

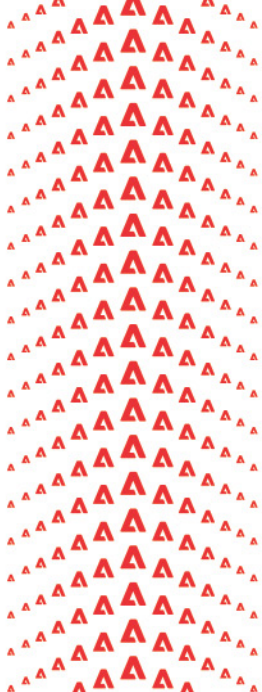
Java 9 to 21 essentials @ work

The new features with examples

Dikran Seropian | June 11, 2024

Agenda

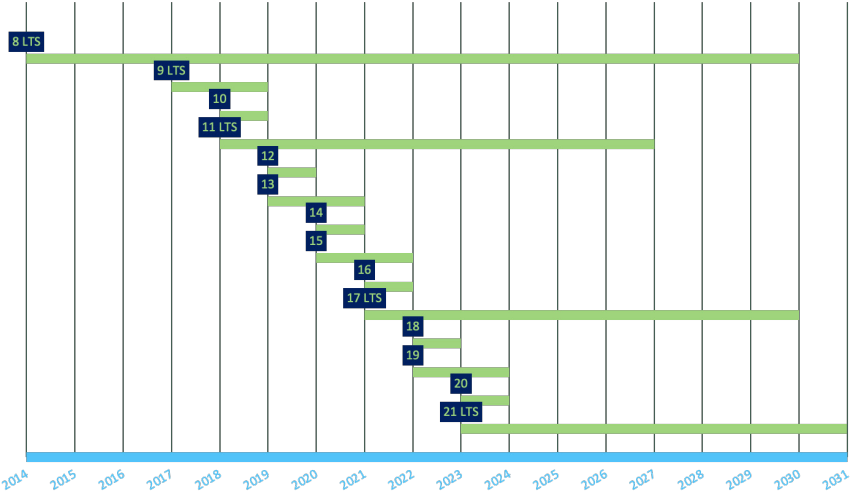
Section	Time
The journey from Java 9 to 21	10 min
Java 9 LTS	10 min
Java 11 LTS	10 min
Java 17 LTS	10 min
Java 21 LTS	10 min
Q & A	10 min



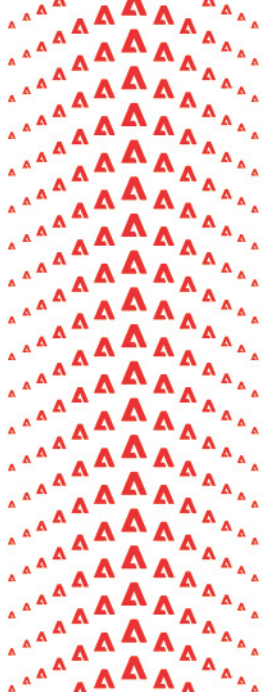
The Journey from Java 9 to 21



The Journey from Java 9 to 21 | Java Delivery Process



Java 9



Java 9 - enhancements & changes

JEP 102: Process API Updates
JEP 110: HTTP 2 Client
JEP 143: Improve Contended Locking
JEP 158: Unified JVM Logging
JEP 165: Compiler Control
JEP 193: Variable Handles
JEP 197: Segmented Code Cache
JEP 199: Smart Java Compilation, Phase Two
JEP 200: The Modular JDK
JEP 201: Modular Source Code
JEP 211: Elide Deprecation Warnings on Import Statements
JEP 212: Resolve Lint and Doclint Warnings
JEP 213: Milling Project Coin
JEP 214: Remove GC Combinations Deprecated in JDK 8
JEP 215: Tiered Attribution for javac
JEP 216: Process Import Statements Correctly
JEP 217: Annotations Pipeline 2.0
JEP 219: Datagram Transport Layer Security (DTLS)
JEP 220: Modular Run-Time Images
JEP 221: Simplified Doclet API
JEP 222: jshell: The Java Shell (Read-Eval-Print Loop)
JEP 223: New Version-String Scheme
JEP 224: HTML5 Javadoc
JEP 225: Javadoc Search
JEP 226: UTF-8 Property Files
JEP 227: Unicode 7.0
JEP 228: Add More Diagnostic Commands 229: Create PKCS12 Keystores by Default
JEP 231: Remove Launch-Time JRE Version Selection
JEP 232: Improve Secure Application Performance
JEP 233: Generate Run-Time Compiler Tests Automatically
JEP 235 Test Class-File Attributes Generated by javac
JEP 236: Parser API for Nashorn
JEP 237: Linux/AArch64 Port
JEP 238: Multi-Release JAR Files
JEP 240: Remove the JVM TI hprof Agent
JEP 241: Remove the jhat Tool
JEP 243: Java-Level JVM Compiler Interface
JEP 244: TLS Application-Layer Protocol Negotiation Extension
245: Validate JVM Command-Line Flag Arguments
JEP 246: Leverage CPU Instructions for GHASH and RSA
JEP 247: Compile for Older Platform Versions
JEP 248: Make G1 the Default Garbage Collector
JEP 249: OCSP Stapling for TLS
JEP 250: Store Interned Strings in CDS Archives
JEP 251: Multi-Resolution Images
JEP 252: Use CLDR Locale Data by Default
JEP 253: Prepare JavaFX UI Controls & CSS APIs for Modularization
JEP 254: Compact Strings
JEP 255: Merge Selected Xerces 2.11.0 Updates into JAXP
JEP 256: BeanInfo Annotations
JEP 257: Update JavaFX/Media to Newer Version of GStreamer
JEP 258: HarfBuzz Font-Layout Engine
JEP 259: Stack-Walking API
JEP 260: Encapsulate Most Internal APIs
JEP 261: Module System
JEP 262: TIFF Image I/O
JEP 263: HiDPI Graphics on Windows and Linux
JEP 264: Platform Logging API and Service
JEP 265: Marlin Graphics Renderer
JEP 266: More Concurrency Updates
JEP 267: Unicode 8.0
JEP 268: XML Catalogs
JEP 269: Convenience Factory Methods for Collections
JEP 270: Reserved Stack Areas for Critical Sections
JEP 271: Unified GC Logging
JEP 272: Platform-Specific Desktop Features
JEP 273: DRBG-Based SecureRandom Implementations
JEP 274: Enhanced Method Handles
JEP 275: Modular Java Application Packaging
JEP 276: Dynamic Linking of Language-Defined Object Models
JEP 277: Enhanced Deprecation
JEP 278: Additional Tests for Humongous Objects in G1
JEP 279: Improve Test-Failure Troubleshooting
JEP 280: Indify String Concatenation
JEP 281: HotSpot C++ Unit-Test Framework
JEP 282: jlink: The Java Linker
JEP 283: Enable GTK 3 on Linux
JEP 284: New HotSpot Build System
JEP 285: Spin-Wait Hints
JEP 287: SHA-3 Hash Algorithms
JEP 288: Disable SHA-1 Certificates
JEP 289: Deprecate the Applet API
JEP 290: Filter Incoming Serialization Data
JEP 291: Deprecate the Concurrent Mark Sweep (CMS) Garbage Collector
JEP 292: Implement Selected ECMAScript 6 Features in Nashorn 294: Linux/s390x Port
JEP 295: Ahead-of-Time Compilation
JEP 297: Unified arm32/arm64 Port
JEP 298: Remove Demos and Samples
JEP 299: Reorganize Documentation

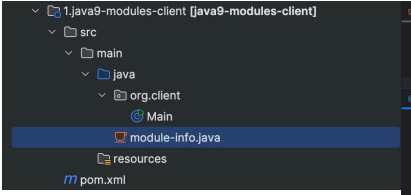


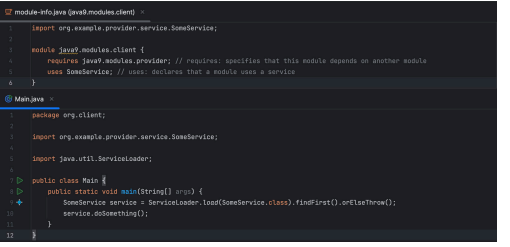
Java 9 - essentials@work

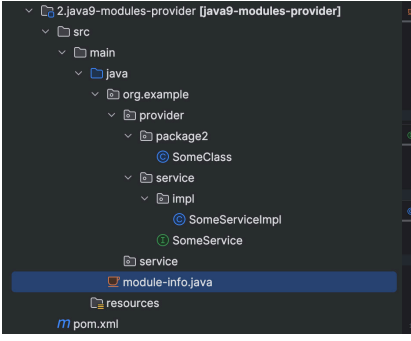
- The Java(9) Platform Module System (JEP 261)
- JShell - The Interactive Java 9 REPL (JEP 222)
- Convenience Factory Methods for Collections (JEP 269)
- Stream API Improvements (JEP 266)
- Private Interface Methods (JEP 213)
- Multi-Resolution Image API (JEP 251)
- The ProcessHandle API (JEP 102)
- CompletableFuture API Improvements(JEP 266)
- Reactive Streams (JEP 266)
- Try-With-Resources Improvement (JEP 213)
- Diamond Operator Extension for Anonymous Inner Class (JEP 213)
- The Optional API Enhancements (JEP 269)
- Enhanced @Deprecated annotation (JEP 277)

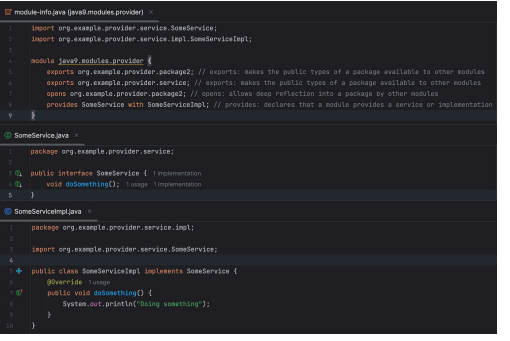
Java 9 - examples

The Java(9) Platform Module System (JEP 261)









Java 9 - examples

JShell - The Interactive Java 9 REPL (JEP 222)

```
sero@Dikrans-MBP-2 dikserop % jshell
|  Welcome to JShell -- Version 21.0.2
|  For an introduction type: /help intro

jshell> int i = 5
i ==> 5

jshell> int j = 12
j ==> 12

jshell> int sum = i + j
sum ==> 17

jshell> System.out.println("sum is:" + sum)
sum is:17
```

Java 9 - examples

Convenience Factory Methods for Collections (JEP 269)




```java
public class Main {
    public static void main(String[] args) {
        // Creating a List with factory method
        List<String> list = List.of("Apple", "Banana", "Cherry");
        System.out.println("List: " + list);

        // Creating a Set with factory method
        Set<String> set = Set.of("Dog", "Cat", "Bird");
        System.out.println("Set: " + set);

        // Creating a Map with factory method
        Map<String, Integer> map = Map.of( k1: "One",  v1: 1,  k2: "Two",  v2: 2,  k3: "Three",  v3: 3);
        System.out.println("Map: " + map);

        //Creating a Map of entries with factory method
        Map<String, Integer> mapOfEntries = Map.ofEntries(
            Map.entry( k: "One",  v: 1),
            Map.entry( k: "Two",  v: 2),
            Map.entry( k: "Three",  v: 3)
        );
        System.out.println("Map of entries: " + mapOfEntries);
    }
}
```

Java 9 - examples

Stream API Improvements (JEP 266)

```java
public class Main {
    public static void main(String[] args) {
        Stream<Integer> stream = Stream.of( ...values: 2, 4, 6, 8, 9, 10, 12);

        // Using takeWhile
        System.out.println("Using takeWhile:");
        stream.takeWhile(n -> n % 2 == 0).forEach(System.out::println);

        // Resetting the stream
        stream = Stream.of( ...values: 2, 4, 6, 8, 9, 10, 12);

        // Using dropWhile
        System.out.println("Using dropWhile:");
        stream.dropWhile(n -> n % 2 == 0).forEach(System.out::println);

        //using Stream.iterate
        System.out.println("Using Stream.iterate:");
        Stream.iterate( seed: 0, i -> i < 10, i -> i + 1).forEach(System.out::println);
    }
}
```

Java 9 - examples

Private Interface Methods (JEP 213)

```java
public interface MyInterface {  no usages  1 implementation
    default void method1() {  no usages  1 override
        commonMethod();
    }

    default void method2() {  no usages
        commonMethod();
    }

    private void commonMethod() {  2 usages
        System.out.println("Common method called");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        MyInterface myInterface = new MyInterface() {
        };
        myInterface.method1();
        myInterface.method2();
    }
}
```

Java 9 - examples

Multi-Resolution Image API (JEP 251)

```
public class Main {
    public static void main(String[] args) {
        // Create images of different resolutions
        BufferedImage img1 = new BufferedImage( width: 64,  height: 64, BufferedImage.TYPE_INT_RGB);
        BufferedImage img2 = new BufferedImage( width: 128,  height: 128, BufferedImage.TYPE_INT_RGB);
        BufferedImage img3 = new BufferedImage( width: 256,  height: 256, BufferedImage.TYPE_INT_RGB);

        // Create a list of images
        List<Image> images = new ArrayList<>();
        images.add(img1);
        images.add(img2);
        images.add(img3);

        // Create a MultiResolutionImage
        MultiResolutionImage multiResImage = new BaseMultiResolutionImage(images.toArray(new Image[0]));

        // Get image of a specific resolution
        Image variant = multiResImage.getResolutionVariant( destImageWidth: 128,  destImageHeight: 128);
        System.out.println("Variant width: " + variant.getWidth( observer: null));
        System.out.println("Variant height: " + variant.getHeight( observer: null));
    }
}
```

Java 9 - examples

The ProcessHandle API (JEP 102)

```java
public class Main {
    public static void main(String[] args) throws Exception {
        ProcessBuilder processBuilder = new ProcessBuilder( ...command: "/Applications/Cryptr.app/Contents/MacOS/Cryptr");
        Process process = processBuilder.start();

        System.out.println("-- process handle --");
        ProcessHandle processHandle = process.toHandle();
        System.out.printf("PID: %s%n", processHandle.pid());
        System.out.printf("isAlive: %s%n", processHandle.isAlive());

        System.out.println("-- process info --");
        ProcessHandle.Info info = processHandle.info();
        info.command().ifPresent(str -> System.out.printf("Command: %s%n", str));
        info.commandLine().ifPresent(str -> System.out.printf("CommandLine: %s%n", str));
        info.arguments().ifPresent(array -> System.out.printf("Arguments: %s%n", Arrays.toString(array)));
        info.startInstant().ifPresent(instant -> System.out.printf("StartInstant: %s%n", instant));
        info.totalCpuDuration().ifPresent(duration -> System.out.printf("CpuDuration: %s millis%n", duration.toMillis()));
        info.user().ifPresent(str -> System.out.printf("User: %s%n", str));

        Thread.sleep( millis: 5000);
        System.out.println("-- destroying --");
        processHandle.destroy();
        //better use onExit to know when process exits (destroy() may not kill the process immediately)
        CompletableFuture<ProcessHandle> future = processHandle.onExit();
        ProcessHandle ph = future.get(); //blocks
        System.out.printf("isAlive: %s%n", ph.isAlive());
    }
}
```

Java 9 - examples

The ProcessHandle API (JEP 102)

```
-- process handle --
PID: 48498
isAlive: true
-- process info --
Command: /Applications/Cryptr.app/Contents/MacOS/Cryptr
CommandLine: /Applications/Cryptr.app/Contents/MacOS/Cryptr
Arguments: [Ljava.lang.String;@b4c966a
StartInstant: 2024-06-11T21:56:29.304Z
User: sero
-- destroying --
isAlive: false

Process finished with exit code 0
```

Java 9 - examples

CompletableFuture API Improvements(JEP 266)

CompletableFuture.orTimeout()

```
CompletableFuture<String> future = CompletableFuture.supplyAsync(() -> {
    // Simulate a long-running task
    try {
        TimeUnit.SECONDS.sleep( timeout: 5);
    } catch (InterruptedException e) {
        throw new IllegalStateException(e);
    }
    return "Result";
});

// This will complete exceptionally with TimeoutException after 2 seconds
future.orTimeout( timeout: 2, TimeUnit.SECONDS) CompletableFuture<String>
        .thenAccept(System.out::println) CompletableFuture<Void>
        .exceptionally(ex -> {
            System.out.println("Exception: " + ex.getMessage());
            return null;
        });
```

Java 9 - examples

CompletableFuture API Improvements(JEP 266)

CompletableFuture.completeOnTimeout()

```
static void completeOnTimeout() {  1 usage
    CompletableFuture<String> future = CompletableFuture.supplyAsync(() -> {
        // Simulate long running task
        try {
            TimeUnit.SECONDS.sleep( timeout: 5);
        } catch (InterruptedException e) {
            throw new IllegalStateException(e);
        }
        return "Result";
    });

    // This will complete with the value "Timeout" after 1 second
    future.completeOnTimeout( value: "Timeout",  timeout: 1, TimeUnit.SECONDS).thenAccept(System.out::println);
```

Java 9 - examples

CompletableFuture API Improvements(JEP 266)

CompletableFuture.delayedExecutor

```
Executor delayed = CompletableFuture.delayedExecutor( delay: 1, TimeUnit.SECONDS);
CompletableFuture<String> future = CompletableFuture.supplyAsync(() -> "Hello World!", delayed);
future.thenAccept(System.out::println);
```

Java 9 - examples

Reactive Streams (JEP 266)

```
// Create a Publisher
SubmissionPublisher<String> publisher = new SubmissionPublisher<>();

// Create a Subscriber and register it with the Publisher
Flow.Subscriber<String> subscriber = new Flow.Subscriber<>() {
    private Flow.Subscription subscription;  2 usages

    @Override
    public void onSubscribe(Flow.Subscription subscription) {
        System.out.println("Subscribing... ");
        this.subscription = subscription;
        subscription.request( n: 1);
    }
    @Override
    public void onNext(String item) {
        System.out.println("Received item: " + item);
        subscription.request( n: 1);
    }
    @Override
    public void onError(Throwable throwable) {
        throwable.printStackTrace();
    }
    @Override
    public void onComplete() {
        System.out.println("Completed");
    }
};
```

```
publisher.subscribe(subscriber);

// Publish items
System.out.println("Publishing items...");
String[] items = {"item1", "item2", "item3"};
for (String item : items) {
    publisher.submit(item);
}

// while subscriber is still processing wait
while (publisher.hasSubscribers()) {
    Thread.sleep( millis: 1000);
}

// Close the Publisher
publisher.close();
```

Java 9 - examples

Try-With-Resources Improvement (JEP 213)

```
final Path path = Files.createTempFile( prefix: "somefile", suffix: "txt");
final BufferedReader reader = new BufferedReader(new FileReader(path.toFile()));
// JEP 213 allows this
try (reader) {
    System.out.println(reader.readLine());
} catch (IOException e) {
    e.printStackTrace();
}
```

Java 9 - examples

Diamond Operator Extension for Anonymous Inner Class (JEP 213)

```
Comparator<String> comparator = new Comparator<>() {
    @Override
    public int compare(String s1, String s2) {
        return s1.compareTo(s2);
    }
};

System.out.println("Comparator result: " + comparator.compare("Hello", "World"));
```

Java 9 - examples

The Optional API Enhancements (JEP 269)

```java
// Example of `or()` method
Optional<String> optional = Optional.of( value: "Hello");
Optional<String> result = optional.or(() -> Optional.of( value: "World"));
System.out.println("Result of or(): " + result.get()); // Outputs: Hello

// Example of `ifPresentOrElse()` method
Optional<String> optionalEmpty = Optional.empty();
optionalEmpty.ifPresentOrElse(
    value -> System.out.println("Value is present: " + value),
    () -> System.out.println("Value is not present") // Outputs: Value is not present
);

// Example of `stream()` method
Optional<String> optionalStream = Optional.of( value: "Hello Stream");
Stream<String> stream = optionalStream.stream();
stream.forEach(System.out::println); // Outputs: Hello Stream
```

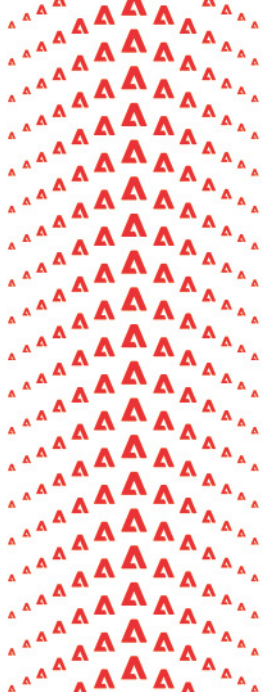
Java 9 - examples

Enhanced @Deprecated annotation (JEP 277)

```java
@Deprecated(since = "1.1", forRemoval = true)
class DeprecatedClass {
    @Deprecated(since = "1.2", forRemoval = true)
    public void deprecatedMethod() {
        System.out.println("This method is deprecated and it may be removed in future versions.");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        DeprecatedClass deprecatedClass = new DeprecatedClass();
        deprecatedClass.deprecatedMethod();
    }
}
```

Java 11



Java 10 + 11 - enhancements & changes

JEP 286: Local-Variable Type Inference
JEP 296: Consolidate the JDK Forest into a Single Repository
JEP 304: Garbage-Collector Interface
JEP 307: Parallel Full GC for G1
JEP 310: Application Class-Data Sharing
JEP 312: Thread-Local Handshakes
JEP 313: Remove the Native-Header Generation Tool (javah)
JEP 314: Additional Unicode Language-Tag Extensions
JEP 316: Heap Allocation on Alternative Memory Devices
JEP 317: Experimental Java-Based JIT Compiler
JEP 319: Root Certificates
JEP 322: Time-Based Release Versioning

JEP 181: Nest-Based Access Control
JEP 309: Dynamic Class-File Constants
JEP 315: Improve Aarch64 Intrinsics
JEP 318: Epsilon: A No-Op Garbage Collector
JEP 320: Remove the Java EE and CORBA Modules
JEP 321: HTTP Client (Standard)
JEP 323: Local-Variable Syntax for Lambda Parameters
JEP 324: Key Agreement with Curve25519 and Curve448
JEP 327: Unicode 10
JEP 328: Flight Recorder
JEP 329: ChaCha20 and Poly1305 Cryptographic Algorithms
JEP 330: Launch Single-File Source-Code Programs
JEP 331: Low-Overhead Heap Profiling
JEP 332: Transport Layer Security (TLS) 1.3
JEP 333: ZGC: A Scalable Low-Latency Garbage Collector (Experimental)
JEP 335: Deprecate the Nashorn JavaScript Engine
JEP 336: Deprecate the Pack200 Tools and API



Java 11 - essentials@work

- Local-Variable Type Inference (JEP 286)
- Unmodifiable Collections (JDK-8177290)
- Optional.orElseThrow() (JDK-8140281)
- HTTP/2 Client (JEP 321)
- Local-Variable Syntax for Lambda Parameters(JEP 323)
- Launch Single-File Source-Code Programs (JEP 330)

Java 11 - examples

Local-Variable Type Inference (JEP 286)

```
public class Main {
    public static void main(String[] args) {
        var message = "Hello world!";
        System.out.println(message);
    }
}
```

Java 11 - examples

Unmodifiable Collections (JDK-8177290)

```java
// Create modifiable collections
List<String> modifiableList = new ArrayList<>(Arrays.asList("A", "B", "C"));
Set<String> modifiableSet = new HashSet<>(Arrays.asList("D", "E", "F"));
Map<String, Integer> modifiableMap = new HashMap<>();
modifiableMap.put("G", 1);
modifiableMap.put("H", 2);
modifiableMap.put("I", 3);

// Create unmodifiable collections using copyOf()
List<String> unmodifiableList = List.copyOf(modifiableList);
Set<String> unmodifiableSet = Set.copyOf(modifiableSet);
Map<String, Integer> unmodifiableMap = Map.copyOf(modifiableMap);

System.out.println("Unmodifiable List: " + unmodifiableList);
System.out.println("Unmodifiable Set: " + unmodifiableSet);
System.out.println("Unmodifiable Map: " + unmodifiableMap);

// Create unmodifiable collections from a stream using Collectors
List<String> unmodifiableListFromStream = modifiableList.stream().collect(Collectors.toUnmodifiableList());
Set<String> unmodifiableSetFromStream = modifiableSet.stream().collect(Collectors.toUnmodifiableSet());
Map<String, Integer> unmodifiableMapFromStream = modifiableMap.entrySet().stream().collect(
        Collectors.toUnmodifiableMap(Map.Entry::getKey, Map.Entry::getValue));

System.out.println("Unmodifiable List from Stream: " + unmodifiableListFromStream);
System.out.println("Unmodifiable Set from Stream: " + unmodifiableSetFromStream);
System.out.println("Unmodifiable Map from Stream: " + unmodifiableMapFromStream);
```

Java 11 - examples

Optional.orElseThrow() (JDK-8140281)

```java
Optional<String> optional = Optional.of( value: "Hello world!");

// Get value from Optional or throw NoSuchElementException if it's empty
String value = optional.orElseThrow();

System.out.println(value);

// Create an empty Optional
Optional<String> emptyOptional = Optional.empty();

// Attempt to get value from empty Optional, this will throw NoSuchElementException
String emptyValue = emptyOptional.orElseThrow();
```

```
Hello world!
Exception in thread "main" java.util.NoSuchElementException Create breakpoint : No value present
    at java.base/java.util.Optional.orElseThrow(Optional.java:377)
    at org.example.Main.main(Main.java:18)
```

Java 11 - examples

HTTP/2 Client (JEP 321)

```
try (HttpClient client = HttpClient.newHttpClient()) {
    HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create("http://example.com"))
            .build();

    // the client will send the request asynchronously
    // when the response is received, it will print the response body
    // the join() method is used to block the current thread until the response is received
    client.sendAsync(request, HttpResponse.BodyHandlers.ofString()) CompletableFuture<HttpResponse<...>>
            .thenApply(HttpResponse::body) CompletableFuture<String>
            .thenAccept(System.out::println) CompletableFuture<Void>
            .join();
```

Java 11 - examples

Local-Variable Syntax for Lambda Parameters(JEP 323)

```java
public class Main {
    public static void main(String[] args) {
        List<String> items = Arrays.asList("Apple", "Banana", "Cherry");

        // Using 'var' in lambda expression
        items.forEach((var item) -> System.out.println(item));
    }
}
```

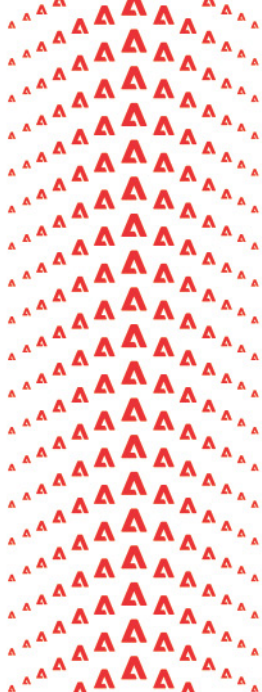
Java 11 - examples

Launch Single-File Source-Code Programs (JEP 330)

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

```
sero@Dikrans-MBP-2 example % java Main.java
Hello world!
```

Java 17



Java 12 -17 - enhancements & changes

JEP 230: Microbenchmark Suite
JEP 334: JVM Constants API
JEP 340: One AArch64 Port, Not Two
JEP 341: Default CDS Archives
JEP 344: Abortable Mixed Collections for G1
JEP 346: Promptly Return Unused Committed Memory from G1
JEP 350: Dynamic CDS Archives
JEP 351: ZGC: Uncommit Unused Memory
JEP 353: Reimplement the Legacy Socket API
JEP 345: NUMA-Aware Memory Allocation for G1
JEP 349: JFR Event Streaming
JEP 352: Non-Volatile Mapped Byte Buffers
JEP 358: Helpful NullPointerExceptions
JEP 361: Switch Expressions
JEP 362: Deprecate the Solaris and SPARC Ports
JEP 363: Remove the Concurrent Mark Sweep (CMS) Garbage Collector
JEP 364: ZGC on macOS
JEP 365: ZGC on Windows
JEP 366: Deprecate the ParallelScavenge + SerialOld GC Combination
JEP 367: Remove the Pack200 Tools and API

JEP 339: Edwards-Curve Digital Signature Algorithm (EdDSA)
JEP 371: Hidden Classes
JEP 372: Remove the Nashorn JavaScript Engine
JEP 373: Reimplement the Legacy DatagramSocket API
JEP 374: Disable and Deprecate Biased Locking
JEP 377: ZGC: A Scalable Low-Latency Garbage Collector
JEP 378: Text Blocks
JEP 379: Shenandoah: A Low-Pause-Time Garbage Collector
JEP 381: Remove the Solaris and SPARC Ports
JEP 385: Deprecate RMI Activation for Removal
JEP 347: Enable C++14 Language Features
JEP 357: Migrate from Mercurial to Git
JEP 369: Migrate to GitHub
JEP 376: ZGC: Concurrent Thread-Stack Processing
JEP 380: Unix-Domain Socket Channels
JEP 386: Alpine Linux Port
JEP 387: Elastic Metaspace
JEP 388: Windows/AArch64 Port
JEP 389: Foreign Linker API (Incubator)
JEP 390: Warnings for Value-Based Classes
JEP 392: Packaging Tool
JEP 393: Foreign-Memory Access API (Third Incubator)
JEP 394: Pattern Matching for instanceof
JEP 395: Records
JEP 396: Strongly Encapsulate JDK Internals by Default

JEP 306: Restore Always-Strict Floating-Point Semantics
JEP 356: Enhanced Pseudo-Random Number Generators
JEP 382: New macOS Rendering Pipeline
JEP 391: macOS/AArch64 Port
JEP 398: Deprecate the Applet API for Removal
JEP 403: Strongly Encapsulate JDK Internals
JEP 407: Remove RMI Activation
JEP 409: Sealed Classes
JEP 410: Remove the Experimental AOT and JIT Compiler
JEP 411: Deprecate the Security Manager for Removal
JEP 415: Context-Specific Deserialization Filters



Java 17 - essentials@work

- Switch Expressions (JEP 361)
- Text Blocks (JEP 378)
- Pattern Matching for instanceof (JEP 394)
- Records (JEP 395)
- Packaging Tool (JEP 392)
- Sealed Classes (JEP 409)
- Strongly Encapsulate JDK Internals (JEP 403)

Switch Expressions (JEP 361)

```
String day = "MONDAY";
//the switch expression returns a value
String typeOfDay = switch (day) {
    case "MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY", "FRIDAY" -> "Weekday";
    case "SATURDAY", "SUNDAY" -> "Weekend";
    default -> {
        //this is new feature
        //it returns a value from the switch expression instead of throwing an exception
        yield "Invalid day";
    }
};
System.out.println(typeOfDay); ;
```

Text Blocks (JEP 378)

```
public class Main {
    public static void main(String[] args) {
        String textBlock = """
                Hello,
                This is an example of a text block in Java 13 and later versions.
                You don't need to escape "double quotes" in this string.
                Text blocks make it easy to work with multiline string literals.
                """;
        System.out.println(textBlock);
    }
}
```

Pattern Matching for instanceof (JEP 394)

```java
Object obj = "Hello, world!";

//old way
if (obj instanceof String) {
    String s = (String) obj;
    System.out.println(s);
}

//new way
if (obj instanceof String str) {
    System.out.println("The length of the string is: " + str.length());
} else {
    System.out.println("The object is not a string.");
}
```

```java
//Point.java
public record Point(int x, int y) { }  4 usages
```

```java
public class Main {
    public static void main(String[] args) {
        Point point1 = new Point( x: 5,  y: 10);
        Point point2 = new Point( x: 5,  y: 10);

        // Using the x() and y() methods
        System.out.println("Point1 coordinates: " + point1.x() + ", " + point1.y());

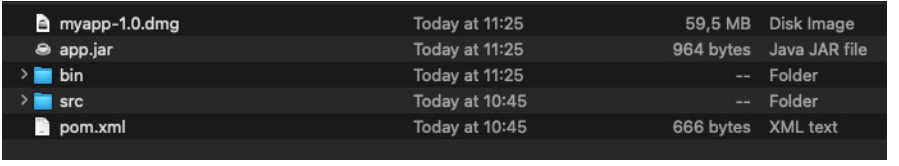
        // Using the equals() method
        System.out.println("Are point1 and point2 equal? " + point1.equals(point2));

        // Using the hashCode() method
        System.out.println("Hashcode of point1: " + point1.hashCode());

        // Using the toString() method
        System.out.println("String representation of point1: " + point1.toString());
    }
}
```

Packaging Tool (JEP 392)

```
sero@Dikrans-MBP-2 25.java17-packaging-tools % javac -d ./bin ./src/main/java/org/example/Main.java
jar cvfe app.jar org.example.Main -C ./bin .
added manifest
adding: org/(in = 0) (out= 0)(stored 0%)
adding: org/example/(in = 0) (out= 0)(stored 0%)
adding: org/example/Main.class(in = 426) (out= 298)(deflated 30%)
sero@Dikrans-MBP-2 25.java17-packaging-tools % jpackage --input . --name myapp --main-jar app.jar --main-class org.example.Main
```



Sealed Classes (JEP 409)

```java
// Sealed classes are classes that have restricted hierarchies.
// They define which other classes or interfaces may extend or implement them.
public sealed abstract class Shape  2 usages  2 inheritors
        permits Circle, Rectangle {
}
```

```java
//classes that extend a sealed class must be final, sealed, or non-sealed.
public final class Circle extends Shape {  1 usage
    private final double radius;  2 usages

    public Circle(double radius) { this.radius = radius; }

    public double getRadius() { return radius; }
}
```

```java
//non-sealed classes are classes that can be extended by any class.
public non-sealed class Rectangle extends Shape {  1 usage
    private final double width;  2 usages
    private final double height;  2 usages

    public Rectangle(double width, double height) {  no usages
        this.width = width;
        this.height = height;
    }

    public double getWidth() { return width; }

    public double getHeight() { return height; }
}
```

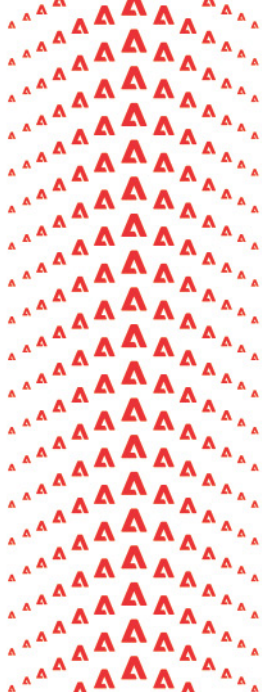
Strongly Encapsulate JDK Internals (JEP 403)

```java
public class Main {
    public static void main(String[] args) {
        Unsafe unsafe = Unsafe.getUnsafe();
    }
}
```

```
Main.java:1: error: package sun.misc is not visible
import sun.misc.Unsafe;
       ^
  (package sun.misc is declared in module java.base, which does not export it to the unnamed module)
1 error
```

```
Exception in thread "main" java.lang.SecurityException Create breakpoint : Unsafe
    at jdk.unsupported/sun.misc.Unsafe.getUnsafe(Unsafe.java:99)
    at org.example.Main.main(Main.java:7)
```

Java 21



Java 18 -21 - enhancements & changes

JEP 400: UTF-8 by Default
JEP 408: Simple Web Server
JEP 413: Code Snippets in Java API Documentation
JEP 416: Reimplement Core Reflection with Method Handles
JEP 418: Internet-Address Resolution SPI
JEP 421: Deprecate Finalization for Removal
JEP 405: Record Patterns (Preview)
JEP 422: Linux/RISC-V Port
JEP 434: Foreign Function & Memory API (Second Preview)
JEP 430: String Templates (Preview)
JEP 431: Sequenced Collections
JEP 439: Generational ZGC
JEP 440: Record Patterns
JEP 441: Pattern Matching for switch
JEP 442: Foreign Function & Memory API (Third Preview)
JEP 443: Unnamed Patterns and Variables (Preview)
JEP 444: Virtual Threads
JEP 445: Unnamed Classes and Instance Main Methods (Preview)
JEP 446: Scoped Values (Preview)
JEP 448: Vector API (Sixth Incubator)
JEP 449: Deprecate the Windows 32-bit x86 Port for Removal
JEP 451: Prepare to Disallow the Dynamic Loading of Agents
JEP 452: Key Encapsulation Mechanism API
JEP 453: Structured Concurrency (Preview)



Java 21 - essentials@work

- UTF-8 by Default (JEP 400)
- Simple Web Server (JEP 408)
- Code Snippets in Java API Documentation (JEP 413)
- Virtual Threads (JEP 444)
- Sequenced Collections (JEP 431)

Preview features to consider

- JEP 405: Record Patterns (Preview)
- JEP 453: Structured Concurrency (Preview)
- JEP 446: Scoped Values (Preview)
- JEP 430: String Templates (Preview)
- JEP 443: Unnamed Patterns and Variables (Preview)
- JEP 445: Unnamed Classes and Instance Main Methods (Preview)

UTF-8 by Default (JEP 400)

```java
// Write a file using the default charset (UTF-8)
Files.writeString(Paths.get( first: "test.txt"), csq: "Hello, world!");

// Read the file using the default charset (UTF-8)
String content = Files.readString(Paths.get( first: "test.txt"));

System.out.println(content);
```

Simple Web Server (JEP 408)

```java
public class Main {
    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress( port: 8000),  backlog: 0);
        server.createContext( path: "/", new MyHandler());
        server.setExecutor(null); // creates a default executor
        server.start();
    }

    static class MyHandler implements HttpHandler {  1 usage
        @Override
        public void handle(HttpExchange t) throws IOException {
            String response = "Hello, world!";
            t.sendResponseHeaders( rCode: 200, response.length());
            OutputStream os = t.getResponseBody();
            os.write(response.getBytes());
            os.close();
        }
    }
}
```

```
sero@Dikrans-MBP-2 29.java21-simple-web-server % curl localhost:8000
Hello, world!%
```

Code Snippets in Java API Documentation (JEP 413)

```java
/**
 * This class provides a method to say hello.
 */
public class Greeting {  2 usages

    /**
     * Says hello to the user.
     *
     * Here's an example of how to use this method:
     *
     * {@snippet :
     * Greeting greeting = new Greeting();
     * greeting.sayHello("John");
     * }
     *
     * @param name the name of the user
     */
    public void sayHello(String name) {  1 usage
        System.out.println("Hello, " + name + "!");
    }
}
```

```java
//create 100000 threads
for (int i = 0; i < 100000; i++) {
    Thread thread = new Thread(() -> {
        try {
            Thread.sleep( millis: 1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    });
    thread.start();
    thread.join();
}
System.out.println("threads started in: " + (System.nanoTime() - start) / 1_000_000 + "ms");
```

```java
start = System.nanoTime();
//create 100000 virtual threads
for (int i = 0; i < 100000; i++) {
    Thread thread = Thread.startVirtualThread(() -> {
        try {
            Thread.sleep( millis: 1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    });
    thread.join();
}
System.out.println("virtual threads started in: " + (System.nanoTime() - start) / 1_000_000 + "ms");
```

Sequenced Collections (JEP 431)

```java
private static void sequencedCollection() {  1 usage
    SequencedCollection<Integer> sequencedCollection = new ArrayList<>();

    sequencedCollection.add(1); // Collection contains: [1]
    sequencedCollection.addFirst( e: 0); // Collection contains: [0, 1]
    sequencedCollection.addLast( e: 2); // Collection contains: [0, 1, 2]

    Integer firstElement = sequencedCollection.getFirst(); // 0
    Integer lastElement = sequencedCollection.getLast(); // 2

    SequencedCollection<Integer> reversed = sequencedCollection.reversed();
    System.out.println(reversed); // Prints [2, 1, 0]

    // Note that any modification to the collection is visible in the methods, including the reversed view.
    sequencedCollection.add(3);
    System.out.println(sequencedCollection); // Prints [0, 1, 2, 3]
    System.out.println(sequencedCollection.reversed()); // Prints [3, 2, 1, 0]
}
```

Sequenced Collections (JEP 431)

```java
private static void sequencedSet() {  1 usage
    SequencedSet<Integer> sequencedSet = new LinkedHashSet<>();

    sequencedSet.add(1); // Collection contains: [1]
    sequencedSet.add(1); // Collection contains: [1]
    sequencedSet.add(2); // Collection contains: [1, 2]

    Integer firstElement = sequencedSet.getFirst(); // 1
    Integer lastElement = sequencedSet.getLast(); // 2

    SequencedSet<Integer> reversed = sequencedSet.reversed();
    System.out.println(reversed); // Prints [2, 1]

    // Note that any modification to the collection is visible in the methods, including the reversed view.
    sequencedSet.add(3);
    System.out.println(sequencedSet); // Prints [1, 2, 3]
    System.out.println(sequencedSet.reversed()); // Prints [3, 2, 1]
}
```

Sequenced Collections (JEP 431)

```
private static void sequencedMap() {  1 usage
    SequencedMap<String, Integer> sequencedMap = new LinkedHashMap<>();

    sequencedMap.put("One", 1); // Map contains: {"One"=1}
    sequencedMap.putFirst( k: "Zero",  v: 0); // Map contains: {"Zero"=0, "One"=1}
    sequencedMap.putLast( k: "Two",  v: 2); // Map contains: {"Zero"=0, "One"=1, "Two"=2}

    String firstKey = sequencedMap.firstEntry().getKey(); // "Zero"
    String lastKey = sequencedMap.lastEntry().getKey(); // "Two"

    SequencedMap<String, Integer> reversed = sequencedMap.reversed();
    System.out.println(reversed); // Prints {"Two"=2, "One"=1, "Zero"=0}

    // Note that any modification to the map is visible in the methods, including the reversed view.
    sequencedMap.put("Three", 3);
    System.out.println(sequencedMap); // Prints {"Zero"=0, "One"=1, "Two"=2, "Three"=3}
    System.out.println(sequencedMap.reversed()); // Prints {"Three"=3, "Two"=2, "One"=1, "Zero"=0}
}
```

Guidelines & resources



- Java 9 to 21: How to Code Like a Pro - O'Reily
- https://github.com/seropian/java9to21-essentials-at-work
- JDK 9 Release Notes
- JDK 10 Release notes
- JDK 11 Release notes
- JDK 12 Release notes
- JDK 13 Release notes
- JDK 14 Release notes
- JDK 15 Release notes
- JDK 16 Release notes
- JDK 17 Release notes
- JDK 18 Release notes
- JDK 19 Release notes
- JDK 20 Release notes
- JDK 21 Release notes



Thank you




Adobe