3,5P

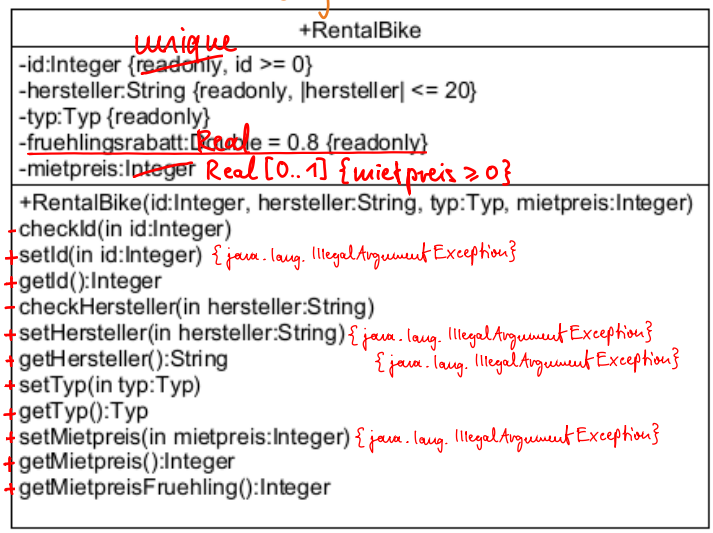
Informatik II – Tutorium 13 – Gruppe 7
Gruppenmitglieder: Hinterreiter, Gallinger, Stein

Übungsblatt 08

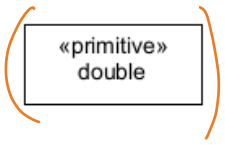
0,5P

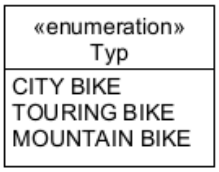
Aufgabe 36a:

Nicht alles muss als readonly deklariert werden



primitives müssen nicht angegeben werden

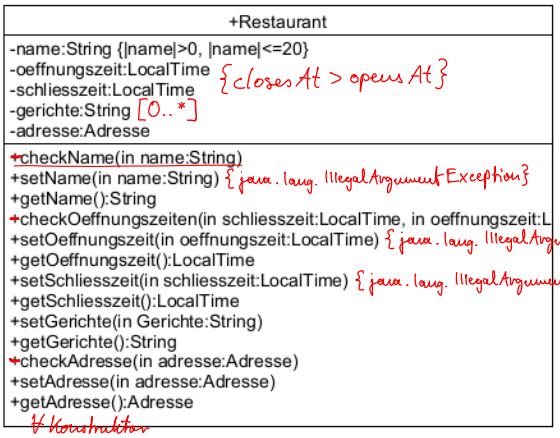


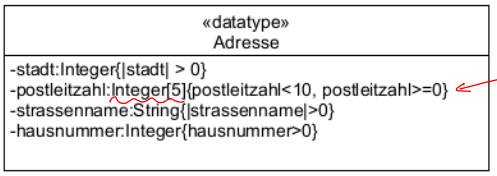


√

Aufgabe 36b:

java.Time.LocalDate vollständig angeben





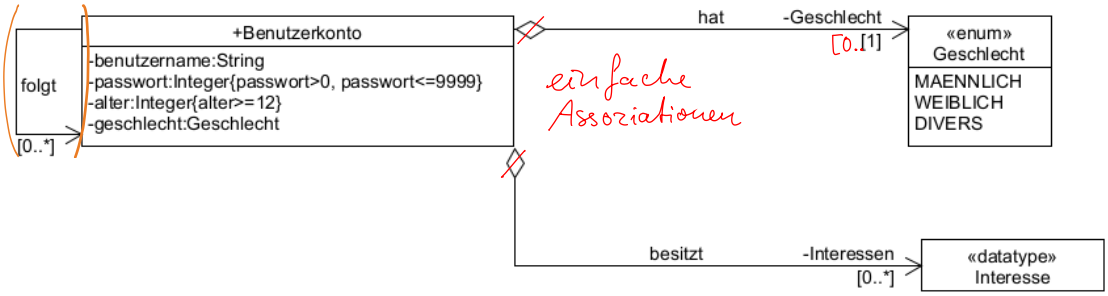
Das wären 5 einzelne Zahlen in einem Array. Umsetzbar, aber wahrscheinlich unmöglich (bitte nicht in der Klausur)

√

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

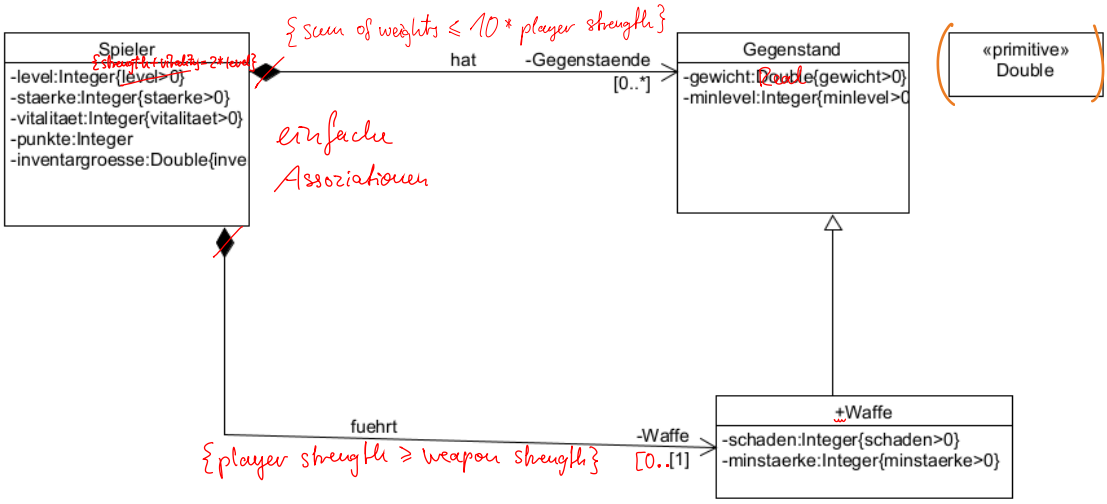
1P

Aufgabe 37a:



✓

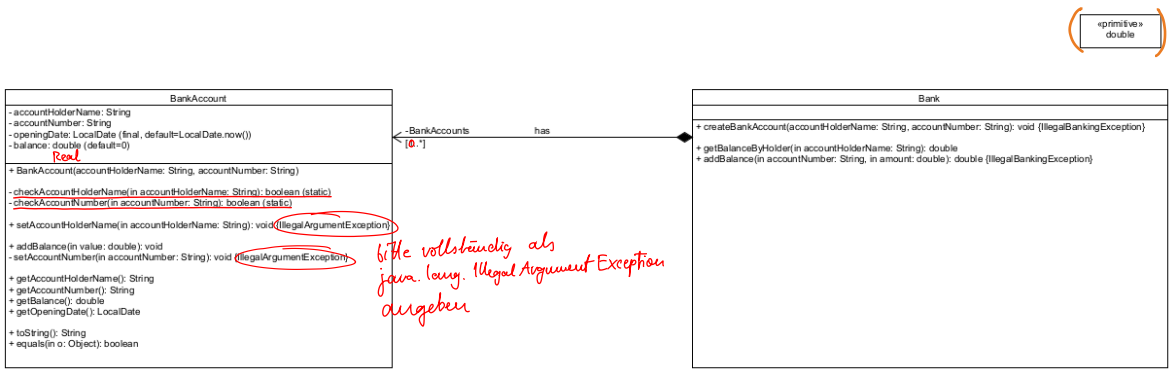
Aufgabe 37b:

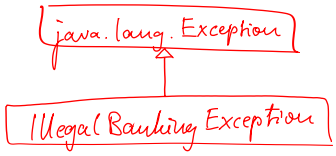


✓

2P

Aufgabe 38 + 39





```java
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
    }
}
```

```java
public class Match {

    // ----------------------------------------

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

    // ----------------------------------------

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

    // ----------------------------------------
}
```

Konstruktor

```java
import java.util.Objects;

class Player {

    // ----------------------------------------

    // Attribute
    private String name;
    private Team team;

    // ----------------------------------------

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

```
        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!

$\sum$ 3,5 P schön!

JH