Eigene Klassen: Beispiel „Autofahren“

Wir wollen jetzt ein Programm schreiben, welches den Benzinverbrauch eines Autos simuliert. Unsere neue Klasse heißt demzufolge Auto.

Schritt 1 - Neue Klasse erzeugen

Erzeugen Sie eine neues leeres Projekt mit der Klasse Auto.

Doppelklicken Sie auf die Klasse Auto um den Quelltext zu bearbeiten. Bereinigen Sie dann den Quelltext so, dass nur noch der Minimalquelltext übrig ist:

```
public class Auto
{
   public Auto()
   {
      …
   }
}
```

Schritt 2 - das Auto kann fahren

Was soll unser Auto alles können? Wir müssen uns zunächst Gedanken darüber machen, welche Methoden die Auto-Objekte benötigen, damit sie "fahren" können.

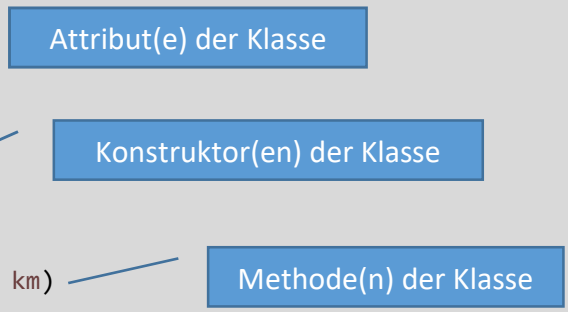
```
public void fahren(double km)
   {
      …
   }
```

Das wäre ein erster Vorschlag für eine Methode zum Fahren. Wir geben beispielsweise den Wert 12.0 als Parameter an, und unser Auto legt eine Strecke von 12.0 km zurück. Natürlich muss diese Streckenveränderung irgendwo gespeichert werden, und damit kommen wir auch schon zum ersten Attribut unserer Klasse, dem Kilometerstand des Tachos.

```
public class Auto
{
   double kmStand;

   public Auto()
   {
      kmStand = 100;
   }

   public void fahren(double km)
   {
      kmStand += km;
   }
}
```



Um nun unsere Klasse Auto auch testen zu können, brauchen wir Hauptprogramm mit einer „public void main()“-Methode, in der wir dann das Auto-Objekt erzeugen und mit dem Auto „fahren“ können (Hinweis: Diese Klasse muss in dem selben src-Ordner wie die Klasse „Auto“ liegen!):

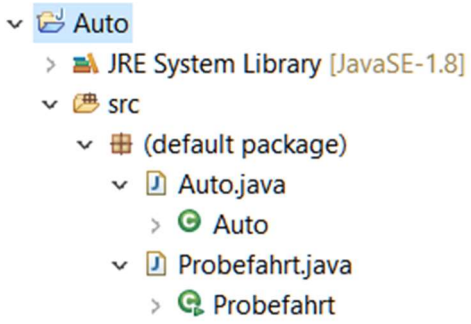
```
public class Probefahrt {

	public static void main(String[] args) {

		…

	}
}
```

Wir haben nun also ein Java-Projekt mit zwei Klassen:



Es soll nun ein Auto mit Hilfe der Klasse „Auto“ erzeugt werden. Das geschieht mit dem Schlüsselwort „new“:

```
public class Probefahrt {

	public static void main(String[] args) {

		Auto kfz1 = new Auto();

	}
}
```

Name der Klasse, von der die Instanz (das Objekt) erzeugt werden soll

selbstgewählter Name für das Objekt. Dieses sind die Namen, anhand derer man später verschiedene Autos unterscheiden kann.

Der Konstruktor, der zum Erzeugen des Objekts benutzt wird. (Wichtig, wenn es mehrere Konstruktoren gibt, s.u.)

Unter dem Objektnamen „kfz1“ steht uns nun ein Auto-Objekt zur Verfügung, welches über alle Attribute und Methoden der Klasse „Auto“ verfügt.
Da der Konstruktor der Klasse den kmStand bei Erzeugung des Objekts immer auf 100 km setzt, haben wir nun ein Auto, das kfz1 heißt und bereits 100 km auf dem Tacho hat.

Wenn nun eine Methode der Klasse benutzt werden soll, muss zunächst der Name des Objekts angegeben werden und dann -durch einen Punkt getrennt- die entsprechende Methode:

```
kfz1.fahren(100.2);
```

ACHTUNG: In der Klasse „Auto“ hatten wir definiert, dass die Methode „fahren“ immer einen Parameter verlangt. Ohne diesen Parameter würde Java keine passende Methode finden und einen Fehler melden.

Nachdem Attribute des Objekts verändert worden sind, sollen die neuen Werte auch abgerufen und auf dem Bildschirm angezeigt werden können. Dafür definieren wir in der Klasse eine weitere Methode „getKilometerstand()“;

```
public class Auto
{
   double kmStand;

   public Auto()
   {
      kmStand = 100;
   }

   public void fahren(double km)
   {
      kmStand += km;
   }

   public double getKilometerstand() {
         return kmStand;
   }
}
```

Die Methode soll den Kilometerstand des Objekts auslesen und zurückgeben. Hier wird definiert, dass der Rückgabewert eine Kommazahl sein soll

Es ist üblich, dass man Methoden, die einen Wert auslesen und zurückgeben mit „get“ benennt (sogenannte ‚Getter‘-Methoden) und Methoden, die Attribute verändern mit „set“ (sogenannte „Setter“-Methoden)

An dieser Stelle wird ein Wert zurückgeliefert, der direkt in der Zeile, von wo aus der Aufruf der Methode erfolgt war, verwertet werden kann.
Im Beispiel wird der Wert direkt in die Klammern der System-Ausgabe gesetzt.
ACHTUNG: Zwei häufige Fehler, die immer wieder passieren:
1. Die return-Anweisung wird vergessen
1. Der Rückgabewert hat nicht das Format, das im Kopf definiert worden ist (hier im Beispiel „double“)

In unserer Testklasse „Probefahrt“ kann nun diese Methode aufgerufen werden:

```
System.out.println(„aktueller Kilometerstand:“+ kfz1.getKilometerstand() );
```

Übungen

Übung 4.1-1 (6 Punkte)

Ergänzen Sie den Quelltext der Klasse Auto um ein Attribut, das die noch vorhandene Benzinmenge in Litern speichert. Der Tank des Autos soll maximal 70 Liter Benzin fassen, und das Auto soll durchschnittlich 7.3 Liter pro 100 km Fahrstrecke verbrauchen.

Verändern Sie nun die Methode fahren derart, dass beim Fahren *tatsächlich* Benzin verbraucht wird. Der Kilometerstand und der Benzinstand des Autos sollen durch eine weitere neue Methode anzeigen in der Konsole ausgegeben werden. Schreiben Sie auch diese Methode.

Übung 4.1-2 (3 Punkte)

Bei der vorherigen Übung sind wir davon ausgegangen, dass der Tank des Autos maximal 70 Liter Benzin fasst. Es gibt aber auch Autos, deren Tank nur 50 Liter fasst; andere Autos wiederum können bis zu 80 oder 90 Liter betankt werden.

Machen Sie Ihr Auto flexibler, so dass der Benutzer der Klasse das maximale Fassungsvermögen des Tanks beim Erzeugen eines Auto-Objektes festlegen kann.

Übung 4.1-3 (2 Punkte)

Auch der Verbrauch des Autos soll nun beim Erzeugen der Auto-Objekte flexibel festgelegt werden können, denn nicht jedes Auto verbraucht im Schnitt 7.3 Liter/100 km. Ergänzen Sie die Klasse entsprechend.