



Miscellaneous

Functional Programming

Pure Function

A pure function has the following attributes:

- The return values are identical for identical arguments
- The function has no side effects e.g no mutation of:
 - local static variables
 - non-local variables
 - mutable reference arguments
 - I/O streams

Referential Transparency


A referentially transparent function only depends on its input. No external state is required nor accessed or mutated.

Future of Java

Java's evolution into 2022: The state of the four big initiatives

Download a PDF of this article This will be a significant year for Java.

As is well known, Java 18 (as JDK 18) will be generally available on March 22, 2022, and Java 19 will be generally available in September.

 <https://blogs.oracle.com/javamagazine/post/java-jdk-18-evolution-valhalla-panama-loom-amber>



Project Valhalla

Project Valhalla is augmenting the Java object model with *value objects* and *user-defined primitives*, combining the abstractions of object-oriented programming with the performance characteristics of simple primitives. These features will be complemented with changes to Java's generics to preserve performance gains through generic APIs.

Main features:

- Value Objects: class instances that only feature `final`-instance-fields and lack object identity.
- User-defined Primitives: support for new, developer-declared primitive types in order to enhance performance and memory footprint
- Classes for basic Primitives: bring value-class features to existing primitives
- Enhanced Generics: provide optimal performance for value classes used with generic APIs

In a nut shell it will be possible to use primitive types as generic type arguments:

```
var numbers = new ArrayList<int>()
```

Current state: 2/3 JEPs are at draft state

Project Panama

Project Panama aims to connect the JVM with foreign (non-Java) libraries

Main features:

- Provide an alternative to the `ByteBuffer` API with similar performance and safety characteristics, but without some of the downsides
- Make native libraries more accessible by replacing the Java Native Interface (JNI) with a more Java-oriented workflow
- Provide basic building blocks that enable other frameworks to define higher-level native interoperability

Current state: the new apis are still incubating temporary preview modules. The vector-api move to a core module will be delayed until the completion of *Valhalla* hence its dependency on primitive- and value-types.

Project Loom

Project Loom takes a two-pronged approach toward improving Java's concurrency model. The goal is to improve the performance of heavily asynchronous code.

| codes like sync, works like async

Main features:

- Introduce threads that are lightweight (meaning you should be able to have millions of them) and user-mode (meaning managed by the JVM, not by the operating system).
- Introduce new concurrency programming models that make use of those lightweight threads.

Current state: Drafts for virtual threads and for a structured concurrency API are drafted and available in an early-access-build.

Project Amber

Project Amber explores smaller, productivity-oriented Java language features. Some of them are part of a larger story, but many are independent of one another.

Features delivered until today:

- Local-variable type inference, or `var` for short, in Java 10
- Text blocks in Java 15
- Records in Java 16
- Parts of a larger pattern matching story
 - `switch` improvements such as arrow syntax and use as expressions in Java 14
 - type patterns in Java 16
 - sealed classes in Java 17

The current focus lies on pattern matching, however Amber aims to improve Java as a whole language.

Current state: the basic pattern matching features should be finished until JDK-19, the following features could be expected:

- Refined patterns with additional conditions and null-checks
- Deconstruction patterns for records and arrays, perhaps even for regular classes
- Custom patterns

Project Valhalla Deutsch

Java Projekt Valhalla ist ein Funktionsbereich, der im Rahmen der Java-Programmiersprache entwickelt wird, um die Leistung und Ausdruckstärke der Sprache durch Hinzufügen neuer Wertetypen und generischer Spezialisierung zu verbessern. Wertetypen bieten eine kompakte und effiziente Darstellung für Objekte, die als Werte verwendet werden, während die generische Spezialisierung es der JVM ermöglicht, spezialisierte Code für generische Typen zu generieren, was die Leistung verbessert.

Generische Spezialisierungen sind ein Konzept in der objektorientierten Programmierung, bei dem die Compiler- und Laufzeitumgebung den generischen Code (Code, der für eine Vielzahl von Typen gültig ist) auf spezifische Typen anpassen kann. Dadurch wird die Leistung verbessert, weil der Code für die spezifischen Typen optimiert werden kann.

Zum Beispiel kann eine generische Klasse, die eine Liste von Elementen verwaltet, für verschiedene Typen spezialisiert werden, wie Integer, String oder Benutzerdefinierte Typen. Durch die Spezialisierung kann der Compiler und die Laufzeitumgebung eine spezialisierte Implementierung für jeden Typ bereitstellen, die die bestmögliche Leistung liefert.