

Übungsblatt 08

0,5 P

Aufgabe 36a:

Nicht alles muss als
readonly deklariert werden

+RentalBike
-id:Integer {readonly, id >= 0} -hersteller:String {readonly, hersteller <= 20} -typ:Typ {readonly} -fruehlingsrabatt:Double = 0.8 {readonly} -mietpreis:Integer
+RentalBike(id:Integer, hersteller:String, typ:Typ, mietpreis:Integer) +checkId(in id:Integer) +setId(in id:Integer) {java.lang.IllegalArgumentException} +getId():Integer +checkHersteller(in hersteller:String) +setHersteller(in hersteller:String) {java.lang.IllegalArgumentException} +getHersteller():String {java.lang.IllegalArgumentException} +setTyp(in typ:Typ) +getTyp():Typ +setMietpreis(in mietpreis:Integer) {java.lang.IllegalArgumentException} +getMietpreis():Integer +getMietpreisFruehling():Integer

primitives müssen
nicht angegeben
werden

«primitive» double

«enumeration» Typ
CITY BIKE TOURING BIKE MOUNTAIN BIKE



Aufgabe 36b:

java.time.LocalDate vollständig angeben

+Restaurant
-name:String { name > 0, name <= 20 } -oeffnungszeit:LocalTime { closes At > opens At } -schliesszeit:LocalTime -gerichte:String [0..*] -adresse:Adresse
+checkName(in name:String) +setName(in name:String) {java.lang.IllegalArgumentException} +getName():String +checkOeffnungszeiten(in schliesszeit:LocalTime, in oeffnungszeit:LocalTime) +setOeffnungszeit(in oeffnungszeit:LocalTime) {java.lang.IllegalArgumentException} +getOeffnungszeit():LocalTime +setSchliesszeit(in schliesszeit:LocalTime) {java.lang.IllegalArgumentException} +getSchliesszeit():LocalTime +setGerichte(in Gerichte:String) +getGerichte():String +checkAdresse(in adresse:Adresse) +setAdresse(in adresse:Adresse) +getAdresse():Adresse

✓ Konstruktor

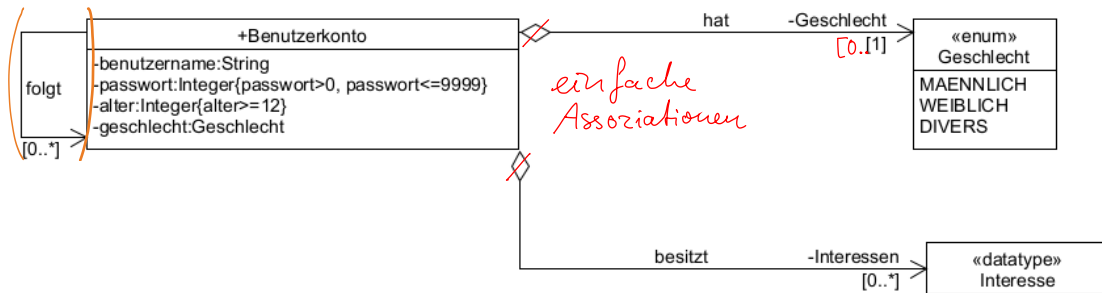
«datatype» Adresse
-stadt:Integer { stadt > 0 } -postleitzahl:Integer { postleitzahl < 10, postleitzahl >= 0 } -strassenname:String { strassenname > 0 } -hausnummer:Integer { hausnummer > 0 }

Das wären 5 einzelne
Zahlen in einem
Array. Unsicher, aber
wahrscheinlich möglich
(bitte nicht in der
Klausur)



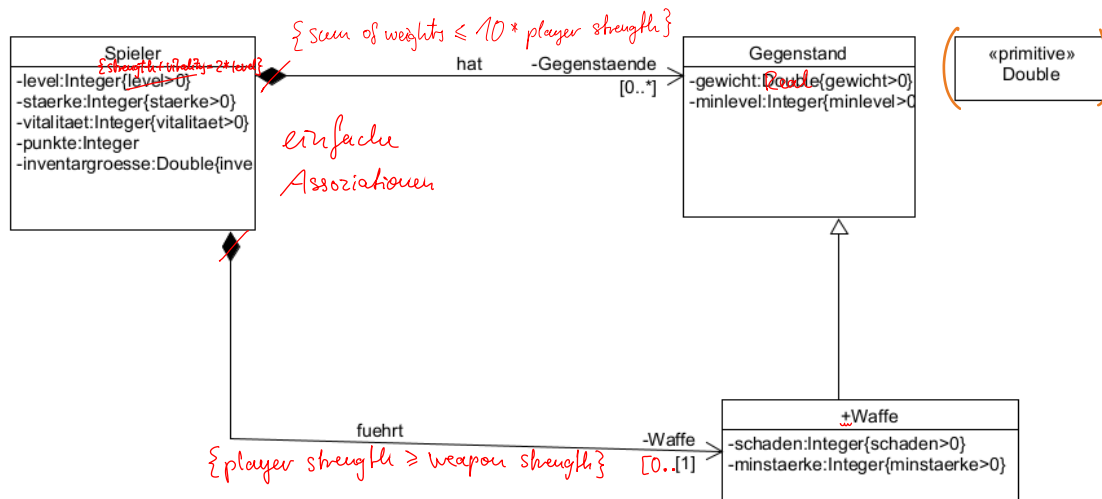
1P

Aufgabe 37a:



✓

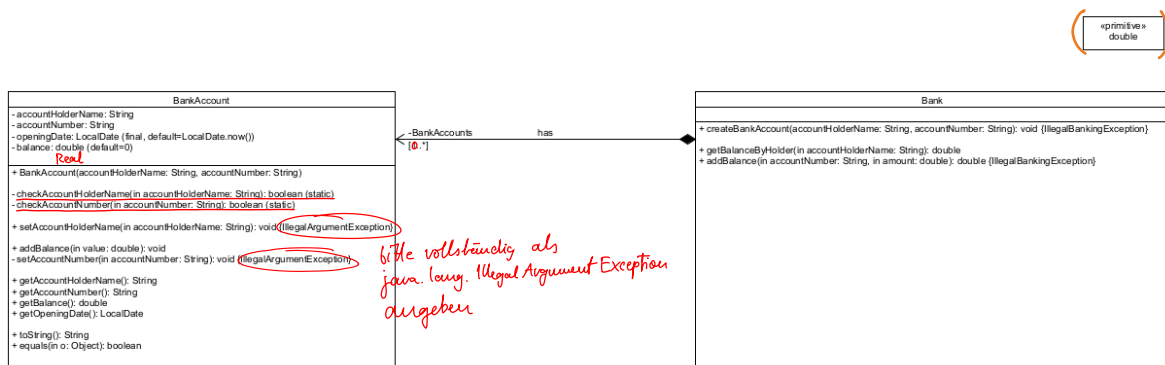
Aufgabe 37b:



✓

2P

Aufgabe 38 + 39



java.lang.Exception
↑
IllegalBankingException

```
public class Main {
    public static void main(String[] args) {

        Team team1 = new Team("Team1", new Player[]{});
        Team team2 = new Team("Team2", new Player[]{});

        Player player1 = new Player("Player1", team1);
        Player player2 = new Player("Player2", team2);

        System.out.println(player1.getTeam().equals(team1)); // Should be
true
        System.out.println(player2.getTeam().equals(team2)); // Should be
true
    }
}
```

```
public class Match {

    // -----

    // Attribute
    private Team team1;
    private Team team2;
    private Team winner;

    // -----

    // {winner == team1 or winner = team2}
    // {team2 != team1}
    // Implement via Setter

    // -----
    // H Konstruktor
}
```

```
import java.util.Objects;

class Player {

    // -----

    // Attribute
    private String name;
    private Team team;

    // -----

    // Konstruktor
    public Player(String name, Team team) {
        this.name = name;
    }
}
```

Informatik II – Tutorium 13 – Gruppe 7
Gruppenmitglieder: Hinterreiter, Gallinger, Stein

```
        setTeam(team);
    }

    // -----

    // setTeam
    public void setTeam(Team team) {
        if (Objects.equals(this.team, team)) {
            return; // Avoid recursion
        }

        this.team = team;
        team.addPlayer(this);
    }

    // -----

    // GetTeam
    public Team getTeam() {
        return team;
    }

    // -----
}
```

```
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

class Team {

    // -----

    // Attribute
    private static Set<String> teamNames = new HashSet<>();

    private String name;
    private Set<Player> players = new HashSet<>();

    // -----

    // Konstruktor
    public Team(String name, Player[] players) {
        if (teamNames.contains(name)) {
            throw new IllegalArgumentException("Team name must be unique");
        }
        this.name = name;
        teamNames.add(name);
        for (Player player : players) {
            addPlayer(player);
        }
    }

    // -----

    public void addPlayer(Player player) {
```

```
        players.add(player);  
        player.setTeam(this);  
    }  
  
    // -----  
  
    // removePlayer  
    public void removePlayer(Player player) {  
        players.remove(player);  
    }  
  
    // -----  
  
    @Override  
    public boolean equals(Object o) {  
        if (this == o) return true;  
        if (o == null || getClass() != o.getClass()) return false;  
        Team team = (Team) o;  
        return Objects.equals(name, team.name);  
    }  
  
    @Override  
    public int hashCode() {  
        return Objects.hash(name);  
    }  
  
    // -----  
}
```

✓ gelungene Umsetzung!

Σ 3,5 P schön!

JP