Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Einführung in die Informatik
Sommersemester 2012

Prof. Dr. Wolfram Burgard
PD Dr. Cyrill Stachniss
Dipl. Ing. Jörg Röwekämper
Dipl. Inf. Markus Kuderer

# Übungsblatt 11

Abgabe bis Freitag, 20.07.2012, 12:00 Uhr

**Hinweis:**

Aufgaben immer per E-Mail (eine E-Mail pro Blatt und Gruppe) an den zuständigen Tutor schicken (Bei Programmieraufgaben Java Quellcode und eventuell benötigte Datendateien).

## Aufgabe 11.1

Für einen Fuhrpark, bestehend aus PKWs, LKWs, Bussen und Fahrrädern, soll einen Klassenhierarchie entworfen werden. Verwenden Sie die folgenden Klassen:

```
Fahrzeug
Kraftfahrzeug
Bus
Fahrrad
PKW
LKW
```

Die unterschiedlichen Fahrzeuge besitzen sowohl gemeinsame als auch unterschiedliche Attribute:

- Jedes Fahrzeug besitze eine Seriennummer.
- Jedes Kraftfahrzeug besitze einen TÜV-Termin.
- Zu jedem Bus gehören die Angaben: Baujahr, amtl. Kennzeichen, Anzahl Sitzplätze, Anzahl Stehplätze sowie die Leistung (ganzzahlig).
- Zu jedem Fahrrad gehören die Angaben: Baujahr und Rahmengröße.
- Zu jedem PKW gehören die Angaben: Baujahr, amtl. Kennzeichen, Anzahl Sitzplätze sowie die Leistung (ganzzahlig).
- Zu jedem LKW gehören die Angaben: Baujahr, amtl. Kennzeichen, Anzahl Sitzplätze, Leistung (ganzzahlig) sowie Zuladung (ganzzahlig).

Darüber hinaus soll die `toString` Methode von jedem Objekt den Typ und zusätzlich die spezifischen Daten des Objektes ausgeben.

1. Implementieren Sie eine Klassenhierarchie. Machen Sie dabei Gebrauch von Vererbung, abstrakten Klassen und Methoden. Vermeiden Sie dabei Wiederholungen.

2. Testen Sie Ihre Implementierung anhand der Klasse `TestHierarchy`, die Sie auf der Homepage zur Übung finden.

3. Visualisieren Sie Ihre Klassenhierarchie als Graphen. Zeichnen Sie ein Rechteck für jede Klasse und einen Pfeil für jede Vererbung (jeweils von der Subklasse zur Superklasse).

**Aufgabe 11.2**

Auf der Vorlesungsseite finden Sie das Projekt GrafikBeispiel, das Grafiken und ein Beispielprogramm enthält. Das Programm öffnet ein Fenster, stellt Grafiken dar und ändert bei Tastendruck die Hintergrundfarbe. Nehmen Sie das Programm als Vorlage für die folgenden Aufgaben.

1. Erweitern Sie die `Map`-Klasse des letzten Übungsblattes um eine Methode `public void draw(Graphics g)`, welche das Spielfeld auf das übergebene `Graphics`-Objekt[1] zeichnet. Verwenden Sie dazu die Grafiken aus o.g. Projekt. Behandeln Sie dabei die Feld-Typen 2, 3, 5 und 6 zunächst als frei begehbare Felder vom Typ 1 (siehe Blatt 10).

2. Schreiben Sie eine Klasse `GameUpdate` mit den folgenden Membervariablen:

   - `double timeSinceStart` – Die Zeit in Sekunden, die seit dem Start des Spiels vergangen sind.
   - `double timeSinceUpdate` – Die Zeit in Sekunden, die seit dem letzten Update vergangen ist.
   - `ArrayList<KeyEvent> keyEvents` – Die KeyEvents, die seit dem letzten Update aufgetreten sind.
   - `ArrayList<MouseEvent> mouseEvents` – Die MouseEvents, die seit dem letzten Update aufgetreten sind.

   (a) Schreiben Sie einen Konstruktor mit folgender Signatur

   ```
   GameUpdate(double timeSinceStart,
   double timeSinceUpdate,
   ArrayList<KeyEvent> keyEvents,
   ArrayList<MouseEvent> mouseEvents),
   ```

   der die Membervariablen initialisiert.

   (b) Schreiben Sie `get`-Methoden für alle Membervariablen.

[1] `http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Graphics.html`

3. Schreiben Sie eine Klasse `GameConfig` mit den folgenden Membervariablen:

   - `int cellHeight` – Die Höhe einer einzelnen Zelle auf der Karte in Pixeln. Standardwert: `20`
   - `int cellWidth` – Die Breite einer einzelnen Zelle auf der Karte in Pixeln. Standardwert: `20`
   - `int windowHeight` – Die Höhe des Fensters in Pixeln. Standardwert: `480`
   - `int windowWidth` – Die Breite des Fensters in Pixeln. Standardwert: `640`
   - `String imageDir` – Das Verzeichnis, in dem die für die Spieldarstellung verwendeten Grafiken liegen. Standardwert: `res/images`
   - `String mapDir` – Das Verzeichnis, in dem die Leveldateien liegen. Standardwert: `res/maps`
   - `String title` – Der Titel des Spiels. Standardwert: `Game`

   (a) Schreiben Sie einen Default-Konstruktor `GameConfig()`, der alle Membervariablen mit den o.g. Standardwerten initialisiert.

   (b) Schreiben Sie eine Methode
   `public static GameConfig getInstance(String filename)`,
   die die gegebene Datei einliest und alle Membervariablen initialisiert. Beachten Sie:

   - Jede Zeile der Datei enthält ein Paar aus einem Bezeichner und einem Wert, separiert durch einen Doppelpunkt.

     $$\langle Bezeichner \rangle : \langle Wert \rangle$$

     Ein Bezeichner ist der Variablennamen ohne Berücksichtigung von Groß- und Kleinschreibung.
   - Alle Membervariablen sollen unter Angabe ihres Bezeichners initialisiert werden können.
   - Alle Membervariablen, die nicht durch die Datei initialisiert werden, sollen mit den Standard-Werten initialisiert werden.
   - Die Reihenfolge in der Datei soll beliebig sein.
   - Bei Mehrfachnennungen soll jeweils der alte Wert überschrieben werden.
   - Bezeichner zu denen keine Membervariable existieren werden ignoriert.
   - Benutzen Sie die Methode `String.trim()`, um Bezeichner und Werte von Leerzeichen am Anfang/Ende zu befreien.

   Eine Beispieldatei finden Sie auf der Vorlesungsseite.
   Hinweis: Verwenden Sie den Default Konstruktor.

   (c) Schreiben Sie eine `get`-Methode für jede Membervariable.