Funktionale und objektorientierte Programmierkonzepte

TECHNISCHE UNIVERSITÄT DARMSTADT

Präsenz-Sprechstunde B

Simon Hock, Nhan Huynh, Daniel Mangold

Überblick

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisatorisches

Interfaces

Abstrakte Klassen

Nützliche Informationen

Klassenattribute vs. Objektattribute

Typumwandlung

Arbeitsphase

Organisatorisches

TECHNISCHE UNIVERSITÄT DARMSTADT

- 01.12.2021 Präsenzsprechstunde im Raum in Raum S103/**103**!
- Aufnahme von Themenwünsche: Bitte spätestens **Montag** Abend Bescheid geben!

Interfaces

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schnittstelle: Trennt, was eine Klasse tut, und wie die Klasse es tut
- Keine Objekte können instanziiert werden
- Zustandslos: Enthält nur Konstanten, Klassenmethoden und nicht implementierte Objektmethoden
  - Konstante: `static final`
- Schlüsselwort: `interface`
- Alles im Interface ist implizit `public`
- Zusätzlich sind alle Methoden implizit `abstract`
- Ausnahme:
  - Seit Java 8: Implementierte Objektmethoden mit dem Schlüsselwort `default`
  - Seit Java 9: Implementierte Methoden mit dem Schlüsselwort `private`

Vertrag

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beschreibung der Funktionalitäten
- Genauere Implementierung bleibt offen
- Entwickler: Freiheit

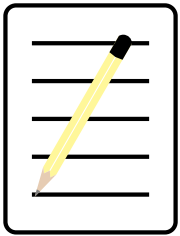

Abbildung: Vertrag

Analogie - Schablone

TECHNISCHE UNIVERSITÄT DARMSTADT

Abbildung:
https://images-na.ssl-images-amazon.com/images/I/71Fey%2BGutvL._AC_SX466_.jpg

Klasse Ingredient

```
1  public class Ingredient {
2
3    private final String name;
4    private final double amount;
5    private final String unit;
6
7    public Ingredient(String name, double amount, String unit) {
8      this.name   = name;
9      this.amount = amount;
10     this.unit   = unit;
11   }
12
13   ...
14 }
```

Klasse Recipe

```
1  public class Recipe {
2
3    private final Ingredient[] ingredients;
4
5    public Recipe(Ingredient[] ingredients) {
6      this.ingredients = ingredients;
7    }
8
9    ...
10 }
```

Interface Cookie

```
1 public interface Cookie {
2
3   Recipe getRecipe();
4
5 }
```

Klasse ShortbreadBiscuit

```java
public class ShortbreadBiscuit implements Cookie {

  @Override
  public Recipe getRecipe() {
    Ingredient[] ingredients = {
      new Ingredient("Unsalted butter", 10, "tbsp"),
      new Ingredient("Confectioners' sugar", 0.5, "cup"),
      new Ingredient("Pure vanilla extract", 0.5, "tbsp"),
      new Ingredient("All-purpose flour 180g", 1.5, "cups"),
      new Ingredient("KKosher salt", 0.5, "tbsp")
    };
    return new Recipe(ingredients);
  }
}
```

Abstrakte Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vorlagen mit Zuständen und Funktionalitäten für Unterklassen
- Keine Objekte können instanziiert werden
- Kann Attribute enthalten
- Kann implementierte und nicht implementierte Objektmethoden enthalten
  - Abstrakte Methode mittels Schlüsselwort `abstract`
- Beliebige Sichtbarkeiten
- Schlüsselwort: `abstract`

Beispiel - Sortierung

TECHNISCHE UNIVERSITÄT DARMSTADT

Interface Sortable

```
1 public interface Sortable {
2
3
4   void sort();
5
6   void sort(int from, int to);
7 }
```

Abstrakte Klasse ArraySort

```
1  public abstract class ArraySort implements Sortable {
2
3    protected final int[] array;
4
5    public ArraySort(int[] array) {
6      this.array = array;
7    }
8
9    @Override
10   public void sort() {
11     sort(0, array.length - 1);
12   }
13
14 }
```

Klasse ArraySelectionSort

```java
public class ArraySelectionSort extends ArraySort {
  public ArraySelectionSort(int[] array) { super(array); }

  @Override
  public void sort(int from, int to) {
    for (int i = from; i < to; i++) {
      int min = i;
      for (int j = i + 1; j <= to; j++)
        if (array[j] < array[min])
          min = j;
      swap(i, min);
    }
  }
}
```

Nützliche Informationen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein Interface kann beliebig viele Interfaces erweitern.
- Eine Klasse kann nur eine Klasse erweitern (Einfachvererbung) und beliebig viele Interfaces implementieren.
- Eine Klasse gilt als abstrakt, falls es eine nicht implementierte Methode besitzt.
- Nicht implementierte Methoden bei Erweiterungen (Interface und abstrakte Klassen) müssen nicht erneut definiert werden.

Beispiel - Erweiterung von Interfaces

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1 public interface Identifiable {
2
3   String getID();
4 }
```

```java
public interface Sortable extends Identifiable {


  void sort();

  void sort(int from, int to);
}
```

Klassenattribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- Statische Attribute sind Eigenschaften, die nicht einzelnen Objekten, sondern deren gesamter Klasse zugeordnet werden.
- Schlüsselwort: `static`
- Formaler Aufbau: Zugriffsmodifikator^{+} `static` Datentyp Bezeichner = Wert;
  - Die Begriffe, die mit einem $^{+}$ (Asterisk) markiert sind, sind optional.
- Zugriff: Klassenname.Klassenattribut

Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class Person {
  private String name;

  public Person(String name) {
    Person.name = name;
  }

  public String getName() {
    return name;
  }
}
```

- Welchen Namen gibt `joe` zurück?
- Welchen Namen gibt `sarah` zurück?

```
1 Person joe   = new Person("Joe");
2 Person sarah = new Person("Sarah");
3
4 System.out.println(joe.getName());
5 System.out.println(sarah.getName());
```

Objektattribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jede Instanz hat ihre eigenen Objektattribute mit eigenen Werten.
- Ohne Schlüsselwort `static`
- Formaler Aufbau: Zugriffsmodifikator$^+$ Datentyp Bezeichner = Wert;
  - Die Begriffe, die mit einem $^+$ (Asterisk) markiert sind, sind optional
- Zugriff: Objekt.Objektattribut

Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class Person {
  private String name;

  public Person(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }
}
```

- Welchen Namen gibt `joe` zurück?
- Welchen Namen gibt `sarah` zurück?

```
1 Person joe   = new Person("Joe");
2 Person sarah = new Person("Sarah");
3
4 System.out.println(joe.getName());
5 System.out.println(sarah.getName());
```

Typumwandlung

TECHNISCHE UNIVERSITÄT DARMSTADT

- engl. Casting
- Primitive Datentypen
- Referenztypen
- Impliziter Cast: Typ muss nicht angegeben werden, sondern die Typumwandlung geschieht automatisch
- Expliziter Cast: Typ muss angegeben werden
- Formaler Aufbau: (<Casting Typ>) Literal/Variable/Objekt

Primitive Datentypen - Widening conversion

TECHNISCHE UNIVERSITÄT DARMSTADT

- „Erweiterungskonvertierung“
- Konvertierung von Typ mit kleinerem (oder schmalerem) in einen Typ mit größerem (oder breiterem) Bereich
- Sichere Konvertierung $\rightarrow$ Kein Datenverlust

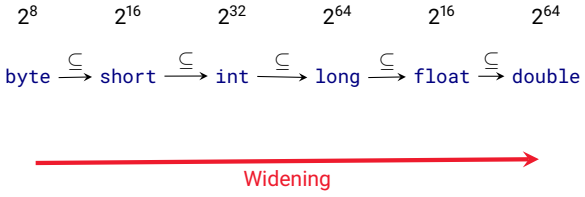

Abbildung: Überblick von widening conversions

Primitive Datentypen - Narrowing conversion

TECHNISCHE UNIVERSITÄT DARMSTADT

- „Verengungskonvertierung“
- Konvertierung von Typ mit größerem (oder breiterem) in einen Typ mit kleinerem (oder schmalerem) Bereich
- Unsichere Konvertierung $\rightarrow$ Datenverlust

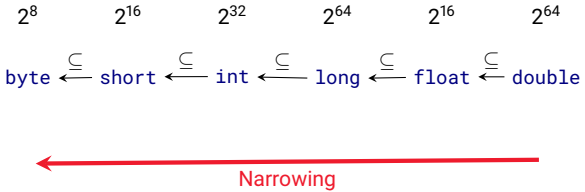

Abbildung: Überblick von narrowing conversions

Primitive Datentypen - Spezialfall

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konvertierung von `byte` nach `char`
- Widening und narrowing
- `byte` wird zu `int` konvertiert
- `int` wird zu `char` konvertiert

Referenztypen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Upcast: Typumwandlung nach oben (bspw. Superklassen)
- Downcast: Typumwandlung nach unten (bspw. Subklassen)
- Fehlermeldung, falls Cast nicht möglich ist!
  - `ClassCastException`
  - Typabprüfung mittels Schlüsselwort `instanceof` möglich
  - Formaler Aufbau: Objekt `instanceof` Typ

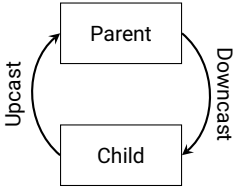

Abbildung: Überblick Upcast und Downcast

Selbstständiges Arbeiten