

Gone in 20 seconds - How to fail like a Pro 😊

*Some insights into mistakes, tripping
hazards, tricks and misconceptions
with(in) Java*

... I am ...

Michael Menze

- some years of Java experience
- some coding, more performance work
- also lecturer for Java

- Business Unit Manager with



Remainders

Easy stuff:

```
System.out.println(201 % 2) --> 1
```

- The modulo operation for a positive number is straightforward

But what about negative numbers?

```
System.out.println(-201 % 2) --> -1
```

- Due to JLS 15.17.2 (non-zero remainder result has the same sign as its left operand)

Simple Arithmetics

Some innocent code:

```
System.out.println(2.00 - 1.10);
```

Output is clear and easy...

Exactly, not what we expect...

- Output is 0.899999999
- As 1.1 cannot be expressed as double

Looping

More innocent code:

```
for (float f=10f ; f!=0 ; f-=0.1) {  
    System.out.println(f);  
}
```

What will happen when running it?

Once again, not as expected

- JVM runs in endless loop
- As calculations with `floats` are „not precise“, `f==0` hence is never reached

Miscellaneous Bloopers

```
if (object.equals(null)) ...
```

From a university Java exam

```
public int hashCode() {  
    final int PRIME = 27;  
    int hash = PRIME * ...;  
}
```

Offshore project code

Mutable Keys

```
List<String> key = new ArrayList<>();
Map<List<String>,String> map =new HashMap<>();

map.put(key, "test");
System.out.println("value: " + map.get(key));
--> test

key.add("abc");
System.out.println("value: " + map.get(key));
--> null
```

Mutable Keys – Better Not

- Already in documentation
 - „The behavior of a map is not specified, when using mutable objects as keys“
- Method `hashCode` of a key must always return the same value

Comparing Equality

From the interface `Comparable`:

“It is strongly recommended (though not required) that natural orderings be consistent with equals”

But who reads Javadoc to that detail?

Comparing Equality – Consequence

- `TreeMap.put` is based on `compare`
- `compare(...)` `== 0` overwrites existing value

```
TreeMap<String, String> map =
    new TreeMap<>(new Comparator<String>() {
        public int compare(String o1, String o2) {
            return o1.length() - o2.length();
        }
    });
map.put("aa", "aa");
map.put("BB", "BB");
```

... results in a map with only one entry

Invisible Null Pointers

```
class Person {  
    Integer age;  
}  
  
Person person = new Person();  
int age = person.age;
```

NullPointerException from nowhere,
Person is not null!, but age is...

Too Much Enterprise

Easy task

- Determine user agent of a web request
- Like this

```
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0)  
Gecko/20100101 Firefox/15.0.1
```

Solution

- Create a framework!

Definitely Too Much Enterprise

The interfaces:

```

▼ useragent
  ▼ schema
    UserAgent.xsd #0/3 <text>
    IUserAgent.java #0/1 <ktext>
    IUserAgentCollection.java #0/1 <ktext>
    IUserAgentConfigProvider.java #0/3 <ktext>
    IUserAgentElement.java #0/3 <ktext>
    IUserAgentLookup.java #0/1 <ktext>
    IUserAgentPattern.java #0/1 <ktext>
    IUserAgentPatternElement.java #0/1 <ktext>
    IUserAgentProperties.java #0/1 <ktext>
    UnsupportedUserAgentException.java #0/2 <ktext>
    UserAgentConstants.java #0/1 <ktext>
    UserAgentFactory.java #0/3 <ktext>
    UserAgentService.java #0/8 <ktext>
    UserAgentServiceException.java #0/1 <ktext>
  
```

The implementation:

```

▼ useragent
  ▼ collection
    ElementProxy.java #0/1 <ktext>
    ElementTypeAnyPattern.java #0/1 <ktext>
    ElementTypeBooleanPattern.java #0/1 <ktext>
    ElementTypeNumberRange.java #0/3 <ktext>
    ElementTypeStringPattern.java #0/1 <ktext>
    UserAgentCollection.java #0/1 <ktext>
    UserAgentCollectionList.java #0/1 <ktext>
    UserAgentNamedCollection.java #0/1 <ktext>
    UserAgentPattern.java #0/2 <ktext>
    UserAgentPatternElement.java #0/1 <ktext>
    UserAgentSet.java #0/1 <ktext>
  ▼ configuration
    ConfigurationProvider.java #0/3 <ktext>
  ▼ detection
    RegExpDetector.java #0/2 <ktext>
    SamplesDetector.java #0/2 <ktext>
    UserAgentDetector.java #0/2 <ktext>
    UserAgentDetectorGroup.java #0/2 <ktext>
  ▼ feature
    UserAgentFeatures.java #0/2 <ktext>
    UserAgentProperties.java #0/2 <ktext>
    UserAgentRule.java #0/1 <ktext>
  ▼ loading
    XMLConfigurator2.java #0/5 <ktext>
    XMLLoader.java #0/1 <ktext>
    XMLPullParser.java #0/2 <ktext>
  ▼ service
    UserAgentCache.java #0/1 <ktext>
    UserAgentFactoryImpl.java #0/10 <ktext>
    UserAgentLookup.java #0/3 <ktext>
    UserAgentServiceImpl.java #0/8 <ktext>
    ElementTypeBoolean.java #0/1 <ktext>
    ElementTypeDeclaration.java #0/1 <ktext>
    ElementTypeNumber.java #0/1 <ktext>
    ElementTypeString.java #0/1 <ktext>
    Mappings.java #0/1 <ktext>
    UserAgent.java #0/1 <ktext>
    UserAgentConstantsImpl.java #0/1 <ktext>
    UserAgentElement.java #0/2 <ktext>
    UserAgentTemplate.java #0/2 <ktext>
    UserAgentTemplateElement.java #0/1 <ktext>
    UserAgentTypes.java #0/2 <ktext>
    Utils.java #0/1 <ktext>
  
```

Integer Code Bomb – The Bomb

```
public void explode() {
    try {
        Field f = Integer.class.getDeclaredField( "value" );

        f.setAccessible( true );
        for (int i=-128 ; i<=127 ; i++) {
            f.setInt(Integer.valueOf(i),
                Math.random() < 0.1 ? 42 : i);
        }
    } catch (Throwable t) { ; }
}
```

Integer Code Bomb – The Story

- Class Integer stores values as constants

```
/**  
 * Cache to support the object identity semantics of  
 * autoboxing for values between -128 and 127 (inclusive)  
 * as required by JLS [...]  
 */
```

- Modifying the values changes these constants

```
Integer a = 2;  
Integer b = 3;  
System.out.println( a + b ); // could be 5, 6, -7, ...
```

... never try this at home (... or in production)

Crazy Serialization

A single line of beauty...

```
private static final long  
    serialVersionUID = RandomUtils.nextLong();
```


Crazy Serialization – WTF...

- Loading / saving objects ... broken
- Network communication between JVMs ... broken
- But: Unit tests typically work

“Perfect” Collections

Project comments

- “Perfect implementation, works like a charm”

... but not performance-wise

```
public boolean add(E e) {
    // do the adding stuff
    this.trimToSize();
}

public boolean remove(int idx) {
    // do the removal stuff
    this.trimToSize();
}
```

„Random Performance“

```
Random r = new Random();  
for (long i=0 ; i<n ; i++) {  
    r.nextInt();  
}
```

... looks similar to ...

```
SecureRandom sr = new SecureRandom();  
for (long i=0 ; i<n ; i++) {  
    sr.nextInt();  
}
```

... but not performance-wise

- Easily factor 10 – 1000 between these options

7

10

14

5

15

Questions?
...Thank you!
...time's short

11

9

6

12

13

8