embarc
Software Consulting GmbH

Java 18 + 19

Auf dem Weg zur nächsten LTS-Version

FALK SIPPACH // EMBARC

Java Forum Nord

Donnerstag, 06.10.2022

1

Java 18 + 19: Der Weg zur nächsten LTS-Version

Seit einigen Jahren kommen nun schon halbjährlich neue Java Major-Releases heraus. Dieses Vorgehen hat sich etabliert und funktioniert erstaunlich gut. Natürlich dürft Ihr nicht den Funktionsumfang von den früheren Versionen (9 und älter) erwarten. Dafür bekommt Ihr als Entwickler aber viel regelmäßiger die aktuellen Änderungen mit. In den Preview-Phasen kann sogar Feedback gegeben und somit die aktive Weiterentwicklung von Java mitgestaltet werden. Alle zwei Jahre erscheinen zudem Long-Term-Support-Versionen, die länger mit Updates und Patches versorgt werden. Auch im Jahr 2022 sind in Form der JEPs wieder einige neue Features dazugekommen oder weiterentwickelt worden, u. a.:

- Pattern Matching und Record Patterns
- Virtual Threads und Structured Concurrency
- UTF-8 by Default
- Simple Web Server
- Code Snippets in Java API Documentation
- Vector API bzw. Foreign Function & Memory API
- und noch einiges mehr

Neben den JDK Enhancement Proposals (JEPs) werfen wir natürlich auch einen Blick auf hilfreiche API-Verbesserungen und Änderungen an der JVM, z. B. bei den Garbage Collectoren. Ihr bekommt einen Überblick über die neusten Entwicklungen im Java-Umfeld und seht heute schon, was Euch in den nächsten Jahren in der täglichen Arbeit erwarten wird.

2

Falk Sippach

- Softwarearchitekt, Berater, Trainer bei embarc
- früher bei Orientation in Objects (OIO), Trivadis

Schwerpunkte:

- Architekturberatung und -bewertung
- Cloud- und Java-Technologien

fs@embarc.de
@sippsack
➔ xing.to/fsi

embarc Software Consulting GmbH

3

Agenda

1 Java – still a thing?

2 Der Weg zum letzten LTS-Release

3 Spannende neue Features

4 Fazit

8

Agenda

1 **Java – still a thing?**

2 Der Weg zum letzten LTS-Release

3 Spannende neue Features

4 Fazit

1

9

Ist Java eigentlich noch **State of the Art**?

10

"Dieses Foto" von Unbekannter Autor ist lizenziert gemäß CC BY-SA

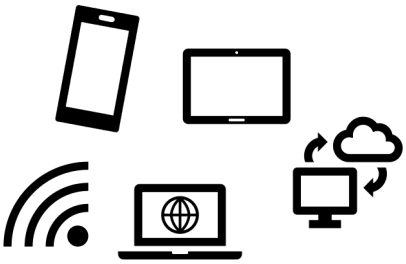
1995

11

12

1995

heute



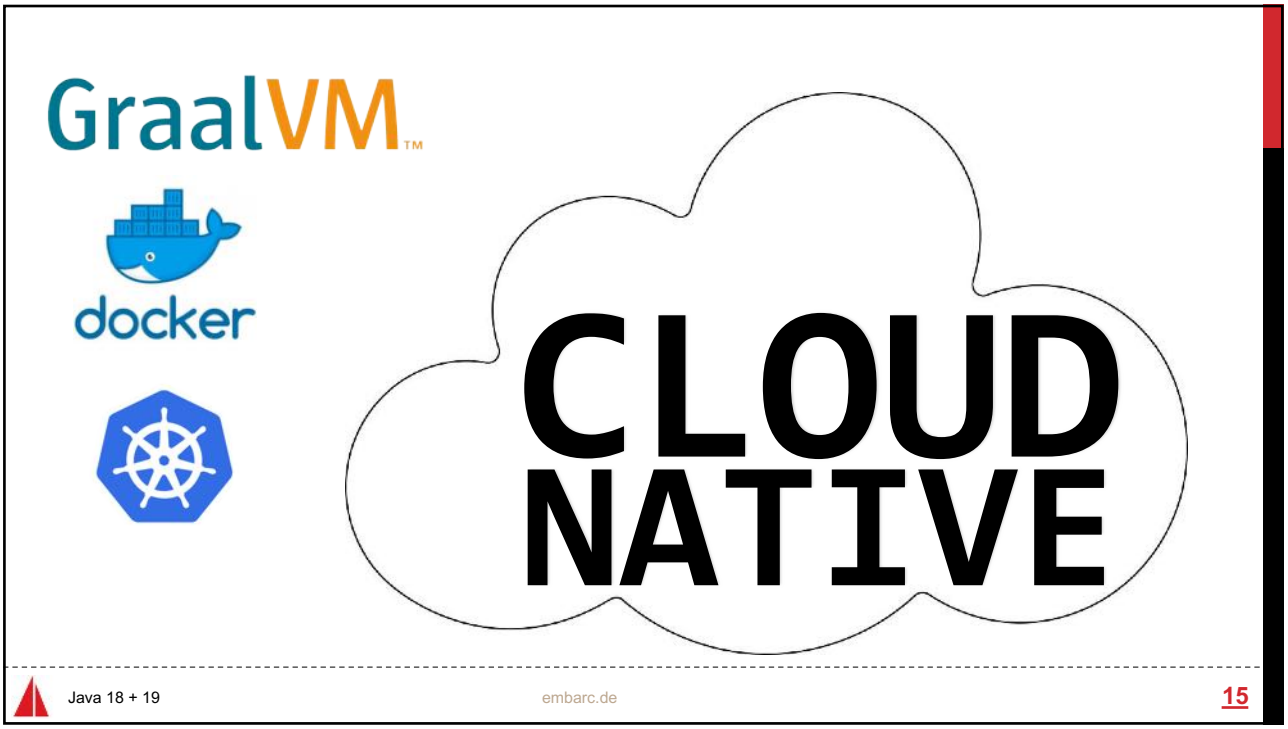
~~1819~~

13

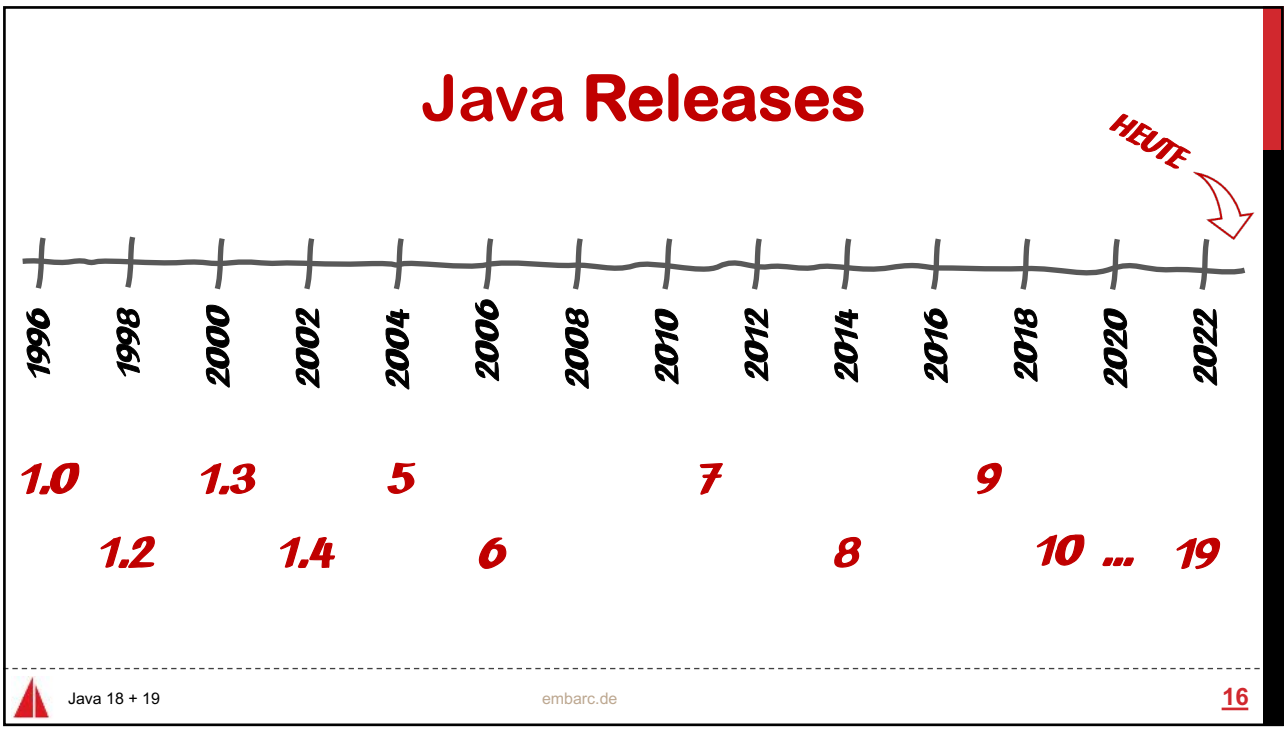
WORA

Write once, run anywhere

14

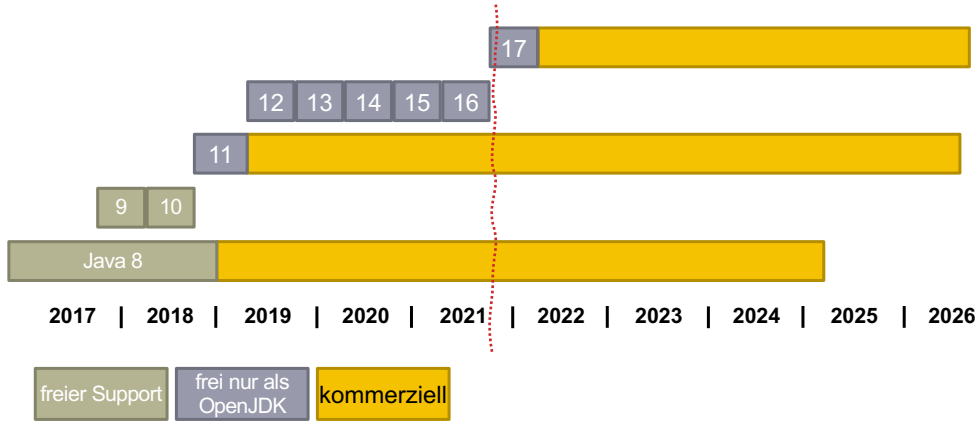

GraalVM
docker
CLOUD
NATIVE
Java 18 + 19
embarc.de
15

15


Java Releases
HEUTE
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
1.0
1.2
1.3
1.4
5
6
7
8
9
10
...
19
Java 18 + 19
embarc.de
16

16

Oracles Free-Lunch is over!



https://www.heise.de/developer/artikel/Wird-Java-jetzt-kostenpflichtig-4144533.html
https://www.oracle.com/technetwork/java/javase/eol-135779.html

17

Agenda

1 Java – still a thing?

2 **Der Weg zum letzten LTS-Release**

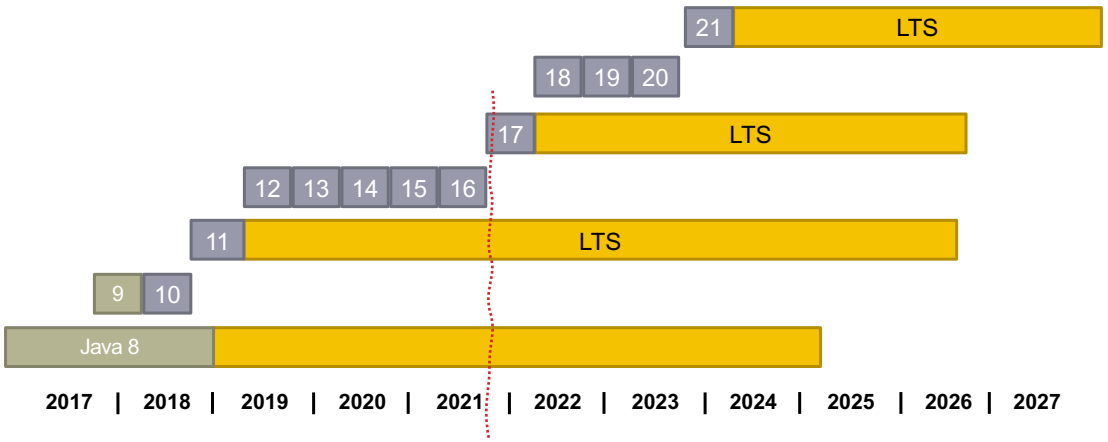
3 Spannende neue Features

4 Fazit

2

18

Release Train



19

Zoo von JDKs

20

09/2018
03/2019
09/2019
03/2020
09/2020
03/2021
09/2021

11

21

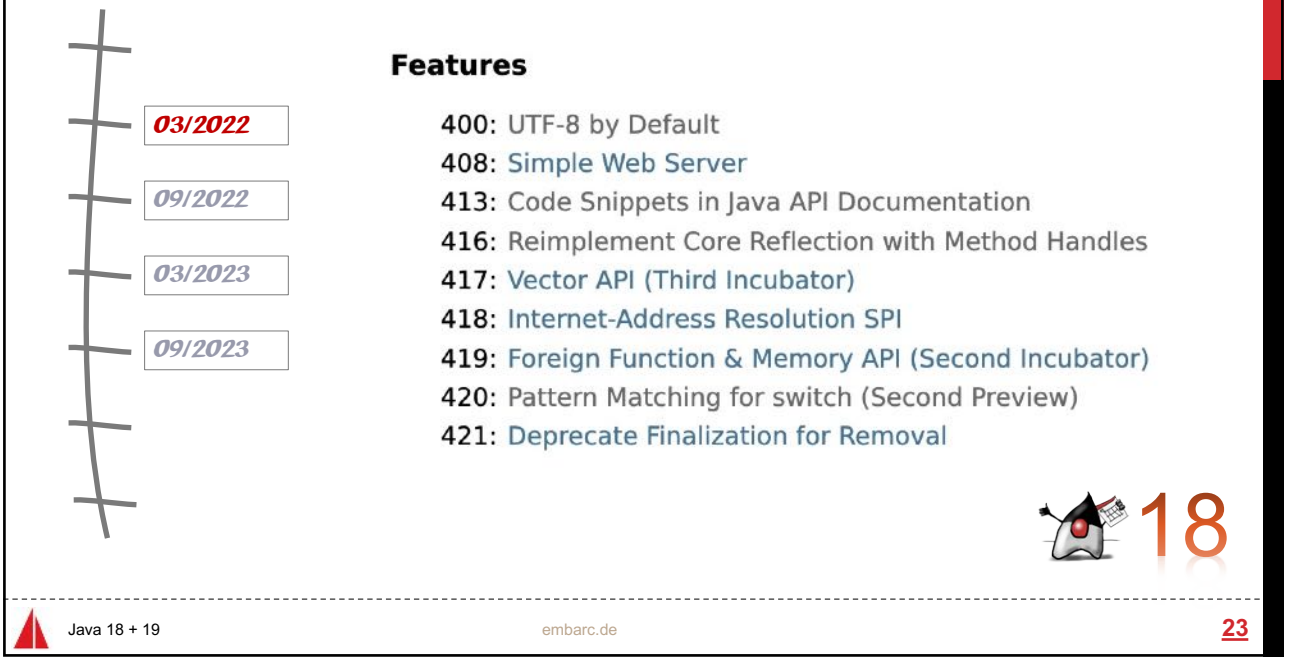
09/2018
03/2019
09/2019
03/2020
09/2020
03/2021
09/2021

Features

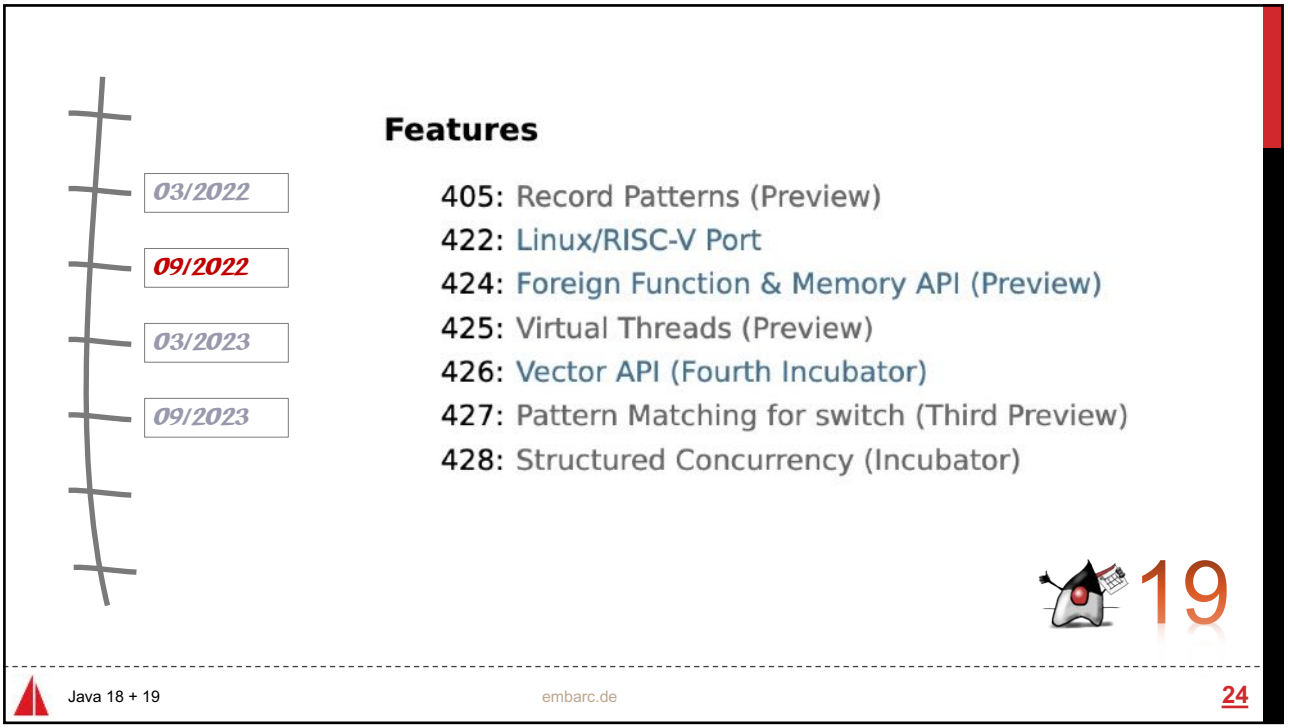
306: Restore Always-Strict Floating-Point Semantics
356: Enhanced Pseudo-Random Number Generators
382: New macOS Rendering Pipeline
391: macOS/AArch64 Port
398: Deprecate the Applet API for Removal
403: Strongly Encapsulate JDK Internals
406: Pattern Matching for switch (Preview)
407: Remove RMI Activation
409: Sealed Classes
410: Remove the Experimental AOT and JIT Compiler
411: Deprecate the Security Manager for Removal
412: Foreign Function & Memory API (Incubator)
414: Vector API (Second Incubator)
415: Context-Specific Deserialization Filters

17

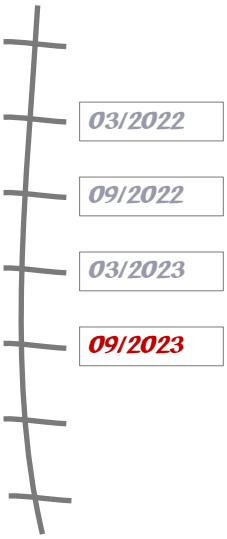
22


03/2022
09/2022
03/2023
09/2023
Features
400: UTF-8 by Default
408: Simple Web Server
413: Code Snippets in Java API Documentation
416: Reimplement Core Reflection with Method Handles
417: Vector API (Third Incubator)
418: Internet-Address Resolution SPI
419: Foreign Function & Memory API (Second Incubator)
420: Pattern Matching for switch (Second Preview)
421: Deprecate Finalization for Removal
18
Java 18 + 19
embarc.de
23

23


03/2022
09/2022
03/2023
09/2023
Features
405: Record Patterns (Preview)
422: Linux/RISC-V Port
424: Foreign Function & Memory API (Preview)
425: Virtual Threads (Preview)
426: Vector API (Fourth Incubator)
427: Pattern Matching for switch (Third Preview)
428: Structured Concurrency (Incubator)
19
Java 18 + 19
embarc.de
24

24



LTS-Release

25

Agenda

1 Java – still a thing?

2 Der Weg zum letzten LTS-Release

3 **Spannende neue Features**

4 Fazit

3

26

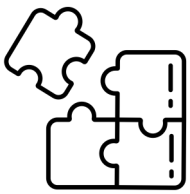
Neue Features

Garbage Collectoren

Interna

API-Änderungen (JDK)

27



Neue Features

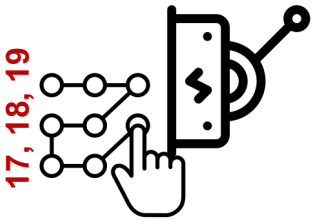
Garbage Collectoren

Interna

API-Änderungen (JDK)

28

17, 18, 19



Pattern Matching for switch

19

Record Patterns

19

Virtual Threads

29

```
/**
 * The following code shows how to use {@code Optional.isPresent}:
 * {@snippet :
 * if (v.isPresent()) {
 *     System.out.println("v: " + v.get()); // @highlight substring="println"
 * }
 *}
 */
public void isPresentDemo() {
```

All Methods	Static Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description	
void	isPresentDemo()	The following code shows how to use Optional.isPresent:	

```
if (v.isPresent()) {
    System.out.println("v: " + v.get());
}
```

18

Code Snippets in JavaDoc

30

jwebserver

```
$ jwebserver
Binding to loopback by default. For all interfaces use "-b 0.0.0.0" or "-b ::".
Serving /home/sven and subdirectories on 127.0.0.1 port 8000
URL http://127.0.0.1:8000/
```

```
HttpServer server = SimpleFileServer.createFileServer(
  new InetSocketAddress(8888), Path.of("out"),
    OutputLevel.INFO);
server.start();
```

Simple Webserver

31

Neue Features

Garbage Collectoren

Internas

API-Änderungen (JDK)

34

Garbage Collectoren

Neue Generation von Low-Latency Garbage Collectoren

moderne Architekturen: Multi-Core + TB RAM

Ziel: kurze GC-Pausen im ms-Bereich

35

Neue Features

Garbage Collectoren

Interna

API-Änderungen (JDK)

36

14, 15, 16, 17, 18, 19

Foreign Linker + Foreign-Memory Access API

37

16, 17, 18, 19

Vector API

38

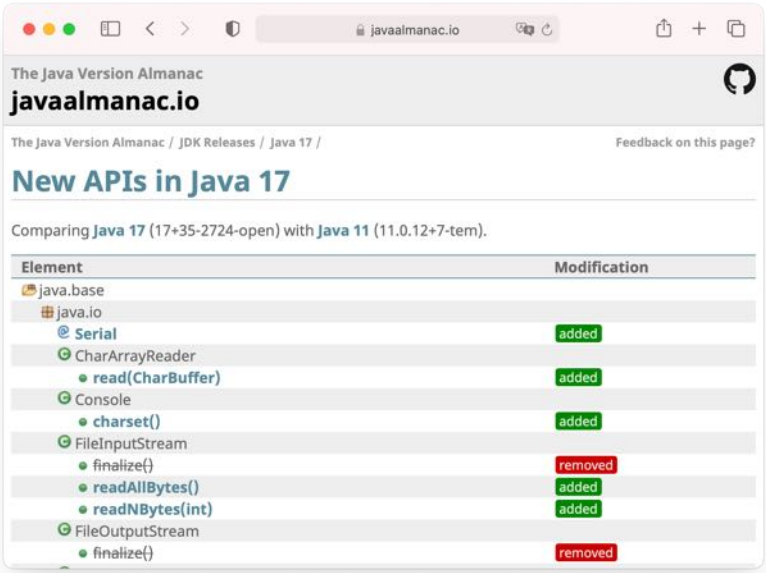
Neue Features

Garbage Collectoren

Interna

API-Änderungen (JDK)

39



JDK

https://javaalmanac.io/jdk/17/apidiff/11/

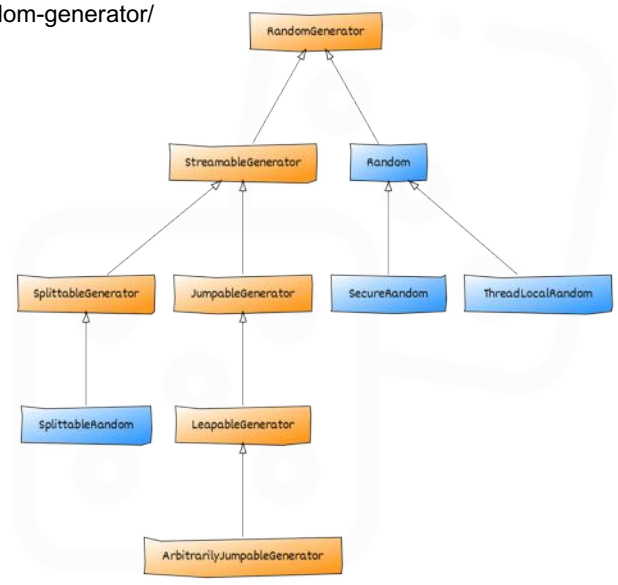
40

```java
var s = Stream.of(1, 2, 3, 4, 5);

var l1 = s.collect(Collectors.toList());

var l2 = s.toList();
```

JDK

41

https://nipafx.dev/java-random-generator/



JDK

42

Auf Wiedersehen …

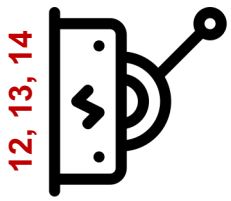
Applet API deprecated for Removal

Security Manager deprecated for Removal

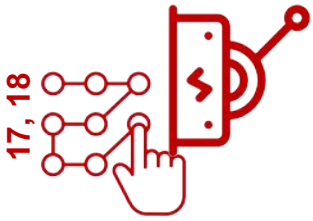
Deprecate Finalization for Removal

Experimental AOT and JIT Compiler

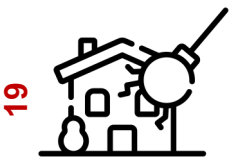
43



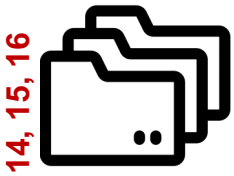
Switch Expression



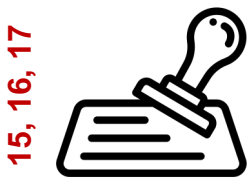
Pattern Matching for switch



Record Patterns



Records



Sealed Classes

44

```java
private static String developerRating( int numberOfChildren ) {
    return switch (numberOfChildren) {
        case 0 -> "open source contributor";
        case 1, 2 -> "junior";
        case 3 -> "senior";
        default -> {
            if (numberOfChildren < 0)
                throw new IndexOutOfBoundsException( numberOfChildren );
            yield "manager";
        }
    };
}
```

Switch Expression

45

```java
package de.sippsack.records;

import java.math.BigDecimal;
import java.util.Currency;

public record MonetaryAmount(BigDecimal value,  Currency currency) {}
```

```
➜  records javap MonetaryAmount.class
Compiled from "MonetaryAmount.java"
public final class de.sippsack.records.MonetaryAmount extends java.lang.Record {
  public de.sippsack.records.MonetaryAmount(java.math.BigDecimal, java.util.Currency);
  public final java.lang.String toString();
  public final int hashCode();
  public final boolean equals(java.lang.Object);
  public java.math.BigDecimal value();
  public java.util.Currency currency();
}
```

Records

46

```
public record MonetaryAmount(BigDecimal value,
      Currency currency) {

    public MonetaryAmount(BigDecimal value) {
        this(value, Currency.getInstance("EUR"));
    }

    public MonetaryAmount times(BigDecimal factor) {
        return new MonetaryAmount(value.multiply(factor),
          currency);
    }
}
```

Records

47

```
MonetaryAmount tenEuro = new MonetaryAmount(BigDecimal.TEN,
Currency.getInstance("EUR"));
MonetaryAmount anotherTenEuro = new MonetaryAmount(BigDecimal.TEN);

System.out.println(tenEuro);
System.out.println(tenEuro.times(BigDecimal.TEN));

System.out.println(tenEuro.equals(anotherTenEuro)); // true
System.out.println(tenEuro.equals(new MonetaryAmount(BigDecimal.TEN,
      Currency.getInstance("USD")))); // false
```

Records

48

feingranulare Steuerung der Vererbungshierarchie

Algebraische Datentypen

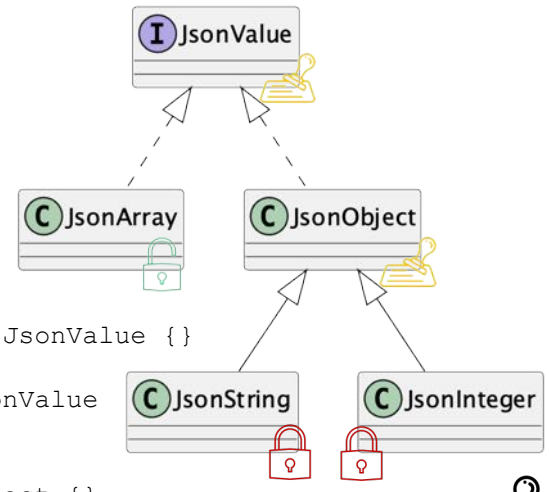
Prüfung der “Exhaustiveness” im switch

Sealed Classes

50

Flexible Klassenhierarchien

```
public sealed interface JsonValue
       permits JsonArray, JsonObject {

    non-sealed class JsonArray implements JsonValue {}

    sealed class JsonObject implements JsonValue
      permits JsonString, JsonInteger {}

    final class JsonString extends JsonObject {}

    final class JsonInteger extends JsonObject {}
}
```



Sealed Classes

51

```java
public sealed interface Expr
        permits ConstantExpr, PlusExpr, TimesExpr, NegExpr {
}
public record ConstantExpr(int i) implements Expr {}
public record PlusExpr(Expr a, Expr b) implements Expr {}
public record TimesExpr(Expr a, Expr b) implements Expr {}
public record NegExpr(Expr e) implements Expr {}
```

Records

Sealed Classes

52

```java
private static boolean isNullOrEmpty(Object o) {
    return o == null ||
           o instanceof String && ((String) o).isBlank() ||
           o instanceof Collection && ((Collection) o).isEmpty();
}
```

```java
private static boolean isNullOrEmpty(Object o) {
    return o == null ||
           o instanceof String s && s.isBlank() ||
           o instanceof Collection c && c.isEmpty();
}
```

```java
System.out.println(isNullOrEmpty(null)); // true
System.out.println(isNullOrEmpty("")); // true
System.out.println(isNullOrEmpty(List.of(1, 2, 3))); // false
```

Pattern Matching for instanceof

53

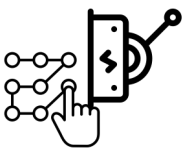
Pattern Matching for instanceof

54

JEP 420: Pattern Matching for switch (Preview)

Type Patterns im switch

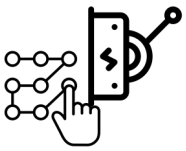
```
String evaluateTypeWithSwitch( Object o ) {
  return switch(o) {
    case String s -> "String: " + s;
    case Collection c -> "Collection: " + c;
    default -> "Something else: " + o;
  };
}
```



55

Exhaustiveness-Prüfung bei Sealed Classes

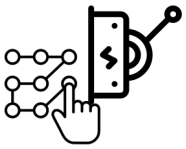
```
public sealed interface Tarif permits Privat, Business, Profi {}

public record Profi(double preisProMinute) implements Tarif {}

public record Business(double preisProMinute) implements Tarif {}

public record Privat(double preisProMinute) implements Tarif {
  public int getNettoMinuten(int minuten) {
    return Math.max(minute - 1, 0);
  }
}
```



56

Exhaustiveness: Kein default notwendig

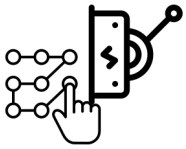
```
double preis = switch(tarif) {
  case Privat p -> p.getNettoMinuten(minuten) *
                                    p.preisProMinute();
  case Business b -> minuten * b.preisProMinute();
  case Profi p -> minuten * p.preisProMinute();
};
```



57

~~Guarded Patterns~~ When Clauses und Null-Check

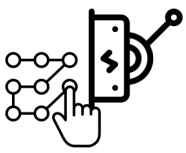
```
boolean isNullOrEmptyWithSwitch( Object o ) {
  return switch(o) {
    case null -> true;
    case String s when s.isBlank() -> true;
    case String s -> false;
    case Collection c when c.isEmpty() -> true;
    default -> false;
  };
}
```



58

Überdeckung bei Vollständigkeitsprüfung

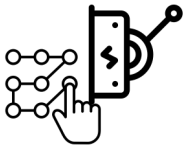
```
String str = ...

switch (str) {

  case String s && s.length() < 5 ->
            System.out.println(s.toUpperCase());
  case "Test" -> System.out.println("constant string");
  ...
}
```



59

Vollständigkeitsprüfung bei Generics

```
sealed interface I<T> permits A, B {}

  final class A<X> implements I<String> {}
  final class B<Y> implements I<Y> {}

  static int testGenericSealedExhaustive(I<Integer> i) {
    return switch (i) {
        // Exhaustive as no A case possible!
        case B<Integer> bi -> 42;
    }
}
```



60

JEP 405: Record Patterns ~~& Array Patterns~~

```
static void printSumOfFirstTwoXCoords(Object o) {
  if (o instanceof ~~Point[] { Point(var x1, var y1),~~
       ~~Point(var x2, var y2), ... }~~) {
    System.out.println(x1 + x2);
  }
}
```

19

Record Patterns

https://openjdk.java.net/jeps/405

61

```
record Point(int x, int y) {
}
record Rectangle(Point p1, Point p2) {
}

Object mayBeAPoint = new Point(0, 0);
Object mayBeARectangle =
    new Rectangle(new Point(0, 1), new Point(10, 6));

if (mayBeAPoint instanceof Point(int x, int y)){
     System.out.println(x + y);
}
```

Record Patterns

63

```
if (mayBeARectangle instanceof Rectangle(Point(int x1, int y1),
        Point(var x2, var y2)) r) {
  int area = Math.abs(x2 - x1) * Math.abs(y2 - y1);
  System.out.println(
    String.format("Fläche des Rechtecks (%s): %s", r, area));
}

if (mayBeARectangle instanceof Rectangle(Point(int x1, int y1),
    Point p2)) {
  System.out.println(
    String.format("Koordinaten des P1 vom Rechteck: (%s; %s)", x1, y1));
}
```

Record Patterns

64

Millionen von Threads …

```
Thread.Builder threadBuilder = createThreadBuilder();

threadBuilder.start(() -> {
    // im Thread auszuführender Code
    System.out.println(Thread.currentThread().isVirtual());
});

private static Thread.Builder createThreadBuilder() {
    return Thread.ofVirtual();
    // return Thread.ofPlatform();
}
```

65

Alt: ExecutorService

```
private static final ExecutorService executor =
    Executors.newCachedThreadPool();

Future<String> task1 = executor.submit(() -> { … }
Future<String> task2 = executor.submit(() -> { … }
Future<String> task3 = executor.submit(() -> { … }

System.out.println(task1.get());
System.out.println(task2.get());
System.out.println(task3.get());
```


66

Structured Concurrency

```
try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {
  Future<String> task1 = scope.fork(() -> { … }
  Future<String> task2 = scope.fork(() -> { … }
  Future<String> task3 = scope.fork(() -> { … }

  scope.join();
  scope.throwIfFailed();

  System.out.println(task1.get());
  System.out.println(task2.get());
  System.out.println(task3.get());
}
```

19

basiert auf Virtual Threads

67

Preview Features müssen aktiviert werden!

```
$ javac --enable-preview -source 19 --add-modules jdk.incubator.concurrent
StructuredConcurrency.java

$ java --enable-preview --add-modules jdk.incubator.concurrent
StructuredConcurrency
```


68

Agenda

1 Java – still a thing?

2 Der Weg zum letzten LTS-Release

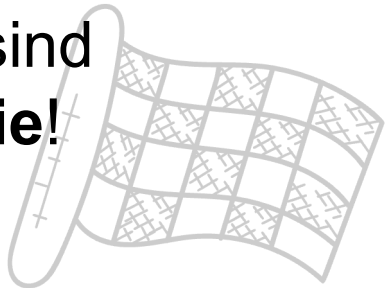
3 Spannende neue Features

4 **Fazit**

4

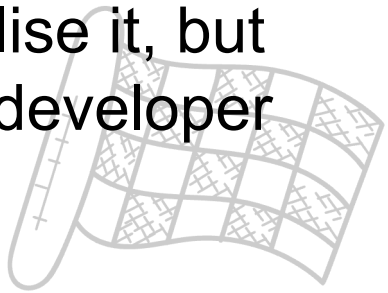
69

Die Java-Welt und
das Ökosystem sind
lebendig wie nie!

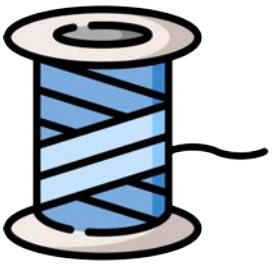


70

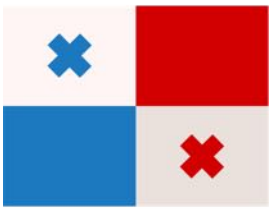
Every feature is not as good as a good developer can utilise it, but as bad as the average developer can misuse it.



71



Was kommt?



72

```
void main() {
  println("Hello World")
}
```

```
public class HelloWorld {
  public static void main(String[] args) {
     System.out.println("Hello World")
  }
}
```

https://twitter.com/BillyKorando/status/1575218151616552960

73

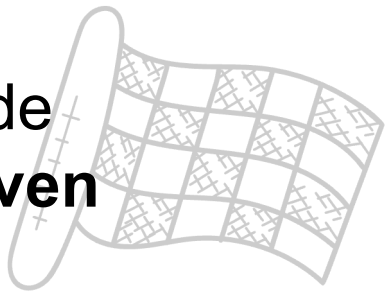
String Templating

```
String query =
    STR."""select * from User u
    where u.\{property} ='\{value}'""";
```

74

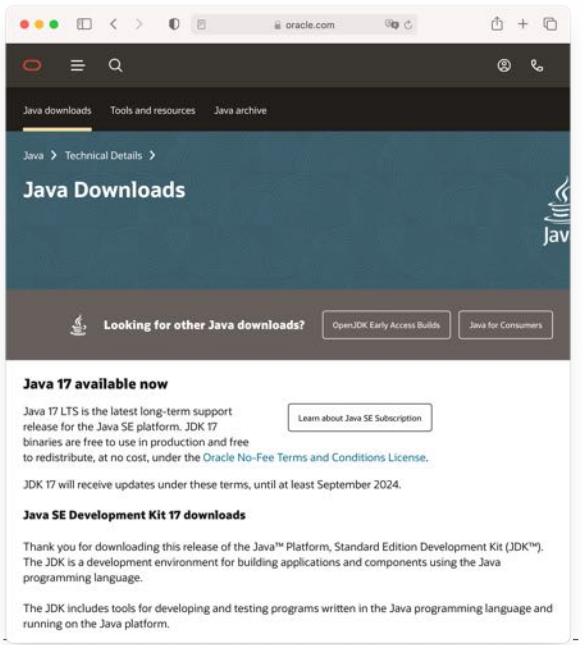
Oracle JDK ist ~~nicht~~ wieder ~~mehr~~ **kostenlos**

Außerdem ~~Aber~~, es gibt jede **Menge Alternativen**

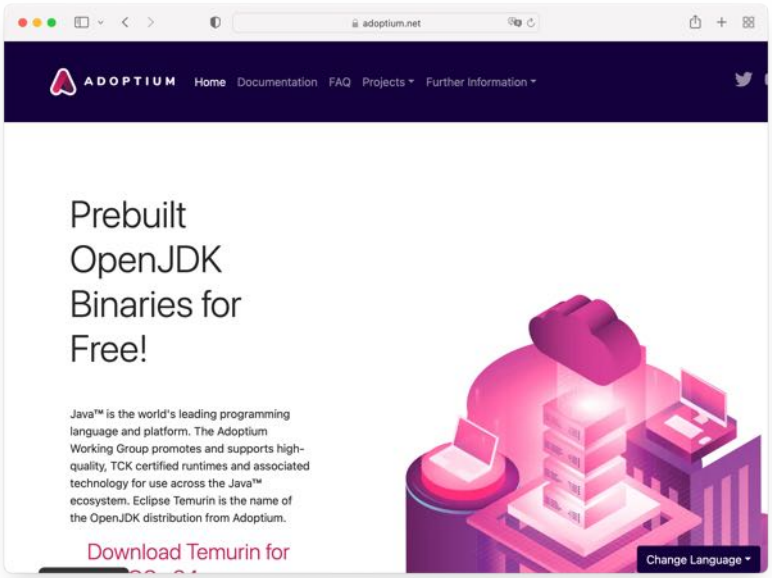


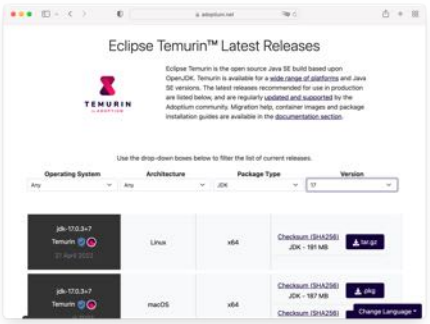
79

> Java 17 LTS is the latest long-term support release for the Java SE platform. **JDK 17 binaries** are **free to use** in **production** and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions License.



80

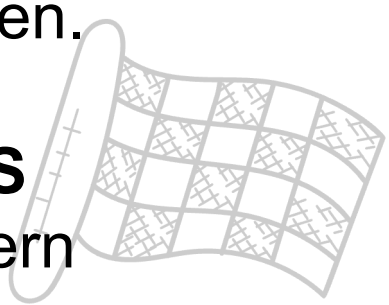




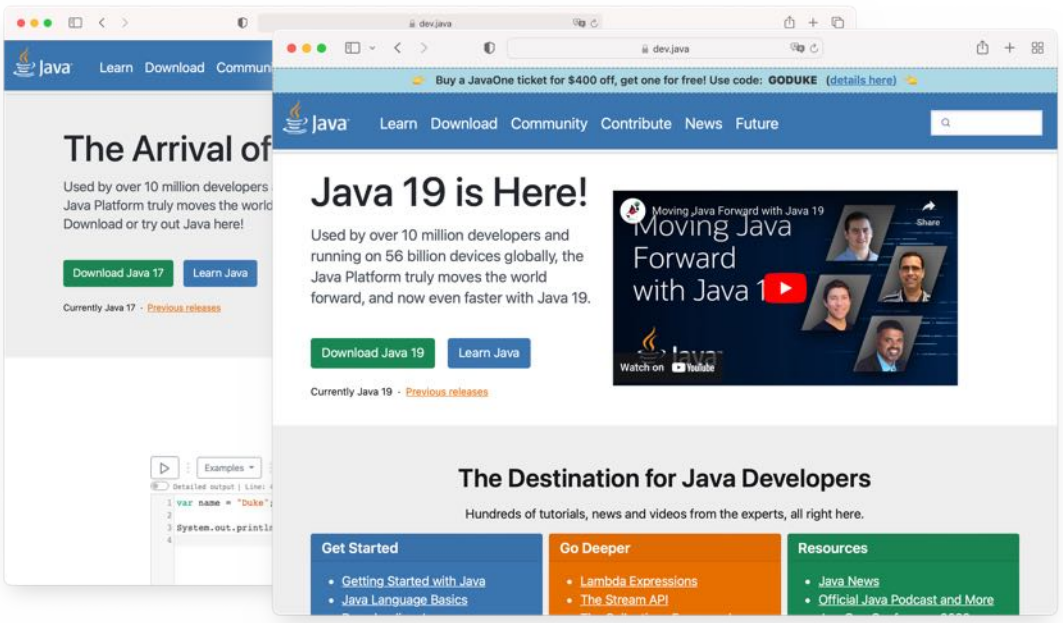
81

Migration/Upgrade
auf **Java 11+** nicht
zu lange aufschieben.

Das **nächste LTS**
Release ist nicht fern
…

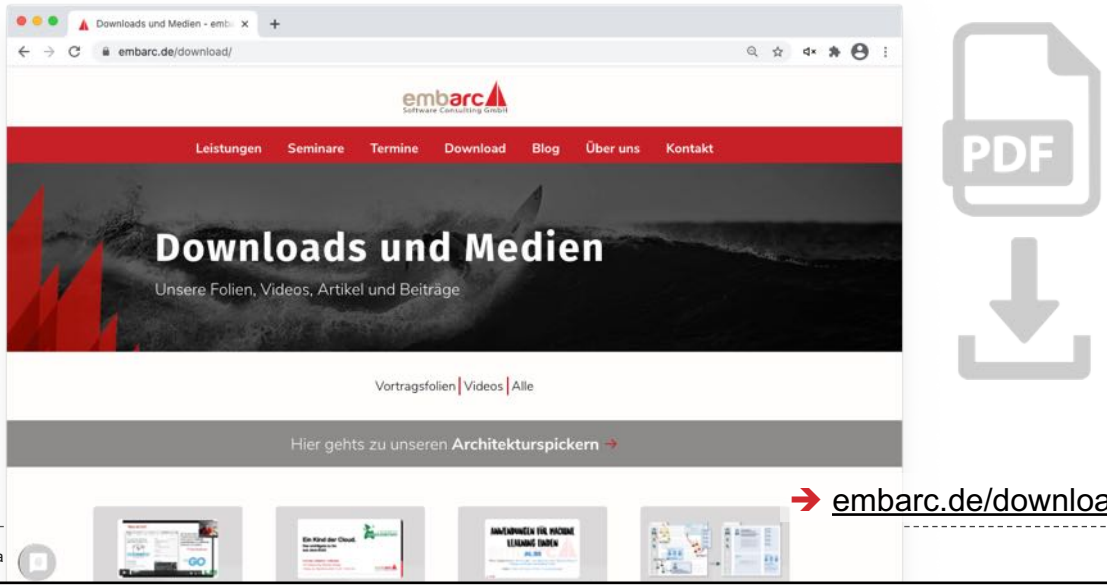


82



83

Folien von heute als PDF zum Download



➔ embarc.de/download/

84

Vielen Dank.

embarc
Software Consulting GmbH

Ich freue mich auf Eure Fragen!

Falk Sippach

fs@embarc.de

@sippsack

➔ xing.to/fsi