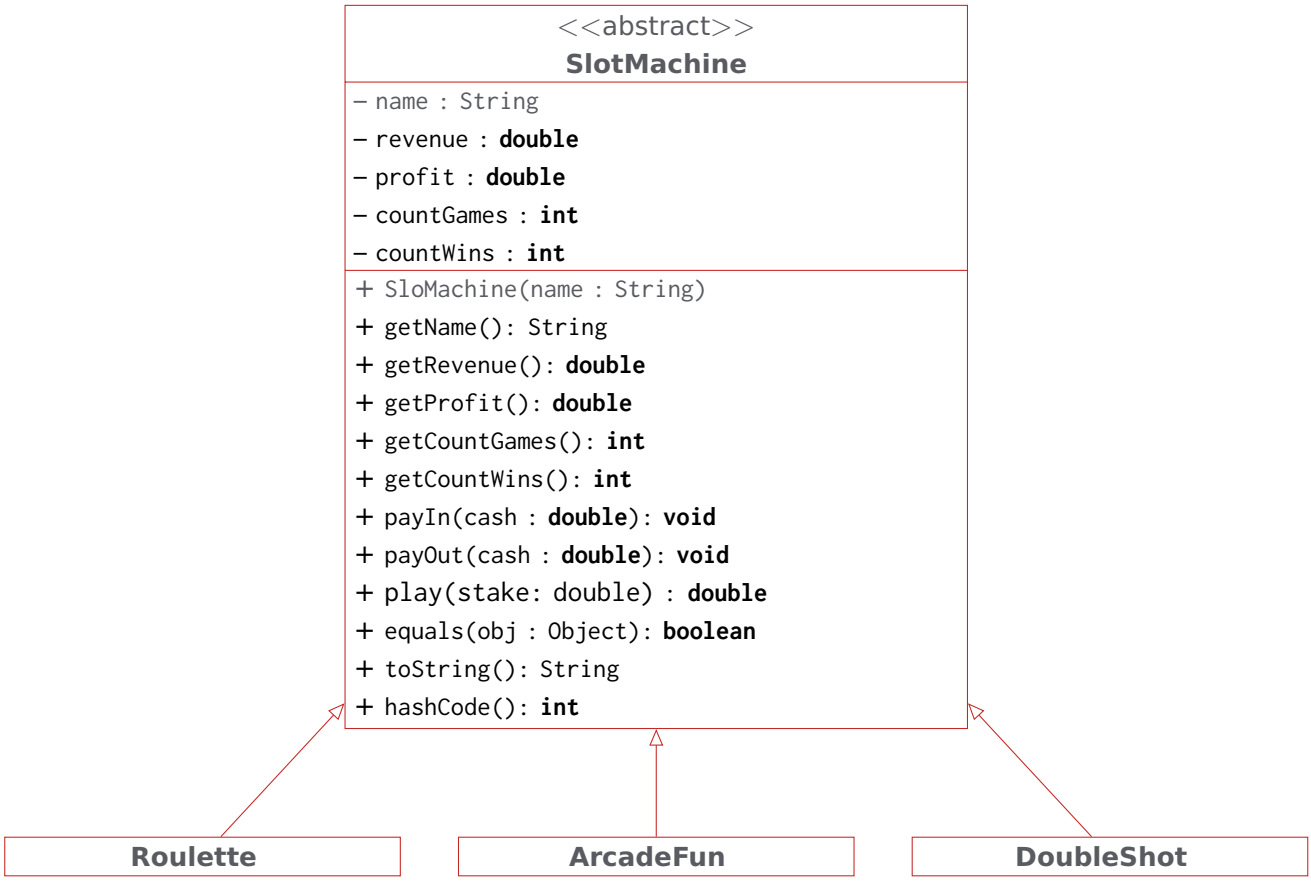
  gespielt werden. In eckigen Klammern sind jeweils die Auszahlungsfaktoren enthalten. Gewinnt ein Spieler das Spiel Noir, erhält er den doppelten Einsatz zurück.
- Der Automat *Double Shot* ist ein klassischer Geldspielautomat: Gewinnt der Spieler, erhält er den doppelten Einsatz zurück. Von jedem Einsatz gehen 10 Prozent in einen Jackpot, der mit einer Wahrscheinlichkeit von 0,1 Prozent gewonnen wird. Der Automat zahlt statistisch über einen langen Zeitraum betrachtet 92 Prozent seiner Einsätze wieder aus. Nur 8 Prozent vom Umsatz sollten also Profit sein. Die ausgezahlten Jackpots zählen ebenfalls zur Auszahlungsquote.
- Ganz neu ist der Automat *Arcade Fun*. Hierbei handelt es sich um keinen Geldspielautomaten. Der Automat soll Jugendliche und jung Gebliebene faszinieren und Freude bereiten ohne dass regulatorische Anforderungen für den Aufsteller entstehen.

Implementieren Sie alle nachfolgenden Ausführungen innerhalb eines Packages `gamblinghall`*.*

Selbstverständlich steht es Ihnen frei, eigene Attribute und Methoden zu ergänzen, wo Sie dies für sinnvoll halten.

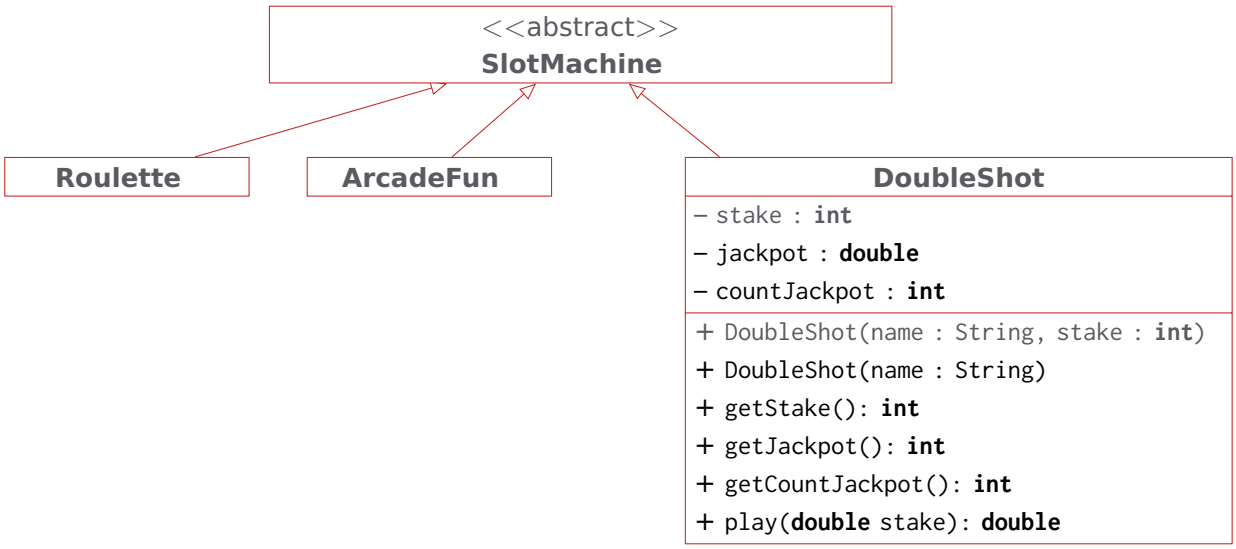
Abstrakte Klassen und Vererbung

Zunächst soll die abstrakte Oberklasse `SlotMachine` implementiert werden:



- Deklarieren Sie die Klasse SlotMachine mit deren Attributen. Das Attribut `name` soll nach der Initialisierung nicht mehr veränderbar sein.
  - `revenue` beinhaltet den bisher erzielten Umsatz des Automaten.
  - `profit` beinhaltet den bisher erzielten Profit des Automaten.
  - `countGames` stellt einen Zähler für alle bisher durchgeführten Spiele dar.
  - `countWins` stellt einen Zähler für alle gewonnenen Spiele dar.
- Implementieren Sie den Konstruktor zur Klasse. Der Parameter `name` darf weder **`null`** noch leer sein. Alle anderen Attribute der Klasse werden bei der Initialisierung auf 0 gesetzt.
- Erstellen Sie die *Getter-Methoden* zu den Klassenattributen. *Setter-Methoden* sollen *nicht implementiert* werden.
- Die Methode `payIn` stellt eine Einzahlung dar. Der Eingabeparameter muss größer als 0 sein.
- Die Methode `payOut` stellt eine Auszahlung dar. Der Eingabeparameter muss größer als 0 sein.
- Die abstrakte Methode `play` wird zwar definiert aber erst durch die ableitenden Klassen implementiert.
- `toString` soll das Attribut `name` ausgeben.
- `hashCode` und `equals` inkludieren (neben der Klasse) das Attribut `name` als unterscheidendes Merkmal.

Die Klasse `DoubleShot`



- Deklarieren Sie die Klasse `DoubleShot` mit deren Attributen und den zugehörigen *Getter-Methoden*. Das Attribut `stake` soll nach der Initialisierung nicht mehr veränderbar sein.
  - `stake` steht für den Einsatz, den ein Spieler pro Spiel einsetzen muss.
  - Im Attribut `jackpot` wird der aktuelle Jackpot gespeichert. Der Jackpot erhöht sich mit jedem Spiel um 10 Prozent des Einsatzes.
  - `countJackpot` zählt, wie oft schon ein Jackpot gewonnen wurde.
- In der Klasse `DoubleShot` sind zwei Konstruktoren zu implementieren: Im Konstruktor `DoubleShot(String name, int stake)` kann der Einsatz `stake` initialisiert werden. Der Einsatz muss ganzzahlig zwischen 1 und 10 sein. Im zweiten Konstruktor wird `stake` nicht mit übergeben. Hier wird `stake` auf `1` gesetzt.
- Die Methode `play` enthält die ganze Logik des Automaten. Implementieren Sie `play` derart, dass der Spieler zufällig gewinnt oder verliert (auch Jackpot). Langfristig müssen die statistischen Vorgaben eingehalten werden (Jackpot wird mit p = 0,1 Prozent gewonnen, Auszahlungsquote 92 Prozent). `play` erwartet einen Parameter `stake`. Da der Automat nur mit einem fixen Einsatz spielt, muss überprüft werden, ob sich die beiden Werte gleichen.
- Überlegen Sie sich, warum `equals` und `toString` nicht implementiert werden müssen.

`main`-Methode (Teil 1)

- Erstellen Sie eine Klasse `GamblingHallMain`, in der Sie Ihre `main`-Methode programmieren.
- Erstellen Sie mehrere `DoubleShot`-Automaten und spielen 100.000 Spiele.
- Wie hat sich der Profit der beiden Automaten entwickelt? Ist er vergleichbar?
- Stimmen die statistischen Vorgaben auch im echten Programmablauf?
- Wie häufig wurden bei den unterschiedlichen Automaten Jackpots ausgezahlt?

Die Klasse `Roulette`

Die Klasse `Roulette` verwendet eine Enumeration `RouletteGameType`. In ihr sind die verfügbaren Roulette-Spiele enthalten. Die Enumeration enthält folgende Werte:

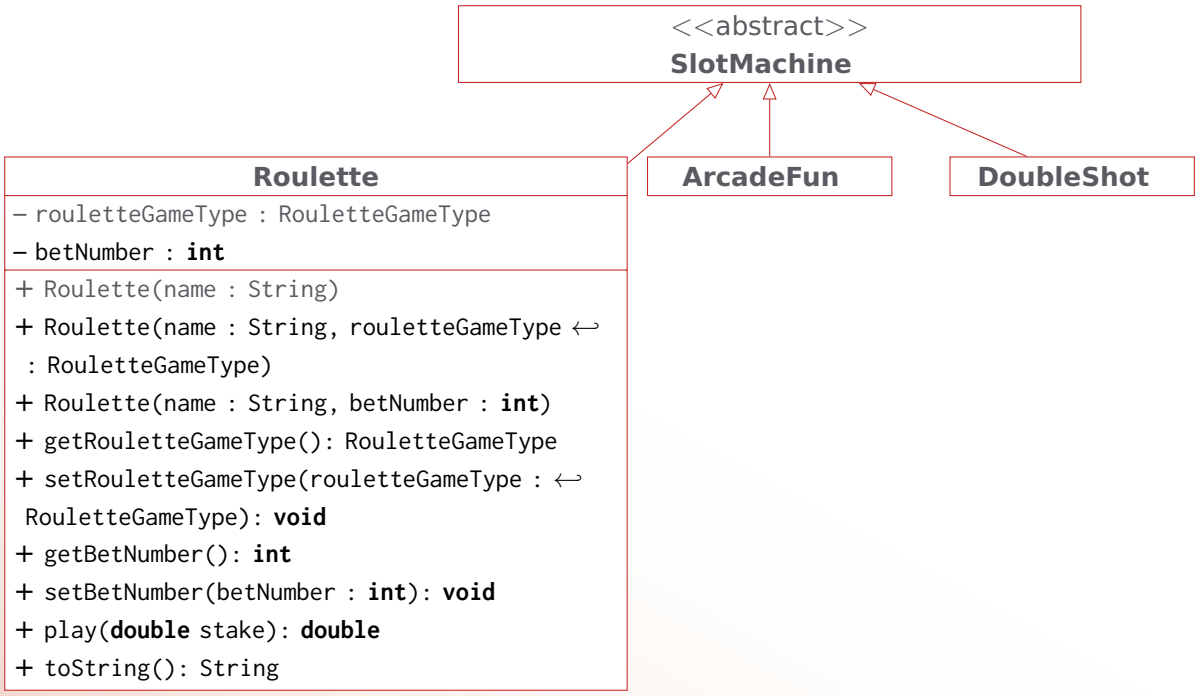
```
NUMBER, RED, BLACK, EVEN, ODD, LOW, HIGH, P12, M12, D12
```

Weiterhin liefert die Enumeration für jeden dieser Werte die Auszahlungsfaktoren sowie die Namen, die in Strings verwendet werden sollen:

Typ Enumeration	Name	Auszahlungsfaktor
NUMBER	Plein	52
RED	Rouge	2
BLACK	Noir	2
EVEN	Pair	2
ODD	Impair	2
LOW	Manque	2
HIGH	Passe	2
P12	Premiere douzaine	3
M12	Moyenne douzaine	3
D12	Derniere douzaine	3

- Implementieren Sie die Enumeration entsprechend der Vorgaben.
- Für Name und Auszahlungsfaktor sollten *Getter-Methoden* in der Enumeration existieren.

Weiterhin ist die Klasse `Roulette` zu implementieren:



- Deklarieren Sie die Klasse `Roulette` mit deren Attributen und den zugehörigen *Getter-Methoden* und *Setter-Methoden*.
  - `rouletteGameType` steht für das Spiel, welches als nächstes gespielt werden soll. `rouletteGameType` kann bei Bedarf vor jedem Spiel verändert werden, den ein Spieler pro Spiel einsetzen muss.
  - Wenn im `rouletteGameType` der Typ `NUMBER` gewählt ist, wird auf eine Nummer zwischen 0 und 36 gesetzt. Die Nummer, auf die gesetzt wird, ist in `betNumber` hinterlegt.
- In der Klasse `Roulette` sind drei Konstruktoren zu implementieren:
  - Im Konstruktor `Roulette(String name)` wird als `RouletteGameType` der Wert `BLACK` verwendet.

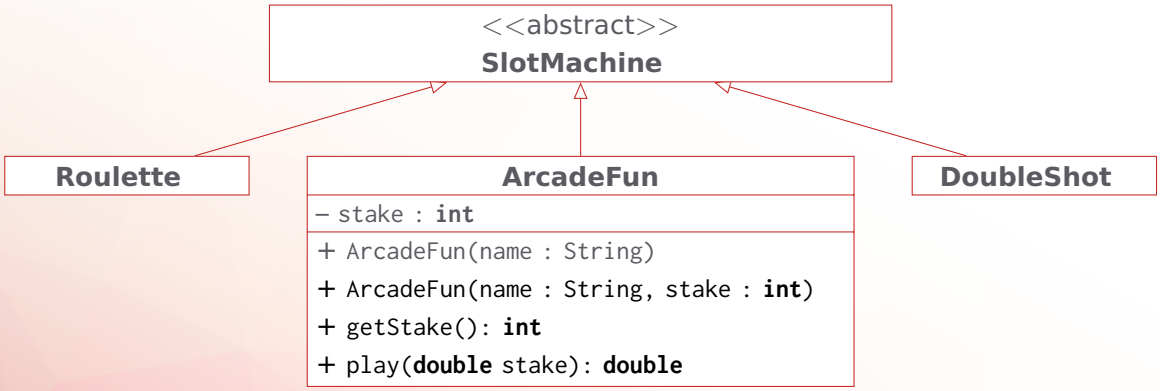
  - Im Konstruktor `Roulette(String name, RouletteGameType rouletteGameType)` wird das gewünschte Spiel ebenfalls übergeben. `rouletteGameType` darf weder **`null`** noch `NUMBER` sein. Wenn auf eine Zahl gesetzt werden soll, wird der dritte Konstruktor verwendet.
  - Im Konstruktor `Roulette(String name, int betNumber)` wird die Zahl, auf die gewettet werden soll, übergeben. Entsprechend ist der `RouletteGameType` hier per default `NUMBER`. Es muss kontrolliert werden, ob `betNumber` zwischen 0 und 36 ist.
- `setRouletteGameType` darf keine **`null`**-Werte entgegennehmen.
- `setBetNumber` darf nur Zahlen zwischen 0 und 36 entgegen nehmen.
- Die Methode `play` enthält die ganze Logik des Automaten. Implementieren Sie `play` derart, dass eine zufällige Zahl zwischen 0 und 36 gezogen wird. Anhand des `RouletteGameType` sowie der gezogenen Zahl sollten Sie überprüfen, ob ein Gewinn vorliegt. Wenn ein Gewinn vorliegt, multipliziert `play` den Einsatz mit dem Auszahlungsfaktor aus `RouletteGameType` und gibt diesen Wert zurück. Wenn der Spieler verloren hat, wird 0 zurückgegeben.
- `toString` soll folgenden beispielhaften String liefern: „Rouletteautomat Monte Carlo: Es wird Plein(8) gespielt“. Monte Carlo ist im Beispiel der Name des Automaten. Plein ist das gespielte Spiel. Nur bei Plein soll in Klammern die gesetzte Nummer ausgegeben werden. Ansonsten soll nur der Name des Spiels ausgegeben werden.

`main`-Methode (Teil 2)

- Erstellen Sie einen `Roulette`-Automaten und spielen 10.000.000 Spiele.
- Unser erster Automat hatte eine Auszahlungsquote von 92 Prozent. Wie hoch ist die Auszahlungsquote des Roulette-Automaten?
- Verändert sich die Auszahlungsquote bei den unterschiedlichen Spielen?

Die Klasse `ArcadeFun`

`ArcadeFun` repräsentiert einfache Automaten wie zum Beispiel Pinball. Der Einsatz bleibt immer gleich. Da der Gewinn des Spielers die Spielfreude ist, wird kein Geld ausbezahlt. Entsprechend einfach gestaltet sich auch die Klasse `ArcadeFun`.



- Deklarieren Sie die Klasse `ArcadeFun`, und deren Attribut `stake`. `stake` repräsentiert den Einsatz bzw. die „Münze“, die in den Automaten geworfen werden muss. `stake` soll nur beim Initialisieren einmalig beschreibbar sein, weswegen für dieses Attribut auch nur eine *Getter-Methode* geschrieben werden muss.
- In der Klasse `ArcadeFun` sind zwei Konstruktoren zu implementieren:
  - Im Konstruktor `ArcadeFun(String name)` wird `stake` mit den Wert 1 belegt.

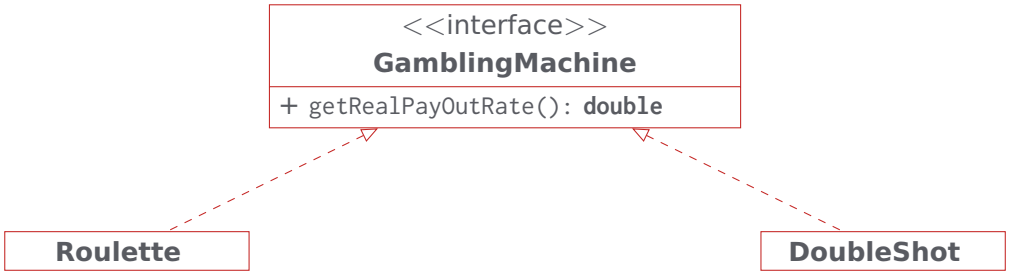
- Im Konstruktor `ArcadeFun(String name, int stake)` wird der gewünschte Einsatz übergeben. `stake` muss ein ganzzahliger Wert zwischen 1 und 10 sein.
- Bei der Methode `play` muss lediglich beachtet werden, dass der übergebene Wert `stake` mit dem Wert aus dem Konstruktor übereinstimmt.

`main`-Methode (Teil 3)

- Erstellen Sie einen oder mehrere `ArcadeFun`-Automaten und spielen Sie einige Spiele.
- Welche Auszahlungsquote hat ein `ArcadeFun`-Automat?

Das Interface `GamblingMachine`

Lange Zeit war die Auzahlungsquote eines Automaten mit einem Mindestwert vom Gesetzgeber reguliert (mindestens 50 Prozent). Damit dieser Wert stets von allen Geldspielautomaten anhand der realen Spieldaten abgerufen werden kann, soll ein Interface implementiert werden:



Das Interface wird von den Klassen `Roulette` und `SlotMachine` implementiert.

`main`-Methode (Teil 4)

- Verwenden Sie die `getRealPayOutRate`-Methode jetzt in Ihren bisherigen Automaten.
- *Testen Sie Ihre Klassen mit den JUnit-Testklassen.*