Agendavorschlag zum Seminar

Java Vertiefung

Dauer: 5 Tage

Autor:
Stephan Karrer

Stand:
25.10.2024

Zu dieser Agenda

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

Java Vertiefung

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 09:00 - 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:

- Internet-Zugang
- Java Development Kit Version 17 oder höher
- Eclipse für Enterprise Developers oder IntelliJ als Entwicklungsumgebung

Sonstiges

- Beispiele´und Übungen werden als Maven-Projekte zur Verfügung gestellt.
- Es wird die jeweils aktuelle JPA-Version benutzt (Jakarta Persistence API 3.x).
- Als Beispielsdatenbank dient H2. Diese kann im Seminar bereit gestellt werden.

1.Tag **Inhalte**

Testen mit JUnit
- Zentrale JUnit-Klassen und Annotationen für JUnit 5 (Jupiter)
- Assertions
- Lebenszyklus einer Testsuite
- Organisation der Testklassen, Sammeln von Testfällen in Suites
- Testen von Exceptions

Bemerkung:
Junit kann dann bei den folgenden Praxisbeispielen durchgängig eingesetzt werden

Vertiefung: Vererbung, Abstrakte Klassen, Interfaces
- Vererbung ist Design-Thema
- Komposition vor Vererbung: Warum Vererbung einschränkt
- Details zur statischen und Instanz-Initialisierung
- Konstruktoren defensiv programmieren
- Private Konstruktoren und Factory-Methoden
- Abstrakte Klasse versus Interface
- Lose Kopplung bei der Codierung, enge Kopplung zur Laufzeit
- Ausblick: Dependency Injection
- Schnittstellenerweiterungen mit Java 8:
  Statische und Default Methoden in Interfaces

Generische Datenstrukturen
- Worum geht es
- Typparameter, generische Klassen und generische Methoden
- Beispiel: Eigene generische Klassen und Interfaces
- Wildcards: Syntax und Sinn
- Unterschied zu den klassischen typisierten Arrays

2.Tag **Inhalte**

Design Patterns
- Was sind Muster in der Software-Entwicklung?
- Kategorien von Mustern

Erzeugungsmuster
- Factory
- Builder
- Prototyp
- Singleton

Strukturmuster
- Adapter
- Bridge
- Decorator
- Facade
- Composite

Verhaltensmuster (Behavioral Pattern)
- Template Method
- Command
- Observer
- Visitor
- Strategy

3.Tag **Inhalte**

Ausnahmebehandlung im Überblick
- Ausnahmen behandeln mit try-catch-finally
- Ausnahmen weiterleiten
- Checked Exceptions und Unchecked Exceptions, RuntimeException
- Erweiterungen mit Java 7
- Eigene Ausnahmen

Datenstrukturen in Java: Collection-Klassen
- Index-sequentielle, verkettete und gehashte Datenstrukturen: Vor- und Nachteile
- Die Bedeutung von equals() und hashcode()
- Struktur der Collection-API: Interfaces, Klassen und Algorithmen
- Das Iterator-Konzept
- Verwendung gehashter Datenstrukturen
- Verwendung von Optional

Einführung in Lambda-Ausdrücke und Streams
- Motivation
- Schreibweisen
- Methodenreferenzen
- Streams von Arrays und Collections
- Terminale und intermediäre Operationen
- Typische Map-Reduce-Operationen
- Parallel-Varianten und Performanz-Aspekte

4.Tag **Inhalte**

Einführung in Threads
- Thread-Konzept
- Runnable und Parameterübergabe bzw. Ergebnisabfrage
- Lebenszyklus von Threads, Starten und Stoppen, Interrupt
- Thread-Scheduling und Prioritäten
- Exceptions: Auffangen von Ausnahmen
- Thread-Gruppen

Verwendung von Thread-Pools
- Runnable und Callable
- Executor-Schnittstellen
- Future als asynchroner Rückgabetyp
- Fork-Join-Pool

Synchronisation
- Synchronisationsproblem
- Mit synchronized Methoden oder kritische Abschnitte schützen
- Die Schnittstelle Lock für explizites Locking
- ReentrantLock, Read/Write-Lock
- Deadlocks vermeiden mit wait/notify bzw. condition/signal

Spezielle Möglichkeiten zur Thread-Koordination im Überblick
- Blocking Queues
- Semaphore
- Completable Future

5.Tag **Inhalte**

Datenbankzugriff mit JDBC
- Datenbanktreiber, JDBC-API
- Eine Beispielverbindung, SQL-Anweisungen ausführen
- DriverManager, ResultSet
- Datentyp-Mapping
- Queries, Parameter
- Transaktionen

Einführung in objekt-relationales Mapping mit JPA
- JPA-Architektur
- Entities und Entity Manager
- Einfaches Mapping mittels Annotationen
- Persistence Context und Persistence Unit
- CRUD-Operationen
- Synchronisation mit der Datenbank
- Transaktionssteuerung
- Abbildung der relationalen Beziehungen
- Eager- versus Lazy-Fetch