Einführung in das Programmieren (3)

8. Objektorientierte Programmierung

8.1. Objekte

„Es ist leichter Dinge aufzuzählen, die Objekte sind, als solche, die keine Objekte sind.“ Descartes (Philosoph aus dem 17. Jahrhundert).

Das **Konzept der objektorientierten Programmierung** ist es, Software in einer Weise zu organisieren, die dem Denkstil unseres objektorientierten Gehirns entspricht. Das menschliche Gehirn möchte über Objekte nachdenken und unsere Gedanken und unser Gedächtnis werden durch Objekte und deren Beziehungen organisiert.

Was ist nun ein Objekt? Ein (reales) Objekt besitzt eine **Identität** (es ist ein einzelnes Ganzes), besitzt einen **Zustand** (es besitzt verschiedene Eigenschaften, die sich ändern können) und es besitzt ein **Verhalten** (es kann Dinge tun und Dinge können etwas mit ihm tun).

Beispiele:
1.) Ein Schüler ist ein Objekt: Jeder Schüler besitzt eine Identität, denn er ist „einmalig“; jeder Schüler besitzt Zustände (ausgeschlafen, hungrig, gut vorbereitet, ...); jeder Schüler besitzt ein Verhalten (Schlafen gehen, lernen, essen, ...).
2.) Ein einzelnes Gummibärchen ist ein Objekt: Es besitzt eine Identität, denn es ist „einmalig“; es besitzt einen Zustand (weich, hart); es besitzt ein Verhalten (wenn auch nicht viel).

8.2. Software-Objekte

Software-Objekte gleichen real existierenden Objekten, d. h. sie haben ebenfalls eine **Identität** (Sie existieren als Teil des Speichers und sind durch einen eindeutigen Namen identifizierbar), einen **Zustand** (Sie besitzen **Attribute**, die softwaremäßig durch **Variablen** dargestellt werden; jedes Attribut besitzt einen jeweils aktuellen **Attributwert**), ein **Verhalten** (Methoden befähigen Objekte, etwas zu tun).

Zur Darstellung von Objekten verwendet man **Objektkarten**:

Allgemeiner Aufbau:

Objektname
Attributname = Attributwert

Beispiel: Schüler

Schülername
ausgeschlafen = ja hungrig = nein vorbereitung = gut

Beispiel: Gummibärchen

Gummibärchen1
farbe = rot alter = 2 essbar = true

8.3. Klassen

Eine **Klasse** ist eine Beschreibung (Plan, Schablone, Kochrezept, ...) für ein mögliches Objekt. Eine Klasse legt damit fest, welche **Attribute und Methoden** ein Objekt besitzt.

Zur Darstellung von Klassen verwendet man **Klassenkarten**:

Allgemeiner Aufbau:

KLASSENNAME
alle Attribute
alle Methoden

Beispiel: Schüler

SCHUELER
ausgeschlafen hunger vorbereitung
geheSchlafen lerne issEtwas

Beispiel: Gummibärchen

GUMMIBAERCHEN
farbe alter essbar
erhoeheAlter pruefeEssbarkeit

8.4. Klassen in Java

In Java können eigene Klassen definiert werden.
Grundlegender Aufbau einer Klasse:

```
class Klassenname {
    //Variable1
    //Varibale2
    // ...

    //Standard-Konstruktor
    Klassenname() {
    }

    //Weitere Konstruktoren

    //Methoden
}
```

In Java Programmen ist die `main()` Methode etwas Besonderes, da der Java Interpreter mit ihr das ganze Programme startet. Die `main()` Methode ist eine statische (`static`) Methode, das bedeutet, dass es nur eine Instanz von ihr gibt und, dass sie (als Teil der Klasse) existiert bevor irgendwelche Objekte erzeugt wurden.
In einer kleinen Applikation könnte `main()` alle notwendigen Anweisungen durchführen. In einer größeren Applikation wird `main()` Objekte erzeugen und deren Methoden verwenden.

Beispiel: Die Klasse Gummibärchen

```
class Gummibaerchen {
   //Variablen
   String farbe = "rot";
   int alter = 1;
   boolean essbar = true;

   //Standard-Konstruktor
   public Gummibaerchen (){
   }
   //Weiterer Konstruktor
   public Gummibaerchen (String farbe){
      this.farbe = farbe;
   }

   //Standard-Methoden get und set
   public String getFarbe (){
      return farbe;
   }
   public setFarbe (String f) {
      farbe = f;
   }
   public int getAlter (){
      return alter;
   }
   public boolean getEssbar() {
      return essbar;
   }
   public setEssbar (boolean zustand){
      essbar = zustand;
   }
   //Weitere Methoden
   public erhoeheAlter(){
      alter = alter + 1;
   }
   public void pruefeEssbarkeit(){
      if (alter > 10){
         essbar = false;
      }
   }
}
```

Eine Klasse erzeugt nicht von sich aus ein Objekt. Um ein Objekt zu erzeugen, wird der `new`-Operator zusammen mit dem Namen der Klasse verwendet. Die Erzeugung eines Objekts, wird **Instantiierung** genannt.

Beispiel: Gummibaerchen gummibaerchen1 = `new` Gummibaerchen();

Hinweis: Es wird ein Objekt der Klasse Gummibaerchen mit dem Bezeichnung „gummibaerchen1“ erzeugt. Dabei wird der Standard-Konstruktor aufgerufen, wodurch die Variablen farbe, alter und essbar auf die Standardwerte gesetzt werden, hier als „rot“, 1 und true.

Beispiel: Gummibaerchen gummibaerchen2 = `new` Gummibaerchen(„gelb“);

Hinweis: Es wird ein Objekt der Klasse Gummibaerchen mit dem Bezeichnung „gummibaerchen2“ erzeugt. Dabei wird der weitere Konstruktor aufgerufen, wodurch die Variablen farbe auf „gelb“ gesetzt wird, die Varibalen alter und essbar weiterhin die Standardwerte bekommen.

Diese Instantiierung könnte zum Beispiel in der wird `main()`- Methode erfolgen.

```
class Hauptprogramm {
  public static void main(String[] args){
     Gummibaerchen gummibaerchen1 = new Gummibaerchen();
     Gummibaerchen gummibaerchen2 = new Gummibaerchen(„gelb“);
  }
}
```

8.5. Punktnotation

Nachdem ein Objekt konstruiert wurde, kann es (gewöhnlich) durch seine eigenen Methoden geändert werden (*nicht* durch seine Konstruktoren). Manche Objekte sind so entworfen worden, dass sie, sobald sie einmal konstruiert wurden, nicht mehr geändert werden können.

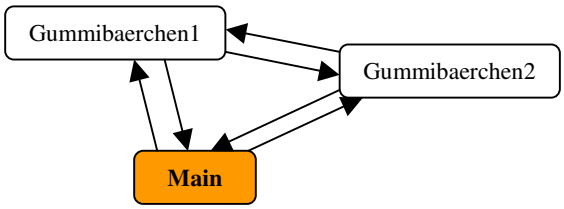
Die beiden Arten von Dingen innerhalb eines Objekts — Variablen und Methoden — werden manchmal auch als **Instanzvariablen** und **Instanzmethoden** oder als Elemente dieses Objekts bezeichnet. Auf die Elemente eines Objekts kann durch die Verwendung der *Punktnotation* zugegriffen werden.

Beispiele: String farbe1 = gummibaerchen1.getFarbe();

gummibaerchen2.erhoeheAlter();

8.6. Kommunizierende Objekte

Ein laufendes Programm ist eine Sammlung von Objekten, die alle ihre eigene Aufgabe erledigen und mit anderen Objekten kommunizieren. In der objektorientierten Programmierung ist es üblich zur Laufzeit weitere Objekte zu erzeugen. Die neuen Objekte könnten zusätzliche Daten handhaben, die das Programm zum Bearbeiten braucht.



8.7. Aufgaben

1. Erstelle ein neues Projekt „inf12_programm80“ und erzeuge die Klassen „Hauptprogramm“, die die main()-Methode enthält, und „Gummibaerchen“ wie oben.
2. Erweitere die Klasse „Gummibaerchen“ um die Methoden zur Ausgabe der Variablen am Bildschirm.
3. Erzeuge mehrere Gummibärchen und wende die möglichen Methoden darauf an.
4. Erzeuge eine Liste von Gummibärchen und schreibe geeignet Methoden dazu.
5. Erstelle eine Klasse „Gummibaerchenliste“ und finde sinnvolle Anwendungsbeispiele für ihren Gebrauch.