Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III
Prof. Dr. A.B. Cremers

Aufgabenblatt 4

— Veröffentlicht am 19.05.2010 —

Abgabe bis 25.05.2010, 13:00 per SVN

Aufgabe 1 *(Einfache Objekte - 10 Punkte)*

Nachdem sich das bisherige Kfz-Verwaltungssystem als unflexibel hinsichtlich der einer Erweiterbarkeit und Wiederverwendbarkeit gezeigt hat, soll das System nun von Grund auf neu entwickelt werden. Diesmal sollen Techniken der objektorientierten Programmierung verwendet werden.

a) Implementieren Sie eine Klasse *Car*. Diese Klasse soll die Felder *licensePlate*, *owner* und *address* umfassen. Diese Felder sollen nur beim Erstellen eines *Car*-Objektes im Konstruktor gesetzt werden. Auf die Felder soll lesender Zugriff mit *get*-Methoden erlaubt sein. Verwenden Sie diese Vorlage:

```
public class Car {
  private String licensePlate;
  ...
  public Car(String licensePlate, String owner, String address) {
    ...
  }
  public String getLicensePlate() {
    ...
  }
  ...
}
```

b) Implementiere Sie eine Klasse *TrafficOffice*, bei der bis zu 1000 *Car*-Objekte registriert werden können. Verwenden Sie folgende Schablone:

```
public class TrafficOffice {
  ...
  public void registerCar(Car car) {
    ...
  }
}
```

c) Fügen Sie der Klasse eine Methode `getCar(String licensePlate)` hinzu, die für ein gegebenes Kennzeichen das zugehörige *Car*-Objekt zurückliefert.

d) Auch die Strafen sollen als Objekte modelliert werden. Entwickeln Sie eine Klasse *Delict*, welche die Felder *fee* und *points* kapselt. Auch hier sollen die Feldwerte nur im Konstruktor gesetzt werden und mittels *get*-Methoden abrufbar sein (vgl. Aufgabenteil a).

Aufgabe 2 *(Einfache Objekte - Teil 2 - 12 Punkte)*

Auf dem letzten Aufgabenblatt haben Sie damit begonnen, ein objektorientiertes Kfz-Verwaltungssystem zu implementieren. In Aufgabe 4 habe Sie die Klassen *Car*, *TrafficOffice* und *Delict* entwickelt. Nun sollen Interaktionen zwischen den Klassen modelliert werden.

a) Erweitern Sie die Klasse *Car* so, dass mit Hilfe einer Methode `registerDelict(Delict delict)` bis zu 500 Verstöße gespeichert werden können. Mit Hilfe von `getDelicts()` soll eine Liste dieser Verstöße abrufbar sein.

b) Entwickeln Sie nun die Klasse *TrafficWarden*, welche mit genau einem *TrafficOffice* interagieren soll. Wie in den Aufgaben 4 des Aufgabenblatts 2 soll auch die Klasse *TrafficWarden* die Funktionen `checkCar(String licensePlate)`, `fine()` und `printTicket()` unterstützten. Gehen Sie bei der Implementierung von dieser Vorlage aus:[1]

```
public class TrafficWarden {
  private TrafficOffice office;
  private Car carToCheck;

  public TrafficWarden(TrafficOffice office) {
    this.office = office;
  }
  ...
}
```

c) Betrachten Sie das Programm aus Aufgabe 4c des Aufgabenblatts 2. Schreiben Sie ein Programm, welches die gleiche Funktionalität hat, aber nur die neu entwickelten Klassen *TrafficOffice*, *TrafficWarden* und *Car* verwendet.

d) Schreiben Sie nun ein Programm, bei dem zwei *Cars* an zwei unterschiedlichen *TrafficOffices* registriert werden. Ein *TrafficWarden* (der nur mit dem ersten *TrafficOffice* kommuniziert) soll das erste Auto überprüfen, eine Strafe registrieren und einen Strafzettel ausstellen. Im Anschluss daran soll ein zweiter *TrafficWarden* (der nur mit dem zweiten *TrafficOffice* kommunizert) das gleiche mit dem zweiten Auto durchführen.

[1] *Hinweis:* Test ob eine Variable *o* ein gültiges Objekt enthält: `(o != null)`.