

Java Wars

Episode IV: Don't lose hope

Werner Eberling

werner.eberling@mathema.de

Oliver Szymanski

oliver.szymanski@mathema.de

www.mathema.de

MATHEMA

Change

```
public class Change {  
  
    public static void main(String[] args) {  
        System.out.println(2.00 - 1.10);  
    }  
}
```

Change

```
public class Change {  
  
    public static void main(String[] args) {  
        System.out.println(2.00 - 1.10);  
    }  
}
```

- ✓ Compiler-Fehler
- ✓ 0.90
- ✓ 0.9
- ✓ 0.8999999999999999

Change

```
public class Change {  
  
    public static void main(String[] args) {  
        System.out.println(2.00 - 1.10);  
    }  
}
```

- x Compiler-Fehler
- x 0.90 (1.10 gibt's als double nicht)
- x 0.9
- ✓ 0.8999999999999999

Change: besser

```
public class Change {  
  
    public static void main(String[] args) {  
        System.out.println(2.00 - 1.10);  
        System.out.printf("%.2f%n", 2.00-1.10);  
    }  
}
```

Change: noch besser

```
public class Change {  
  
    public static void main(String[] args) {  
        System.out.println(2.00 - 1.10);  
        // rechnet immer noch falsch!  
        System.out.printf("%.2f%n", 2.00-1.10);  
  
        System.out.println(  
            new BigDecimal(2.00).subtract(  
                new BigDecimal(1.10)));  
    }  
}
```

Change: jetzt aber - oder?

```
public class Change {  
  
    public static void main(String[] args) {  
  
        System.out.println(  
            new BigDecimal("2.00").subtract(  
            new BigDecimal("1.10")));  
    }  
}
```

Conditioner

```
public class DosEquis {  
  
    public static void main(String[] args) {  
        char x = 'X';  
        int i = 0;  
        System.out.println(true ? x : 0);  
        System.out.println(false ? i : x);  
    }  
}
```

Conditioner

```
public class DosEquis {  
  
    public static void main(String[] args) {  
        char x = 'X';  
        int i = 0;  
        System.out.println(true ? x : 0);  
        System.out.println(false ? i : x);  
    }  
}
```

- ✓ 'X' \n 'X'
- ✓ Kompilierfehler
- ✓ Nichts davon
- ✓ 0 \n 0

Conditioner

```
public class DosEquis {  
  
    public static void main(String[] args) {  
        char x = 'X';  
        int i = 0;  
        System.out.println(true ? x : 0);  
        System.out.println(false ? i : x);  
    }  
}
```

```
x 'X' \n 'X'  
x Kompilierfehler  
✓ 'X' \n 88  
x 0 \n 0
```

Calculate

```
public class Calculate {  
  
    public static void main (String[] args) {  
        short x = 0;  
        int i = 123456;  
  
        x+=i;  
        x = x + i;  
    }  
}
```

Calculate

```
public class Calculate {  
  
    public static void main (String[] args) {  
        short x = 0;  
        int i = 123456;  
  
        x+=i;  
        x = 0;  
        x = x + i;  
    }  
}
```

- ✓ 123456
- ✓ -7616
- ✓ Kompilierfehler in Zeile 3
- ✓ Kompilierfehler in Zeile 5

Calculate

```
public class Calculate {  
  
    public static void main (String[] args) {  
        short x = 0;  
        int i = 123456;  
  
        x+=i;  
        x = 0;  
        x = x + i;  
    }  
}
```

- x 123456
- x -7616
- x Kompilierfehler in Zeile 3
- ✓ Kompilierfehler in Zeile 5

Browstertest

```
public class BrowserTest {  
    public static void main(String[] args) {  
        System.out.println(":iexplore:");  
        http://google.com  
        System.out.println(":maximize:");  
    }  
}
```

Browserstest

```
public class BrowserTest {  
    public static void main(String[] args) {  
        System.out.println(":iexplore:");  
        http://google.com  
        System.out.println(":maximize:");  
    }  
}
```

- ✓ :iexplore: \n :maximize:
- ✓ Kompilierfehler
- ✓ Browser wird geöffnet (Windows)
- ✓ Runtimefehler (!Windows)

Browstertest

```
public class BrowserTest {  
    public static void main(String[] args) {  
        System.out.println(":iexplore:");  
        http://google.com  
        System.out.println(":maximize:");  
    }  
}
```

- ✓ :iexplore: \n :maximize:
- x Kompilierfehler
- x Browser wird geöffnet (Windows)
- x Runtimefehler (!Windows)

Finally – or not?

```
static int test() throws Exception {  
    try {  
        return 1;  
    } catch (Exception e) {  
        return 2;  
    } finally {  
        return 3;  
    }  
}
```

Finally – or not?

```
static int test() throws Exception {  
    try {  
        return 1;  
    } catch (Exception e) {  
        return 2;  
    } finally {  
        return 3;  
    }  
}
```

- ✓ Kompilierfehler
- ✓ 1
- ✓ 2
- ✓ 3

Moment, erst mal mit Exceptions

```
static void test() throws Exception {  
    try {  
        throw new Exception("1");  
    } catch (Exception e) {  
        throw new Exception("2");  
    } finally {  
        throw new Exception("3");  
    }  
}
```

Moment, erst mal mit Exceptions

```
static void test() throws Exception {  
    try {  
        throw new Exception("1");  
    } catch (Exception e) {  
        throw new Exception("2");  
    } finally {  
        throw new Exception("3");  
    }  
}
```

- ✓ Kompilierfehler
- ✓ 1
- ✓ 2
- ✓ 3

Moment, erst mal mit Exceptions

```
static void test() throws Exception {  
    try {  
        throw new Exception("1");  
    } catch (Exception e) {  
        throw new Exception("2");  
    } finally {  
        throw new Exception("3");  
    }  
}
```

x Kompilierfehler
x 1
x 2
✓ 3

Finally – or not?

```
static int test() throws Exception {  
    try {  
        return 1;  
    } catch (Exception e) {  
        return 2;  
    } finally {  
        return 3;  
    }  
}
```

- x Kompilierfehler
- x 1
- x 2
- ✓ 3

Catch & Co.

```
public static void main (String[] args) {  
    try {  
        System.out.println("hello world");  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

Catch & Co.

```
public static void main (String[] args) {  
    try {  
        System.out.println("hello world");  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

- ✓ Laufzeitfehler
- ✓ „hello world“
- ✓ Kompilierfehler
- ✓ Nichts davon

Catch & Co.

```
public static void main (String[] args) {  
    try {  
        System.out.println("hello world");  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

- x Laufzeitfehler
- x „hello world“
- ✓ Kompilierfehler
- x Nichts davon

More Catch & Co.

```
public static void main (String[] args) {  
    try {  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

More Catch & Co.

```
public static void main (String[] args) {  
    try {  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

- ✓ Laufzeitfehler
- ✓ Nichts
- ✓ Kompilierfehler
- ✓ „hello world“

More Catch & Co.

```
public static void main (String[] args) {  
    try {  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

- × Laufzeitfehler
- ✓ Nichts
- × Kompilierfehler
- × „hello world“

Was müssen wir fangen?

```
public class Exceptions {  
  
    interface I1 { void f() throws IOException; }  
    interface I2 { void f() throws InterruptedException; }  
    interface I3 extends I1, I2 { }  
  
    static public class T implements I3 {  
        public void f() { System.out.println("hello world"); }  
    }  
  
    public static void main (String[] args) {  
        (new T()).f();  
    }  
}
```

Was müssen wir fangen?

```
public class Exceptions {  
  
    interface I1 { void f() throws IOException; }  
    interface I2 { void f() throws InterruptedException; }  
    interface I3 extends I1, I2 { }  
  
    static public class T implements I3 {  
        public void f() { System.out.println("hello world"); }  
    }  
  
    public static void main (String[] args) {  
        (new T()).f();  
    }  
}
```

- ✓ Passt schon
- ✓ Kompiliert nicht
- ✓ Nichts davon
- ✓ „hello world“

Was müssen wir fangen?

```
public class Exceptions {  
  
    interface I1 { void f() throws IOException; }  
    interface I2 { void f() throws InterruptedException; }  
    interface I3 extends I1, I2 { }  
  
    static public class T implements I3 {  
        public void f() { System.out.println("hello world"); }  
    }  
  
    public static void main (String[] args) {  
        (new T()).f();  
    }  
}
```

- ✓ Passt schon
- × Kompiliert nicht
- × Nichts davon
- ✓ „hello world“

Copy und IO im Weitesten

```
public static void main(String[] args) throws IOException {  
    if (args == null || args.length != 2) return;  
    InputStream in = null;  
    OutputStream out = null;  
    try {  
        in = new FileInputStream(args[0]);  
        out = new FileOutputStream(args[1]);  
        byte[] buf = new byte[1024];  
        int n;  
        while ((n = in.read(buf)) >= 0) {  
            out.write(buf, 0, n);  
        }  
    } finally {  
        in.close();  
        out.close();  
    }  
}
```

Copy und IO im Weitesten

```
public static void main(String[] args) throws IOException {
    if (args == null || args.length != 2) return;
    InputStream in = null;
    OutputStream out = null;
    try {
        in = new FileInputStream(args[0]);
        out = new FileOutputStream(args[1]);
        byte[] buf = new byte[1024];
        int n;
        while ((n = in.read(buf)) >= 0) {
            out.write(buf, 0, n);
        }
    } finally {
        in.close();
        out.close();
    }
}
```

- ✓ Das würde nie kopieren
- ✓ Alles prima
- ✓ Kein throws bei „main“
- ✓ Geht manchmal

Copy und IO im Weitesten

```
public static void main(String[] args) throws IOException {
    if (args == null || args.length != 2) return;
    InputStream in = null;
    OutputStream out = null;
    try {
        in = new FileInputStream(args[0]);
        out = new FileOutputStream(args[1]);
        byte[] buf = new byte[1024];
        int n;
        while ((n = in.read(buf)) >= 0) {
            out.write(buf, 0, n);
        }
    } finally {
        in.close();
        out.close();
    }
}
```

- × Das würde nie kopieren
- × Alles prima
- × Kein throws bei „main“
- ✓ Geht manchmal

Mind the Gap

```
public class Gap {
    private static final int GAP_SIZE = 10 * 1024;

    public static void main(String args[]) throws IOException {
        File tmp = File.createTempFile("gap", ".txt");
        OutputStream out = new BufferedOutputStream(
            new FileOutputStream(tmp));

        out.write(1);
        out.write(new byte[GAP_SIZE]);
        out.write(2);
        out.close();
        InputStream in = new BufferedInputStream(
            new FileInputStream(tmp));

        int first = in.read();
        in.skip(GAP_SIZE);
        int last = in.read();
        System.out.println(first + last);
    }
}
```

Mind the Gap

```
public class Gap {
    private static final int GAP_SIZE = 10 * 1024;

    public static void main(String args[]) throws IOException {
        File tmp = File.createTempFile("gap", ".txt");
        OutputStream out = new BufferedOutputStream(
            new FileOutputStream(tmp));

        out.write(1);
        out.write(new byte[GAP_SIZE]);
        out.write(2);
        out.close();
        InputStream in = new BufferedInputStream(
            new FileInputStream(tmp));

        int first = in.read();
        in.skip(GAP_SIZE);
        int last = in.read();
        System.out.println(first + last);
    }
}
```

- ✓ 1
- ✓ 3
- ✓ 12
- ✓ Mal so mal so

Mind the Gap

```
public class Gap {
    private static final int GAP_SIZE = 10 * 1024;

    public static void main(String args[]) throws IOException {
        File tmp = File.createTempFile("gap", ".txt");
        OutputStream out = new BufferedOutputStream(
            new FileOutputStream(tmp));

        out.write(1);
        out.write(new byte[GAP_SIZE]);
        out.write(2);
        out.close();
        InputStream in = new BufferedInputStream(
            new FileInputStream(tmp));

        int first = in.read();
        in.skip(GAP_SIZE);
        int last = in.read();
        System.out.println(first + last);
    }
}
```

- ✓ 1 (praktisch)
- × 3
- × 12
- ✓ Mal so mal so (Spek)

Oddity

```
public class Oddity {  
    public static void main(String[] args) {  
        System.out.println(isOdd(3));  
    }  
    public static boolean isOdd(int i) {  
        return i % 2 == 1;  
    }  
}
```

Oddity

```
public class Oddity {  
    public static void main(String[] args) {  
        System.out.println(isOdd(3));  
    }  
    public static boolean isOdd(int i) {  
        return i % 2 == 1;  
    }  
}
```

- ✓ Nie
- ✓ Immer
- ✓ In 50% der möglichen Eingabe
- ✓ In 75% der möglichen Eingaben

Oddity

```
public class Oddity {  
    public static void main(String[] args) {  
        System.out.println(isOdd(3));  
    }  
    public static boolean isOdd(int i) {  
        return i % 2 == 1;  
    }  
}
```

- Nie
- Immer
- In 50% der möglichen Eingabe
- In 75% der möglichen Eingaben

Long Division

```
public class LongDivision {  
    public static void main(String[] args) {  
        final long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;  
        final long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;  
        System.out.println(MICROS_PER_DAY / MILLIS_PER_DAY);  
    }  
}
```

Long Division

```
public class LongDivision {  
    public static void main(String[] args) {  
        final long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;  
        final long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;  
        System.out.println(MICROS_PER_DAY / MILLIS_PER_DAY);  
    }  
}
```

- ✓ 1000
- ✓ Exception
- ✓ 5
- ✓ Long.MIN_VALUE

Long Division

```
public class LongDivision {  
    public static void main(String[] args) {  
        final long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;  
        final long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;  
        System.out.println(MICROS_PER_DAY / MILLIS_PER_DAY);  
    }  
}
```

- x 1000
- x Exception
- ✓ 5
- x Long.MIN_VALUE

Elementary

```
public class Elementary {  
    public static void main(String[] args) {  
        System.out.println(12345 + 54321);  
    }  
}
```

Elementary

```
public class Elementary {  
    public static void main(String[] args) {  
        System.out.println(12345 + 54321);  
    }  
}
```

- ✓ 66666
- ✓ 17777
- ✓ -123
- ✓ Nichts davon

Elementary

```
public class Elementary {  
    public static void main(String[] args) {  
        System.out.println(12345 + 54321);  
    }  
}
```

- x 66666
- ✓ 17777
- x -123
- x Nichts davon

Line Printer

```
public class LinePrinter {  
  
    public static void main(String[] args) {  
        // Note: \u000A is Unicode representation of linefeed (LF)  
        char c = 0x000A;  
        System.out.println(c);  
    }  
}
```

Line Printer

```
public class LinePrinter {  
  
    public static void main(String[] args) {  
        // Note: \u000A is Unicode representation of linefeed (LF)  
        char c = 0x000A;  
        System.out.println(c);  
    }  
}
```

- ✓ <Leerzeile>
- ✓ "0x000A"
- ✓ "10"
- ✓ Nichts davon

Line Printer

```
public class LinePrinter {  
  
    public static void main(String[] args) {  
        // Note: \u000A is Unicode representation of linefeed (LF)  
        char c = 0x000A;  
        System.out.println(c);  
    }  
}
```

- x <Leerzeile>
- x "0x000A"
- x "10"
- ✓ Nichts davon

Hello World (?)

```
\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020\u0020\u0020
\u0063\u006c\u0061\u0073\u0073\u0020\u0055\u0067\u006c\u0079
\u007b\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020\u0020
\u0020\u0020\u0020\u0020\u0020\u0073\u0074\u0061\u0074\u0069\u0063
\u0076\u0066\u0069\u0064\u0020\u0064\u0061\u0069\u006e\u0028
\u0053\u0074\u0072\u0069\u006e\u0067\u005b\u005d\u0020\u0020
\u0020\u0020\u0020\u0020\u0020\u0061\u0072\u0067\u0073\u0029\u007b
\u0053\u0079\u0073\u0074\u0065\u0064\u002e\u0066\u0075\u0074
\u002e\u0070\u0072\u0069\u006e\u0074\u006c\u006e\u0028\u0020
\u0022\u0048\u0065\u006c\u006c\u0066\u0020\u0077\u0022\u002b
\u0022\u0066\u0072\u006c\u0064\u0022\u0029\u003b\u007d\u007d
```

Hello World (?)

```
\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020\u0020
\u0063\u006c\u0061\u0073\u0073\u0020\u0055\u0067\u006c\u0079
\u007b\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020
\u0020\u0020\u0020\u0020\u0073\u0074\u0061\u0074\u0069\u0069\u0063
\u0076\u0066\u0069\u0064\u0020\u0064\u0061\u0069\u006e\u0028
\u0053\u0074\u0072\u0069\u006e\u0067\u005b\u005d\u0020\u0020
\u0020\u0020\u0020\u0020\u0061\u0072\u0067\u0073\u0029\u007b
\u0053\u0079\u0073\u0074\u0065\u0064\u002e\u0066\u0075\u0074
\u002e\u0070\u0072\u0069\u006e\u0074\u006c\u006e\u002
\u0022\u0048\u0065\u006c\u006c\u0066\u0020\u0077\u002
\u0022\u0066\u0072\u006c\u0064\u0022\u0029\u003b\u007
```

- ✓ kompiliert nicht
- ✓ Schreibt "Hello world"
- ✓ wirft Exception
- ✓ nichts davon

Hello World (?)

```
\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020\u0020\u0020
\u0063\u006c\u0061\u0073\u0073\u0020\u0055\u0067\u006c\u0079
\u007b\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020\u0020
\u0020\u0020\u0020\u0020\u0020\u0073\u0074\u0061\u0074\u0069\u0063
\u0076\u0066\u0069\u0064\u0020\u0064\u0061\u0069\u006e\u0028
\u0053\u0074\u0072\u0069\u006e\u0067\u005b\u005d\u0020\u0020
\u0020\u0020\u0020\u0020\u0020\u0061\u0072\u0067\u0073\u0029\u007b
\u0053\u0079\u0073\u0074\u0065\u0064\u002e\u0066\u0075\u0074
\u002e\u0070\u0072\u0069\u006e\u0074\u006c\u006e\u002
\u0022\u0048\u0065\u006c\u006c\u0066\u0020\u0077\u002
\u0022\u0066\u0072\u006c\u0064\u0022\u0029\u003b\u007
```

- x kompiliert nicht
- ✓ Schreibt "Hello world"
- x wirft Exception
- x nichts davon

Pain, Gain, Main

```
public class PGMain {  
    private static Random rnd = new Random();  
    public static void main(String[] args) {  
        StringBuffer word = null;  
        switch (rnd.nextInt(2)) {  
            case 1: word = new StringBuffer('P');  
            case 2: word = new StringBuffer('G');  
            default: word = new StringBuffer('M');  
        }  
        word.append('a');  
        word.append('i');  
        word.append('n');  
        System.out.println(word);  
    }  
}
```

Pain, Gain, Main

```
public class PGMain {  
    private static Random rnd = new Random();  
    public static void main(String[] args) {  
        StringBuffer word = null;  
        switch (rnd.nextInt(2)) {  
            case 1: word = new StringBuffer('P');  
            case 2: word = new StringBuffer('G');  
            default: word = new StringBuffer('M');  
        }  
        word.append('a');  
        word.append('i');  
        word.append('n');  
        System.out.println(word);  
    }  
}
```

- ✓ immer "Pain"
- ✓ immer "Gain"
- ✓ immer "Main"
- ✓ Etwas anderes

Pain, Gain, Main

```
public class PGMain {  
    private static Random rnd = new Random();  
    public static void main(String[] args) {  
        StringBuffer word = null;  
        switch (rnd.nextInt(2)) {  
            case 1: word = new StringBuffer('P');  
            case 2: word = new StringBuffer('G');  
            default: word = new StringBuffer('M');  
        }  
        word.append('a');  
        word.append('i');  
        word.append('n');  
        System.out.println(word);  
    }  
}
```

- x immer "Pain"
- x immer "Gain"
- x immer "Main"
- ✓ Etwas anderes

Looper

```
public class Looper {  
    public static void main(String[] args) {  
        // i;  
        // j;  
        while (i <= j && j <= i && i != j) {  
            // ...  
        }  
    }  
}
```

Looper

```
public class Looper {  
    public static void main(String[] args) {  
        // i;  
        // j;  
        while (i <= j && j <= i && i != j) {  
            // ...  
        }  
    }  
}
```

x i?
x j?

Looper

```
public class Looper {  
    public static void main(String[] args) {  
        // i;  
        // j;  
        while (i <= j && j <= i && i != j) {  
            // ...  
        }  
    }  
}
```

- ✓ Integer i = new Integer(0);
- ✓ Integer j = new Integer(0);

Dogs Life (I)

```
public class Pet {
    public final String name;
    public final String food;
    public final String sound;
    public Pet(String name, String food, String sound) {
        this.name = name;
        this.food = food;
        this.sound = sound;
    }
    public void eat() {
        System.out.println(name + ": Mmmmm, " + food);
    }
    public void play() {
        System.out.println(name + ": " + sound + " " + sound);
    }
    public void sleep() {
        System.out.println(name + ": Zzzzzzz...");
    }
    ...
}
```

Dogs Life (II)

```
...
public void live() {
    new Thread() {
        public void run() {
            while (true) {
                eat();
                play();
                sleep();
            }
        }
    }.start();
}

public static void main(String[] args) {
    new Pet("Fido", "beef", "Woof").live();
}

}
```

Dogs Life (II)

```
...  
public void live() {  
    new Thread() {  
        public void run() {  
            while (true) {  
                eat();  
                play();  
                sleep();  
            }  
        }  
    }.start();  
}
```

pu

}

}

- ✓ terminiert ohne Ausgabe
- ✓ "Fido: Mmmmm beef\nFido: Woof Woof\nFido: Zzzzzzz..."...
- ✓ Exception
- ✓ Kompiliert nicht

Dogs Life (II)

```
...
public void live() {
    new Thread() {
        public void run() {
            while (true) {
                eat();
                play();
                sleep();
            }
        }
    }.start();
}
}
}
}
```

- x terminiert ohne Ausgabe
- x "Fido: Mmmmm beef\nFido: Woof Woof\nFido: Zzzzzzz..."...
- x Exception
- ✓ Kompiliert nicht

Static Init

```
public class StaticInit {
    private static boolean init = false;
    static {
        Thread t = new Thread(new Runnable() {
            public void run() {
                init = true;
            }
        });
        t.start();
        try {
            t.join();
        } catch (InterruptedException e) {
            throw new AssertionError(e);
        }
    }

    public static void main(String[] args) {
        System.out.println(init);
    }
}
```

Static Init

```
public class StaticInit {
    private static boolean init = false;
    static {
        Thread t = new Thread(new Runnable() {
            public void run() {
                init = true;
            }
        });
        t.start();
        try {
            t.join();
        } catch (InterruptedException e) {
            throw new AssertionError(e);
        }
    }

    public static void main(String[] args) {
        System.out.println(init);
    }
}
```

- ✓ AssertionError
- ✓ true
- ✓ true/false
- ✓ Nichts davon

Static Init

```
public class StaticInit {
    private static boolean init = false;
    static {
        Thread t = new Thread(new Runnable() {
            public void run() {
                init = true;
            }
        });
        t.start();
        try {
            t.join();
        } catch (InterruptedException e) {
            throw new AssertionError(e);
        }
    }

    public static void main(String[] args) {
        System.out.println(init);
    }
}
```

- x AssertionError
- x true
- x true/false
- ✓ Nichts davon

Twisted

```
public class Twisted {
    private final String name;

    Twisted(String name) {
        this.name = name;
    }
    private String name() {
        return name;
    }
    private void reproduce() {
        new Twisted("reproduce") {
            void printName() {
                System.out.println(name());
            }
        }.printName();
    }
    public static void main(String[] args) {
        new Twisted("main").reproduce();
    }
}
```

Twisted

```
public class Twisted {
    private final String name;

    Twisted(String name) {
        this.name = name;
    }
    private String name() {
        return name;
    }
    private void reproduce() {
        new Twisted("reproduce") {
            void printName() {
                System.out.println(name());
            }
        }.printName();
    }
    public static void main(String[] args) {
        new Twisted("main").reproduce();
    }
}
```

- ✓ Compilefehler
- ✓ "reproduce"
- ✓ "main"
- ✓ nichts davon

Twisted

```
public class Twisted {
    private final String name;

    Twisted(String name) {
        this.name = name;
    }
    private String name() {
        return name;
    }
    private void reproduce() {
        new Twisted("reproduce") {
            void printName() {
                System.out.println(name());
            }
        }.printName();
    }
    public static void main(String[] args) {
        new Twisted("main").reproduce();
    }
}
```

- × Compilefehler
- × "reproduce"
- ✓ "main"
- × nichts davon

ShortSet

```
public class ShortSet {
    public static void main(String args[]) {
        Set<Short> s = new HashSet<Short>();
        for (short i = 0; i < 100; i++) {
            s.add(i);
            s.remove(i - 1);
        }
        System.out.println(s.size());
    }
}
```

ShortSet

```
public class ShortSet {  
    public static void main(String args[]) {  
        Set<Short> s = new HashSet<Short>();  
        for (short i = 0; i < 100; i++) {  
            s.add(i);  
            s.remove(i - 1);  
        }  
        System.out.println(s.size());  
    }  
}
```

- ✓ 1
- ✓ 100
- ✓ Exception
- ✓ Nichts davon

ShortSet

```
public class ShortSet {  
    public static void main(String args[]) {  
        Set<Short> s = new HashSet<Short>();  
        for (short i = 0; i < 100; i++) {  
            s.add(i);  
            s.remove(i - 1);  
        }  
        System.out.println(s.size());  
    }  
}
```

- x 1
- ✓ 100
- x Exception
- x Nichts davon

UrlSet

```
public class UrlSet {
    private static final String[]
        URL_NAMES = { "http://javapuzzlers.com",
                    "http://apache2-snort.skybar.dreamhost.com",
                    "http://www.google.com",
                    "http://javapuzzlers.com",
                    "http://findbugs.sourceforge.net",
                    "http://www.cs.umd.edu" };

    public static void main(String[] args)
        throws MalformedURLException {
        Set<URL> favorites = new HashSet<URL>();
        for (String urlName : URL_NAMES)
            favorites.add(new URL(urlName));
        System.out.println(favorites.size());
    }
}
```

UrlSet

```
public class UrlSet {
    private static final String[]
        URL_NAMES = { "http://javapuzzlers.com",
                    "http://apache2-snort.skybar.dreamhost.com",
                    "http://www.google.com",
                    "http://javapuzzlers.com",
                    "http://findbugs.sourceforge.net",
                    "http://www.cs.umd.edu" };

    public static void main(String[] args)
        throws MalformedURLException {
        Set<URL> favorites = new HashSet<URL>();
        for (String urlName : URL_NAMES)
            favorites.add(new URL(urlName));
        System.out.println(favorites.size());
    }
}
```

- ✓ 4
- ✓ 5
- ✓ 6
- ✓ Nichts davon

UrlSet

```
public class UrlSet {
    private static final String[]
        URL_NAMES = { "http://javapuzzlers.com",
                    "http://apache2-snort.skybar.dreamhost.com",
                    "http://www.google.com",
                    "http://javapuzzlers.com",
                    "http://findbugs.sourceforge.net",
                    "http://www.cs.umd.edu" };

    public static void main(String[] args)
        throws MalformedURLException {
        Set<URL> favorites = new HashSet<URL>();
        for (String urlName : URL_NAMES)
            favorites.add(new URL(urlName));
        System.out.println(favorites.size());
    }
}
```

- ✓ 4 (mit Internet)
- ✓ 5 (ohne Internet)
- × 6
- ✓ Nichts davon

Thread-Test

```
public class Test extends junit.framework.TestCase {
    int number;

    public void test() throws InterruptedException {
        number = 0;
        Thread t = new Thread(new Runnable() {
            public void run() {
                assertEquals(2, number);
            }
        });
        number = 1;
        t.start();
        number++;
        t.join();
    }
}
```

Thread-Test

```
public class Test extends junit.framework.TestCase {
    int number;

    public void test() throws InterruptedException {
        number = 0;
        Thread t = new Thread(new Runnable() {
            public void run() {
                assertEquals(2, number);
            }
        });
        number = 1;
        t.start();
        number++;
        t.join();
    }
}
```

- ✓ immer rot
- ✓ immer grün
- ✓ mal grün/mal rot
- ✓ hängt

Thread-Test

```
public class Test extends junit.framework.TestCase {
    int number;

    public void test() throws InterruptedException {
        number = 0;
        Thread t = new Thread(new Runnable() {
            public void run() {
                assertEquals(2, number);
            }
        });
        number = 1;
        t.start();
        number++;
        t.join();
    }
}
```

- x immer rot
- ✓ immer grün
- x mal grün/mal rot
- x hängt

Ich hoffe, es hat a bisserl Spaß gemacht ,)

Vielen Dank!

werner.eberling@mathema.de

oliver.szymanski@mathema.de