14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Lucky Seven

Kleine Sprachänderungen in Java 7

Michael Wiedeking

MATHEMA Software GmbH

Agenda

- Things we won't talk about
- Things that may come
- Q & A

Things we won't talk about

- Java Modules
- Annotations of Types
- Sorry, no Closures

But: Remember Galadriel's Mirror

A basin filled with water in which one may see visions of the past, present and future, appearing in The Lord of the Rings.

Galadriel, an Elf ruler of Lothlórien, invites the story's hero, Frodo Baggins, to look into it.

Galadriel cannot predict what the mirror will show and does not guarantee that its visions will come to pass.

Fun with Java – Integer Literals

Multiples of 37

```
int[ ] table = {
    000, 037, 074,
    111, 148, 185,
    222, 259, 296,
    333, 370
};
```

Integer Literals

- Before
  - int i = 0xCAFE;
  - int l = 0xBABEL;
- After
  - int i = 0xCAFE_BABE;
  - int j = 0b0110_1100_0111_0101;
  - long l = 123_456_789_012L;
  - byte b = 0xFFu;

Indexing

- Before

```
List<String> list = …;
String firstElement = list.get(0);

Map<Integer, String> map = new HashMap<>();
map.put(Integer.valueOf(1), "One”);
```

- After

```
List<String> list = …;
String firstElement = list[0];

Map<Integer, String> map = new HashMap<>();
map[Integer.valueOf(1)] = "One";
```

Indexing

```
// New superinterfaces to indicate syntax?
interface GettableFromInt<E> {
    E get(int);
}
interface Gettable<V> {
    V get(Object);
}
interface Settable<E> {
    E set(int, E);
}
interface Puttable<K, V> {
    V put(K, V);
}
```

Indexing

java.util.List<E> extends GettableFromInt<E>, Settable<E>, …

java.util.Map<K, V> extends Gettable<V>, Puttable(K, V), …

Collection Literals

- Before

```
List<String> list =
        Collections.asList(new String[ ] {"a", "b", "c”});
```

- After

```
List<String> list = ["a", "b", "c"];
String firstElement = list[0];
Map<Integer, String> map = {1 : "One"};
```

Better Diagnostics

- Before

```
interface List<E> {}
class Test {
    <T> void merge(List<T> l1, List<T> l2) {}
    void test(List<? extends Test> list) {
        merge(list, list);
    }
}
```

```
Test.java:6: <T>merge(List<T>,List<T>) in Test
   cannot
be applied to (List<capture#339 of ? extends
Test>,List<capture#768 of ? extends Test>)
    merge(list, list);
    ^
1 error
```

Better Diagnostics

- After

```
Test.java:6: method merge in class Test cannot be
applied to given types
    merge(list, list);
    ^
  required: List<T>,List<T>
  found: List<CAP#1>,List<CAP#2>
  where T is a type-variable:
    T extends Object declared in method
   <T>merge(List<T>,List<T>)
  where CAP#1,CAP#2 are fresh type-variables:
    CAP#1 extends Test from capture of ? extends Test
    CAP#2 ex1 error
```

Type Literals

- Before

```
Type type = String.class
```

- After

```
Type<List<String>> x = List<String>.class;
```

Type Inference (1/2)

- Before

```
Map<String, List<String>> anagrams =
    new HashMap<String, List<String>>();
```

- After

```
Map<String, List<String>> anagrams =
    new HashMap<>();
```

Type Inference (2/2)

- Before

```
timeWaitsFor(Collections.<Man>emptySet());
```

- After

```
timeWaitsFor(Collections.emptySet());
```

Automatic Resource Block Management

- Before

```
try {
    InputStream in = …
    OutputStream out = …
     ... // Perform action with in and out
} catch (Exception e) {
     in.close();
     out.close();
}
```

Automatic Resource Block Management

- Before

```
try {
    …
} catch (Exception e) {
    if (in != null) {
        in.close();
    }
    if (out != null) {
        out.close();
    }
}
```

Automatic Resource Block Management

- Before

```
if (in != null) {
    try {
        in.close();
    } catch (IOException ioe) {
        // ???
    }
}
if (out != null) {
    // … out …;
}
```

Automatic Resource Block Management

- After

```
// Start a block using two resources, which will
// automatically be cleaned up when the block
// exits scope
do (InputStream in = …; OutputStream out = …)
{
    ... // Perform action with in and out
}
```

Language Level XML Support

- Before

– / –

- After

```
elt.appendChild(
    <muppet>
        <name>Kermit</name>
    </muppet>
);
```

JavaBean Property Support

- Before

```
private String name;
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
```

- After

```
public property String name;
a->Name = b->Name;
```

Operator Overloading

- Before

```
BigDecimal a = new BigDecimal("123…890");
BigDecimal b = …
BigDecimal c = …

a = a.mul(b).add(c);
```

- After

```
BigDecimal a = …
BigDecimal b = …
BigDecimal c = …

b = a * b + c;
```

Strings in switch Statement

- Before

```
static boolean booleanFromString(String s) {
    if (s.equals("true")) {
        return true;
    } else if (s.equals("false")) {
        return false;
    } else {
        throw new IllegalArgumentException(s);
    }
}
```

Strings in switch Statement

- After

```
static boolean booleanFromString(String s) {
    switch(s) {
    case "true":
        return true;
    case "false":
        return false;
    default:
        throw new IllegalArgumentException(s);
    }
}
```

Comparisons for Enums

- Before

```
– / –
```

- After

```
boolean isRoyalty(Rank r) {
    return r >= Rank.JACK && r != Rank.ACE;
}
```

Chained Invocation

- Before

```
Factory factory = new Factory();
factory.setSomething(something);
factory.setOther(other);
Thing thing = factory.result();
```

- After

```
class Factory {
    void setSomething(Something something) { ... }
    void setOther(Other other) { ... }
    Thing result() { ... }
}

Thing thing = new Factory()
    .setSomething(something)
    .setOther(other)
    .result();
```

Extension Methods

- Before

– / –

- After

```
import static java.util.Collections.sort;
List<String> list = ...;
list.sort();    // looks like call to List.sort(), but
really is call to static method Collections.sort().
```

Enhanced Null Handling

- Before

```
public String getPostcode(Person person) {
  if (person != null) {
      Address address = person.getAddress();
      if (address != null) {
        return address.getPostcode();
      }
   }
   return null;
}
```

Enhanced Null Handling

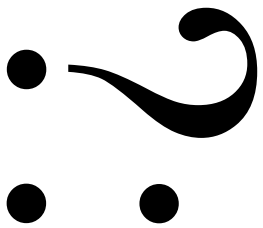
- After

```
public String getPostcode(Person person) {
    return person?.getAddress()?.getPostcode();
}
```

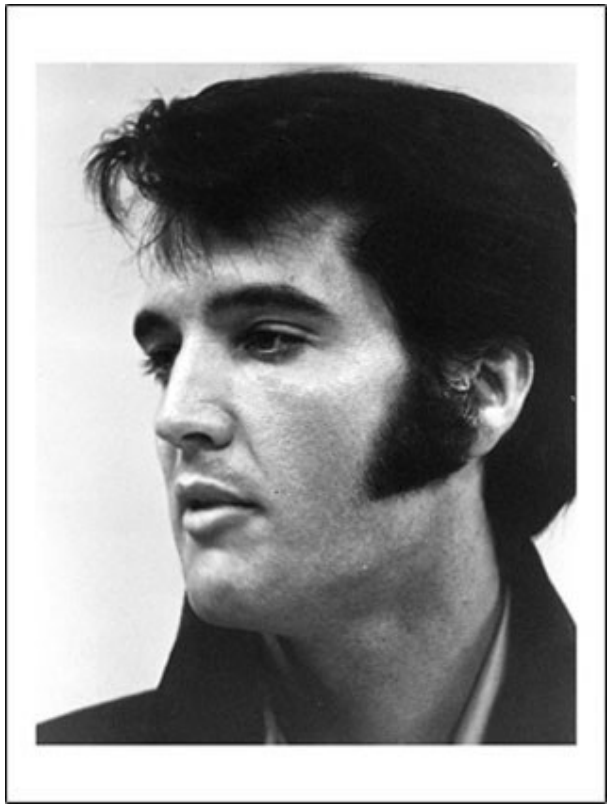
Do you know what this might be?

?:

Maybe this helps?



Do you know this man?



Null Safe Operator (Elvis-Operator)

- Before

```
String s = returnImagePathOrNull(id);
if (s == null) {
    s = “ghost.gif”;
}
```

- After

```
String s = returnImagePathOrNull(id) ?: “ghost.gif”;
```

Enhanced Null Handling

- Before

```
String str = getStringMayBeNull();
str = (str == null ? "" : str);
```

- After

```
String str = getStringMayBeNull() ?: "";
```

Improved catch Clause

- Before

```
try {
    return clazz.newInstance();
} catch (InstantiationException e) {
    throw new AssertionError(e);
} catch (IllegalAccessException e) {
    throw new AssertionError(e);
}
```

Improved catch Clause

- After

```
try {
    return clazz.newInstance();
} catch (InstantiationException | IllegalAccessException e) {
    throw new AssertionError(e);
}
```

Improved catch Clause

- After

```
try {
    doable.doIt();
} catch (final Throwable ex) {
    logger.log(ex);

    // surrounding method can declare only
     // checked Exceptions thrown by doIt()
    throw ex;
}
```

Types and Generics

- Reified Generics — NO
- Type Literals — NO
- JSR 308 Annotations on Java Types — YES
- Type Inference — YES

Language Proposals

- Closures NO
- Automatic Resource Management Blocks NO
- Language level XML support NO
- JavaBean property support NO

Miscellaneous Language

• BigDecimal operator support	NO
• Strings in switch statements	HMM
• Comparisons for Enums	HMM
• Chained invocations	NO?
• Extension methods	NO?
• Improved catch	YES
• Null-safe handling	YES

Thank you!

Q & A

Vielen Dank!

Michael Wiedeking

MATHEMA Software GmbH