Interfaces

Stephanie Böhning

Programmieren 2 - Media Systems

Hochschule für Angewandte Wissenschaften Hamburg

Warum Interfaces

- Interfaces sind eine Möglichkeit der „Mehrfachvererbung“
- Definiert Schnittstellen, um innerhalb eines Programms feste Schnittstellen zu definieren. Damit können mehrere Programmierer gut an einem Projekt arbeiten und man kann Interfaces dazu nutzen Programme modular aufzubauen.

Beispiel

Wir wollen eine App programmieren, in der alle Mathe 3 Inhalte enthalten sind:

- Komplexe Zahlen
- Integrale
- Gaußsches Trapez
- Tangentenberechnung…

Dabei soll sich jede Gruppe um eine Berechnung kümmern.

Beispiel

- Alle Themen benötigen eine Berechnung
- Wir haben eine variable Anzahl von Eingaben und immer ein Ergebnis.
- Die Datentypen des Ergebnisses und die Eingaben sind je nach Thema verschieden

Beispiel

Komplexe Zahlen

[+][3 + 4i][7 - 8i] = [10 - 4i]

Integrale

$[x^2 + 6x + 7] = [1/3*x^3 + 3*x^2 + 7x]$

Gaußsches Trapez

[x1, y1][x2, y2][x3, y3][x4, y4] = [A] (Flächeninhalt)

Tangentenberechnung…

[Formel][x] = [Steigung]

Beispiel

```
public interface Calculable {
     Object calc(List<Object> list);
}
```

Das Interface könnte dafür so aussehen.

Alle Gruppen könnten unabhängig von einander arbeiten und jeder weiß wie die Schnittstelle aussieht.

Beispiel

Für komplexe Zahlen könnte die Klasse so aussehen:

```
public class ComplexNumbers implements Calculable {

	@Override
	Object calc(List<Object> list) {
		//hier werden die einzelnen Argumente
		der Liste ausgewertet und eine neue
		komplexe Zahl berechnet und zurückgegeben
	}
}
```

Abstrakte Klassen und Interfaces

- Erbt eine Klasse von einer anderen z.B. Apfelbaum erbt von Pflanze, dann ist der Apfelbaum eine Pflanze.

- Interfaces werden oft auch so eingesetzt, dass sie eine Eigenschaft beschreibt für die sie Methoden deklariert, z.B. das Interface Essbar mit der Methode ***String getEssbareFrucht()***

- So bekommt die Klasse Apfelbaum Methoden von Pflanze wie ***int getGröße()*** aber auch ***String getEssbareFrucht()*** vom Interface Essbar

Abstrakte Klassen und Interfaces

Laubbaum:

- Essbare Frucht, verliert Blätter (Apfelbaum)
- verliert Blätter (Bittermandel)
- Ist Essbar (Blätter), verliert Blätter (Birke)
- Verliert keine Blätter (spanische Eiche)
- Essbare Frucht, verliert keine Blätter (Olivenbaum)

Tannen:

- Essbare Frucht, verliert keine Blätter (Fichte)
- Verliert Blätter (Lärche)
- Verliert keine Blätter (Kiefer)

Kräuter:

- Ist essbar (Petersilie)

Abstrakte Klassen und Interfaces

Bei dem Beispiel versagt die Vererbung, es gibt zu viele Ausnahmen.

Möchte man wissen welche Bäume essbar sind, essbare Früchte haben oder Blätter verlieren, helfen nur Interfaces weiter.

```
//genauso wie man Listen von Klassen haben kann, kann man Listen von Interfaces haben
List<Essbar> essbares;
List<EssbareFrucht> essbareFruechte;
List<SommerGruen> sommerGruenes;
```

Beispiel eines Interfaces

```
public interface Movable {
      void moveItem();
}
```

Implementiert eine Klasse dieses Interface, so muss sie die Methode moveItem () überschreiben.

Beispiel der Implementierung eines Interfaces

```
public class Ball implements Movable{
    private int xVelocity
    private int yVelocity;
    private int x;
    private int y;

    @Override
    public void moveItem() {
        x += xVelocity;
        y += yVelocity;
    }
}
```

Gibt es Fragen ???