Einführung in die Programmierung

by André Karge

Übung - Polymorphie Teil 2

letzte Woche

- Methoden überladen
- Generics

diese Woche

- Vererbung
- Interfaces
- Besprechung Übungsblatt 8

Polymorphie

Vererbung - Insel 5.8

Definition

- Grundprinzip der Objektorientierten Programmierung (OOP)
- Klassen *erben* Attribute und Methoden von übergeordneten Klassen
- Stichwort *extends*
- in der Regel spricht man von *Parent*-classes und *Child*-classes (siehe Vorlesung)

Vererbung - Insel 5.8

Sichtbarkeit von Attributen und Methoden

- Kennen wir schon aus früheren Übungen: *public*, *protected*, *private*
- **public** = Attribut / Methode einer Klasse mit diesem Keyword sind von allen anderen Klassen aus direkt sicht- und aufrufbar
- **protected** = Attribut / Methode einer Klasse mit diesem Keyword sind von allen anderen Klassen nicht sichtbar, außer von *child*-Klassen
- **private** = Attribut / Methode einer Klasse mit diesem Keyword sind von allen anderen Klassen nicht sichtbar, einschließlich *child*-Klassen

Vererbung - Insel 5.8

Overriding

- Methoden von Parent-classes können in Child-classes überschrieben werden
- Stichwort: *@Override*
- wird vor den Methodenheader geschrieben
- Wenn mehrere Klassen ein und die selbe Funktion durchführen können sollen, aber der Ablauf der Funktion unterschiedlich ist, sollte man in der übergeordneten Instanz angeben, dass es diese Funktion gibt und dann in den untergeordneten Instanzen konkret definieren
- Beispiel: alle 2D Körper haben eine Funktion *berechneFläche*, nur die Fläche eines Quadrates wird anders berechnet als die eines Kreises

Vererbung - Insel 5.8

Beispiel - Parent Class

```
public class Koerper {
  protected String name;
  public Koerper(String name) {
    this.name = name;
  }
  public float getArea() {
    return 0;
  }
  public void printName() {
    System.out.println(this.name);
  }
}
```

Vererbung - Insel 5.8

Beispiel - Child Classes

```
public class Quadrat extends Koerper { // abgeleitet von Koerper (heißt: Zugriff auf name,
  private float laenge;                // std. Konstr., getArea())
  public Quadrat(String name, float l) {
    super(name); // Aufruf des Konstruktors der parent-class
    this.laenge = l;
  }
  @Override // Überschreibung der vorhandenen Parent-Methode
  public float getArea() {
    return this.laenge * this.laenge;
  }
}
public class Circle extends Koerper {
  private float radius;
  public Circle(String name, float radius) {
    super(name);
    this.radius = radius;
  }
  @Override
  public float getArea() {
    return (float)(Math.PI * Math.pow(this.radius, 2));
  }
```

Vererbung - Insel 5.8

Beispiel Instanziierung

```
// Virtuelle Methoden
Quadrat q = new Quadrat("eins", 42); // neues Quadrat
Koerper k = new Quadrat("zwei", 5); // ein Quadrat ist automatisch auch ein Körper

Circle c = new Circle("kreis1", 12);
Koerper ck = new Circle("kreis2", 42);


q.printName();
System.out.println(q.getArea()); // Methode wurde in Koerper definiert
k.printName();
System.out.println(k.getArea());
c.printName();
System.out.println(c.getArea());
ck.printName();
System.out.println(ck.getArea());
```

Vererbung - Insel 5.8

Übungsaufgabe - 15min

1. Schreiben Sie ein Programm, das aus folgenden Klassen besteht: **Koerper2d, Kreis, Quadrat, Rechteck, Dreieck**
2. Nutzen Sie **Koerper2d** als Parent und den Rest als Child (möglicherweise gibt es auch eine Klasse die von einer anderen Klasse erben kann?)
3. Definieren sie die Funktionen für den Flächeninhalt und den Umfang der Koerper
4. Testen Sie ihr Programm in einer entsprechenden main-Methode

Interfaces

Interfaces - Insel 6.7

Beschreibung

- Setzt eine Bedingung für die Instanziierung von Objekten (Methoden im Interface **müssen** implementiert werden)
- Methoden können jedoch unterschiedlich realisiert sein
- Stichwort: *interface* und *implements*

```
public interface IKoerper {
  public float getArea(); // Was muss eine Klasse an Methoden Implementieren
  public void printName();
}
public class Circle implements IKoerper { // Implementation eines Interfaces
  public float getArea() { // konkrete Implementation des Interfaces
    return (float)(Math.PI * Math.pow(this.radius, 2));
  }
  public void printName() { // Ebenso wie hier
    System.out.println("Hallo! Ich bin " + this.name);
  }
}
```

Optionaler Zusatzbeleg für die Weihnachtspause

Optionaler Zusatzbeleg

1. Schreiben Sie ein Konsolen-Tetris (wikipedia) (Tastatursteuerung)
2. Nutzen Sie für ihr Spiel alle Mittel, die Sie bisher gelernt haben (Schleifen, Kontrollstrukturen, Arrays, Listen, Vererbung, Random, Enum, usw.)
3. Wenn eine Reihe voll ist, dann wird sie entfernt, die Reihen darüber rutschen nach und der Spieler bekommt Punkte (Es können maximal 4 Reihen gleichzeitig verschwinden - Punkte: 1=40, 2=100, 3=300, 4=1200)
4. Lassen Sie die Breite und Höhe des Spielfeldes einstellbar
5. Lassen Sie den Spieler mehrfach spielen und behalten Sie Überblick über die Anzahl der gewonnenen / verlorenen Spiele

Fröhliche Weihnachten!