Vorkurs Programmieren

HTW Berlin
SoSe 2016

Inhalt Vorlesung 4

1. Wiederholung (Woche)
2. OOP
    1. Abstrakte Klassen
    2. Überdecken und Überschreiben
    3. Dynamisches Binden
3. Sonstiges
    1. Enums
    2. Annotations
4. Design Patterns
    1. Singleton

1.) Wiederholung

1.) Interfaces

Interfaces: Zusammenfassung

- Legt Eigenschaften nach außen fest
  - z.B.: `Resizable`, `Comparable`, `Sortable`
- Mehrere Eigenschaften via Interface implementierbar
  - Eine Art Schablone: Interfaces verdecken alle Methoden, außer für die Eigenschaft notwendigen
- Trennt Implementierung von Deklaration
  - Trennen von *“Was ist implementiert”* und *“Wie ist es Implementiert”*.
- Warum Einfach-Klassenvererbung, aber Mehrfach-Schnittstellenvererbung?

1.) Wiederholung

1.) Fehlerbehandlung

Fehlerbehandlung: Zusammenfassung

- Stoppt/Verlässt aktuellen Programmcode sofort (vgl. `Return`)
- Undefinierten Zustand beheben
- `try-catch-finally` möglichst genau
- Eigene Fehler erzeugen bei eigenen Modulen *oder* klare Fehlermeldungen erzeugen

Arten von Exceptions

- Zur Kompilierzeit bekannt:
  - geprüfte (checked) Exceptions, müssen deklariert werden (via `throws`)
  - abgeleitet von `Throwable`
  - bekannt, dass diese geworfen werden können
  - z.B.: `IOException`
- Zur Laufzeit
  - ungeprüfte (unchecked) Exception, können deklariert werden (via `throws`)
  - abgeleitet von `RuntimeException`
  - abhängig vom Zustand, sollten i.d.R. nicht auftreten
  - z.B.: `NumberFormatException`

1.) Wiederholung

1.) Design Patterns

Was sind Design Patterns?

- Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software

- Nicht nur wichtig bei OOP

  - Bsp.: definierter Such-Algorithmus

- *Design* Pattern

  - Muster Zusammenarbeiten mehrerer Klassen

  - *Software-Architektur-Design*

Eigenschaften

- Erprobt
  - auf Fehler und Nebeneffekte
  - auf Nutzen
- Bekannt
  - Entwickler verstehen Code schneller
- Effizienz
  - sparen Zeit
  - Code lässt sich leichter wiederverwenden

Abgrenzung

- Nur Vorlage (Klassen- / Objektnamen, Verwendung)
- Menschenverstand nutzen
- Anti-Pattern: Muster von schlechtem Code

- Bsp.: `“Hello” == “Hello”` vs.
  `“Hello”.equals(“Hello”)`

- Tipp: neues Feature entdeckt?
  > Google nach *Patterns / Anti-Patterns*

2.) OOP

Weitere Aspekte

Abstrakte Klassen

- Neben Interfaces und Elternklasse weitere Möglichkeit
- Zeigt an, dass Klasse nicht alle Interface-Methoden implementieren muss.
- Syntax: `abstract class <name> {}`
- Können nicht selber initialisiert werden

Überdecken und Überschreiben

- Bei der Vererbung von Elternklasse zu Kindklasse:
  - Methoden: werden überschrieben
    d.h. können nur durch super aufgerufen werden, aber nicht beim casten
  - Felder: werden überdeckt
    d.h. können durch casten wieder sichtbar gemacht werden.

Dynamisches Binden

- Beruht auf dem Überschreiben von Methoden
- Erst zur Laufzeit ist bekannt, welches Funktion genau aufgerufen wird.
  Vergleiche: Factory

3.) Sonstiges

Enums

- Enumerations (Aufzählungen)
- Definiert endliche Aufzählung
- Z.B.: Montag, Dienstag, Mittwoch, …

```
public enum Weekday{
  MONDAY, TUESDAY, WEDNESDAY,
  THURSDAY, FRIDAY, SATURDAY, SUNDAY
}

Weekday day = Weekday.SATURDAY

if (day == Weekday.MONDAY) { ... }
```

Annotations

- Meta-Anweisungen im Code
- Geben Anweisungen für IDE, Compiler, Programmierer, andere Module, …

```
@Override

@Deprecated

@Test
```

4.) Design Patterns

Singleton

Singleton

- Nützlich wenn ein Objekt nicht mehrmals erzeugt werden kann / soll.
- Instanz wird über `static getInstance()` zurück gegeben.

```
public class Hello {

    private static Hello instance;

    public static Hello getInstance() {
        if(instance == null) {
            instance = new Hello();
        }

        return instance;
    }

    // ...
}
```

Überblick und Zusammenfassung

Was haben wir gemacht?

- Grundlagen der Programmierung kennen gelernt.
  - Variablen, Funktionen, Basis-Datentypen

Was haben wir gemacht?

- Grundlagen der OOP kennen gelernt.
  - Klassen, Objekte, Vererbung

Was haben wir gemacht?

- Datenstrukturen und Algorithmen
  - Sortieren, Listen, Maps

Was haben wir gemacht?

- Best Practices und Design Patterns
- Testen
- Factory, Singleton

Was kommt noch?

- Viel mehr Details zum Verständnis
- Daten und Programme von und mit anderen Servern verarbeiten
- Gleichzeitige Programmabläufe
- Grafische Benutzeroberflächen
- Datenbanken

Ende

Vielen Dank.

Viel Erfolg und Spaß im Studium.