MT AG

data . cloud . mobile

Besserer Java Code, außerhalb der Automatismen

Wolfgang Nast

Agenda

- Voraussetzung
- Strukturieren
- Collections und Lambdas
- Optional und Objects
- Builder und CodeChains
- Exception und Lambda
- Zusammenfassung

Im Überblick

Technologie-orientiert
Branchen-unabhängig

Hauptsitz
Ratingen

240
Beschäftigte

MT AG

Zertifizierter
Partner von
Oracle,
Microsoft
und SAP

Inhabergeführte
Aktiengesellschaft

Gründungsjahr
1994

Ausbildungs-
betrieb

kununu
TOP COMPANY
www.kununu.com
VON MITARBEITERN EMPFOHLEN!

Niederlassungen
Frankfurt am Main, Köln

MT AG

Voraussetzung

- IDE (Eclipse, Intellij, NetBeans)
  - Warnings
  - Formatter
- Maven, Gradle
- Checkstyle
- Findbugs
- Sonarqube

Strukturen

- try catch finally
- synchronized und Lock
- Generic Parameter

try catch finally

Üblich

```
InputStream inStream = null;
   try {
      inStream = new FileInputStream (Datei);
      //Logic;
   } catch (IOException e) {//Handling 1}
   } finally {
      try{
         if(inStream != null){
            instream.close();
         }
     } catch (IOException e) {//Hadling 2}
   }
```


MT AG

try catch finally

Vereinfacht

```
try (InputStream inStream = new FileInputstream(Datei)) {
   //Logic;
   return Ergebnis;
} catch (IOException e){
   //Handling 1+2
}
```

Synchronized und Lock

Üblich

```
public Wert block(){
    Wert retVal = new Wert();
    synchronized (retVal) {
        //Logic;
    }
    return retVal;
}
```

Synchronized und Lock

Modern umfangreich

```
public Wert block(){
    Lock lock = new ReentrantLock();
    try {
        lock.lock();
        //Logic;
        return Ergebnis;
    } finally{
        lock.unlock();
    }
}
```


MT AG

Lock as Resource Teil 1

Hilfsklasse

```
public class LockRes() implements Closeable{
   private Lock lock;
   public LockRes(Lock lock) {
        this.lock = lock;
       lock.lock();
   }
   public LockRes(Lock lock, boolean inter) throws
           InterruptedException {
       this.lock = lock;
       lock.lockInterruptibly();
   }
```


MT AG

Lock as Resource Teil 2

Hilfsklasse

```
    @Override
      public void close() {
         lock.unlock();
      }
}
```


MT AG

Lock as Resource ohne Exception

Hilfsklasse verwenden

```
public Wert block(){
   Lock lock = new ReentrantLock();
   try (LockRes bl = new LockBlock(lock)){
      //Logic;
      return Ergebnis;
   }
}
```

Lock as Resource mit Exception

Hilfsklasse verwenden

```
public Wert block(){
   Lock lock = new ReentrantLock();
   try (LockRes bl = new LockBlock(lock, true)){
      //Logic;
      return Ergebnis;
   } catch(InterruptedException e) {
     //Unterbrochen
     return null;
   }
}
```


MT AG

Generic Parameter

Üblich

```
public Object update(Updateable daten){
   try {
      return em.update(daten);
   }
   return null;
}
   daten1 = (Daten1)update(daten1);
   daten2 = (Daten2)update(daten2);
```

Generic Parameter

Vereinfacht

```
public <T extends Updateable>
  T update(T daten){
   try {
      return em.update(daten);
   }
   return null;
}
   daten1 = update(daten1);
   daten2 = update(daten2);
```


MT AG

Generic Parameter

Impliziter Cast

```
@SuppressWarnings("unchecked")
public <T> T aufruf(int pos){
   return (T)daten[pos];
}
   Daten1 daten1 = aufruf(1);
   Daten2 daten2 = aufruf(2);
```

Collection und Lambdas

- List und HashMap
- For Schleifen
- Streams
- Supplier (Command Pattern)

Vector und List

Üblich

```
    Vector werte = new Vector();
    werte.addElement(wert1);
    werte.addElement(wert2);



for(int i = 0; i <werte.size(); ++i) {
    Object wert = werte.get(i);
    //Logic
}
```


MT AG

Vector und List

Verbessert

```
List<?> werte = List.of(wert1, wert2);


werte.forEach(wert -> {
   //Logic
});
```


MT AG

Hashtable und HashMap

Üblich

```
Hashtable map = new Hashtable();
map.put(key1, wert1);
map.put(key2, wert2);
for(Enumeration keyEnum = map.keys(); keyEnum.hasMoreElements(); ) {
    Object key = keyEnum.nextElement();
    Object wert = map.get(key);
    //Logik
}
```

Hashtable und HashMap

Verbessert

```
Map<?, ?> map = Map.of(key1, wert1,
   key2, wert2);

map.forEach((key, wert) -> {
   //Logik
});

//Nützliches
map.merge(key, val, String::concat);
map.getOrDefault(key, defWert);
```

Streams

Üblich

```
Collection<Integer> zahlen = List.of(1, 3, 6);
zahlen.stream().filter(x -> x != null)
    .mapToInt(x -> x)
    .forEach(System.out::println);

List<Float> zahlen2 = List.of(1.1f, 3.7f, 6.54f);
zahlen2.stream()
    .filter(Objects::nonNull)
    .filter(x -> x > 0)
    .forEach(System.out::println);
```

Streams

Verbessert

```
IntStream zahlenSt = IntStream.of(1, 3, 6);
zahlenSt.forEach(System.out::println);

DoubleStream.of(1.1, 3.7, 6.54)
    .filter(x -> x > 0)
    .forEach(System.out::println);
```

Supplier (Command Pattern)

Üblich

```
Logger log = Logger.getLogger("Main");
   //logging
   if (log.isLoggable(Level.WARNING)){
      log.warning("Text: " + wert.getTeil());
   }
```


MT AG

Supplier (Command Pattern)

Verbessert

```
Logger log = Logger.getLogger("Main");
//logging
log.warning(() -> "Text: " + wert.getTeil());
```

Optional and Objects

- Optional
- Comparator
- equals und hashCode
- requireNonNull
- isNull und nonNull

Optional

Üblich

```
if (wert != null) {
    if (wert.getText() != null) {
        if (wert.getText().getTeile() != null) {
            //Logic
        }
    }
}
```

Optional

Verbessert

```
Optional.ofNullable(wert)
   .map(Wert::getText)
   .map(Text::getTeile)
   .ifPresent(x -> {
       //Logic
});
```


MT AG

Comparator

```
public class WerteKlasse implements Compareable<WerteKlasse> {
   //Attribute
   public static Comparator<WerteKlasse> comp =
   Comparator.comparingLong(WerteKlasse::getId)
   .thenComparing(WerteKlasse::getText);

    @Override
    public int compareTo(WerteKlasse o) {
        return comp.compare(this, o);
    }
```

equals

equals mit compareTo

```
 @Override
 public boolean equals(Object obj) {
    if (this == obj) {
       return true;
    }
    if (obj != null && !getClass().equals(obj.getClass())) {
       return false;
    }
    return compareTo((WerteKlasse)obj) == 0;
}
```

hashCode

```
@Override
public int hashCode() {
    return Objects.hash(id, text);
}
```

requireNonNull(@NotNull)

Üblich

```
  public void aufruf(Wert wert) {
    if (wert == null) {
       throw new NullPointerException(msg);
    }
    //Logic
}
```

requireNonNull(@NotNull)

Verbessert

```
 public void aufruf(Wert wert) {
    Objects.requireNonNull(wert, msg);
     //Logic
}
```

isNull und nonNull

Üblich

```
if (wert != null) {
    if (wert.getSub() == null) {
        //Logic
    }
}
```

isNull und nonNull

Verbessert

```
 stream().filter(Objects::nonNull)
     .map(Wert::getSub)
     .filter(Objects::isNull)
     .forEach(x -> {
         //Logic
})
```


MT AG

Builder und CodeChains

- Konstruktor
- Builder

Konstruktor Teil 1

```
public class Werte {
    private final long id;
    private final String name;
    private final String vorname;
    public Werte(long id, String name, String vorname) {
        this.id = id;
        this.name = name;
        this.vorname = vorname;
    }
```

Konstruktor Teil 2

```
 public long getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public String getVorname() {
        return vorname;
    }
}
```

Builder Teil 1

```
public class WerteBuilder {
    private long id = 0;
    private String name;
    private String vorname;
    public WerteBuilder() {
    }
    public WerteBuilder withId(long id) {
        this.id = id;
        return this;
    }
```

Builder Teil 2

```
 public WerteBuilder withName(String name) {
        this.name = name;
        return this;
    }
    public WerteBuilder withVorname(String vorname) {
        this.vorname = vorname;
        return this;
    }
    public Werte build() {
        return new Werte(id, name, vorname);
    }
}
```

Konstruktor und Builder

Aufruf

```
Wert wert1 = new Wert(12, "Müller", "Wolfgang");


Wert wert1 = new WerteBuilder()
    .withId(12)
    .withVorname("Wolfgang")
    .withName(„Müller").build();
```

Exception und Lambdas

- Checked Exception in Lambdas

Checked Exceptions in Lambdas

Beispiel Methode

```
public String trim2(String text) throws IOException, InterruptedException {
    if (text == null) {
      throw new IOException("Text is null!");
    }
    if (text.length() == 0) {
      throw new InterruptedException("Text is empty!");
    }
    return text.trim();
  }
```

Checked Exceptions in Lambdas

Beispiel Methode

```
Stream.of("Teil 1", " Teil 2 ", "Teil 3", "", null)
    .map(HideException.hide(main::trim2))
    .forEach(System.out::println);
```

Checked Exceptions in Lambdas

Aufruf Beispiel Üblich

```
Stream.of("Teil 1", " Teil 2 ", "Teil 3", "", null).map( x -> {
   try {
      main.trim2();
   } catch(IOException |
           InterruptedException e){
      //Logic Exception
   }
})).forEach(System.out::println);
```

Checked Exceptions in Lambdas

Aufruf Beispiel mit Hilfsklasse

```
try {
   Stream.of("Teil 1", " Teil 2 ", "Teil 3", "", null).map(HideException
      .hide(main::trim2))
      .forEach(System.out::println);
} catch (HideException e) {
   //Logic Exception
}
```

Checked Exceptions in Lambdas

Hilfsklasse Teil 1

```
@FunctionalInterface
public interface HideException<Parameter, Return, Except extends Exception> {
  Return call(Parameter value) throws Except;
  default Return callDef(Parameter value) {
    try {
      return call(value);
    } catch (RuntimeException e) {
      throw e;
    } catch (Exception e) {
      throw new HideException(e);
  }}
```

Checked Exceptions in Lambdas

Hilfsklasse Teil 2

```
static <Parameter, Return, Except extends Exception> Function<T, R>
    hide(HideException<Parameter, Return, Except> caller) {
    return caller::callDef;
  }
}
```

Zusammenfassung

- Lambdas(Command Pattern)
- Code Chaining
- Exception handling

Vorträge der MT AG

Dienstag, 21. 11.17

JavaScript Tuning in modernen Web-Applikationen
16.00 Uhr | Foyer Tokio
Till Albert

Mittwoch, 22. 11.17

Ein Snapshot ist kein Backup
8.00 Uhr | Raum Shanghai
Angelina Weinschenk

Pimp my iGrid
15.00 Uhr | Raum Kopenhagen
Moritz Klein

Jetlag: Oracle JET und APEX
9.00 Uhr | Raum Kopenhagen
Oliver Lemm

APEX (Hoch)verfügbar? Darf etwas Open Source sein?
15.00 Uhr | Raum Seoul
Ernst Leber

Mit Augenhöhe und Aufmerksamkeit Projekte zum Erfolg führen
13.00 Uhr | Raum Kiew
Carsten Firus

OWB-ODI Migration: Fallstricke & Lösungen im Praxisbericht
16.00 Uhr | Raum Oslo
Jürgen Günter

Java Script und Offline First
15.00 Uhr | Raum Kopenhagen
Kai Donato

Wieder verschätzt?
17.00 Uhr | Raum Singapur
Oliver Lemm

APEX open Mic Night
20.30 Uhr | Raum Istanbul

Donnerstag, 23. 11.17

Besserer Java Code, außerhalb der Automatismen
9.00 Uhr | Raum Helsinki
Wolfgang Nast

Java 9: Endlich Jigsaw!
10.00 Uhr | Raum Budapest
Salem Ben Nasr

So bringen Sie Ihr DWH Projekt zum Scheitern
12.00 Uhr | Raum Stockholm

Java Script und PL/SQL – das dynamische Duo für APEX
12.00 Uhr | Raum St. Petersburg
Moritz Klein

JSON in Java mit Schema und JsonPath
14.00 Uhr | Raum Oslo

Freitag, 24. 11.17

Ihre Datenbank startet nicht? Oder Anatomie des Startup-Prozesses einer Oracle-Datenbank
09.00 Uhr | NCC
Ernst Leber
Workshop

MT AG

MT AG

data . cloud . mobile

Wolfgang Nast
Telefon: +49 2102 30961 – 0
wolfgang.nast@mt-ag.com