

Einführung in Java mit BlueJ

Arbeitsblatt 5.2

Datenkapselung

mehr unter: www.u-helmich.de/inf/BlueJ/seiten/seite05.html

Datenkapselung ist ein sehr wichtiges Konzept in der objektorientierten Programmierung. Bild 1 zeigt eine graphische Darstellung einer solchen Datenkapsel. Die eigentlichen Daten befinden sich im Innern der Kapsel, von außen sind sie nicht zugänglich. Will man die Daten verändern, so muss man sich dazu bestimmter Operationen bedienen. In Bild 1 sind diese Operationen wie eine Schicht um die Daten herum angeordnet.

Vergleichen wir eine solche Datenkapsel mit einer Java-Klasse, so fallen sofort einige Gemeinsamkeiten ins Auge. Die Daten im Innern der Datenkapsel entsprechen den **Attributen** der Java-Klasse, die Operationen, die die Daten umgeben, sind mit den **Methoden** der Klasse gleichzusetzen: Operationen, die die Daten verändern, entsprechen den verändernden Methoden, und Operationen, die Informationen über die Daten liefern, entsprechen den abfragenden oder sondierenden Methoden.

Wofür ist die Datenkapselung nun gut? Wenn Sie in einem Auto fahren, so wollen Sie sich auf den Verkehr konzentrieren und nicht ständig darüber nachdenken, wie die Gangschaltung oder das Gaspedal funktionieren. Das würde sie nur ablenken und zu Fehlern verführen. Es reicht, wenn Sie wissen, wo das Gaspedal ist und wie man es bedient bzw. wie Sie die Gänge richtig einlegen. Die Einzelheiten der Mechanik sehen sie nicht, denn sie sind vor Ihnen verborgen.

Analog verfährt man, wenn man programmiert. Wenn Sie eine Linie zeichnen wollen, so reicht es völlig aus, wenn Sie wissen, dass es den drawLine-Befehl gibt und wie er aufgerufen wird. Mit welchem mathematischen Algorithmus der Befehl die Linie zeichnet, interessiert nicht.

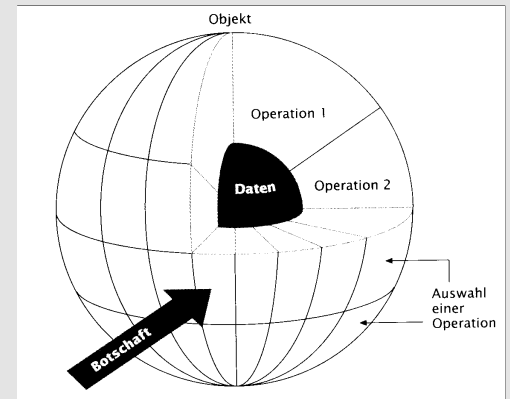
Dieses seit langem bewährte Programmierprinzip heißt Datenkapselung. Die inneren Details (also die Attribute und bestimmte Algorithmen) einer Klasse werden eingekapselt und dadurch vor der Außenwelt verborgen (**information hiding**). Die Interaktion mit der Außenwelt geschieht ausschließlich über definierte **Schnittstellen (interfaces)**: Verändernde und abfragende Methoden.

Bild 2 und 3 zeigen den Quelltext von zwei Klassen Gangschaltung und Auto. Das Attribut gs der Klasse Auto ist ein Objekt der Klasse Gangschaltung. Diese Klasse hat wiederum ein Attribut gang, welches als privat deklariert ist. Dies hat zur Folge, dass die Klasse Auto nicht korrekt übersetzt werden kann; der Compiler liefert eine Fehlermeldung: "*gang has private access in Gangschaltung*". Der System.out.println()-Befehl hat keinen Zugriff auf das Attribut gang des Objektes gs, denn das Attribut wurde als privates Attribut deklariert. Würde man dagegen schreiben

```
System.out.println(gs.getGang());
```

so würde das Programm laufen. Bei getGang() handelt es sich nämlich um eine abfragende Methode, die als public deklariert wurde. Alternativ hätte man natürlich das Attribut gang der Klasse Gangschaltung als public deklarieren können. Dann wäre dieses Attribut aber nicht mehr gekapselt. Das wäre dann so, als würden Sie beim Autofahren ständig die Mechanik der Gangschaltung vor sich sehen.

Bei der Datenkapselung verfolgt man also das Ziel, die Attribute einer Klasse vor unerlaubten Zugriffen von außen zu schützen. Ein so geschütztes Attribut kann nicht aus Versehen verändert werden, was bei einem großen Programm sonst unweigerlich früher oder später passieren würde. In der Tat ist die Datenkapselung entwickelt worden, als die Anwendungsprogramme immer größer und unübersichtlicher wurden. Die Datenkapselung stellt einen wirksamen Schutz gegen die eigenen Programmierfehler dar: Was man nicht kennt oder sieht, kann man auch nicht aus Versehen verändern.



1 Das Datenkapsel-Modell von Balzert

```
public class Gangschaltung
{
    private int gang;

    public Gangschaltung()
    {
        gang = 1;
    }

    public int getGang()
    {
        return gang;
    }
}
```

2 Die Klasse Gangschaltung

```
public class Auto
{
    int benzinstand, kmstand;
    Gangschaltung gs;

    public Auto(int benz, int km)
    {
        benzinstand = benz;
        kmstand = km;
        gs = new Gangschaltung();
    }

    public void ausgabe()
    {
        System.out.println(gs.gang);
    }
}
```

3 Die Klasse Auto