ORACLE

Java SE 8 New Features

Dirk Reinemann
Working Student Sales Consulting

Program Agenda

1. Language
2. Libraries
3. Internationalization
4. Security
5. Platform
6. Virtual Machine

Die Java Standard Edition 8 Spezifikation befindet sich noch in der Entwicklung!

1. Language

Lambda Expressions

Motivation

- Von vielen Sprachen unterstützt (Ruby, Groovy, Scala, …)

  → Akzeptanz und Vertrautheit
- Bereitstellung von Internal Iteration

Lambda Expressions

Internal Iterations

- Datenstruktur erhält Quellcode zum Ausführen
  - Wahl der geeigneten Reihenfolge
  - Parallele Verarveitung
  - Partitionierung über mehreren Kernen

→ Optimierte Ausführung

Lambda Expressions

Virtual Extension Methods

- Erweiterung bestehender Interfaces durch Methoden
  - Gewährleistung der Abwärtskompatibilität
  - Merhfachvererbung des Verhaltens
    - Nicht Zustand!

```
public interface Set<T> extends Collection<T> {

    public T reduce(Reducer<T> reducer) default Collections.<T>setReducer();
}
```

Lambda Expressions

Ziele

- Vereinfachte Arbeit mit abtrakten und hochperformanten Bibliotheken
  - Ressourcenverbrauch
  - Erstellung
  - Migration
- Verbesserte Unterstützung von Multicore
  - Internal Iteration

Generalized Target-Type Inference

- Reduzierung expliziter Typenangaben in Methodenaufrufen

```
class List<E> {

   static <Z> List<Z> nil() { ... };

   static <Z> List<Z> cons(Z head, List<Z> tail) { ... };
}

List<String> ls = List.nil();

Java SE 7: List.cons(42, List.<Integer>nil());
```

Annotations on Java Types

- Erweiterte Nutzung von Annotationen
  - Verwendung von Typen kennzeichnen
    - Kritischer Fehler

```
void monitorTemperature() throws @Critical TemperatureException { ... }
```

  - Type Checkers zur Ermittlung von Fehlern bei der Kompilierung
    - Null Pointer

```
String myString = (@NonNull String) str;
```

Access to Parameter Names at Runtime

- Abfrage der Parameternamen von Methoden und Konstruktoren zur Laufzeit
  - Eliminierung überflüssiger Annotationen
  - Automatische Generierung von Template Code

```
@Override
private int doSomething(String first, String second) { ... }
```

- Reflection: Ja Nein

Repeating Annotations

- Mehrere Annotationen des selben Typs an einem Element
  - Vermeidung von Containern

→ Verbesserung der Lesbarkeit des Quelltextes

```
@Schedule(dayOfMonth="last")
@Schedule(dayOfWeek="Fri", hour="23")
public void doPeriodicCleanup() { ... }
```

2. Libraries

Bulk Data Operations for Collections

- Massenoperationen für große Datenmengen
- Verschiedene Verabeitungsformen
  - Seriell
  - Parallel
    - Nutzung der Fork-Join Implementierung
    - Festlegung der Ausführungszeit
      - Lazy
      - Eager

Concurrency Updates

- Skalierbare Variablen
  - *DoubleAccumulator, DoubleAdder, LongAccumulator, LongAdder*
    - Bessere Skalierung bei häufigem und parallelem Zugriff
- Hashmap Caches
  - *ConcurrentHashMap*
    - Verbesserte Schlüsselerzeugung
    - Unterstützung großer Datenmengen
    - Verbessertes Scannen und Aufräumen

Date and Time API

- Unterstützung verschiedener Standards
  - Datum, Zeit, Zeitzone, Instant
  - ISO-8601, CLDR, BCP47
- Erweiterbares Kalendersystem

JDBC 4.2

- Generische Setter und Update Methoden
  - *ResultSet, PreparedStatement, CallableStatement*
  - Unterstützung neuer Datentypen
- Spezifizierung von Properties für die Java EE Umgebung
  - *DataSource*

Parallel Array Sorting

- Sortierung von Arrays Parallel
  - *java.util.Arrays*
  - *parallelSort*
  - Primitiven Typen außer boolean

→ 30% Geschwindigkeitssteigerung

Base64 Encoding and Decoding

- Java SE 7
  - *sun.misc.BASE64Encoder*
  - *sun.misc.BASE64Decoder*
  - Nutzung inoffizieller API
- Java SE 8
  - *java.util.Base64.Encoder, java.util.Base64.Decoder*
  - *encode, encodeToString, decode, wraps*
  - Nutzung standardisierter API

Reduce Core-Library Memory Usage

- Reduzierung der Größe von Objekten
  - Helferklasse für ungenutzte Felder (Refelection)
- Deaktivierung des Reflection Compiler
  - Keine Bytecodegenerierung durch Reflektion Compiler
- Sonstige Speicherreduzierungen
  - Anpassung der Startgröße von Tabellen, Caches, Buffers

→ Performanceberücksichtigung

Charset Implementation Improvements

- Verringerung der Größe von Zeichensätzen
- Erzeugung von Zeichensätzen aus Textdateien zur Laufzeit
- Performanceverbesserung beim Kodieren und Dekodieren

Enhance Core Libraries with Lambdas

"Oh cool, they added a Lambda here, and that lets me solve my problem more easily."

Stuart W. Marks
Principal Member of Technical Staff

3. Internationalisation

Unicode 6.2

- Unterstützung des Unicode 6.2 Standards

4. Security

Configurable Secure Random-Number Generation

- Bessere Implementierung von *SecureRandom*
- Anwendungen können sich unter Linux aufhängen
  - Java Virtual Machine nutzt /dev/random
  - Blockiert das System

Enhanced Certificate Revocation-Checking API

- Ausfall beim Kontakt mit Server ist ein fataler Fehler
  - Alles oder Nichts Prinzip
  - *java.security.cert*
- Neue Klassen
  - *RevocationChecker*
  - *RevocationParameters*

Leverage CPU instructions for AES Cryptography

- Verbesserung der Performance der AES Verschlüsselung und Entschlüsselung

SHA-224 message digests

- Eweiterung des Java Development Kits um die SHA-224 Algorithmen

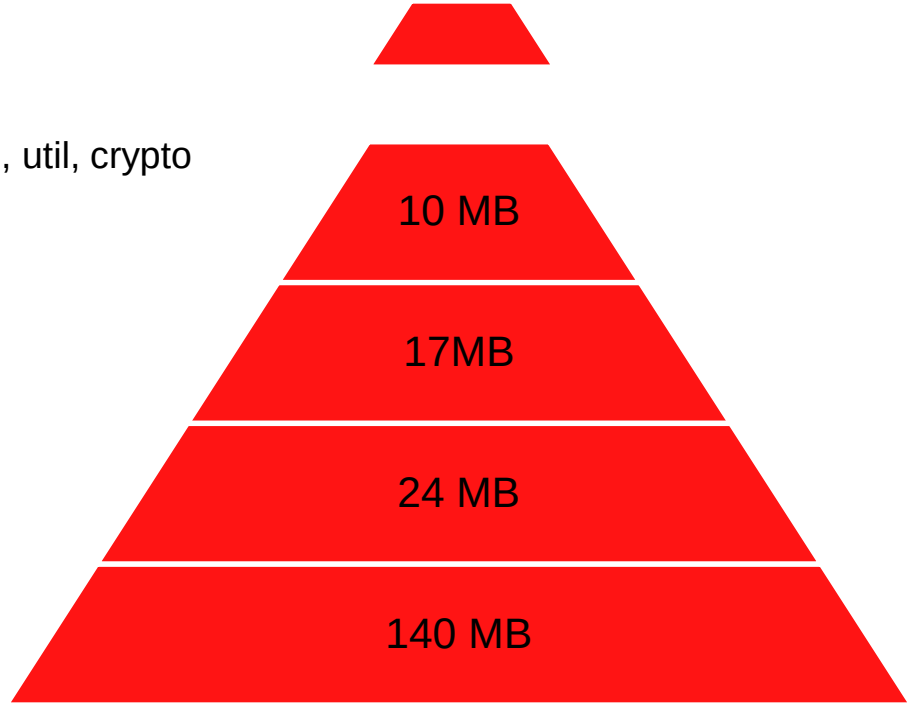
5. Platform

Compact Profiles

- Definition von Profilen als Untermenge des Java Runtime Environment
- 3 Compact Profiles
  - Aufeinander aufbauend
  - Identifikation zur Laufzeit möglich
  - Toolerweiterung zur Erstellung von Anwendungen für ein Profill

Compact Profiles

- Compact 1
  - io, lang, math, nio, net, security, text, time, util, crypto
- Compact 2
  - rmi, sql, transaction, xml
- Compact 3
  - management, naming, tools
- Java SE 8
  - *



Prepare for Modularization

- Überarbeitung hinderlicher Quelltextteil
  - Class Loading
  - Service Provider
- Kommandozeilentool
  - Überprüfung von Abhängigkeiten der Anwendung
- Anpassung der Pfade im Java Verzeichnis
  - Relativer statt absoluter Pfade

Launch JavaFX Applications

- Erweiterung des Java Kommandozeilentools um Java FX Initialisierungslogik
  - Musste bisher in die Anwendung integriert werden
  - Ausführung von Java FX Anwendungen einfacher

6. Virtual Machine

Lambda-Form Representation For Method Handles

- Verbesserung der Method Handles und Invokedynamic
  - Performance, Qualität, Portabilität
- Reduzierung des Assembly Codes in der Java Virtual Machine
- Reduzierung der nativen Aufrufe während einer Methode Handle Verarbeitung

Remove The Permanent Generation

- Entfernung der Permanenten Generation
  - Kein Tuning mehr notwendig
- Allokation der Metadaten einer Klasse im nativen Speicher
- Strings und statische Klassenvariablen in den Heap
- Teil der HotSpot und JRockit Konvergenz

Enhanced Verification Errors

- Zusätzliche Informationen bei der Verifizierung von Java Bytecode
  - Bisher nur *VerrifyError*
  - Für Tools und Packages die Bytecode verändern

Reduce Class Metadata Footprint

- Reduzierung der Metadaten von Klassen
- Nutzung der Techniken von CVM und Java ME CDC

→ 25% möglich

Nashorn JavaScript Engine

- JavaScript Engine für das Java Development Kit
- Unterstützung der *javax.script* API
- JavaScript Code von Java aufrufen und umgekehrt
  - Direktes Mapping zu *JavaBeans*
- Kommandozeilentool
  - Evaluierung von JavaScript Code in Shebang Scripts
- Verbesserung der Performance im vergleich zu Rhino

Autoconf-Based Build System

- Erhöhung der Build Geschwindigkeit
- Vereinfachung des Build Quelltextes
- Genaue und reproduzierbare Build Ausgabe
- Überarbeitung der Makefile Struktur
- Umsetzung von Autoconf
- Parallele Kompilierung
- Inkrementelle Builds

ORACLE

Vielen Dank für Ihre Aufmerksamkeit

Haben Sie noch Fragen?