

Die nächste Version
mit LTS-Support

Java 21

Die nächste LTS-Version

FALK SIPPACH // EMBARC



1

Java 21 - Die nächste Version mit LTS-Support

Alle halben Jahre erscheinen neue Major-Releases, alle zwei Jahre als LTS mit längerer Unterstützung (Long Term Support). Viele arbeiten aber noch mit der früheren LTS-Versionen 8 oder sind gerade erst auf 11 umgestiegen. Wir wollen uns daher anschauen, was sich seitdem im Java-Universum getan hat. Das sind so spannende Themen wie Pattern Matching und Virtual Threads. Aber es sind noch viele weitere Features in Arbeit oder der Planung, wie die Vector API, die Foreign Function & Memory API, die String Interpolation oder die Primitive bzw. Value Types.

Neben diesen verschiedenen 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

🐦 @sipsack

🔗 → [xing.to/fsi](https://www.xing.to/fsi)



Java™
Champions



Agenda



- 1 Java – still a thing?
- 2 Ist Java 21 ein LTS-Release?
- 3 Viele spannende neue Features
- 4 Fazit

Agenda



1

- 1 Java – still a thing?
- 2 Ist Java 21 ein LTS-Release?
- 3 Viele spannende neue Features
- 4 Fazit



Die nächste Version mit LTS-Support

embarc.de

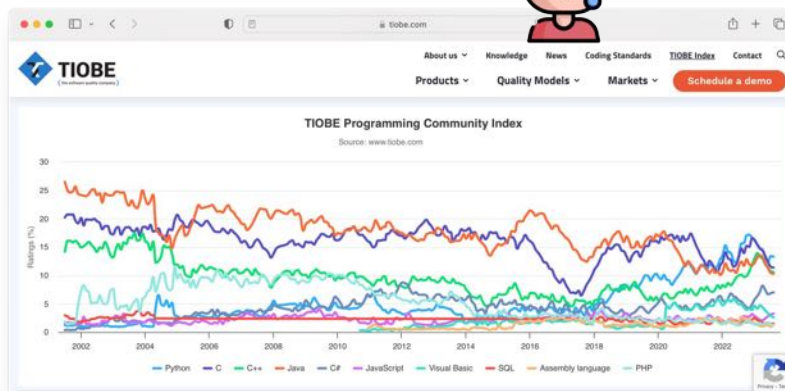
8

8

Is Java still hyped or still relevant at all?



nur noch Platz 4



Aug 2023	Aug 2022	Change	Programming Language	Ratings	Change
1	1		Python	13.33%	-2.30%
2	2		C	11.41%	-3.35%
3	4		C++	10.63%	+0.49%
4	3		Java	10.33%	-2.14%
5	5		C#	7.04%	+1.64%
6	8		JavaScript	3.29%	+0.89%
7	6		Visual Basic	2.63%	-2.26%
8	9		SQL	1.53%	-0.14%
9	7		Assembly language	1.34%	-1.41%
10	10		PHP	1.27%	-0.09%
11	21		Scratch	1.22%	+0.63%
12	15		Go	1.16%	+0.20%
13	17		MATLAB	1.05%	+0.17%
14	18		Fortran	1.03%	+0.24%
15	31		COBOL	0.96%	+0.1



Die nächste Version mit LTS-Support

embarc.de

9

9

Traue keiner Statistik, die Du nicht selbst ...



Die nächste Version mit LTS-Support

embarc.de

10

10

PYPL: Popularity of Programming Language



immerhin schon Platz 2



Rank	Change	Language	Share	1-year trend
1		Python	27.99 %	+0.1 %
2		Java	15.9 %	-1.1 %
3		JavaScript	9.36 %	-0.1 %
4		C#	6.67 %	-0.4 %
5		C/C++	6.54 %	+0.3 %
6		PHP	4.91 %	-0.4 %
7		R	4.4 %	+0.2 %
8		TypeScript	3.04 %	+0.2 %
9	↑↑	Swift	2.64 %	+0.6 %
10		Objective-C	2.15 %	+0.1 %
11	↑↑	Rust	2.12 %	+0.5 %
12	↓↓↓	Go	2.0 %	-0.1 %
13	↓	Kotlin	1.78 %	-0.0 %



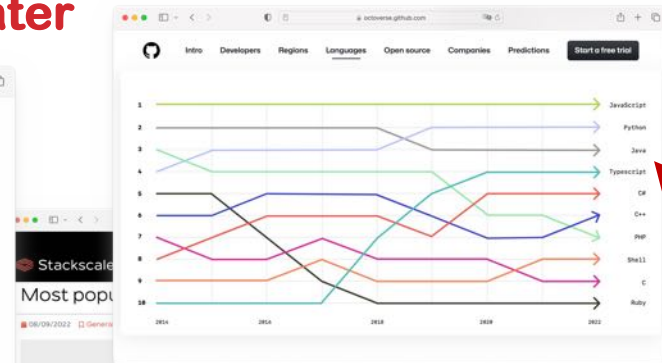
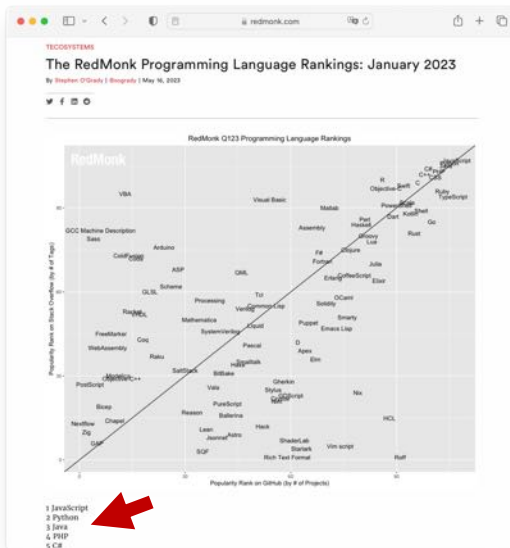
Die nächste Version mit LTS-Support

embarc.de

11

11

Viele Statistiken später



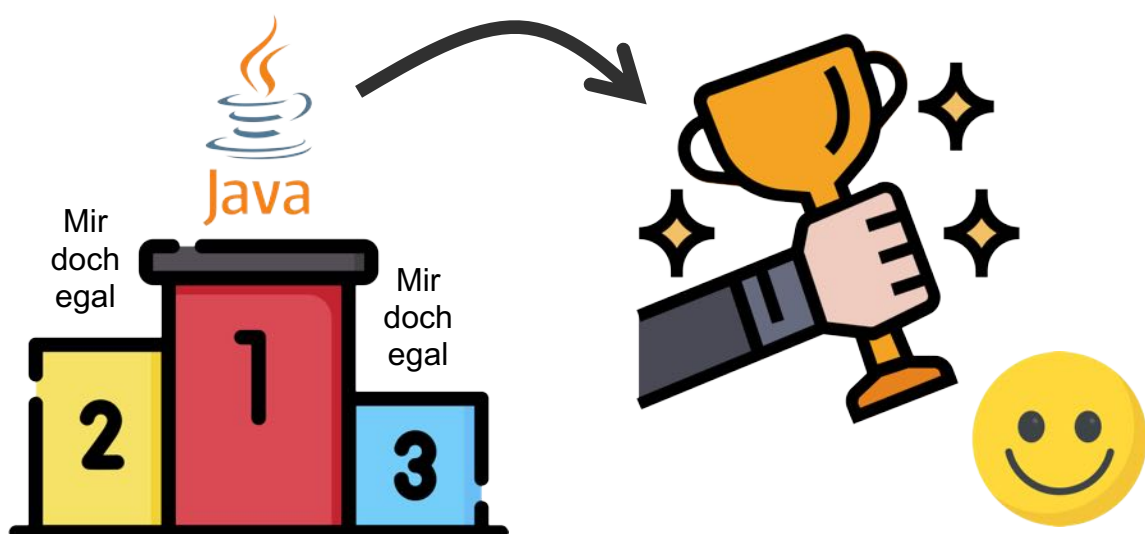
Die nächste Version mit LTS-Support

embarc.de

12

12

FPSI: Falks Programmiersprachen Index



Die nächste Version mit LTS-Support

embarc.de

13

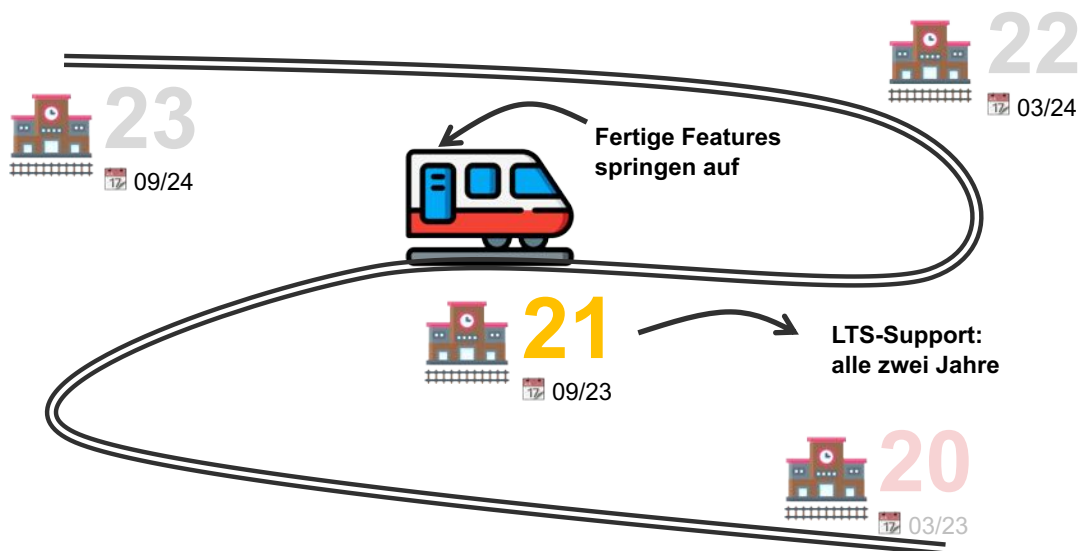
13

Warum ist Java auch nach fast 30 Jahren immer noch so populär?



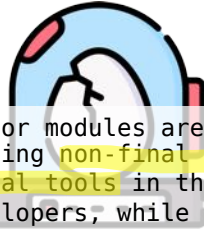
14

Halbjährlicher Release-Train



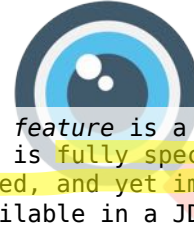
15

Inkubator und Preview



Incubator modules are a means of putting **non-final APIs** and **non-final tools** in the hands of developers, while the APIs/tools progress towards **either finalization or removal in a future release.**

<https://openjdk.org/jeps/11>



A *preview feature* is a new feature [...] that is **fully specified, fully implemented, and yet impermanent.** It is available in a JDK feature release to provoke developer feedback based on real world use; this may lead to it **becoming permanent in a future Java SE Platform.**

<https://openjdk.org/jeps/12>



Agenda

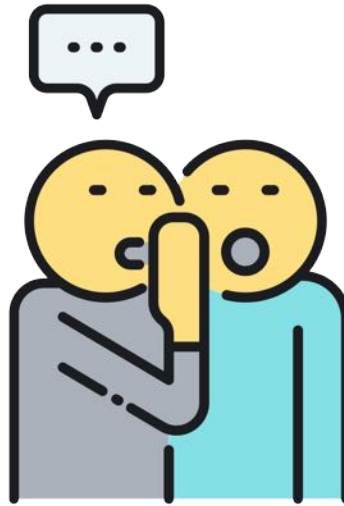


- 1 Java – still a thing?
- 2 **Ist Java 21 ein LTS-Release?**
- 3 Viele spannende neue Features
- 4 Fazit

2



Ist 21 Java überhaupt ein LTS-Release?



Die nächste Version mit LTS-Support

embarc.de

18

18

Ist Java 21 überhaupt ein LTS-Release?

Inside Java
News and views from members of the Java team at Oracle

Shows: Podcast | Newscast | JEP Café | Sip Of Java [dev.java](#) | [About](#) | [Jobs](#)

Sort by: Date | Author | Tag

Java 21 is no LTS Version - Inside Java Newscast #52
Nicolai Parlog on July 6, 2023

Ansehen auf [YouTube](#)

Nicolai Parlog [@nipafx](#)

[#Java21](#) is no long-term support version!

Don't worry, there'll be plenty of JDK 21 builds that get free, timely updates and lots of companies that offer commercial support, but that doesn't make 21 "an LTS version". Let's talk semantics! [👉](#)

Nicolai Parlog [@nipafx](#)

Before we get into it, what I describe here is nothing new. The same holds for 17 and 11 and probably also for 8 and before but I didn't check whether there are any differences for these older versions.

Ok, now let's tackle this:
"Java 21 is a long-term support version"



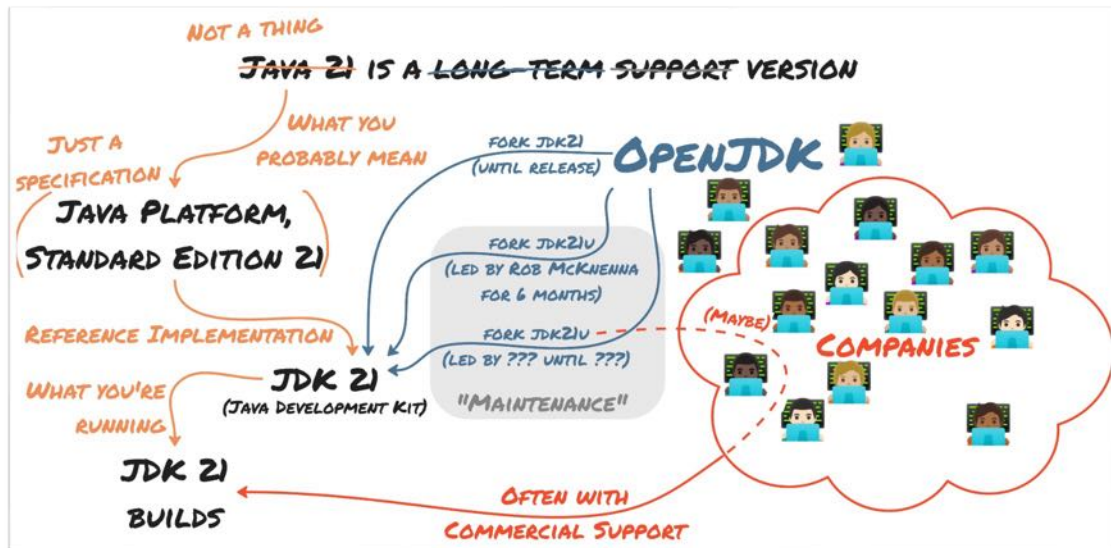
Die nächste Version mit LTS-Support

embarc.de

19

19

Diverse JDK 21 Build erhalten LTS ...



<https://twitter.com/nipafx/status/1676908785313492992?t=9K-kilkLOSbS6RFT9OYKzw&s=03>

20

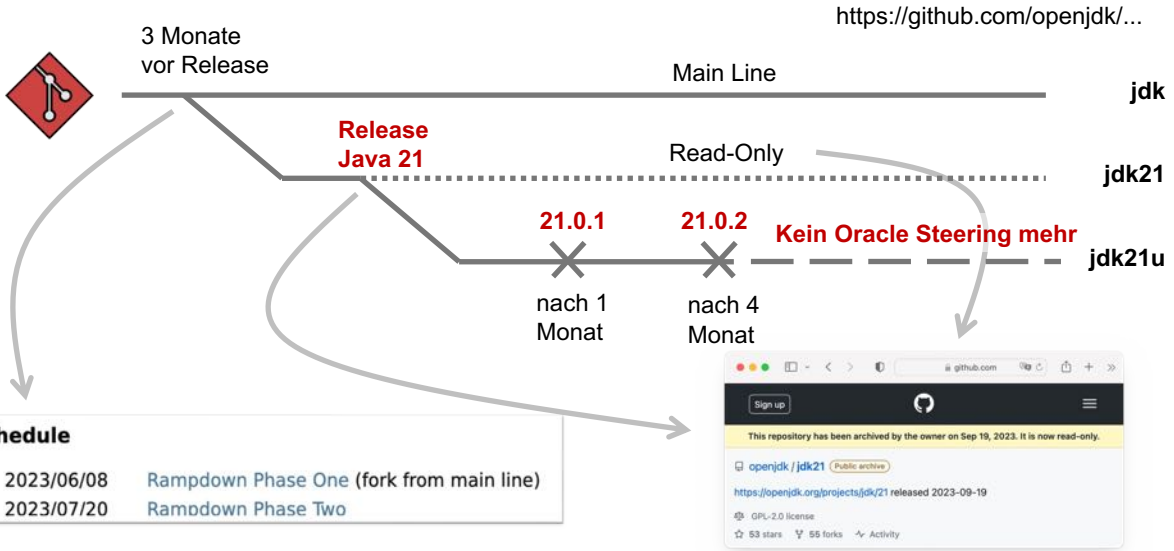
Zoo von JDKs

OpenJDK



21

JEP 3: JDK Release Process



22

Apropos LTS Release ...

Available Java Versions for macOS 64bit

Vendor	Use	Version	Dist	Status	Identifier	
Corretto		21		amzn		21-amzn
...						
Temurin		8.0.372		sem		8.0.372-sem
		20.0.2		tem		20.0.2-tem
		20.0.1		tem		20.0.1-tem
		19		local only		19-tem
		17.0.8		installed		17.0.8-tem

Warum gibt es 10 Tage nach dem Java 21 Release noch keine Temurin Variante? 🤔

OpenJDK

OCTLA Signatories List

The following organizations and individuals have signed the OpenJDK Community TCK License Agreement (OCTLA) and been granted access to the JCK.

- Signatories for Java SE 21, or later
 - Amazon.com Services LLC
 - Azul Systems, Inc.
 - BellSoft
 - MicroDoc Software GmbH
 - Microsoft
- Signatories for Java SE 9 - Java SE 20
 - Amazon.com Services, Inc.
 - Ampere Computing LLC

Screenshots vom 29.09.2023

Die nächste Version mit LTS-Support

embarc.de

https://openjdk.org/groups/conformance/JckAccess/jck-access.html 23

23

The screenshot shows a browser window at adoptium.net. The page features a dark blue navigation bar with the Adoptium logo and links for Home, Marketplace, Documentation, FAQ, Projects, and Further Information. A pink notification banner at the top states: "28th of September 2023: We are awaiting access to the new Java 21's specification tests before formally releasing Eclipse Temurin 21. For further information including how to use our early access builds, please read our blog." The main content area has a white background with the title "Eclipse Temurin 21 release delay" in bold black text. Below the title, it says "September 28, 2023 – posted by Adoptium PMC" and includes social media icons for Twitter, LinkedIn, Facebook, YouTube, and Email. The text of the post reads: "At Adoptium we were as excited as everyone else to welcome the release of Java 21 last week. We have been testing OpenJDK 21 extensively, and have successfully built and AQAvit verified the OpenJDK 21 GA source code release (based on the jdk-21+35 tag). Our community is committed to ensuring that all our builds are compliant to the Java specification before declaring them an official Eclipse Temurin release. We believe it is important to ensure that every official release is secure and high-quality as determined by AQAvit, and is". At the bottom of the post, there are buttons for "Cookie settings" and "Sprache ändern". The footer of the browser window shows a red triangle icon, the text "Die nächste Version mit LTS-Support", the URL "embarc.de", the full URL "https://adoptium.net/de/blog/2023/09/temurin21-delay/", and the page number "24".

24

Agenda



- 1 Java – still a thing?
- 2 Ist Java 21 ein LTS-Release?
- 3 Viele spannende neue Features**
- 4 Fazit

3

The slide features a large red number "3" on the left side, which is underlined. The agenda items are listed in a numbered format, with the third item, "Viele spannende neue Features", highlighted in bold black text. A clock icon is positioned in the top right corner. The footer contains a red triangle icon, the text "Die nächste Version mit LTS-Support", the URL "embarc.de", and the page number "25".

25

BIG

Java 21 is ...

Die nächste Version mit LTS-Support embarc.de **26**

26

LTS

Java 17	Java 18	Java 19	Java 20	...
14 JEPs	9 JEPs	7 JEPs	7 JEPs	
<i>Pattern Matching</i> Sealed Classes Records <i>Foreign Function & Memory API</i> <i>Vector API</i>	<i>Pattern Matching</i> <i>Foreign Function & Memory API</i> <i>Vector API</i> Simple Web Server JavaDoc Code Snippets	<i>Pattern Matching</i> <i>Foreign Function & Memory API</i> <i>Vector API</i> <i>Virtual Threads</i> <i>Structured Concurrency</i>	<i>Pattern Matching</i> <i>Foreign Function & Memory API</i> <i>Vector API</i> <i>Virtual Threads</i> <i>Structured Concurrency</i> <i>Scoped Values</i>	

Legende: **final** - *Preview/Inkubator*

Die nächste Version mit LTS-Support embarc.de **27**

27

Features **Pattern Matching**
JEPs added *Unnamed Patterns*
Foreign Function & Memory API
JEPs targeted to *Vector API*
Virtual Threads / Structured Concurrency /
Scoped Values
String Templates
Sequenced Collections
Unnamed Classes and Instance Main Methods

Java **21: 15** JEPs

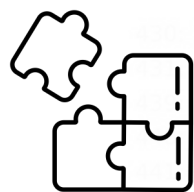


Die nächste Version mit LTS-Support

embarc.de

28

28



Neue Features



Garbage Collectoren



Interna



API-Änderungen (JDK)



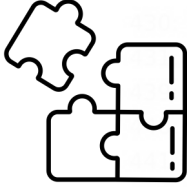
Die nächste Version mit LTS-Support

embarc.de


29

29


Features




Neue Features




Garbage Collectoren



Interna



API-Änderungen (JDK)



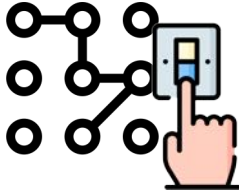
Die nächste Version mit LTS-Support

embarc.de

30


30

17 - 21




Pattern Matching for switch

19 - 21



Record Patterns


21



Unnamed Patterns and Variables


Preview!

19 - 21



Virtual Threads


21



String Templates


Preview!


21



Unnamed Classes & Instance Main Method

Preview!





Die nächste Version mit LTS-Support

embarc.de

31

31

Unnamed Classes & Instance Main Method

```
void main() {
    System.out.println("Hello World")
}
```



wird zu

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



shell> java --source 21 --enable-preview Main.java

In Kombination mit **Launch Single-File Source-Code Programs** (JEP 330)



Die nächste Version mit LTS-Support

embarc.de

32

32

Launch-Protokoll

```
protected static void main() {
    System.out.println("protected static void main()");
}
```

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



Gewinner!



Die nächste Version mit LTS-Support

embarc.de

33

33

Launch-Protokoll

➔ Gewinner!

```
public static void main() {
    System.out.println("protected static void main()");
}

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



Die nächste Version mit LTS-Support

embarc.de

34

34

String Templates (String Interpolation)

```
String ersetzung = 'Ersetzungen'
String s = "Ein GString kann ${ersetzung} enthalten!"

println "Hallo ${user.name}!"
```



Triggerwarnung: Diese Folie ist Original von 2009!



Die nächste Version mit LTS-Support

embarc.de

35

35

Strings in Java

- String-Verkettung mit "+", StringBuffer/-Builder, String.format(), ...
- ~~JEP 326: Raw String Literals (Preview)~~ in Java 12 (keine Sonderzeichen escapen)
- Text-Blocks ab Java 13 (mehrzeilige Strings)
- 430: **String Templates** (Preview) ab Java 21



```
User user = new User("dieter@develop.de",
    new Role("admin"));

String info = STR."User '{user.email()}' hat die
Rolle '{user.role().name()}';

// User 'dieter@develop.de' hat die Rolle 'admin'
System.out.println(info);
```



Warum `\{variable}`?

Pre Java 21

```
public static void main(String[] args) {
    String s = "\{args}";
}
```

Kompiler-Fehler:
java: illegal escape character

Abwärtskompatibilität!



```
String title = "My Web Page";
String text = "Hello, world";
String html = STR."\"
    <html>
    <head>
    <title> \{
        // Comment
        "My company: " + title
    }
    </title>
</head>
<body>
    <p>\{text}</p>
</body>
</html> \"";
```



FormatProcessor

```
double zahl = 2.0;

String formatted = FMT."Zahl: %7.2f\{zahl}";

System.out.println(formatted); // Zahl: 2.00
```



RAW Processor

```
var template = RAW."Today is day
\{LocalDate.now().getDayOfYear()} of year
\{LocalDate.now().getYear()}.";

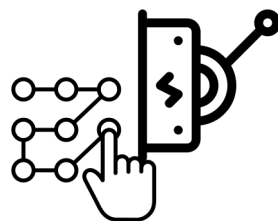
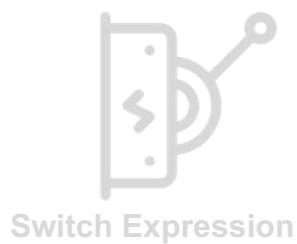
System.out.println(template.fragments());
System.out.println(template.values());
String result = template.process(STR);
System.out.println(result); // Today is day ...
```



Custom Processor

```
StringTemplate.Processor<ResultSet, SQLException> SQL = template -> {
    String sql = template.fragments().stream().collect(Collectors.joining(" ? "));
    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    ...
    return preparedStatement.executeQuery();
};

var result = SQL.""
    select id, email, enabled from Users u
    where u.email = \{user.email()}
    and u.enabled = \{user.enabled()};
result.next();
System.out.println(STR."Found: \{result.getString("email")});
```



Pattern Matching

„Pattern matching is a mechanism for **checking a value against a pattern**. A successful match can also **deconstruct a value into its constituent parts**.

It is a more **powerful version of the switch statement in Java** and it can likewise be used in place of a series of if/else statements.“



<https://docs.scala-lang.org/tour/pattern-matching.html>



Die nächste Version mit LTS-Support

embarc.de

44

44

Neues Programmierparadigma?

The screenshot shows an article on InfoQ.com. The title is "Data Oriented Programming in Java". It is dated JUN 20, 2022 and is 22 MIN READ. The author is Brian Goetz, Java Language Architect at Oracle. It is reviewed by Daniel Bryant, Dev-Rel & Technical Leader at InfoQ News Manager. Key Takeaways include: "Project Amber has brought a number of new features to Java in recent years. While each of these features are self-contained, they are also designed to work together. Specifically, records, sealed classes, and pattern matching work together to enable easier data-oriented programming in Java."



<https://www.infoq.com/articles/data-oriented-programming-java/>

<https://slides.nipafx.dev/patterns/#/>

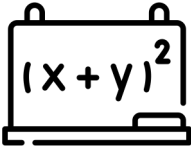


Die nächste Version mit LTS-Support

embarc.de


45

45




Algebraische Datentypen

Sealed Classes




Summentypen


Records



Produkttypen



es gibt noch mehr: Quotienten-, Aufzählungstypen, ...



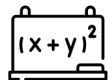
Die nächste Version mit LTS-Support

embarc.de

46

46

Liste als algebraischer Datentyp



>> data List a = Nil | Cons a (List a)

[]


:: oder :

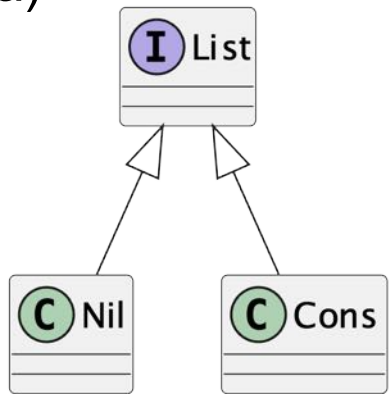
= Construct


Cons 1 (Cons 2 (Cons 3 Nil))

1:2:3:[]

[1, 2, 3]







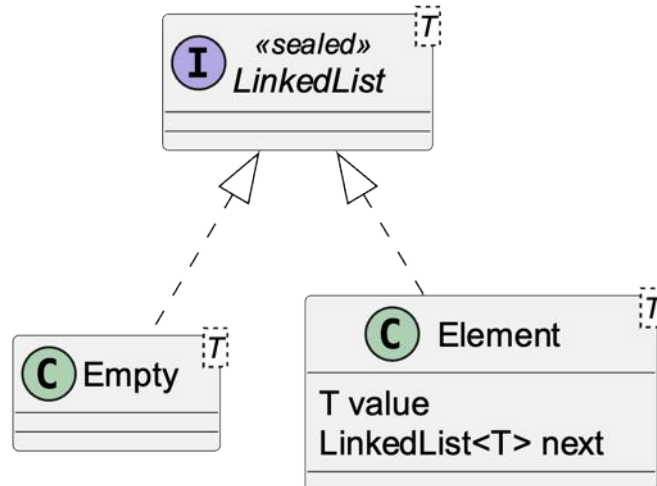
Die nächste Version mit LTS-Support

embarc.de

47

47

Algebraische Datentypen



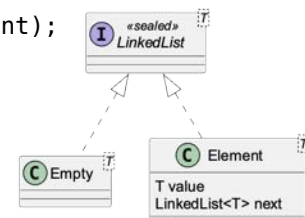
```

public sealed interface LinkedList<T>
    permits LinkedList.Element, LinkedList.Empty {

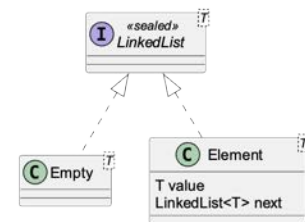
    record Element<T>(T value, LinkedList<T> next)
        implements LinkedList<T> {}

    final class Empty<T> implements LinkedList<T> {}

    static <T> LinkedList<T> of(T... values) {
        if (values.length == 0) return new LinkedList.Empty<>();
        LinkedList<T> current = new LinkedList.Empty<>();
        for (int i = values.length - 1; i >= 0; i--) {
            current = new LinkedList.Element<>(values[i], current);
        }
        return current;
    }
}
  
```



```
LinkedList<Integer> list = of(1, 2, 3);
System.out.println(contains(5, list)); // false
System.out.println(contains(1, list)); // true
```



Die nächste Version mit LTS-Support

embarc.de

50

50

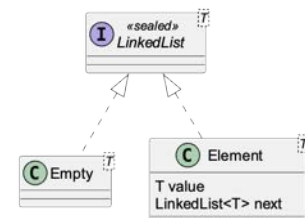
```
static <T> boolean contains(T value, LinkedList<T> list) {
    return switch (list) {
        case Empty empty -> false;
        case Element<T>(T v, var tail)
            when Objects.equals(v, value) -> true;
        case Element<T>(T v, var tail) -> contains(value, tail);
    };
}
```

switch-Expression

Type Pattern
(Pattern Matching for instanceof)

Record Pattern

when-Clause
(Guarded Pattern)



Die nächste Version mit LTS-Support

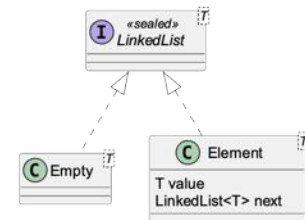
embarc.de

51

51


```
static <T> boolean contains(T value, LinkedList<T> list) {
    return switch (list) {
        case Empty _ -> false;
        case Element<T>(T v, _)
            when Objects.equals(v, value) -> true;
        case Element<T>(_, var tail) -> contains(value, tail);
    };
}
```

Unnamed Pattern



Unnamed Variables

```
try {
    var result = 5 / 0;
} catch(ArithmeticException _) {
    System.out.println("Division nicht möglich");
}

System.out.println(new ArrayList<>(List.of(1, 2, 3))
    .stream()
    .map(_ -> 42)
    .toList()); // [42, 42, 42]
```



```

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();
}

```



Die nächste Version mit LTS-Support

embarc.de

54

54

```

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

```



Die nächste Version mit LTS-Support

embarc.de

56

56

feingranulare Steuerung der Vererbungshierarchie

Algebraische Datentypen

Prüfung der "Exhaustiveness" im switch



Sealed
Classes



Die nächste Version mit LTS-Support

embarc.de

57

57

Flexible Klassenhierarchien

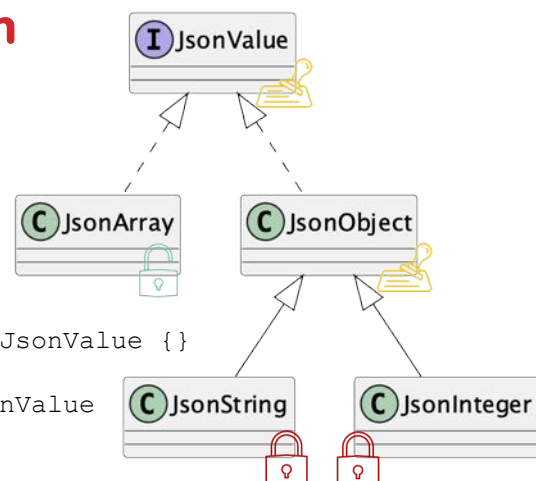
```
public sealed interface JsonValue
    permits jsonArray, JsonObject {

    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



Die nächste Version mit LTS-Support

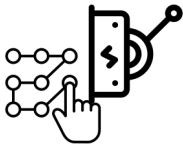
embarc.de

58

58

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;
    };
}
```



Die nächste Version mit LTS-Support

embarc.de

61

61

Virtual Threads

JVM features and APIs for supporting easy-to-use, high-throughput, lightweight concurrency and new programming models.



Die nächste Version mit LTS-Support

embarc.de

64

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();
}
```



Die nächste Version mit LTS-Support

embarc.de

65

65

Aber es geht gar nicht um Millionen von Threads

- **klassisches, gleiches Thread-API** 😊
- **Vereinfachter Umgang mit einer Menge von teilweise blockierenden Aufgaben (I/O)**
 - blockiert ein Virtual Thread, wird er geparkt und der nächste ist dran
 - schnell erzeugt und leichtgewichtig -> kein Pooling notwendig, jede Aufgabe in einem neuen Thread starten
- **Virtual Threads sind nicht schneller oder effizienter (laufen auch nur auf Carrier/Platform Threads)** 😞
- **nicht für sehr rechenintensive Tasks (die nicht blockieren)**



Die nächste Version mit LTS-Support

embarc.de

66

66

Bisher: 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());
```



Die nächste Version mit LTS-Support

embarc.de

67

67

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());
}
```



basiert auf
Virtual Threads



Die nächste Version mit LTS-Support

embarc.de

68

68

Scoped Values

```
public final static ScopedValue<SomeUser> CURRENT_USER
    = ScopedValue.newInstance();

public void someControllerAction(HttpServletRequest request) {
    SomeUser user = authenticate(request);
    ScopedValue.where(CURRENT_USER, user)
        .run(() -> someService.processService());
    someService.processService();
}

void processService() {
    System.out.println(CURRENT_USER
        .orElseThrow(() -> new RuntimeException("not valid")));
}
```



Loom – Chancen und Limitierungen

- nicht für die Verarbeitung großer Datenmengen bzw. schwere Berechnungen
- Leichter für "naive" Programmierung von Threads
- Webcontainer können Virtual Threads einsetzen, um Speicher zu sparen
 - mehr parallele Requests
 - aber Limitierung bleiben Ressourcen außerhalb der JVM (Datenbanken, ..)



Für wen ist es gut?



Virtual Threads unter der Haube

Wir alle, die Webanwendungen bauen und davon **indirekt profitieren**.



Nebenläufige Programmierung

Entwickler, die **sehr viele** und/oder **blockierende Aufgaben** implementieren müssen (einfacheres API).



Die nächste Version mit LTS-Support

embarc.de

71

71



Neue Features



Garbage Collectoren



Interna



API-Änderungen (JDK)



Die nächste Version mit LTS-Support

embarc.de

72

72

Neue Generation von Low-Latency Garbage Collectoren

moderne Architekturen: Multi-Core + TB RAM

Ziel: kurze GC-Pausen im ms-Bereich



Die nächste Version mit LTS-Support

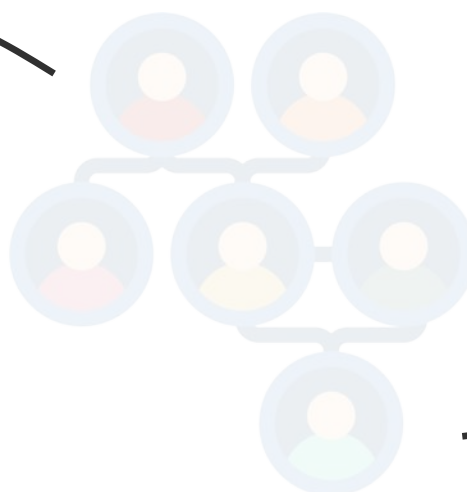
embarc.de

73

73

JEP 439: Generational ZGC

Old-Gen:
hat schon
mehrere GC-
Läufe überlebt,
wird länger leben



Young-Gen:
wird im GC-Lauf
prozessiert




Die nächste Version mit LTS-Support

embarc.de


74

74


Features




Neue Features




Garbage Collectoren



Interna



API-Änderungen (JDK)




Die nächste Version mit LTS-Support

embarc.de

75

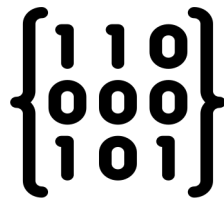
75

14 – 21 ff.





Foreign Linker + Foreign-Memory Access API

14 – 21 ff.



Vector API






Die nächste Version mit LTS-Support

embarc.de


76

76


Features



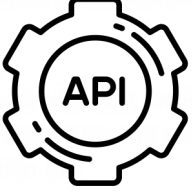
Neue Features




Garbage Collectoren



Interna



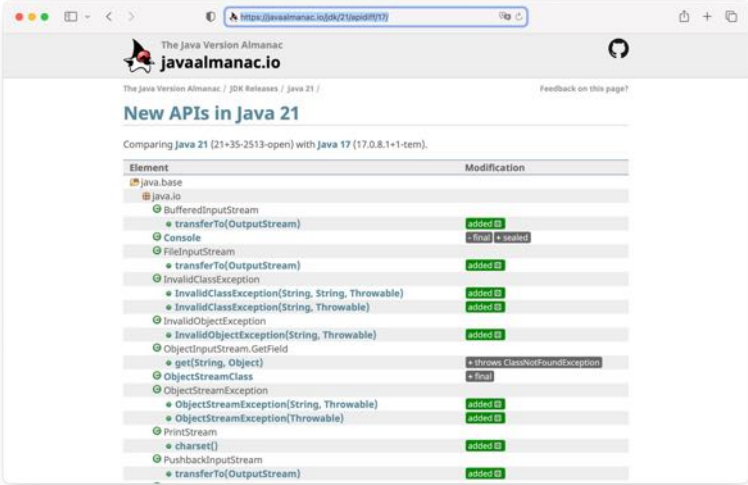
API-Änderungen (JDK)




Die nächste Version mit LTS-Support

embarc.de

77



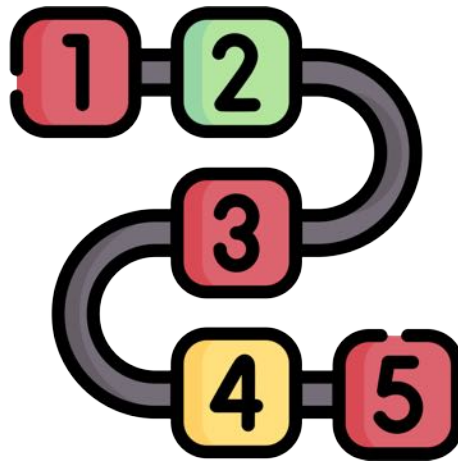
<https://javaalmanac.io/jdk/21/apidiff/17/>



Die nächste Version mit LTS-Support

embarc.de

78



Sequenced Collection

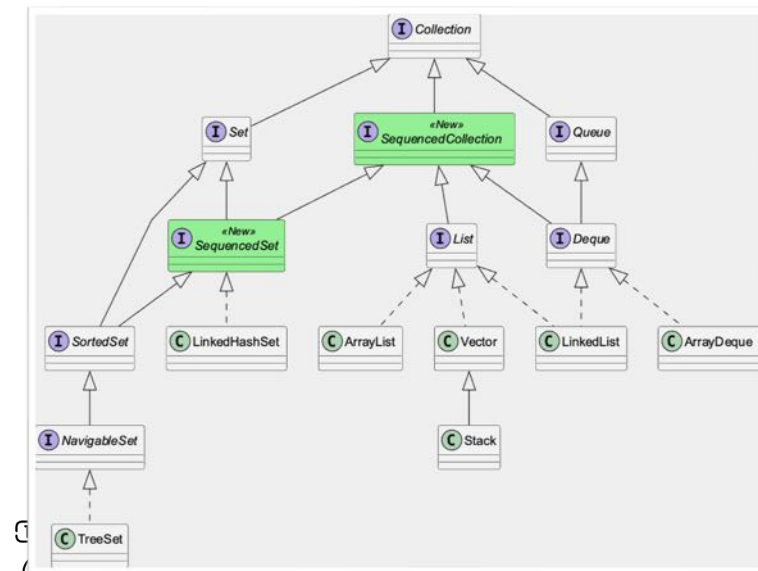


Die nächste Version mit LTS-Support

embarc.de

79

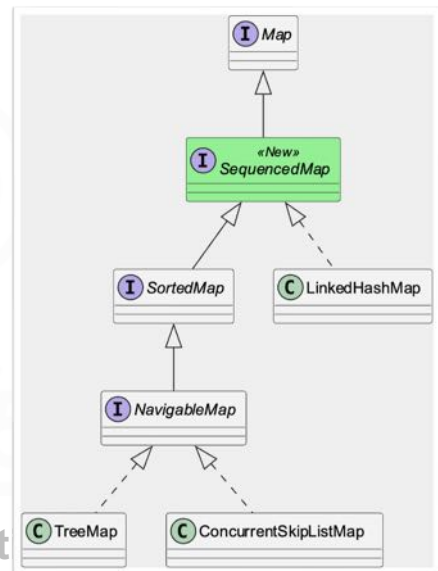
79



Die nächste Version mit LTS-Support

embarc.de

80



80

```

List<Integer> list = new ArrayList<>(List.of(1, 2, 3, 4, 5));
List<Integer> reversed = list.reversed();
System.out.println(reversed); // 5, 4, 3, 2, 1

list.addFirst(0);
list.addLast(6);

System.out.println(list.getFirst()); // 0
System.out.println(list.getLast()); // 6

```



```

SequencedMap<Integer, String > map =
    new LinkedHashMap<>(Map.of(1, "a", 2, "b"));

SequencedMap<Integer, String> reversedMap = map.reversed();
System.out.println(map); // {2=b, 1=a}
System.out.println(reversedMap); // {1=a, 2=b}

map.putFirst(0, "o");
map.putLast(3, "c");

System.out.println(map.pollFirstEntry()); // 0=o
System.out.println(reversedMap); // {3=c, 2=b, 1=a}

System.out.println(map.sequencedKeySet()
    .reversed()); // [3, 2, 1]

```



Aktivierung Preview + Inkubator Features

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

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



Welche Änderungen gab es noch seit Java 17?



Features

<https://openjdk.org/projects/jdk/21/>

- 430: String Templates (Preview)
- 431: Sequenced Collections
- 439: Generational ZGC
- 440: Record Patterns
- 441: Pattern Matching for switch
- 442: Foreign Function & Memory API (Third Preview)
- 443: Unnamed Patterns and Variables (Preview)
- 444: Virtual Threads
- 445: Unnamed Classes and Instance Main Methods (Preview)
- 446: Scoped Values (Preview)
- 448: Vector API (Sixth Incubator)
- 449: Deprecate the Windows 32-bit x86 Port for Removal
- 451: Prepare to Disallow the Dynamic Loading of Agents
- 452: Key Encapsulation Mechanism API
- 453: Structured Concurrency (Preview)

JDK 21 will be a long-term support (LTS) release from most vendors. For a complete list of the JEPs integrated since the previous LTS release, JDK 17, please see [here](#).

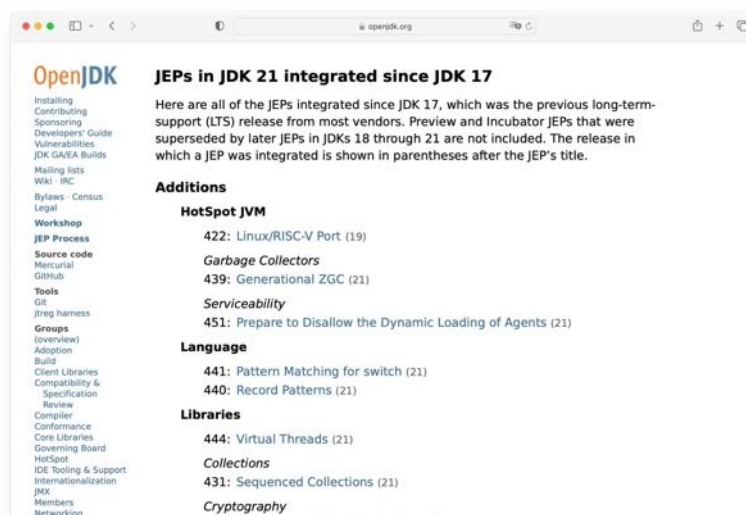


Die nächste Version mit LTS-Support

embarc.de

85

85



OpenJDK **JEPs in JDK 21 integrated since JDK 17**

Here are all of the JEPs integrated since JDK 17, which was the previous long-term-support (LTS) release from most vendors. Preview and Incubator JEPs that were superseded by later JEPs in JDKs 18 through 21 are not included. The release in which a JEP was integrated is shown in parentheses after the JEP's title.

Additions

HotSpot JVM

- 422: Linux/RISC-V Port (19)

Garbage Collectors

- 439: Generational ZGC (21)

Serviceability

- 451: Prepare to Disallow the Dynamic Loading of Agents (21)

Language

- 441: Pattern Matching for switch (21)
- 440: Record Patterns (21)

Libraries

- 444: Virtual Threads (21)
- 431: Sequenced Collections (21)
- Cryptography



Die nächste Version mit LTS-Support

embarc.de

<https://openjdk.org/projects/jdk/21/jeps-since-jdk-17>

86

86

```
/**
 * 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:	
		<pre>if (v.isPresent()) { System.out.println("v: " + v.get()); }</pre>	



Code Snippets in JavaDoc



Die nächste Version mit LTS-Support

embarc.de

87

87

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.createServer(
    new InetSocketAddress(8888), Path.of("out"),
    OutputLevel.INFO);
server.start();
```



Simple Webserver



Die nächste Version mit LTS-Support

embarc.de

88

88

Auf Wiedersehen ...

- 🎯 **Applet API** deprecated for Removal
- 🎯 **Security Manager** deprecated for Removal
- 🎯 Deprecate **Finalization** for Removal
- 💀 Experimental **AOT and JIT Compiler**



Agenda



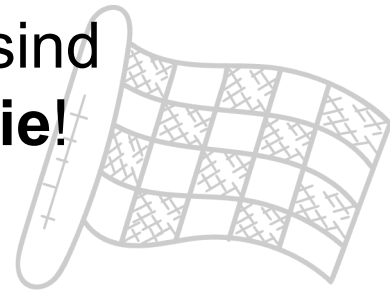
- 1 Java – still a thing?
- 2 Ist Java 21 ein LTS-Release?
- 3 Viele spannende neue Features
- 4 **Fazit**

4





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



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



Was könnte in Java 22 konkret kommen?

JEP 447: Statements before super() (Preview)

```
public PositiveBigInteger(long value) {
    // Invalid in Java 21
    if (value <= 0) throw new
        IllegalArgumentException("non-positive value");
    super(value);
}
```

JEP 455: Primitive types in Patterns, instanceof, and switch (Preview)

JEP draft: Ahead of Time Compilation for the Java Virtual Machine



Die nächste Version mit LTS-Support

embarc.de

94

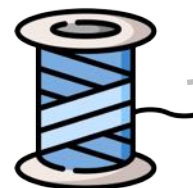
94

Pattern Matching +
kleine Syntax-
Verbesserungen



Amber

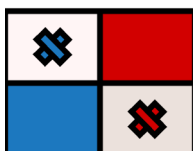
Virtual Threads



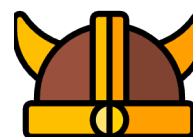
Loom

Was
kommt
noch?

Vector API +
Foreign Function
& Memory API



Panama



Valhalla

Value Types



Die nächste Version mit LTS-Support

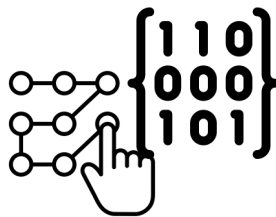
embarc.de

95

95

Array Patterns

```
static void printFirstTwoStrings(Object o) {
    if (o instanceof String[] { String s1, String s2, ... }) {
        System.out.println(s1 + s2);
    }
}
```



Die nächste Version mit LTS-Support

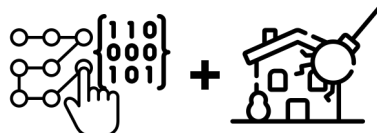
embarc.de

96

96

Kombination: Array & Records 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);
    }
}
```



Die nächste Version mit LTS-Support

embarc.de

97

97

Record Patterns in foreach-Schleifen



MerlinBoe
@MBoegie

...

Latest news for our @Baselone workshop!! Awesome iteration with `for(Point(var x, var y) : points)` getting possible 🎉🎉🎉 #java #patterns

[Tweet übersetzen](#)

OpenJDK @OpenJDK · 19 Std.

New candidate JEP: 432: Record Patterns (Second Preview):
openjdk.org/jeps/432 #openjdk #java

8:18 vorm. · 19. Okt. 2022 · Twitter for iPhone



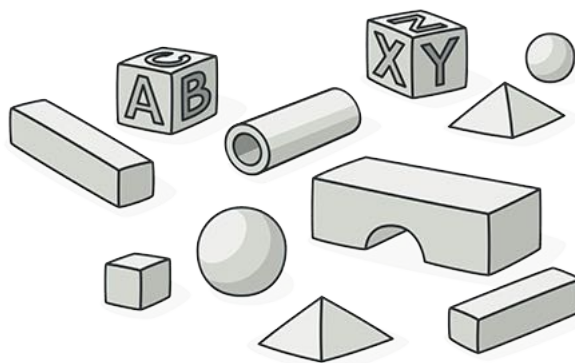
Die nächste Version mit LTS-Support

embarc.de

98

98

Primitive Obsession



<https://refactoring.guru/smells/primitive-obsession>



Die nächste Version mit LTS-Support

embarc.de

99

99

Project Valhalla

"Advanced Java VM and Language feature candidates."

- Java kennt **primitive** Typen und **Klassen**
- **Custom Types** nur über **Klassen**
 - haben **Identität** und sind **Referenzen**
 - **Mutable, extra Memory** für Header, Locking/Synchronisierung, **Heap vs. Stack** (Speicher-Indirektion), **Nullable**

Nicht alle benutzer-definierten Typen brauchen das



Die nächste Version mit LTS-Support

embarc.de

100

100

Value Types



```
value class Point {
    private int x;
    private int y;

    public implicit Point();

    public Point(int x, int y) { .. }

    public Distance distanceTo(Point p) { .. }
}
Point! point = .. // can't be null
```



Die nächste Version mit LTS-Support

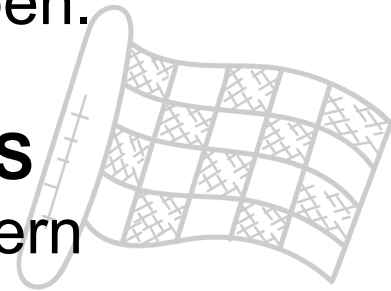
embarc.de

101

101

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

**Das nächste LTS
Release ist nicht fern**



...

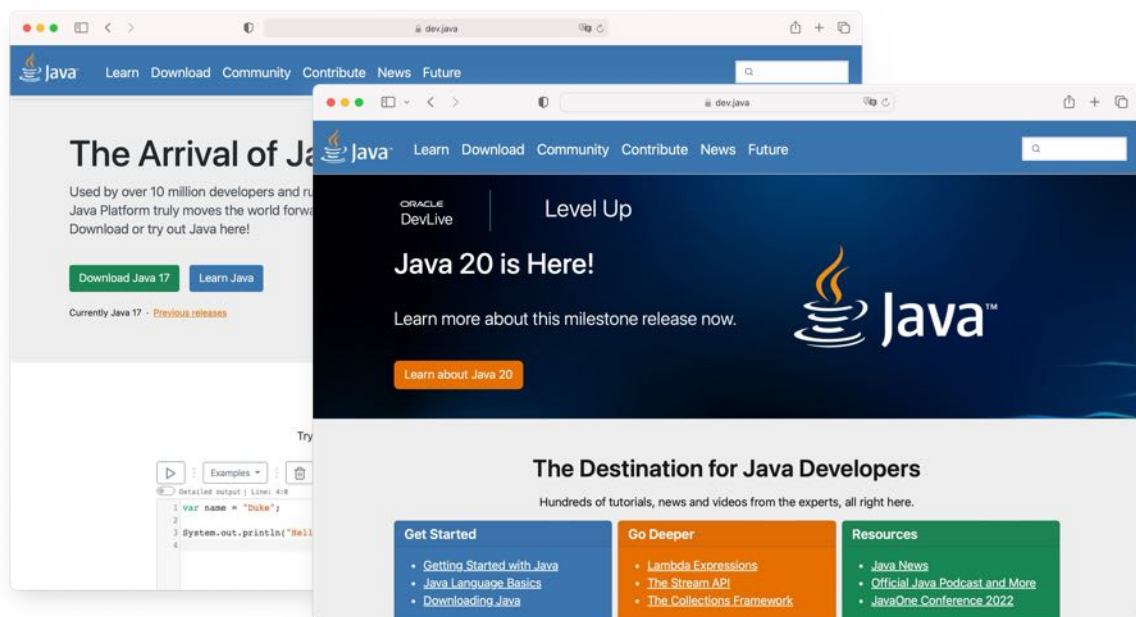


Die nächste Version mit LTS-Support

embarc.de

102

102



Die nächste Version mit LTS-Support

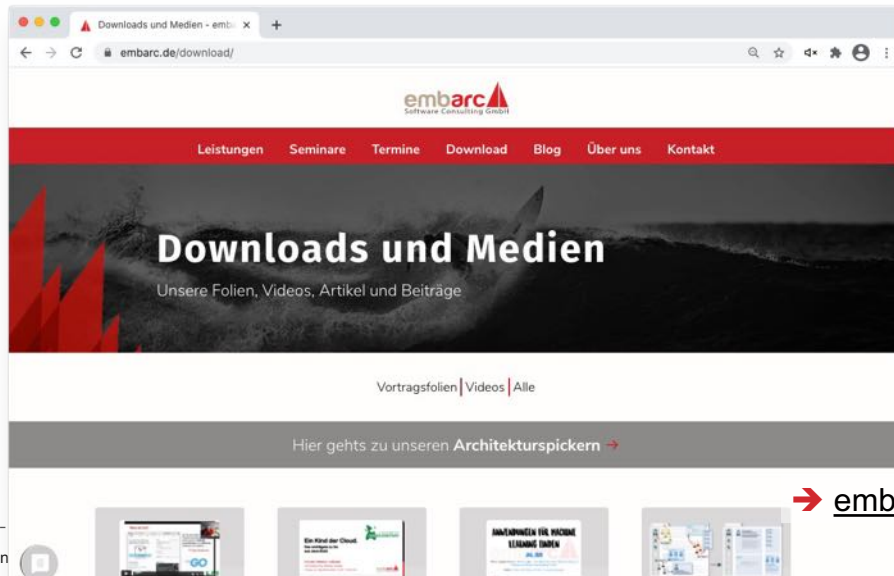
embarc.de

<https://dev.java/>

103

103


Folien von heute als PDF zum Download




→ embarc.de/download/

104


104

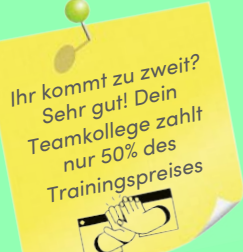
 **SOCREATORY**
The Software Creators' Academy

EINE KOLLABORATION VON **INOQ** 

Java Update Trainings


9 bis 11, 17, 21

Sprecht mich an 


Ihr kommt zu zweit?
Sehr gut! Dein
Teamkollege zahlt
nur 50% des
Trainingspreises

User Preismodell – Flexibel und transparent

Aktuelle
Termine:
socreatory.com



105

Vielen Dank.

Ich freue mich auf Eure Fragen!



Falk Sippach



fs@embarc.de



@sipp sack



→ [xing.to/fsi](https://www.xing.to/fsi)



Die nächste Version mit LTS-Support

embarc.de

106