Ausgabe 01/2021

Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

D: 4,90 EUR | CH: 9,80 CHF | A: 5,60 EUR | Benelux: 5,80 EUR

4 191978 304903 01

Java aktuell

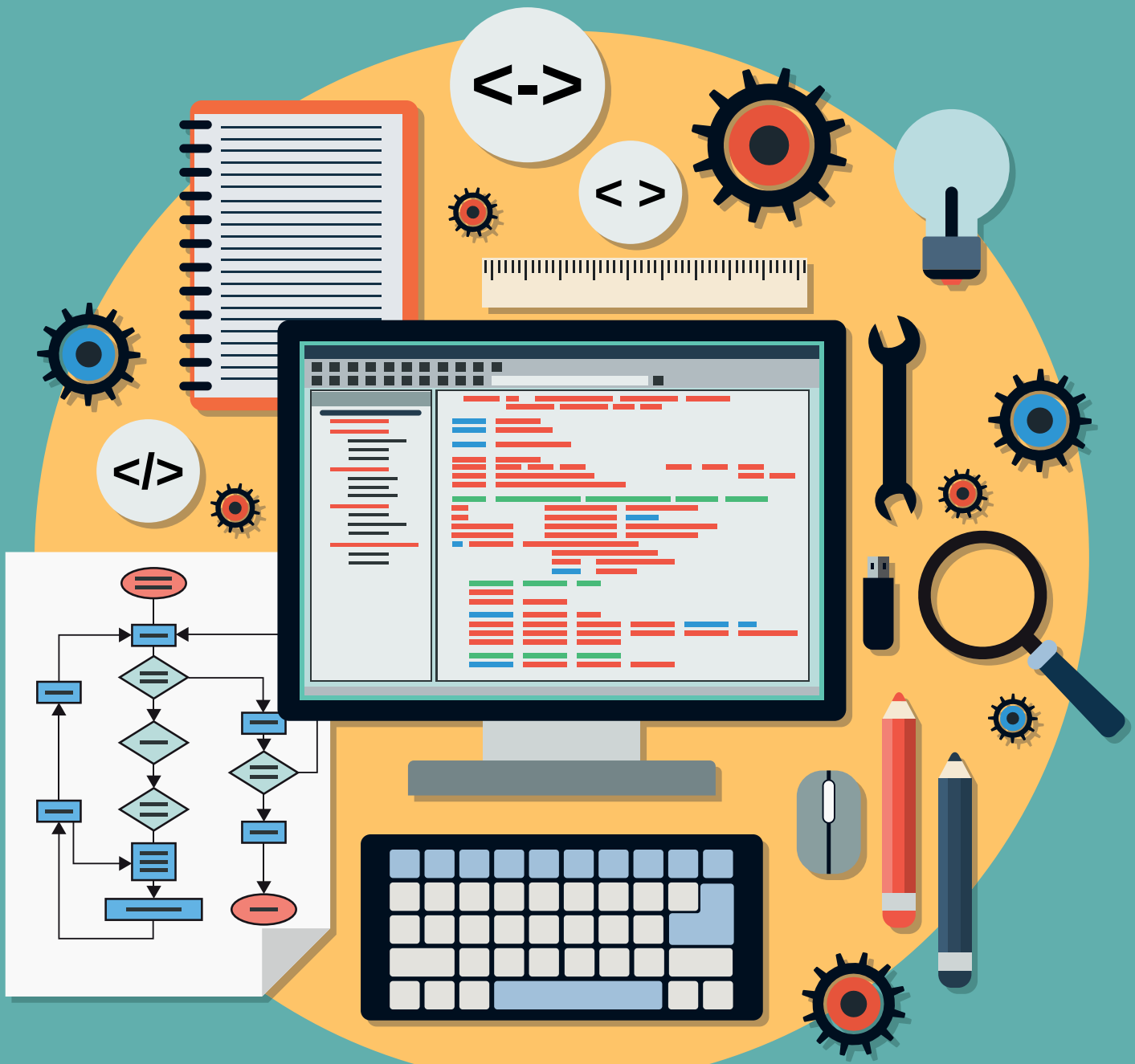
iJUG Verbund
www.ijug.eu

Java 15
Die Features im Überblick

Java-Libraries
fritz2, Vavr und Kotlin Arrow

Spring
Spring Data JPA, Spring HATEOAS und Spring mit React

WERKZEUGE
Tools & Frameworks



iJUG Verbund

Werden Sie Mitglied im iJUG!

Ab 15,00 EUR im Jahr erhalten Sie

30 % Rabatt auf Tickets der JavaLand

Jahres-Abonnement der Java aktuell

Mitgliedschaft im Java Community Process

www.ijug.eu

Unbekannte Kostbarkeiten des SDK Heute: Unterschiede in Arrays, Dateien und Puffern

Bernd Müller, Ostfalia

Das Java SDK enthält eine Reihe von Features, die wenig bekannt sind. Wären sie bekannt und würden sie verwendet, könnten Entwickelnde viel Arbeit und manchmal sogar zusätzliche Frameworks einsparen. Wir wollen in dieser Reihe derartige Features des SDK vorstellen: die unbekannten Kostbarkeiten.

Eine häufige Anforderung ist die Überprüfung, ob zwei Dinge gleich sind. Alternativ kann aber auch das Wissen, dass zwei Dinge sich unterscheiden und worin sie sich unterscheiden, sinnvoll eingesetzt werden. In den Java-Versionen 9, 11 und 12 fanden eine ganze Reihe von `mismatch()`-Methoden Einzug in das JDK, die wir uns heute anschauen.

Die Klasse Arrays

Wir beginnen den `mismatch()`-Reigen mit der Klasse `Arrays`. Sie bekam mit Java 9 eine ganze Reihe überladener `mismatch()`-Methoden spendiert. Die Methoden existieren für primitive Typen sowie für den Typ `Object` jeweils in zwei Varianten. In der ersten werden die beiden Arrays übergeben, in der zweiten zusätzlich noch jeweils ein Start- und ein End-Index für den Vergleichsbereich. Zur Veranschaulichung geben wir die beiden Methoden für Object-Arrays an *(siehe Listing 1)*.

Das *Listing 2* enthält kleine JUnit-Tests, die die Verwendung verdeutlichen. Die Methoden geben, so wie die weiteren noch vorzustellenden Methoden, jeweils den Index des ersten Unterschieds zurück.

Eine weitere Version existiert für generische Typen. Durch Javas Type Erasure kann die Vergleichsmethode nicht zur Compile-Zeit ermittelt werden; deshalb wird dieser Methode als zusätzlicher Parameter noch die Comparator-Methode übergeben, sodass sich die Signatur wie in *Listing 3* gezeigt darstellt.

Die Version mit Start- und End-Indizes existiert ebenfalls. Da Listen einfach in Arrays überführt werden können, können auch Unterschiede in Listen über den Umweg eines Arrays einfach bestimmt werden.

Die Klasse Files

Die Klasse Files bekam mit Java 12 eine `mismatch()`-Methode. Die beiden Parameter sind `Path`-Instanzen: `static long mismatch(Path path, Path path2)`.

```
static int mismatch(Object[] a, Object[] b)
static int mismatch(Object[] a, int aFromIndex, int aToIndex, Object[] b, int bFromIndex, int bToIndex)
```

Listing 1

```
@Test
public void intArrayMismatch() {
     int[] array1 = {1, 2, 3, 4, 5};
     int[] array2 = {1, 2, 0, 4, 5};
     Assertions.assertSame(2, Arrays.mismatch(array1, array2));
}

@Test
public void objectArrayMismatch() {
     String[] array1 = {"a", "b", "c", "d", "e"};
     String[] array2 = {"a", "b", "x", "d", "e"};
     Assertions.assertSame(2, Arrays.mismatch(array1, array2));
}

@Test
public void listMismatch() {
     List<String> list1 = List.of("a", "b", "c", "d", "e");
     List<String> list2 = List.of("a", "b", "x", "d", "e");
     Assertions.assertSame(2, Arrays.mismatch(list1.toArray(), list2.toArray()));
}

@Test
public void fileMismatch() throws IOException {
     Path path1 = Paths.get("src", "test", "resources", "file1");
     // Inhalt: abcde
     Path path2 = Paths.get("src", "test", "resources", "file2");
     // Inhalt: abxde
     Assertions.assertSame(2l, Files.mismatch(path1, path2));
}

@Test
public void bufferMismatch() {
     CharBuffer buffer1 = CharBuffer.allocate(5).put("abcde").rewind();
     CharBuffer buffer2 = CharBuffer.allocate(5).put("abxde").rewind();
     Assertions.assertSame(2, buffer1.mismatch(buffer2));
}
```

Listing 2

```
static <T> int mismatch(T[] a, T[] b, Comparator<? super T> cmp)
```

Listing 3

Die Buffer-Klassen

Unterklassen der abstrakten Klasse `java.nio.Buffer`, also etwa `ByteBuffer` und `IntBuffer`, erhielten mit Java 11 jeweils eine `mismatch()`-Methode, die als Parameter einen Puffer desselben Typs erwarten. Beispielhaft für die Klasse CharBuffer also: `int mismatch(CharBuffer that)`.

Im Vergleich zu den anderen `mismatch()`-Varianten fällt auf, dass die `Buffer`-Methoden nicht „static" sind. Wir bereits angekündigt, zeigt das *Listing 2* einige einfache Beispiele, die selbsterklärend sind.

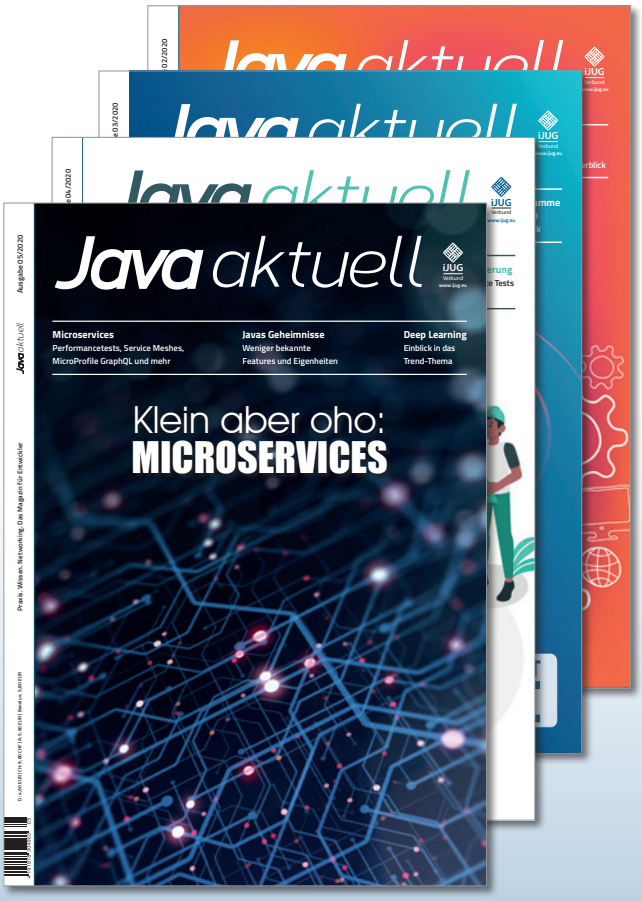
Zusammenfassung

Seit den letzten Java-Versionen ist es deutlich einfacher, Unterschiede in Arrays, Dateien und Puffern zu finden. Die Familie der `mismatch()`-Methoden liefert bei Unterschieden jeweils den Index des ersten Unterschieds oder -1, falls kein Unterschied existiert.

Bernd Müller
Ostfalia
bernd.mueller@ostfalia.de

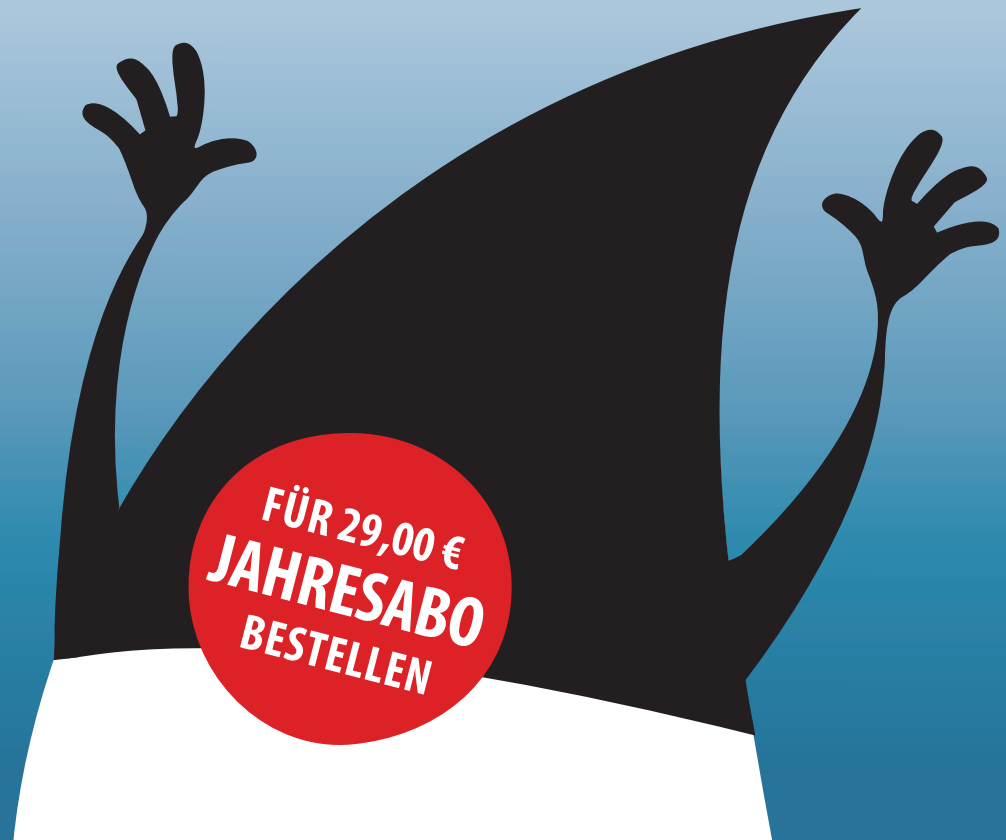
Nach seinem Studium der Informatik und der Promotion arbeitete Bernd Müller für die IBM und die HDI Informationssysteme. Er ist Professor, Geschäftsführer, Autor mehrerer Bücher zu den Themen JSF und JPA, sowie Speaker auf nationalen und internationalen Konferenzen. Er engagiert sich im iJUG und speziell in der JUG Ostfalen.

Java aktuell



Mehr Informationen zum Magazin und Abo unter:

https://www.ijug.eu/de/java-aktuell



iJUG Verbund
www.ijug.eu

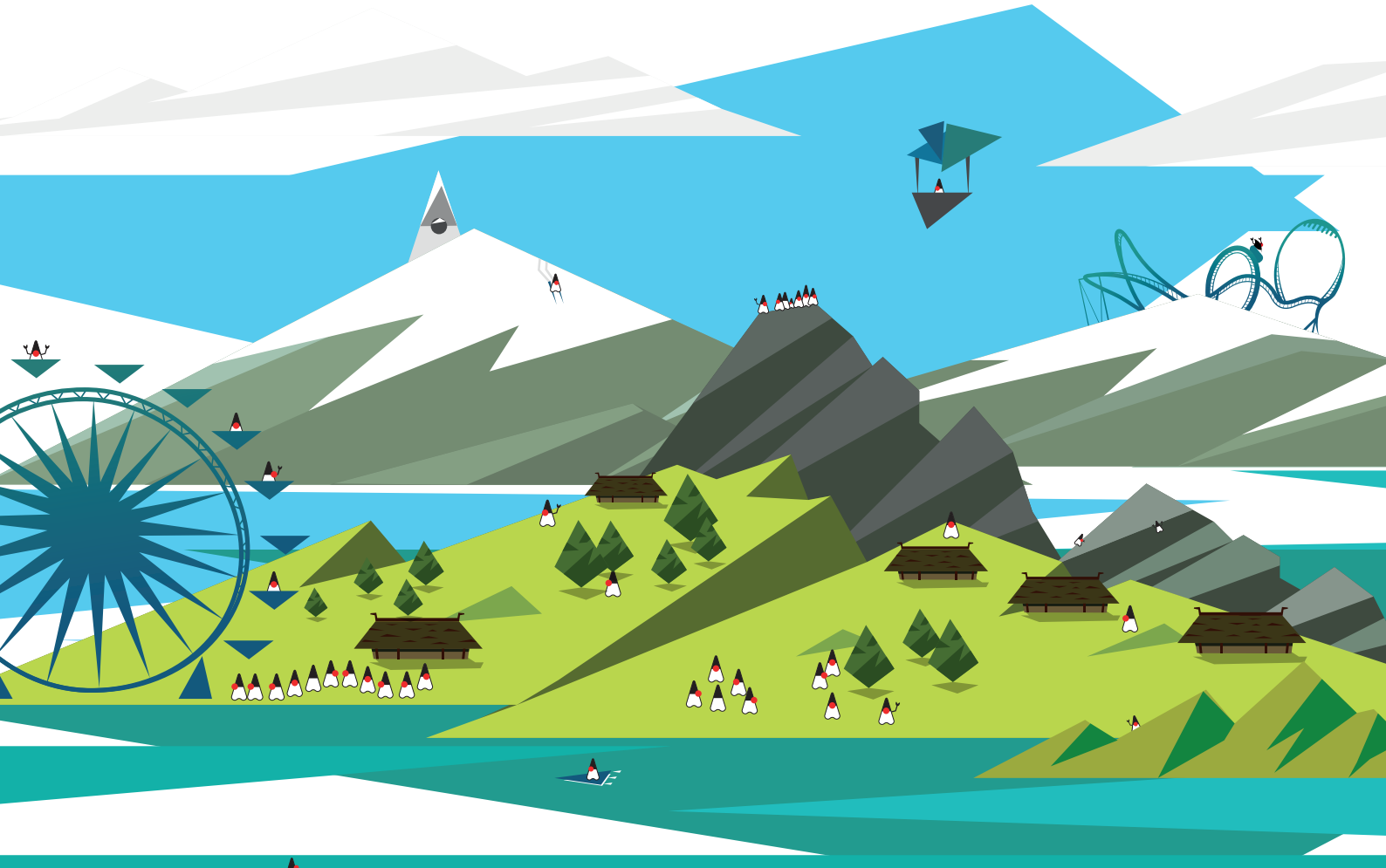
JavaLand

16. - 18. März 2021
in Brühl bei Köln

Programm jetzt online

Hybride Veranstaltung

Was die JavaLand als Plattform für Wissentransfer und Networking ausmacht, kannst du vor Ort im Phantasialand oder online erleben. Als Teilnehmer entscheidest du selbst!



Präsentiert von: DOAG Heise Medien

Community Partner: iJUG Verbund

www.javaland.eu