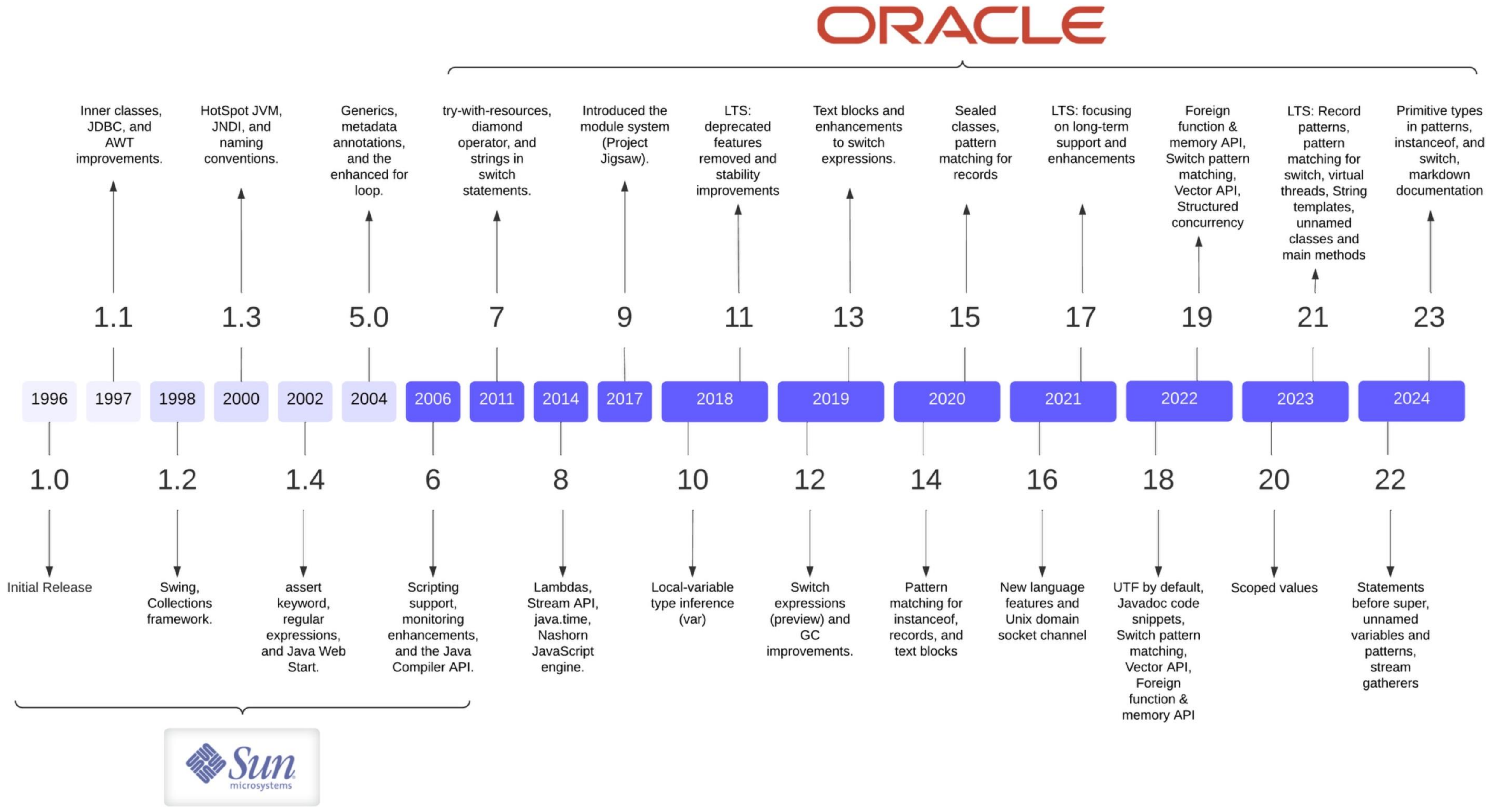
Java 8 bis 23

Simon Martinelli
martinelli.ch

Resources

• Java Developer Portal	https://dev.java
• Open JDK	https://openjdk.org
• Friends of OpenJDK	https://foojay.io
• Java User Groups	https://dev.java/community/jugs/
• JUG Switzerland	https://jug.ch

Next Event: Monday, 25.11.2024, 18:00 in Bern


ORACLE
Inner classes, JDBC, and AWT improvements.
HotSpot JVM, JNDI, and naming conventions.
Generics, metadata annotations, and the enhanced for loop.
try-with-resources, diamond operator, and strings in switch statements.
Introduced the module system (Project Jigsaw).
LTS: deprecated features removed and stability improvements
Text blocks and enhancements to switch expressions.
Sealed classes, pattern matching for records
LTS: focusing on long-term support and enhancements
Foreign function & memory API, Switch pattern matching, Vector API, Structured concurrency
LTS: Record patterns, pattern matching for switch, virtual threads, String templates, unnamed classes and main methods
Primitive types in patterns, instanceof, and switch, markdown documentation
1.1
1.3
5.0
7
9
11
13
15
17
19
21
23
1996
1997
1998
2000
2002
2004
2006
2011
2014
2017
2018
2019
2020
2021
2022
2023
2024
1.0
1.2
1.4
6
8
10
12
14
16
18
20
22
Initial Release
Swing, Collections framework.
assert keyword, regular expressions, and Java Web Start.
Scripting support, monitoring enhancements, and the Java Compiler API.
Lambdas, Stream API, java.time, Nashorn JavaScript engine.
Local-variable type inference (var)
Switch expressions (preview) and GC improvements.
Pattern matching for instanceof, records, and text blocks
New language features and Unix domain socket channel
UTF by default, Javadoc code snippets, Switch pattern matching, Vector API, Foreign function & memory API
Scoped values
Statements before super, unnamed variables and patterns, stream gatherers
Sun microsystems

API Changes

- **The Java Version Almanac**
  https://javaalmanac.io

  Maintained by Marc R. Hoffmann from Bern
  (JaCoCo Project Lead)

Maven

```xml
<properties>
    <!-- >= JDK 9 -->
    <maven.compiler.release>21</maven.compiler.release>
    <!-- < JDK 9 -->
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
</properties>


<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
        <!-- overrides properties -->
        <source>8</source>
        <target>8</target>
        <release>21</release>

        <!-- To test preview features -->
        <compilerArgs>--enable-preview</compilerArgs>
    </configuration>
</plugin>
```

Text Blocks (Basics) (Preview 13, Standard 15)

```java
// Text blocks start with a """ followed by optional whitespaces and a newline.
// The simplest example looks like this:
String sql1 = """
        select * from employee""";

// Trailing whitespaces are removed
String sql2 = """
        select *
        from employee""";

// \s (escape sequence) can be used to keep the blank after *
String sql3 = """
        select * \s
        from employee""";

// Use \ to continue next line without line break
String sql3 = """
        select * \
        from employee""";
```

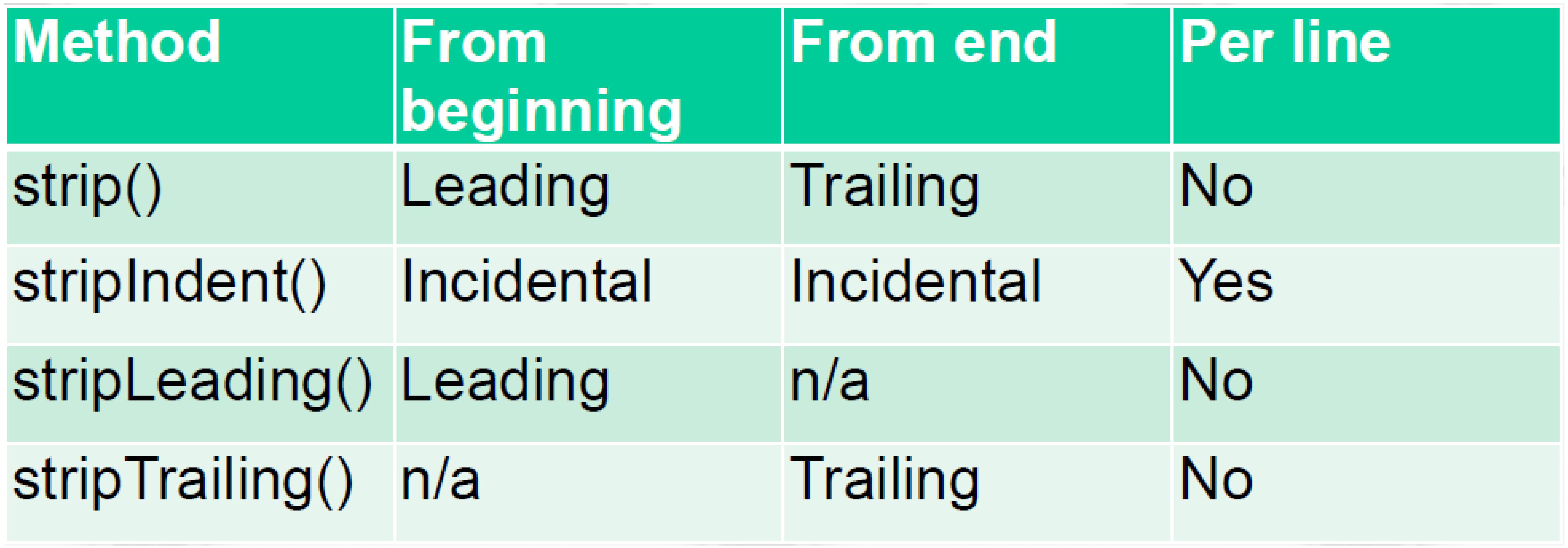
Text Block (Indention)

```
// A text block differentiates incidental white space from essential white space

String html1 = """
        <html>
            <body>
               <p>Hello World.</p>
            </body>
        </html>
        """;

String html2 = """
        <html>
            <body>
               <p>Hello World.</p>
            </body>
        </html>
""";
```

String Methods (Standard 15)



Method	From beginning	From end	Per line
strip()	Leading	Trailing	No
stripIndent()	Incidental	Incidental	Yes
stripLeading()	Leading	n/a	No
stripTrailing()	n/a	Trailing	No

String Methods (Standard 12)

```
// This method is equivalent to String.format(this, args). The advantage is that, as an
instance method,
// it can be chained off the end of a text block:
String formatted(Object... args)

String output = """
    Name: %s
    Phone: %s
    Address: %s
    Salary: $%.2f
    """.formatted(name, phone, address, salary);


// Allows to transform a String with a Function
public <R> R transform(Function<? super String,? extends R> f)

String lower = "ABC".transform(s -> s.toLowerCase());
```

Switch Expression (Preview 13, Standard 14)

```java
// Switch Statement

int numLetters;

switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        numLetters = 6;
        break;
    case TUESDAY:
        numLetters = 7;
        break;
    case THURSDAY:
    case SATURDAY:
        numLetters = 8;
        break;
    case WEDNESDAY:
        numLetters = 9;
        break;
    default:
        throw new IllegalStateException(
                      "Invalid day: " + day);
}
```

```java
// Switch Expression

int numLetters = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> 6;
    case TUESDAY -> 7;
    case THURSDAY, SATURDAY -> 8;
    case WEDNESDAY -> 9;
    default -> throw new IllegalStateException(
                              "Invalid day: " + day);
};
```

Switch Expression yield

```java
Day day = Day.WEDNESDAY;
int numLetters = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> {
        System.out.println(6);
        yield 6;
    }
    case TUESDAY -> {
        System.out.println(7);
        yield 7;
    }
    case THURSDAY, SATURDAY -> {
        System.out.println(8);
        yield 8;
    }
    case WEDNESDAY -> {
        System.out.println(9);
        yield 9;
    }
    default -> throw new IllegalStateException("Invalid day: " + day);
};
```

Switch Statement yield

```java
Day day = Day.WEDNESDAY;
int numLetters = switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        System.out.println(6);
        yield 6;
    case TUESDAY:
        System.out.println(7);
        yield 7;
    case THURSDAY:
    case SATURDAY:
        System.out.println(8);
        yield 8;
    case WEDNESDAY:
        System.out.println(9);
        yield 9;
    default:
        throw new IllegalStateException("Invalid day: " + day);
};
```

Records

Preview: 14
Final: 16

```java
record Rectangle(
        double length, double width) {
}
```

```java
public final class Rectangle {
    private final double length;
    private final double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    double length() { return this.length; }
    double width()  { return this.width; }

    public boolean equals...
    public int hashCode...

    public String toString() {...}
}
```

Records with Constructor

```
record Rectangle(double length, double width) {

    public Rectangle {
        if (length <= 0 || width <= 0) {
             throw new java.lang.IllegalArgumentException(
                     String.format("Invalid dimensions: %f, %f", length, width));
        }
    }
}
```

Records Mutability

```
public record Book (String title, int numPages, List<String> chapters) {
}


// chapters is a mutable collection!

Book book = new Book("Breaking and entering", 289, chapters);

System.out.println(book.title());
System.out.println(book.toString());

chapters.add("2");
book.chapters().add("3");
```

Records Immutable Collection

```
public record Book (String title, int numPages, List<String> chapters) {

    public Book {
        chapters = List.copyOf(chapters);
    }
}
```

Sealed Classes (Preview 15, Standard 17)

```
public abstract sealed class Shape
        permits Circle, Rectangle, Square, WeirdShape { ... }

public final class Circle extends Shape { ... }

public sealed class Rectangle extends Shape
        permits TransparentRectangle, FilledRectangle { ... }
public final class TransparentRectangle extends Rectangle { ... }
public final class FilledRectangle extends Rectangle { ... }

public final class Square extends Shape { ... }

public non-sealed class WeirdShape extends Shape { ... }
```

Pattern Matching for instanceof
(Preview 14, Standard 16)

```java
// Without Pattern Matching

if (shape instanceof Rectangle) {
    Rectangle r = (Rectangle) shape;
    return 2 * r.length() + 2 * r.width();
} else if (shape instanceof Circle) {
    Circle c = (Circle) shape;
    return 2 * c.radius() * Math.PI;
}
```

```java
// With Pattern Matching

if (shape instanceof Rectangle r) {
    return 2 * r.length() + 2 * r.width();
} else if (shape instanceof Circle c) {
    return 2 * c.radius() * Math.PI;
}
```

Pattern Matching Flow Scope

```java
// Compiles
if (num instanceof Double d1 && d1.intValue() % 2 == 0) {
    System.out.println(d1.intValue());
}

// Does not compile! d2 might not be Double
if (num instanceof Double d2 || d2.intValue() % 2 == 0) {
    System.out.println(d2.intValue());
}
```

Pattern Matching for switch (Preview 17)

```java
static double getPerimeter(Shape shape) throws IllegalArgumentException {
    return switch (shape) {
                case Rectangle r -> 2 * r.length() + 2 * r.width();
                case Circle c    -> 2 * c.radius() * Math.PI;
                default          -> throw new IllegalArgumentException("Unrecognized shape");
    };
}


// Selector Expresssion Type
static void typeTester(Object obj) {
    switch (obj) {
        case null     -> System.out.println("null");
        case String s -> System.out.println("String");
        case Color c  -> System.out.println("Color with " + c.values().length + " values");
        case Point p  -> System.out.println("Record class: " + p.toString());
        case int[] ia -> System.out.println("Array of int values of length" + ia.length);
        default       -> System.out.println("Something else");
    }
}
```

Helpful NullPointerExceptions (14)

```
static void foo(String name) {
    if (name.equals("Simon")) { // Line 52
      // ...
    }
}
```

```
Exception in thread "main" java.lang.NullPointerException:
    Cannot invoke "String.equals(Object)" because "name" is null
        at io.seventytwo.edu.Examples.foo(Examples.java:52)
        at io.seventytwo.edu.Examples.main(Examples.java:16)
```

Compact Number Formatting

```java
NumberFormat fmt = NumberFormat.getCompactNumberInstance(Locale.ENGLISH,
NumberFormat.Style.SHORT);

System.out.println(fmt.format(1000));
System.out.println(fmt.format(100000));
System.out.println(fmt.format(1000000));


// Output
1K
100K
1M
```

Day Period Support Added

```java
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("B");

System.out.println(dtf.format(LocalTime.of(8, 0)));
System.out.println(dtf.format(LocalTime.of(13, 0)));
System.out.println(dtf.format(LocalTime.of(20, 0)));
System.out.println(dtf.format(LocalTime.of(23, 0)));
System.out.println(dtf.format(LocalTime.of(0, 0)));


// Output
in the morning
in the afternoon
in the evening
at night
midnight
```

Stream.toList()

```
// Old (mutable list)
List<String> stringList =  Stream.of("a", "b", "c").collect(Collectors.toList());

// New (immutable List)
List<String> stringList =  Stream.of("a", "b", "c").toList();
```

Local Records, Enums, Interfaces

```java
List<Merchant> findTopMerchants(List<Merchant> merchants, int month) {

    // Local record
    record MerchantSales(Merchant merchant, double sales) {}

    return merchants.stream()
            .map(merchant -> new MerchantSales(merchant, computeSales(merchant, month)))
            .sorted((m1, m2) -> Double.compare(m2.sales(), m1.sales()))
            .map(MerchantSales::merchant)
            .collect(toList());
}
```

Sealed Classes

Preview: 15
Final: 17

- Sealed classes and interfaces restrict which other classes or interfaces may extend or implement them

```
abstract sealed class Shape
          permits Circle, Rectangle, Square {
}
```

Preview: 12
Final: 14

Switch Expression and Sealed Classes

- Exhaustive: the switch expression must cover all cases

```
abstract sealed class Shape
        permits Circle, Rectangle, Square {
}
```

```
String result = switch (color) {
    case RED -> "Red";
    case GREEN -> "Green";
    case BLUE -> "Blue";
};
```

Record Patterns

Preview: 20
Final: 21

- Use to test whether a value is an instance of a record class type and, if it is, recursively perform pattern matching on its component values

```
record Pair(int left, int right) {}

if (p instanceof Pair(int left, int right)) {
   ...
}
```

Virtual Threads

Preview: 19
Final: 21

- A **platform thread** is implemented as a thin wrapper around an operating system (OS) thread.
  - The number of available platform threads is limited to the number of OS threads.
- A **virtual thread** isn't tied to a specific OS thread.
  - A virtual thread still runs code on an OS thread.
  - When code running in a virtual thread calls a blocking I/O operation, the Java runtime suspends the virtual thread until it can be resumed.
  - The OS thread associated with the suspended virtual thread is now free to perform operations for other virtual threads.

Exercises

- https://github.com/simasch/java23-update

Unnamed Classes and Instance Main Methods

Preview: 21
Final:

```
String greeting = "Hello, World!";

void main() {

  System.out.println(greeting);

}


$ java HelloWorld.java
```

Unnamed Classes and Instance Main Methods

Preview: 21
Final:

```
// java --enable-preview Main.java
record Point(int x, int y) {}
record Rectangle(int width, int height) {}
record Circle(int radius) {}

void main() {
    var point = new Point(1, 2);
    println(describePoint(point));

    var shape = new Circle(5);
    var area = calculateArea(shape);
    println("Area of the shape: %s !%n".formatted(area));
}
```

Preview: 22
Final:

Statements before super()

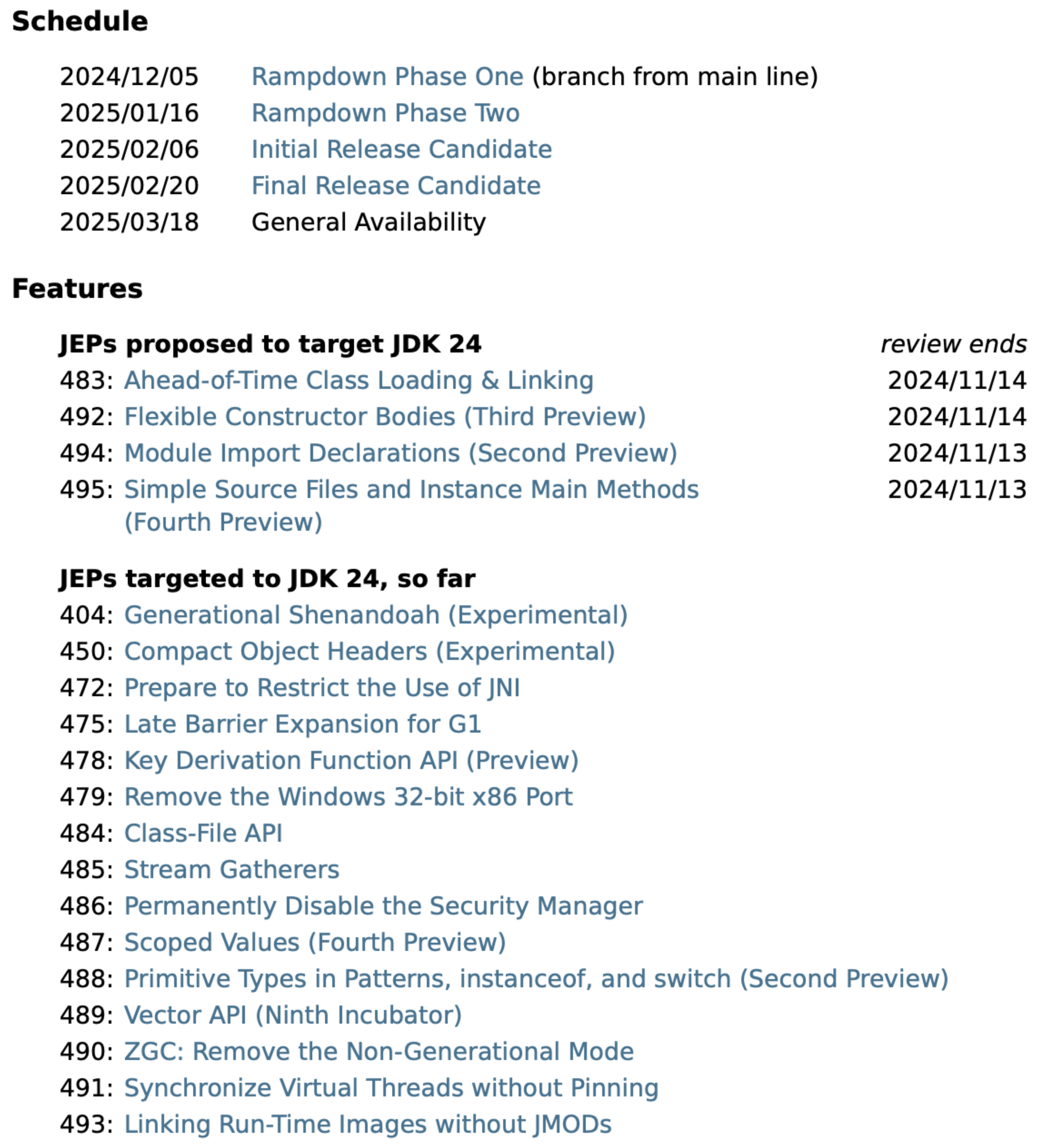
```
public class PositiveBigInteger extends BigInteger {

  // before
  public PositiveBigInteger(long value) {
    super(value); // Potentially unnecessary work
    if (value <= 0)
      throw new IllegalArgumentException("non-positive value");
  }

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

Outlook Java 24

- https://openjdk.org/projects/jdk/24/



Schedule

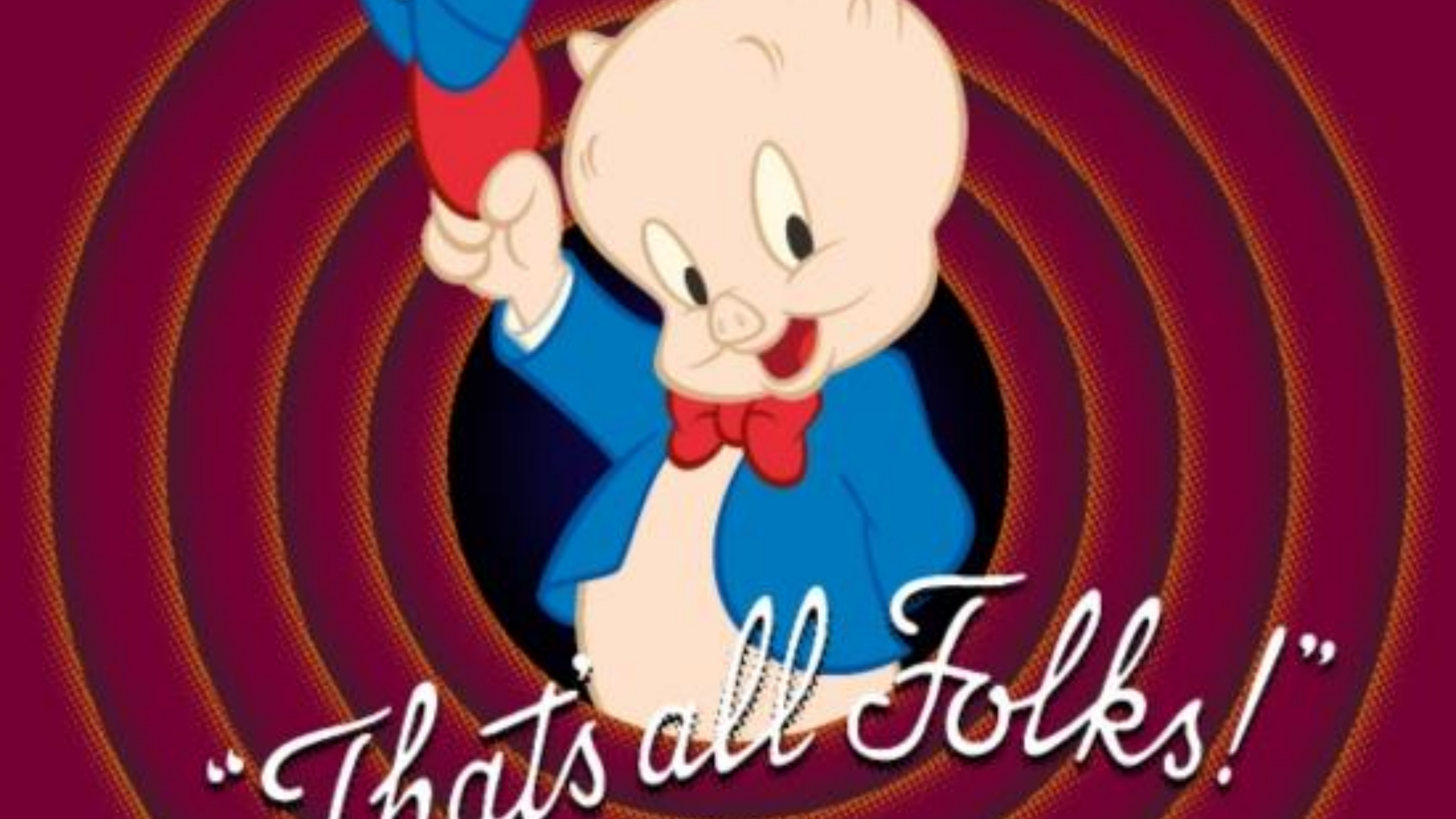
2024/12/05	Rampdown Phase One (branch from main line)
2025/01/16	Rampdown Phase Two
2025/02/06	Initial Release Candidate
2025/02/20	Final Release Candidate
2025/03/18	General Availability

Features

JEPs proposed to target JDK 24	*review ends*
483: Ahead-of-Time Class Loading & Linking	2024/11/14
492: Flexible Constructor Bodies (Third Preview)	2024/11/14
494: Module Import Declarations (Second Preview)	2024/11/13
495: Simple Source Files and Instance Main Methods (Fourth Preview)	2024/11/13

JEPs targeted to JDK 24, so far

404: Generational Shenandoah (Experimental)
450: Compact Object Headers (Experimental)
472: Prepare to Restrict the Use of JNI
475: Late Barrier Expansion for G1
478: Key Derivation Function API (Preview)
479: Remove the Windows 32-bit x86 Port
484: Class-File API
485: Stream Gatherers
486: Permanently Disable the Security Manager
487: Scoped Values (Fourth Preview)
488: Primitive Types in Patterns, instanceof, and switch (Second Preview)
489: Vector API (Ninth Incubator)
490: ZGC: Remove the Non-Generational Mode
491: Synchronize Virtual Threads without Pinning
493: Linking Run-Time Images without JMODs


"That's all Folks!"