Lehrstuhl Computergrafik
Prof. Dr.-Ing. Hendrik P.A. Lensch
Patrick Wieschollek, Sebastian Herholz, Fabian Groh

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Informatik II

Übungsblatt 6

Ausgabe: Di, 31. Mai 2016 - **Abgabe: Di, 07. Juni 2016 - 12:00 Uhr**

Aufgaben

6.1 RC-Racing Cars (1 + 1 + 1 + 3 + 2 + 4 + 2 + 2 + 2 + 4 = 22 Punkte)

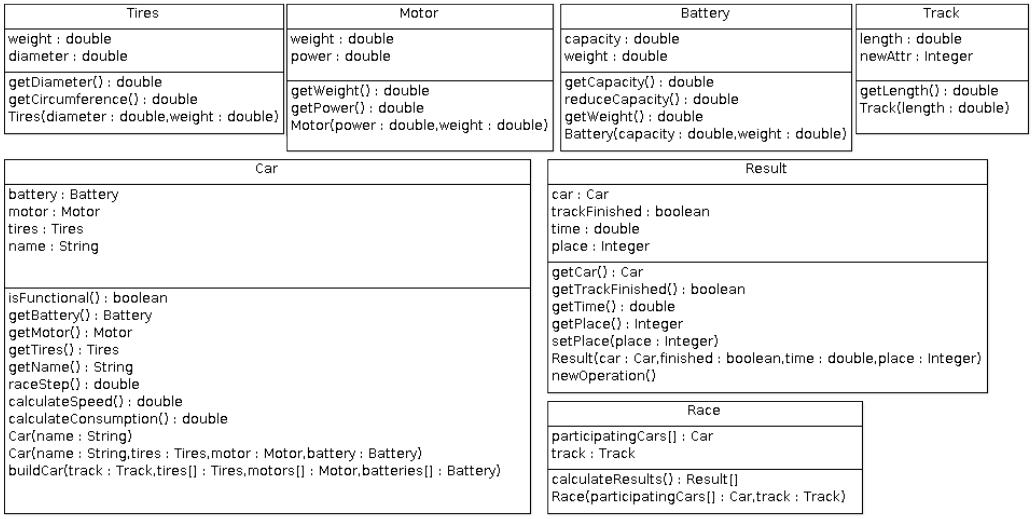

Figure 1: Klassendiagram der einzelen Klassen.

In dieser Aufgabe sollen Sie weitere Erfahrungen mit der objektorientierten Programmierung sammeln. Es geht um das Bauen von ferngesteuerten Autos, um an einem Rennen teilzunehmen.

Ein Rennen besteht aus einer Liste der teilnehmenden Autos und einer Strecke mit der Länge `length`.
Ein Auto besteht aus verschiedenen Teilen: Reifen, Motor und Batterie. Alle Teile haben unterschiedliche Eigenschaften wie z.B. Durchmesser, Gewicht, Leistung und Verbrauch . All diese Eigenschaften beeinflussen die Reichweite des RC-Cars und die Zeit, die es benötigt um die Strecke mit Länge `length` zu fahren.

Achtung: Beachten Sie, dass alle Felder als private deklariert sein müssen.
Achtung: Beachten Sie, dass alle Klassen unter dem Paket `race` zugeteilt sein müssen.

a) Schreiben Sie nun eine Klasse `Track` mit dem Feld double `length`. Denken Sie auch an einen geeigneten Konstruktor und die Methode `getLength()`.

b) Implementieren Sie die Klasse `Tires`, welche die Felder double `diameter`, sowie double `weight` beinhalten soll.

Schreiben Sie auch einen geeigneten Konstruktor, welcher den beiden Feldern sofort Werte zuweist. Implementieren Sie außerdem eine Methode `getDiameter`, welche den Wert des Feldes `diameter` zurückgibt, und eine Methode `getWeight`, welche den Wert des Feldes `weight` zurückgibt. Zusätzlich soll eine Funktion `getCircumference` erstellt werden, die den Umfang eines Rades zurück gibt.

c) Implementieren Sie die Klasse `Motor`, welche die Felder double `power, sowie` double `weight` beinhalten soll.
Schreiben Sie auch einen geeigneten Konstruktor, welcher den beiden Feldern Werte zuweist. Implementieren Sie außerdem eine Methode `getPower`, welche den Wert des Feldes `power` zurückgibt, und eine Methode `getWeight`, welche den Wert des Feldes `weight` zurückgibt.

d) Implementieren Sie die Klasse `Battery`, welche als Felder einen double `capacity` und einen double `weight` haben soll.
Schreiben Sie auch einen geeigneten Konstruktor, welcher den beiden Feldern Werte zuweist. Implementieren Sie außerdem eine Methode `getCapacity`, welche den Wert des Feldes `capacity` zurückgibt, und eine Methode `getWeight`, welche den Wert des Feldes `weight` zurückgibt. Schreiben Sie auch die Methode `reduceCapatiy`, welche einen double Wert als Argument bekommt, dieser Wert soll von der Kapazität der Batterie abgezogen werden. Wenn die Batterie nicht genügend Kapazität mehr hat, soll die Funktion `false` züruck geben, andernfalls `true`.

e) Implementieren Sie die Klasse `Car`, welche als Felder `name` vom Typ String, `tires` vom Typ Tires, einen `motor` vom Typ Motor und eine `battery` vom Typ Battery haben soll.
Schreiben Sie auch einen geeigneten Konstruktor, welcher den Feldern Werte zuweist. Zudem soll auch ein Konstruktor implementiert werden, welcher nur den Namen setzt. Implementieren Sie außerdem alle get-Methoden für die Attribute des Autos.
Die Funktion `isFunctional` soll prüfen, ob das Auto fahrbereit ist. Ein Auto ist fahrbereit, wenn es einen Motor, Reifen und eine Batterie besitzt. Zusätzlich muss die Kapazität der Batterie ≥ 0 sein.

f) Berechnen Sie, welche Geschwindigkeit ein Auto mit gegebenen Teilen erreicht und welchen Energieverbrauch es hat.
Schreiben Sie hierfür eine Methode `calculateSpeed` in der Car-Klasse, welche berechnen soll, wie viel Strecke pro Sekunde im Rennen zurückgelegt wird. Hierfür soll die Methode das Ergebnis folgender Formel zurück geben:

$$\frac{\text{Reifenumfang} \cdot \text{Power des Motors}}{\text{Gesamtgewicht des Autos}} \cdot 1000 = \text{Strecke pro raceStep}$$

Implementieren Sie außerdem eine Methode `calculateConsumption` in der Car-Klasse, welche berechnen soll, wieviel Energie das Auto pro Sekunde im Rennen verbraucht. Hierfür soll die Methode das Ergebnis folgender Formel zurück geben:

$$\frac{\text{Power des Motors} \cdot \text{Gesamtgewicht des Autos}}{100} = \text{verbrauchte Enegie pro raceStep}$$

Diese Methoden sollen public sein.

g) Implementieren Sie die Function `raceStep`, welche das Auto eine Sekunde fahren läßt und die dabei zurückgelegte Strecke zurückgibt. Beachten Sie, dass das Auto nur fahren kann, wenn seine Batterie noch genügeng Kapazität hat. Nach einem `raceStep` muss die Kapazität der Batterie angepasst werden. Sollte das Auto keine Energie mehr haben, soll 0 zurückgegeben werden.

h) Um ein Auto für eine gegebene Strecke zu optimieren soll die Funktion `buildCar` implementiert werden. Diese Funktion bekommt den `Track` und Listen möglicher Teile übergeben. Überlegen Sie sich eine geeignete Methode, um die Teile zuzuweisen. Für eine optimale Zuweisung sollten die Formeln für den Verbrauch und Geschwindigkeit berücksichtigt werden.

i) Implementieren Sie noch eine weitere Klasse `Result`, mit den Feldern (car: Car), (trackFinished: boolean), (time: double), (place: int). Ein Result beinhaltet eine Referenz auf das Auto, einen

`boolean`, ob das Auto es ins Ziel geschafft hat, die Zeit die es gebraucht hat und die Platzierung. Vergessen Sie auch hier den Konstruktor und die `get`-Methoden nicht.

j) Nun fehlt noch eine Klasse `Race` mit den Feldern `Car[] participatingCars` und `Track track`. Denken Sie auch an einen geeigneten Konstruktor und `get`-Methoden.
Erstellen Sie in dieser Klasse auch folgende Methode:

- `calculateResults()`, welche ein Result-Array zurückgibt. Diese Methode soll für jedes Auto die Methode `raceStep` aufrufen, bis alle Autos im Ziel sind oder nicht mehr fahren können. Prüfen sie mit Hilfe der `isFunctional` Funktion, ob ein Auto noch fahrbereit ist. Speichern Sie für jedes Auto ein Result mit Car-Referenz, ob das Auto ins Ziel kam, welche Zeit es für die Strecke gebraucht hat (soll 0 sein, falls das Auto nicht ins Ziel kam) und die Platzierung in das Array.
  Sortieren sie den `Result[]` Array anhand der Platzierung.

k) Um das Rennen jetzt zu starten, rufen sie die `main` Methode der Klasse `Main` auf. Sie können diese Methode anpassen, um unterschiedliche Konfigurationen zu testen.

Tipp: Es ist hilfreich sich vor dem Implementieren ein Klassendiagramm mit den Feldern und Methoden inklusive Signaturen zu erstellen.
Tipp: Um Ihr Programm zu testen, können Sie in der main-Methode die verschiedenen Methoden sinnvoll aufrufen und sich die Ergebnisse ausgeben lassen.

6.2 Vererbung (3 + 2 + 3 = 8 Punkte)

Hier sehen Sie 3 Codebeispiele. Welche Ausgabe erhalten Sie jeweils und erklären Sie wie diese zustande kommt?
Bachten: Diese Aufgabe muss als seperate PDF abgegeben werden!

i) Codebeispiel 1:

```
1        public class FirstClass{
2                public int calculate(int a, int b){
3                        return a+b;
4                }
5        }
```

```
1        public class SecondClass extends FirstClass{
2                public int calculate(int a, int b){
3                        return a*b;
4                }
5        }
```

```
1        public class Main{
2                public static void main(String[] args){
3                        FirstClass thing = new SecondClass();
4                        System.out.println(thing.calculate(2,3));
5                }
6        }
```

ii) Codebeispiel 2:

```
1        public class FirstClass{
2                public int divide(int x, int y){
3                        return x/y;
4                }
5        }
```

```
public class SecondClass extends FirstClass{
        public float divide(float x, float y){
                return x/y;
        }
}
```

```
public class Main{
        public static void main(String[] args){
                SecondClass thing = new SecondClass();
                System.out.println(thing.divide(5f, 3f));
                System.out.println(thing.divide(5, 3));
        }
}
```

iii) Gehen Sie hier auf den Unterschied der beiden Ausgaben ein:
Codebeispiel 3.1:

```
public class Money {
        double currentMoney;
        public Money(double value){
                currentMoney = value;
        }
        public void reduceMoney(double value){
                this.currentMoney -= value;
        }

        public static void main(String[] args) {
                Money moneyOne = new Money(1000);
                Money moneyTwo = moneyOne;

                moneyTwo.reduceMoney(500);
                System.out.println(moneyTwo.currentMoney);
                System.out.println(moneyOne.currentMoney);
        }
}
```

Codebeispiel 3.2:

```
public class Money {
        double currentMoney;
        public Money(double value){
                currentMoney = value;
        }
        public Money(Money m){
                currentMoney = m.currentMoney;
        }

        public void reduceMoney(double value){
                this.currentMoney -= value;
        }

        public static void main(String[] args) {
                Money moneyOne = new Money(1000);
                Money moneyTwo = new Money(moneyOne);
                moneyTwo.currentMoney = moneyOne.currentMoney;

                moneyTwo.reduceMoney(500);
                System.out.println(moneyTwo.currentMoney);
                System.out.println(moneyOne.currentMoney);
        }
}
```