Programmierung II mit Java

Inhaltsverzeichnis

Tag 1 - Willkommen, Setup & Einführung in Klassen

09:15 - 10:30 module-intro

```
1. Welcome
2. Fachlicher Schwerpunkt des Kurses (Zugdisposition)
2.1. Kontext
2.2. Klassenmodell
3. IDE Setup
3.1. Lokales Arbeitsverzeichnis: Der Workspace
3.2. Programmiersprache Java
3.3. IDE IntelliJ
3.4. Quellcodeverwaltung: Git
3.5. Herunterladen der Kursinhalte (Git mit IntelliJ)
```

10:45 - 11:45 module-tools-n-help

```
1. Projektverwaltung mit Maven & Projektstruktur
2. Test-Driven-Development (TDD)
3. Generierung von Quellcode mit der IDE
4. Autocompletion
```

```
5. Icons & Symbole in der IDE
6. Tipps, Tricks, Probleme & Lösungen
```

	11:45 - 12:15	module-classes

```
1. Organisation & Nutzung von Java Klassen
2. Klassen & Instanzen
```

Tag 2 - Klassen & Interfaces

	09:15 - 10:00	module-classes

```
1. Organisation & Nutzung von Java Klassen
2. Klassen & Instanzen
3. Vererbung
4. Überladen & Übersteuern (Überschreiben)
5. Polymorphismus
```

	10:00 - 10:30	module-object-contract

```
1. Der Objektvertrag
1.1. Objektidentität mit hashCode()
1.2. Objektgleichheit mit equals()
1.3. Objektrepräsentation mit toString()
```

	10:45 - 12:15	module-interfaces

```
1. Implementierung mit Interfaces
1.1. Erzeugungsregeln
1.2. Was kann mit Interfaces eigentlich erreicht werden?
2. Abstrakte Klassen
```

Tag 3 - Abstraktionen & Datentypen

	09:15 - 10:30	module-interfaces

```
1. Implementierung mit Interfaces
1.1. Erzeugungsregeln
1.2. Was kann mit Interfaces eigentlich erreicht werden?
2. Abstrakte Klassen
```

10:30 - 11:45 module-datatypes

```
1. Datentypen
2. Typumwandlungen
2.1. Implizite Typumwandlung
2.2. Explizite Typumwandlung (Casting)
3. Typumwandlung von komplexen Datentypen (Klassen)
```

Tag 4 - Kontrollstrukturen, Sichtbarkeiten & Collections

09:15 - 10:00 module-visibility

```
1. Access Modifier (Sichtbarkeit)
1.1. default
1.2. public
1.3. protected
1.4. private
```

10:00 - 11:00 module-loops

```
1. Kontrollstrukturen
1.1. Konditionalsausdrücke
1.1.1. if - else-if - else
1.1.2. switch case
1.2. Schleifen
1.2.1. for (each)
1.2.2. (do) while
```

11:00 - 12:15 module-collections

```
1. Arrays
2. Collections Framework
2.1. Java Collections API Interfaces
2.2. Special Java Collections Classes
2.3. Synchronized Wrappers
2.4. Unmodifiable Wrappers
```

Tag 5 - Associations, Java Generics & Enums

10:45 - 12:15 module-associations

```
1. Assoziationen
1.1. Beziehungsarten
1.2. Aggregation & Komposition
1.3. Navigierbarkeit
1.4. One-to-One-Assoziation
1.5. One-to-Many-Assoziation
1.6. Many-to-Many-Assoziation
```

	10:00 - 12:15	module-generics

```
1. Java Generics
1.1. Basics & Einführung
1.2. Bounded Generics
1.3. Wildcards
```

	09:15 - 10:00	module-enums

```
1. Java Enums
1.1. Zusätzliche Methoden
1.2. Enums in Switch-Ausdrücken
1.3. Zusätzliche Datenfelder
```

Tag 6 - Funktionale Programmierung

	09:15 - 12:15	module-funcprogramming

```
1. Funktionale Programmierung in Java
1.1. Imperative Programmierung
1.2. Deklarative Programmierung
1.3. Lambda Ausdrücke
1.4. Streaming (API)
```

Tag 7 - Ausnahmen, Fehlerbehandlung, Records & Statics

	09:15 - 11:00	module-exceptions

```
1. Exception Handling
1.1. Exceptions Hierarchie
1.2. Exception Handling
1.3. Mehrere Exceptions
```

```
1.4. Der finally Block
1.5. throws und throw
```

	11:00 - 11:30	module-records

```
1. Java Records
1.1. Ziel/Zweck
1.2. Basics
```

	11:30 - 12:15	module-statics

```
1. Statische Klassen und Variablen
```

Tag 8 - Referenzsemantik, Javadoc, Clean Code

	09:15 - 10:30	module-semantics

```
1. Referenzsemantik
1.1. Wertsemantik - pass by value
1.2. Referenzsemantik - pass reference
```

	10:45 - 11:15	module-javadoc

```
1. Javadoc
1.2. Theorie & Einführung
1.3. Erzeugen von Javadoc
```

	11:15 - 12:15	module-clean-code

```
1. OOP Design Prinzipien
1.1. Single Responsibility Principle
1.2. Open/Closed Principle
1.3. Liskov's Substitution Principle
1.4. Interface Segregation Principle
1.5. Dependency Inversion Principle
1.6. Don't Repeat Yourself Principle (DRY)
1.7. Keep it simple and stupid (KISS)
1.8. Composition Over Inheritance Principle
1.9. Loose coupling between components
1.10. Immutability
1.11. Document your public API
```

```
2. OOP Best Practices & Code Smells
```

Tag 9 - Code-Qualität, Prüfungsvorbereitung

```
09:15 - 10:30          module-patterns
```

```
1. Qualität in Softwareprojekten
2. Erzeugungsmuster (Creational Patterns)
3. Strukturmuster (Structural Patterns)
4. Verhaltensmuster (Behavioral Patterns)
5. Architekturmuster (Architectural Patterns)
6. Refactoring
```

```
10:45 - 12:15          Prüfungsvorbereitung & Prüfungsorganisation
```

```
1. Alle Themen im "Schnelldurchlauf"
2. Organisation der Prüfung
3. Zeit für Fragen
```

Tag 10 - Prüfungstage/-block

Ausgestaltung und Modi der Prüfungen aktuell noch nicht final festgelegt. Es gibt aber folgende **Idee**. Die **Prüfung** besteht insgesamt aus zwei Teilen:

```
Dauer          : 1h
Kandidat:Innen : 3-4er Gruppen
```

1. Prüfung der **Programmierkenntnisse**, entweder ...
   a. als 3-4er Gruppe am vorbereiteten Rechner gemeinsam & live Progemmieraufgaben lösen, mit Unit-Tests und Klassen in /src Paketen, oder ...
   b. manuell schriftlich auf Papier, individuell
2. Prüfung des **Grundverständnisses**: vorbereiteter Use-Case, konzeptionell mündliches Gespräch, Gruppenarbeit an der Tafel, Fragen und Antworten

Optionale Themen

Datenbankprogrammierung mit Java

```
hh:mm - hh:mm          module-databases
```

```
1. Datenbank Programmierung
1.1. Preparation
1.2. Setup H2 Database
1.3. Theorie & Einführung
1.3.1. SQL-Datenbanken
1.3.2. Structured Query Language (SQL)
1.3.3. NoSQL-Datenbanken
1.3.4. SQL oder NoSQL?
1.3.5. ORM Objektrelationales Mapping
1.3.6. JPA - Jakarta Persistence API
```

Parallele Programmierung mit Threads

hh:mm - hh:mm module-threads

```
1. Parallele Programmierung mit Threads
1.1. Thread & Runnable
1.2. Synchronized
```

Integration von Software/-komponenten

hh:mm - hh:mm module-integration

```
1. Was versteht man unter "Integration"?
2. Integrationskriterien
3. Integrationsoptionen bzw. -muster
4. Synchrone Integration mit (REST)
5. Asynchrone Integration mit Nachrichten (Messaging)
```