Simon Martinelli, 72 Services LLC

About @simas_ch

1972 — 1995 — 2000 — 2007 — 2009 — 2011 — 2012 — 2013 — 2021

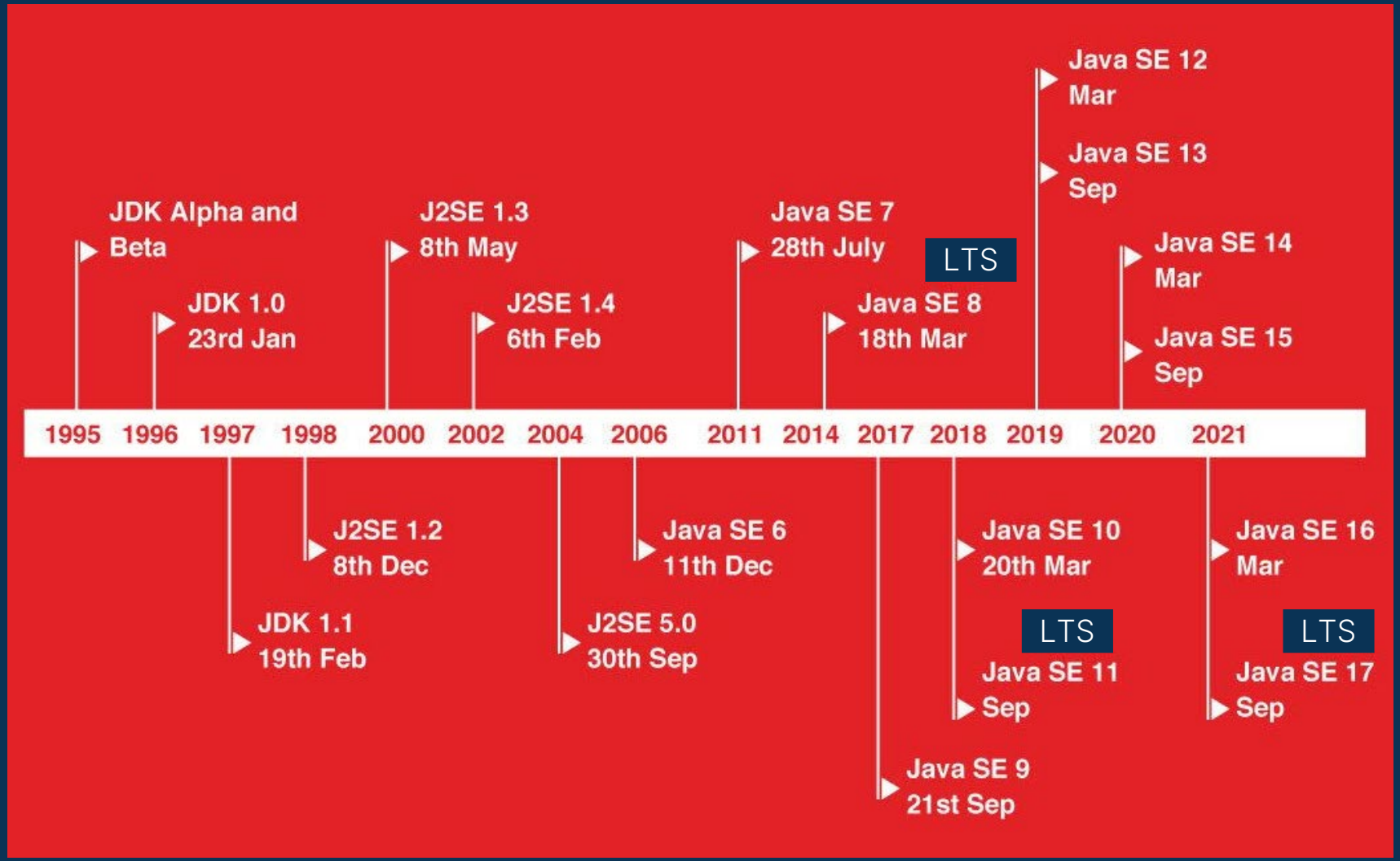
1995: COBOL — SBB CFF FFS

2007: Bern University of Applied Sciences

2011: JSR-352 Java Batch

2013: JSR-354 Money/Currency

Evolution of Java



https://en.wikipedia.org/wiki/Java_version_history

API Changes

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

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

```
// 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""";
```

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>
""";
```

Text Blocks (Methods)

```
// 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);

// The stripIndent method removes incidental white space from a multi-line string, using the same algorithm used by the
// Java compiler. This is useful if you have a program that reads text as input data and you want to strip indentation in
// the same manner as is done for text blocks.
String stripIndent()

// The translateEscapes method performs the translation of escape sequences (\b, \f, \n, \t, \r, \", \', \\ and octal
// escapes) and is used by the Java compiler to process text blocks and string literals.
// This is useful if you have a program that reads text as input data and you want to perform escape sequence processing.
// Note that Unicode escapes (\uNNNN) are not processed.
String translateEscapes()
```

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 Statement yield

```
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);
};
```

Switch Expression yield

```
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);
};
```

Records (Preview 14)

```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));
        }
    }
}
```

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 for switch (Preview 17)

```
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 Support

```
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

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

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

Local Records, Enums, Interfaces

```
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());
}
```

72 Services GmbH

Tulpenweg 1
2575 Täuffelen, Switzerland

+41 32 544 88 88
info@72.services
https://72.services