Problemorientiertes Programmieren RoboCode

Klassen in Java

Eva Eibenberger und Christian Rieß

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Datentypen

- Wir kennen bisher
  - 8 **Primitive Datentypen**: `byte, short, int, long, float, double, char, boolean`

```
int zaehler1 = 1;
double kommaZahl = 1.5;
char zeichen = 'a';
```

aber auch andere

  - z.B. die Klasse String

```
String s1 = "Hallo";
String s2 = "Welt";
String s3 = s1.concat(s2);
```

Wie definiert eigene Datentypen?

… über **Klassen**

- Unser Beispiel: Eine Klasse zur Beschreibung einer Kaffeemaschine

  - Welche Eigenschaften (**Attribute**) sollte eine Kaffeemaschine haben?

  - Welche Funktionalitäten (**Methoden**) sollte eine Kaffeemaschine haben?

Klasse Kaffeemaschine

- Bezeichnung
- Standort
- Tankvolumen
- Wasserstand
- Betriebszustand (an/aus)

- Kaffee kochen
- Anschalten/ausschalten
- Wasser nachfüllen

Instanzen einer Klasse

Klasse Kaffemaschine

Attribute:
- Bezeichnung
- Tankvolumen
- Wasserstand
- Betriebszustand (an/aus)

Methoden:
- Kaffee kochen
- Anschalten/ausschalten
- Wasser nachfüllen

- Klassen sind „Konstruktionspläne“
- Ein **Objekt** wird auch **Instanz** einer Klasse genannt

Bezeichnung:	Saeco Xelsis
Tankvolumen:	30 Tassen
Wasserstand:	20
Betriebszustand:	aus

Bezeichnung:	Bosch Solitaire
Tankvolumen:	12 Tassen
Wasserstand:	0
Betriebszustand:	aus

Bezeichnung:	Jura F90
Tankvolumen:	20 Tassen
Wasserstand:	20
Betriebszustand:	ein

Beispiel: Erstellen einer Klasse

```
public class Kaffeemaschine {

    public String bezeichnung;     // Attribut
    private int tankvolumen;       // Attribut
    private int wasservorrat;      // Attribut
    private int zustand            // Attribut (0 aus, 1 an)

    // Methoden
    public void initialisieren(int tv, int wv, int z) {
        tankvolumen = tv;
        wasservorrat = wv;
        zustand = z;
    }
    public int kochen(int tassen) {
        if (zustand == 0) {
            tassen = 0;
        }
        if (tassen > wasservorrat) {
            tassen = wasservorrat;
        }
        wasservorrat -= tassen;
        return tassen;
    }
}
```

Beispiel: Erstellen einer Klasse

```
public class Kaffeemaschine {

    public String bezeichnung;      // Att
    private int tankvolumen;        // Att
    private int wasservorrat;       // Att
    private int zustand             // Att

    // Methoden
    public void initialisieren(int tv, i
        tankvolumen = tv;
        wasservorrat = wv;
        zustand = z;
    }
    public int kochen(int tassen) {
        if (zustand == 0) {
            tassen = 0;
        }
        if (tassen > wasservorrat) {
            tassen = wasservorrat;
        }
        wasservorrat -= tassen;
        return tassen;
    }
}
```

Klassen

- Name beginnt mit einem Großbuchstaben
- Stehen in einer eigenen Datei mit gleichem Namen: `Kaffemaschine.java`
- In jeder Datei i.d.R. nur eine Klasse

Attribute

- durch Variablen in der Klasse realisiert
- können Modifikatoren haben: z.B. `public` oder `private`
- `private`: Variable ist nur innerhalb der eigenen Klasse sichtbar, d.h. kann nur dort verwendet werden
- `public`: in allen Klassen sichtbar

Auch Methoden können Modifikatoren haben: z.B. `public` oder `private`

Beispiel: Erzeugen von Objekten

```
public class Beispielprogramm {

    public static void main (String[] args) {

        // Erzeugen einer Instanz
        Kaffeemaschine kaffeeM = new Kaffeemaschine();

        // Zugriff auf Variablen der Instanz
        kaffeeM.bezeichnung = „Jura F90";

        // Zugriff auf Methoden der Instanz
        kaffeeM.initialisieren(30, 30, 1);
        int anzahlTassen = 5;
        kaffeeM.kochen(anzahlTassen);

    }
}
```

Erzeugen von Objekten:
- „Instanziieren“, „Erzeugen einer Instanz“
- Mit dem Schlüsselwort **new**

Zugriff auf Attribute/Methoden einer Instanz:
- Mittels Objektnamen und Instanzvariablen/-methode

Konstruktoren einer Klasse

```
...     // Erzeugen einer Instanz
        Kaffeemaschine kaffeeM = new Kaffeemaschine();
        kaffeeM.bezeichnung = "Jura F90";
        kaffeeM.initialisieren(30, 30, 1);
...
```

- Oft müssen die Attribute/Variablen einer Klasse direkt nach dem Instanziieren mit einem Anfangswert belegt werden (Initialiiseren)

→ Umsetzung mittels **Konstruktor**(en)

  - Eine spezielle Methode
  - Konstruktoren tragen den Namen ihrer Klasse
  - Konstrukturen werden beim Instanziieren eins Objekts aufgerufen

Konstruktoren einer Klasse

```
public class Kaffeemaschine {

    public String bezeichnung;      // Attribut
    private int tankvolumen;        // Attribut
    private int wasservorrat;       // Attribut
    private int zustand             // Attribut (0 aus, 1 an)

    // Konstruktoren
    public Kaffeemaschine() {
        bezeichnung = "noname";
        tankvolumen = 15;
        wasservorrat = 0;
        zustand = 0;
    }
    public Kaffeemaschine(String bz, int tv, int wv, int z) {
        bezeichung = bz;
        tankvolumen = tv;
        wasservorrat = wv;
        zustand = z;
    }
    // andere Methoden
    // ...
}
```

Eine Klasse kann mehrere Konstuktoren haben:

- gleicher Name, aber unterschiedliche Übergabeparameter

Erzeugen von Objekten

- Vorher:

```
... // Erzeugen einer Instanz
    Kaffeemaschine kaffeeM = new Kaffeemaschine();

    // Initialisieren der Attribute
    kaffeeM.bezeichnung = "Jura F90";
    kaffeeM.initialisieren(30, 30, 1);
...
```

- Mit selbstgeschriebenem Konstruktor

```
... // Erzeugen einer Instanz und Initialisieren der Attribute
    Kaffeemaschine kaffeeM = new Kaffeemaschine();
...
```

```
... // Erzeugen einer Instanz und Initialisieren der Attribute
    Kaffeemaschine kaffeeM = new Kaffeemaschine("Jura F90",30,30,1);
...
```

Übersetzen und Ausführen des Beispiels

- Übersetzen des Programms
  - Alle nötigen .java-Dateien müssen übersetzt werden

```
faui00a [~/test]> javac Kaffeemaschine.java Beispielprogramm.java
```

```
faui00a [~/test]> javac *.java
```

  - Der Compiler übersetzt alle noch nicht übersetzten .java-Dateien, die von zu übersetzenden .java-Dateien direkt referenziert werden

```
faui00a [~/test]> javac Beispielprogramm.java
```

- Ausführen des Programms
  - Interpreter startet die Programmausführung mit der Methode `main` der ihm übergebenen Klasse

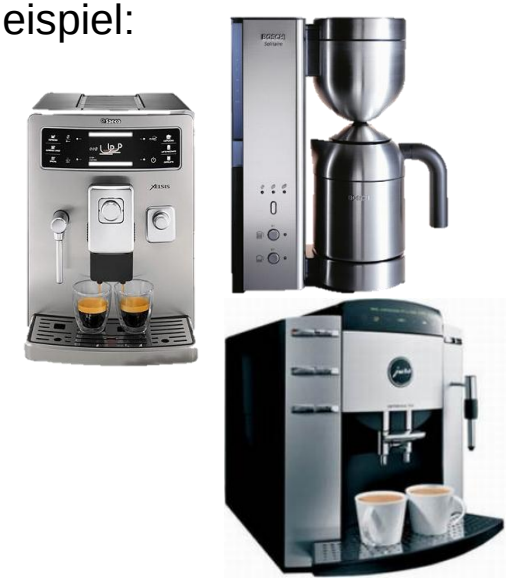
```
faui00a [~/test]> java Beispielprogramm
Vollautomat Saeco – 5 Tassen
```

```
faui00a [~/test]> java Kaffeemaschine
Exception in thread "main" java.lang.NoSuchMethodError: main
```

Innere Klassen

- Idee:
  - Innere Klassen werden in einer umgebenden Klasse definiert
  - Einsatz als Hilfsklasse, welche von „außen“ nicht unbedingt angesprochen werden muss
- Nützliches Konzept für RoboCode

- Beispiel:



- Jede Kaffeemaschine hat einen Heizkörper
- Nur wenn das der Heizkörper aufgeheizt ist, kann Wasser erhitzt werden
- Der Benutzer hat kein Interesse den Heizkörper direkt zu bedienen

→ Klasse `Heizkoerper` ist eine innere Klasse der Klasse `Kaffeemaschine`

Innere Klassen – Beispiel

Instanziieren der inneren Klasse z.B. im Konstruktor oder in einer anderen Methode der äußeren Klasse

Innere Klasse

```
public class Heizkoerper{

    private float temperatur;

    public int erhitzeWasser(int anzTassen) {
        if (temperatur < 100) {
            aufheizen();
        }
        ...
    }
    private void aufheizen() {
        ...
    }
}
```

Innere Klassen – Beispiel

Instanziieren der inneren Klasse z.B. im Konstruktor oder in einer anderen Methode der äußeren Klasse

Innere Klasse

```
public class Kaffeemaschine {

    // Attribute
    private Heizkoerper heizK;
    ...

    // Konstruktor
    public Kaffeemaschine() {
        heizK = new Heizkoerper();
        ...
    }

    // Methoden
    public int kochen(int tassen) {
        if (tassen > wasservorrat) {
            tassen = wasservorrat;
        }
        heizK.erhitzeWasser(tassen);
        ...
    }

    public class Heizkoerper{

        private float temperatur;

        public int erhitzeWasser(int anzTassen) {
            if (temperatur < 100) {
                aufheizen();
            }
            ...
        }
        private void aufheizen() {
            ...
        }
    }
}
```