Agendavorschlag zum Seminar

Grundlagen der Java-Programmierung

Dauer: 5 Tage

Autor:
Stephan Karrer

Stand:
1.10.2018

Zu dieser Agenda

Diese Agenda ist ein Vorschlag bzgl. der Inhalte und Unterrichtszeiten für das Seminar

Grundlagen der Java-Programmierung

Bitte überprüfen Sie, ob diese Konzeption Ihren Erfordernissen entspricht. Änderungen oder Ergänzungen hinsichtlich der Seminarinhalte oder des Zeitplans nehmen wir in Absprache mit Ihnen, soweit möglich, gerne vor.

Zeitliche Struktur des Seminars

Wir gehen von 8:30 -16:30 täglicher Seminarzeit aus. Individuelle Regelungen zu den Unterrichtszeiten und den Pausenregelungen stimmen wir mit Ihnen ab.

Hard/Software - Medien

Die Teilnehmer benötigen PCs unter Windows oder Linux mit folgender zusätzlicher Software:

- Java Development Kit Version 8 (JDK 8)
- Eclipse als Entwicklungsumgebung
- Internet-Zugang

1.Tag **Inhalte**

Einführung in Java
- Java Compiler, Bytecode,
- Virtuelle Maschine (JVM), Java-Laufzeitumgebung
- Write Once, Run Anywhere‹ (WORA) in der Praxis
- Java SE und Java EE, Java ME

Erste Schritte mit der Entwicklungsumgebung Eclipse
- Views, Editoren, Perspektiven
- Workspace-Konzept
- Projekte anlegen
- Java-Codierungen erzeugen
- Helferlein

Datentypen und Kontrollstrukturen
- Syntax von Java-Programmen
- Programmausführung
- Variablen und Datentypen
- Ausdrücke und Operatoren
- Fallunterscheidungen und Schleifen
- Methoden
- Pakete und Namensräume
- Erste Praxisübungen

OO-Grundprinzip: Klassenbildung und Kapselung
- Kapselung
- Sichtbarkeits-Level
- Objekterzeugung und Konstruktoren
- Speicherfreigabe, Garbage-Collection
- Zugriff auf Eigenschaften
- Beans-Konventionen
- Referenz-Semantik bei Objekten

2.Tag **Inhalte**

OO-Grundprinzip: Klassenbildung und Kapselung
- Identität und Objektvergleiche
- Implementierung von equals() und toString()
- Die this-Referenz
- Überladen von Methoden
- Statische Elemente
- Konstanten
- Private Konstruktoren, Fabrikmethoden

OO-Grundprinzip: Vererbung
- Generalisierung und Spezialisierung
- Vererbung in Java
- Beispiel anhand eines Applets
- Elternklasse gibt Schnittstelle vor
- Überschreiben oder Verdecken
- Überschreiben oder Überladen
- Polymorphismus in Java
- Typ-Anpassung und –prüfung
- Verkettung der Konstruktoren
- Vorhandene Klassen um Kindklassen erweitern
- Probleme hierbei?

3.Tag **Inhalte**

OO-Prinzip: Bedeutung von Schnittstellen
- Abstrakte Klassen
- Beispiele
- Interfaces in Java
- Sortieren auf Basis der Interfaces „Comparable“ und „Comparator“
- Schnittstellen erweitern, Marker-Interfaces

Arrays in Java
- Schreibweisen, Deklaration und Initialisierung
- Arrays von Primitiven und Objekten
- Mehrdimensionale Arrays

Aufzählungstypen (Enum)
- Interface als Konstantenpool?
- Enum-Konzept ab Java 5
- Umsetzung eines einfachen Aufzählungstyps
- Enums können durchaus komplexer werden

Ausnahmebehandlung
- Ausnahmen behandeln mit try-catch-finally
- Ausnahmen weiterleiten
- Checked Exceptions und Unchecked Exceptions, RuntimeException
- Praktische Umsetzung
- Eigene Ausnahmen

4.Tag **Inhalte**

Wrapperklassen und Boxing
- Aufgaben der Wrapperklassen
- Autoboxing und Autounboxing ab Java 5
- Vorsicht: Objekte haben Referenzsemantik, Primitive haben Wertsemantik

Generische Datenstrukturen
- Worum geht es
- Typparameter, generische Klassen und generische Methoden
- Wie sieht das in der Dokumentation aus
- Wildcards: Syntax und Sinn

Einführung - Datenstrukturen in Java: Collection-Klassen
- Index-sequentielle, verkettete und gehashte Datenstrukturen: Vor- und Nachteile
- Struktur der Collection-API: Interfaces, Klassen und Algorithmen
- Das Iterator-Konzept
- Verwendung gehashter Datenstrukturen

Innere Klassen
- Formen
- Syntax
- Beispiele für die Verwendung

5.Tag **Inhalte**

Dateizugriff und I/O

- Zugriff auf Dateien, Verzeichnisse und Dateiattribute
- Möglichkeiten ab Java 7
- Binäre Datenströme (XXXStream)
- Lesen/Schreiben von Text-Formaten (Reader, Writer)
- Objektserialisierung
- Properties

Eine erste GUI für unseren bisherigen Programmcode:
Tabellarische Darstellung unserer Beispielsdaten

- Verwendung von JTable aus der Swing-Bibliothek
- Umsetzung eines Table-Models