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 straight-forward

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)

Source: Java Puzzlers, JLS

Simple Arithmetics

Some innocent code:

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

Output is clear and easy…

Exactly, not what we expect…

- Output is `0.89999999`
- As `1.1` cannot be expressed as `double`

Source: Java Puzzlers, JLS

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 `float`s are „not precise“, `f==0` hence is never reached

Miscalleneous 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

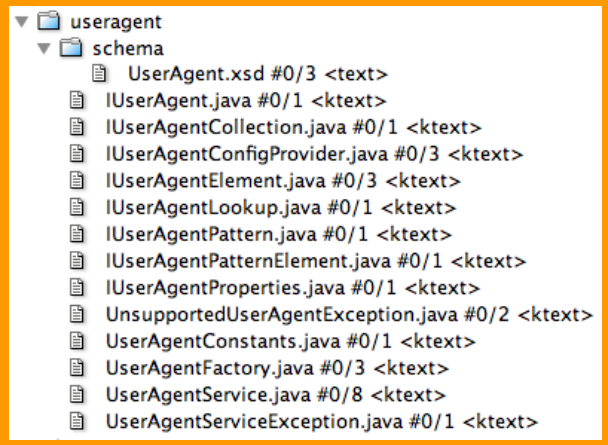
```
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0)
Gecko/20100101 Firefox/15.0.1
```

Solution

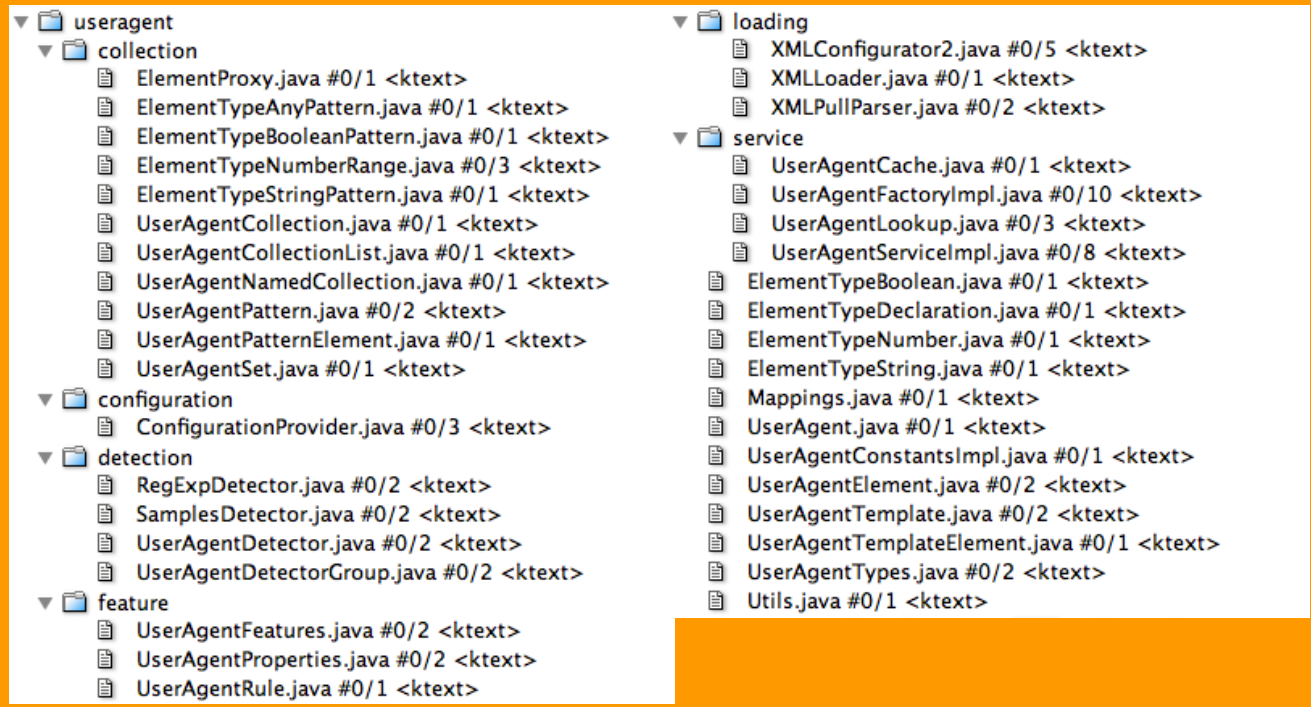
- Create a framework!

Definitely Too Much Enterprise

The interfaces:



The implementation:



Source: http://thedailywtf.com/Articles/The-Enterprise-User-Agent.aspx

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)

Source: http://thedailywtf.com/Articles/Disgruntled-Bomb-Java-Edition.aspx

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

Source: http://thedailywtf.com/Articles/Serializalicious.aspx

“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