TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

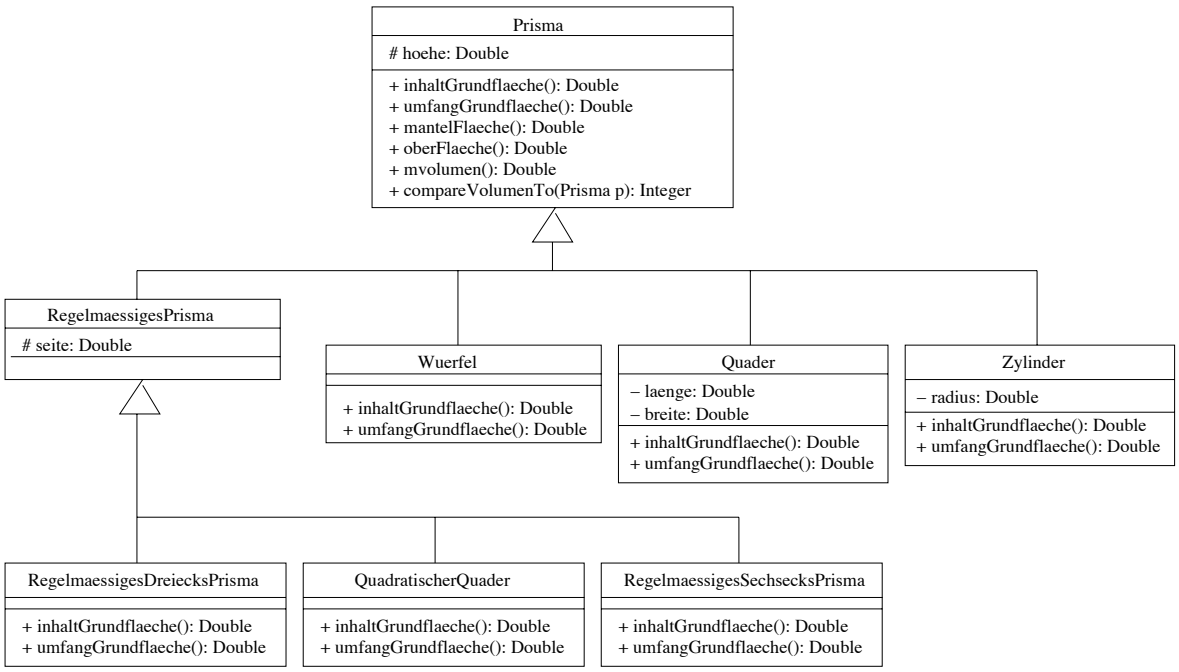
**Lehrstuhl für Sprachen und Beschreibungsstrukturen**
**Praktikum Grundlagen der Programmierung**
F. Forster, T. Gawlitza, A. Flexeder

WS 2007/2008
Lösungsvorschläge zu Blatt 8
7. Dezember 2007

# Übungen zu Praktikum Grundlagen der Programmierung

## Aufgabe 40 Vererbung (Lösungsvorschlag)

a) Eine geeignete Modellierung würde folgendermaßen aussehen:




b) Die Operationen `inhaltGrundflaeche()` und `umfangGrundflaeche()` sind von der Grundfläche des konkreten Prismas abhängig und können erst dort implementiert werden. Alle anderen Operationen können bereits in der Klasse `Prisma` implementiert werden und diese Implementierung an alle Unterklassen zur Wiederverwendung vererben. Dabei stützen sie sich teilweise auf die beiden Operationen `inhaltGrundflaeche()` und
`umfangGrundflaeche()` ab.

c) Die Oberklasse `Prisma` für beliebige gerade Prismen:

```
public class Prisma {
   // Gemeinsames Attribut aller Prismen ist die Hoehe:
   protected double hoehe;

   // Konstruktor:
   public Prisma (double h) {
      hoehe = h;
   }

   // Die Operationen umfangGrundflaeche() und inhaltGrundflaeche()
   // sind abhaengig von der Form des konkreten Prismas und werden
   // hier nur aus technischen Gruenden definiert;
   // In den verschiedenen Unterklassen muessen diese Methoden dann
   // ueberschrieben werden. Später werden sie
   // durch abstrakte Methoden ersetzt:

public double umfangGrundflaeche(){
return 0;
}
public double inhaltGrundflaeche(){
return 0;
}

  // Die anderen Operationen koennen bereits in dieser Klasse implementiert
  // werden:

  public double mantelFlaeche() {
     return umfangGrundflaeche() * hoehe;
  }

  public double oberFlaeche() {
     return mantelFlaeche() + 2 * inhaltGrundflaeche();
  }

  public double volumen() {
     return inhaltGrundflaeche() * hoehe;
  }

  public int compareVolumenTo(Prisma p) {
     double v1 = volumen();
     double v2 = p.volumen();

     if (v1 < v2)
        return -1;
     else if (v1 > v2)
        return 1;
     else
        return 0;
  }
}
```

Die Klasse `Wuerfel` als Unterklasse von `Prisma`:

```
public class Wuerfel extends Prisma {
```

```
  // Konstruktor:
  public Wuerfel(double h) {
     super(h);
  }

   // Implementation der Operationen umfangGrundflaeche() und
   // inhaltGrundflaeche():

  public double umfangGrundflaeche() {
     return 4 * hoehe;
  }

  public double inhaltGrundflaeche() {
     return hoehe * hoehe;
  }
}
```

Die Klasse `Quader` als Unterklasse von `Prisma`:

```
public class Quader extends Prisma {

  // Zusaetzlich sind die Attribute Laenge und Breite der Grundflaeche
  // noetig:
  private double laenge;
  private double breite;

  // Konstruktor:
  public Quader(double h, double l, double b) {
     super(h);
     laenge = l;
     breite = b;
  }

   // Implementation der Operationen umfangGrundflaeche() und
   // inhaltGrundflaeche():

  public double umfangGrundflaeche() {
     return 2 * (laenge + breite);
  }

  public double inhaltGrundflaeche() {
     return laenge * breite;
  }
}
```

Die Klasse `Zylinder` als Unterklasse von `Prisma`:

```
public class Zylinder extends Prisma {

  // Zusaetzlich ist das Attribute Radius Grundflaeche noetig:
  private double radius;

  // Konstruktor:
```

```
  public Zylinder (double h, double r) {
     super(h);
     radius = r;
  }

   // Implementation der Operationen umfangGrundflaeche() und
   // inhaltGrundflaeche():

  public double umfangGrundflaeche() {
     return 2 * radius * Math.PI;
  }

  public double inhaltGrundflaeche() {
     return radius * radius * Math.PI;
  }
}
```

Die Klasse `RegelmaessigesPrisma` als Unterklasse von `Prisma` und Oberklasse für alle regelmäßigen Prismen:

```
public class RegelmaessigesPrisma extends Prisma {

   // alle Grundflaechen regelmaessiger Prismen sind gleichseitige
   // Vielecke. Als zusaetzliches Attribut genuegt die Seitenlaenge:
   protected double seite;

   // Konstruktor:
   public RegelmaessigesPrisma(double h, double s) {
      super(h);
      seite = s;
   }
}
```

Die Klasse `RegelmaessigesSechsecksPrisma` als Unterklasse von `RegelmaessigesPrisma`:

```
public class RegelmaessigesSechsecksPrisma extends RegelmaessigesPrisma {

  // Konstruktor:
  public RegelmaessigesSechsecksPrisma(double h, double s) {
     super(s, h);
  }

   // Implementation der Operationen umfangGrundflaeche() und
   // inhaltGrundflaeche():

  public double umfangGrundflaeche() {
     return 6 * seite;
  }

  public double inhaltGrundflaeche() {
     return 3 * seite * seite * Math.sqrt(3) / 2;
  }
}
```

## Aufgabe 41 Überladen vs. Überschreiben (Lösungsvorschlag)

a) zur Compilierzeit:
   betrachte die Signatur der Funktion: Funktionsname und Typ der Parameter (nur die Deklaration ist entscheidend)
b) zur Laufzeit:
   Entscheidung welche Funktion (welcher Klasse) aufgerufen wird

Folgende Ausgaben werden produziert:

- bin Methode f in Klasse A
- bin Methode g in Klasse A
- bin Methode f in Klasse B
- bin Methode g in Klasse A
- bin Methode g in Klasse C; bin Methode f in Klasse A
- bin Methode g in Klasse C; bin Methode f in Klasse B
- bin Methode f in Klasse A; bin Methode f in Klasse D mit Typ A
- bin Methode f in Klasse B; bin Methode f in Klasse D mit Typ A
- bin Methode f in Klasse B; bin Methode f in Klasse D mit Typ B