Geplante Agenda

Java Fortgeschritten

Dauer: 10 Tage

Autor:
Stephan Karrer

Stand:
20.03.2022

1.Tag **Inhalte**

Wiederholung und 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
- Schnittstellenerweiterungen mit Java 8:
  Statische und Default Methoden in Interfaces

Innere Klassen

- Arten: static, instance, lokal, anonym
- private oder public
- typische Verwendungen

Speicherverbrauch und Garbage-Collection

- Arbeitsweise des Garbage-Collectors
- Inwieweit haben wir Einfluß
- Unnötige Objekterzeugung vermeiden
- Memory Leaks

2.Tag **Inhalte**

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

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

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
- Unschönes Design: NotSupportedOperation

3.Tag **Inhalte**

Datenstrukturen in Java: Fortsetzung
- Verwendung gehashter Datenstrukturen
- Navigable Sets
- Queues
- Immutable Collections

Spezielles zu Strings
- Pattern-Matching
- Reguläre Ausdrücke
- Ausgabe-Formatierung

Die Klasse Optional
- Optional statt null
- Erzeugung und Nutzung von Optional
- Optional für Primitive

4.Tag **Inhalte**

Lambda-Ausdrücke und funktionale Programmierung

- Motivation
- Schreibweisen
- Methodenreferenzen
- Konstruktorreferenzen
- Zugriff auf die Umgebung: effektiv final
- Was ist mit Checked Exceptions
- Möglichst zustandslos und keine Seiteneffekte
- Funktionale Schnittstellen in java.util.function
- Die Basis: Supplier, Function, Consumer
- Primitiv-Varianten: Boxing vermeiden
- Nutzen der statischen und default-Methoden in den Schnittstellen

5.Tag **Inhalte**

Übungstag

6.Tag **Inhalte**

Streams

- Motivation
- Arbeitsweise
- Streams und das Builder-Pattern
- Stream-Erzeugung
- Intermediate Operations
- Terminal Operations: Collectoren und Reducer
- Short-Circuit Operations
- zustandsbehaftete Operationen
- Streams mit Primitiven: Boxing vermeiden
- Eigene Collectoren und Reducer
- Wie arbeiten parallele Streams
- Bringt die Parallelverarbeitung was
- Fallstricke

7.Tag **Inhalte**

Datum und Zeit mit Java (Schwerpunkt Date-Time-API ab Version 8)
- Die alten Varianten: Date, Calendar, GregorianCalendar
- java.util.Date ist Basis für java.sql.Date
- Maschinenzeit versus Kalender
- Die Klassen LocalDate, LocalTime und LocalDateTime
- Weitere Klassen und Berechnungsmöglichkeiten
- Zeitzonen berücksichtigen
- Zeitdauern (Period, Duration)
- Parsen und Formatieren von Datumszeitwerten
- Konvertierungen zwischen der klassischen API und Date-Time-API

Berücksichtigung nationaler Konventionen
- Locale-Objekte
- Nationale Ausgabeformate
- RessourceBundle-Mechanismus

Multi-Threading Basis
- Grundsätzliches
- Wie wird das durch die VM auf die jeweilige Systemplattform abgebildet
- Die Klasse Thread und das Interface Runnable
- Thread-Attribute
- Threads koordinieren: sleep, join, interrupt
- Deprecated: suspend und stop
- Terminierung und Exception-Handler
- Threadgruppen

Thread-Pools
- Executor-Schnittstelle und konkrete Executoren
- Callable statt Runnable
- Die Option auf die Zukunft: Future als Rückgabe
- Erweiterung durch CompletableFuture
- CompletableFuture als Alternative zu Streams?
- Spezialfall: Fork-Join-Pool

8.Tag **Inhalte**

Thread-Safety
- Probleme beim parallelen Zugriff
- Was ist in Java atomar
- Spezialfälle und Atomic Types
- Basissperrmechanismus: synchronized
- Wirkungsweise und korrekte Verwendung
- Verwendung von Lock-Objekten
- Schreib- und Lesesperren
- Deadlocks und ihre Vermeidung
- wait/notify aus Object und Condition bei Lock-Objekten
- Thread-Safe Datenstrukturen

Optional: Threads Spezielles
- Spezielle Synchronsationsmechanismen: Semaphore, Barrier, …
- Zeitsteuerung via Timer und Timertask
- ShutdownHook
- ThreadLocal

-

9.Tag **Inhalte**

Ab Java 9: Modularisiertes JDK
- Die Ziele von Projekt Jigsaw
- Überblick über die Systemmodule
- Classpath vs. Module Path
- Einschränkungen: Interne APIs, Deep Reflection
- Compiler- und Laufzeit-Optionen zur Übersteuerung
- Umstrukturierungen im JDK
- Remove-EE (was entfällt und welche Alternativen gibt es)
- Ausblick: Modularisierung eigener Anwendungen

Zeitpuffer zur Ergänzung vorhandener Themen

10.Tag Inhalte

Übungstag