

Java versions

A migration strategy
by Albert Attard



Agenda

1. What's so unique about Java version migration?
2. What can go wrong?
3. A migration strategy

Code examples

All code examples can
be downloaded from:

<https://bit.ly/3bBcw4g>



What's so unique about Java version migration?

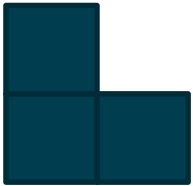


1 feature

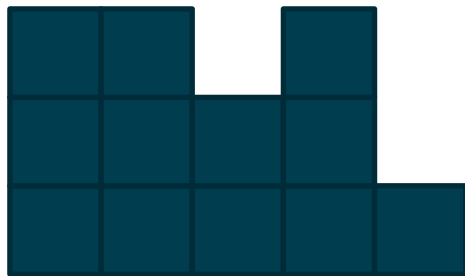


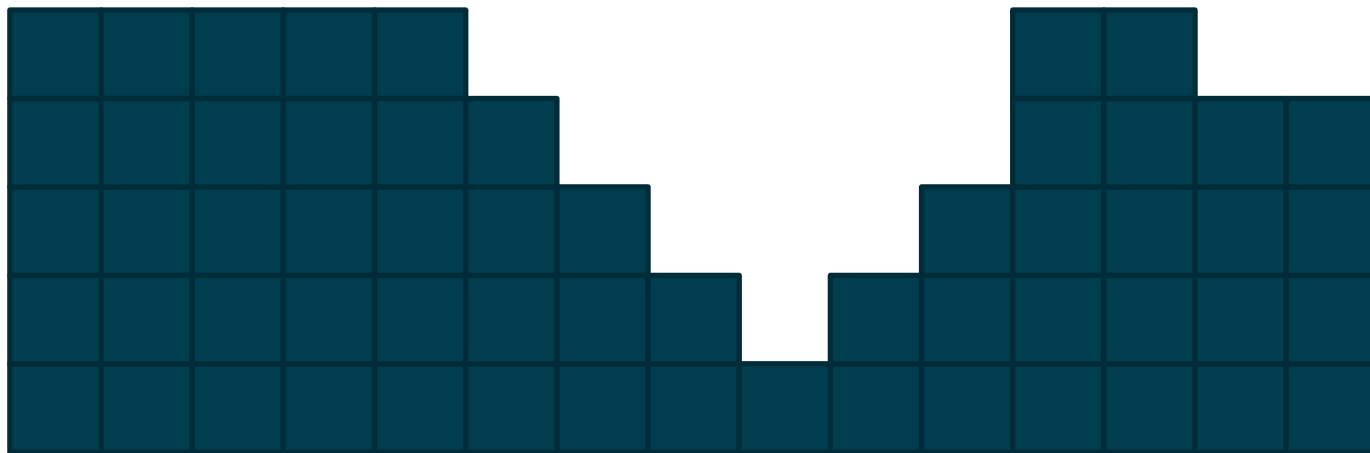
2 features

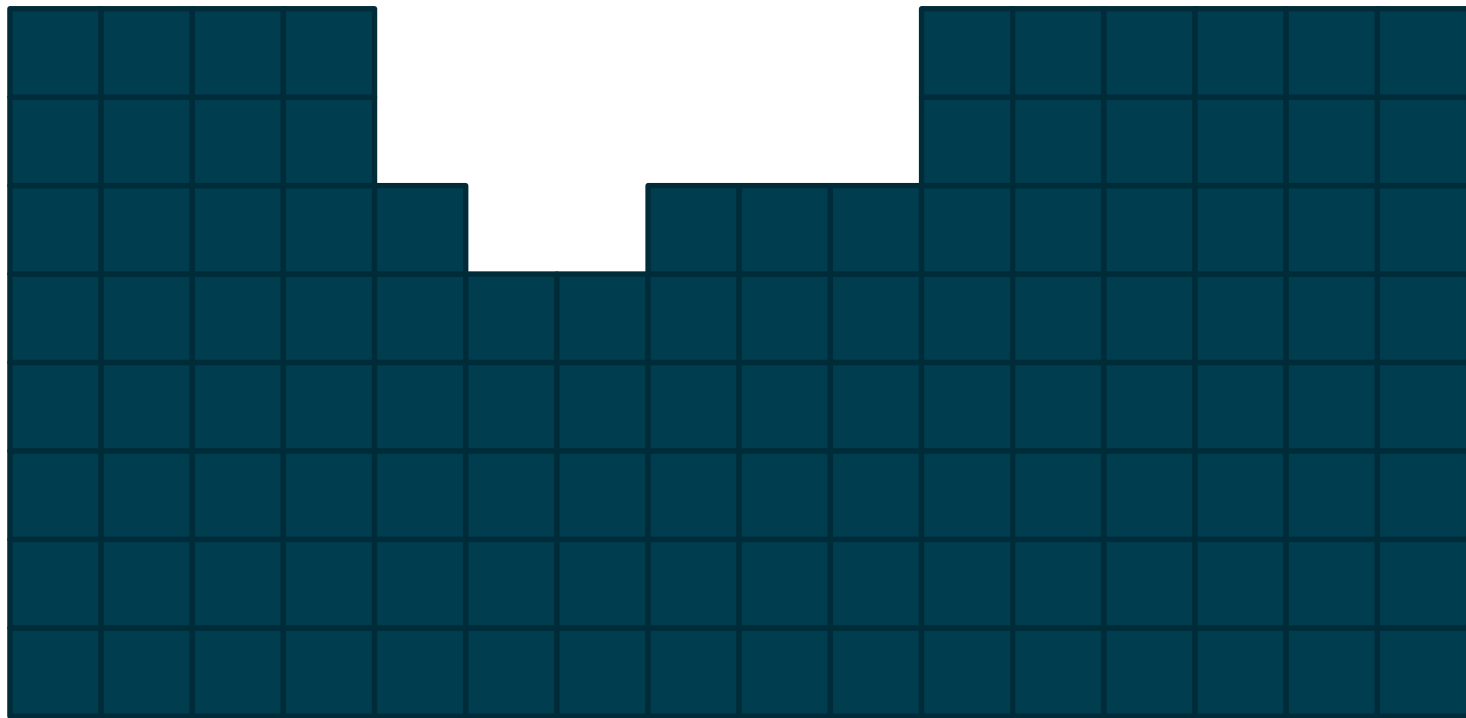


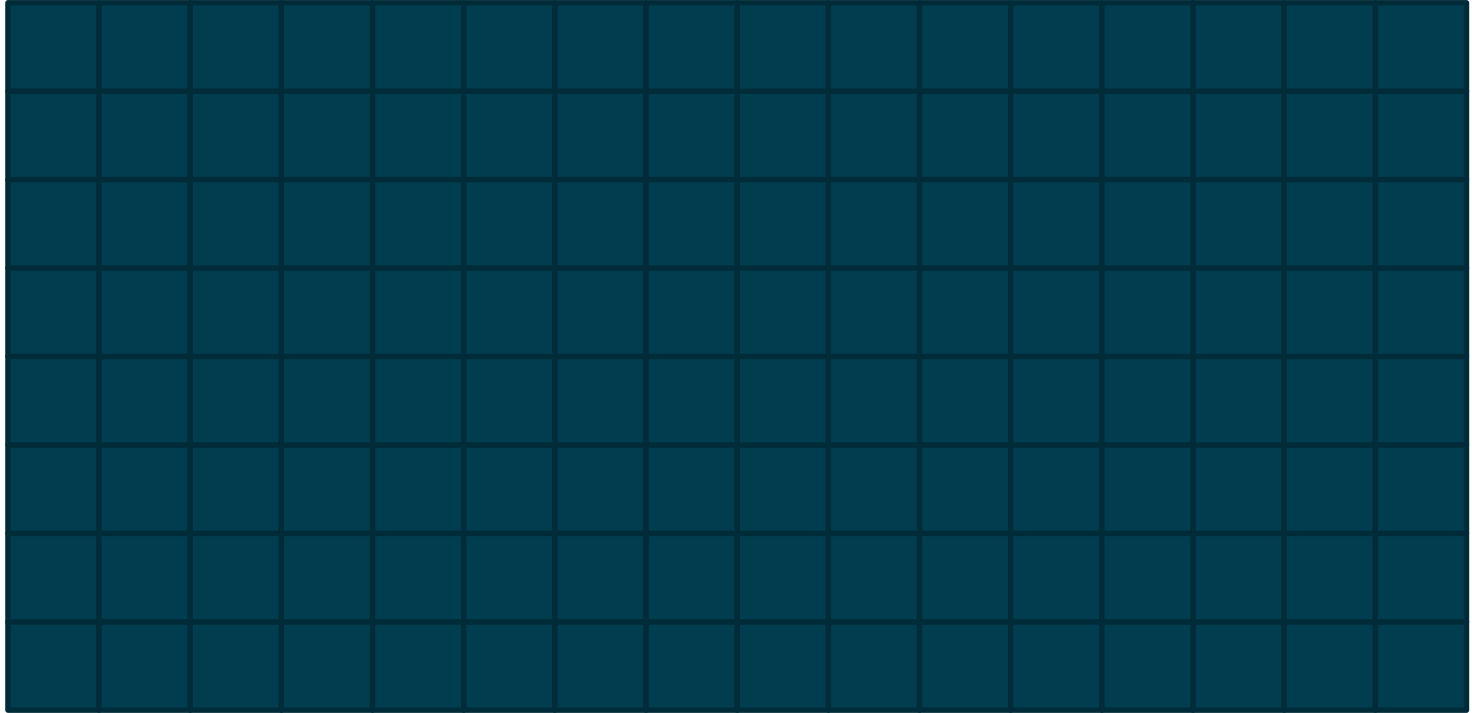


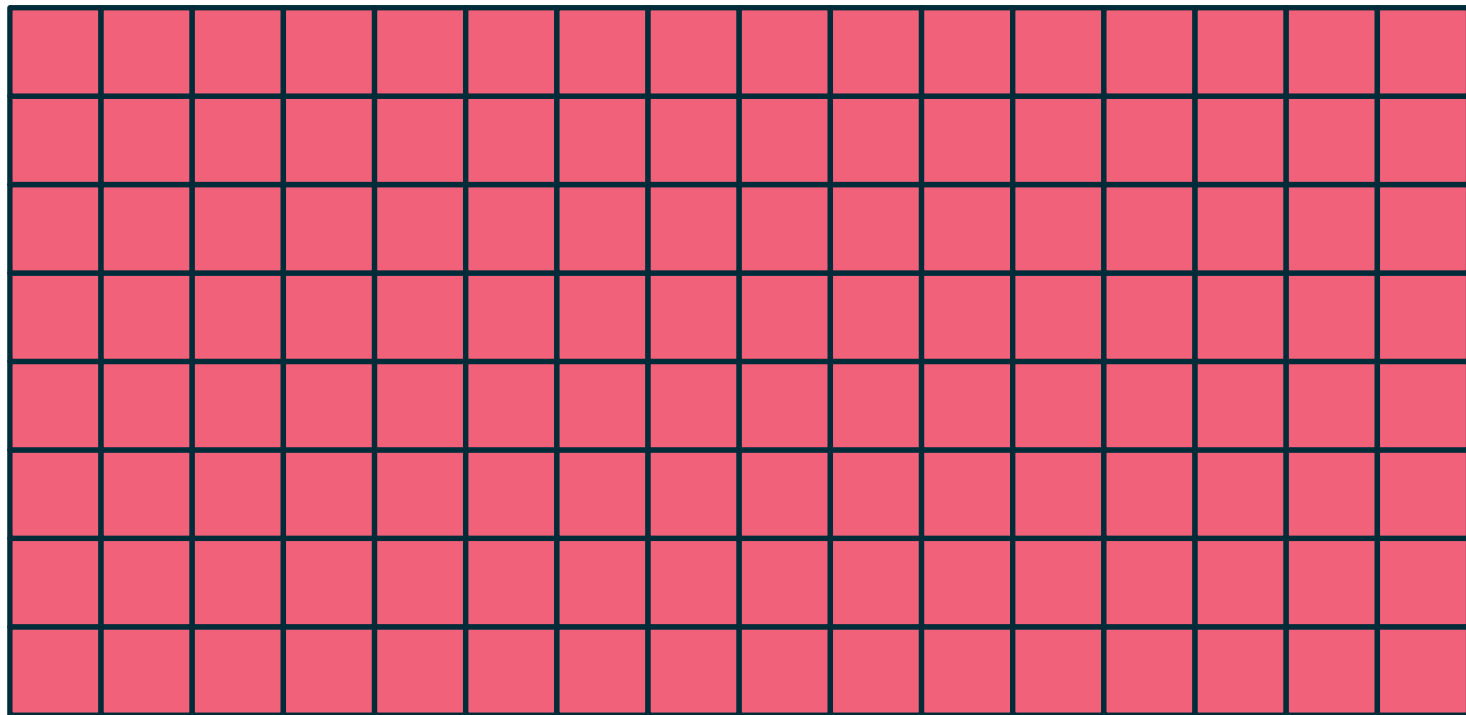












What can go wrong?



Isn't Java backwards-compatible?

```
package demo;

import java.rmi.MarshalledObject;
import java.rmi.RemoteException;
import java.rmi.activation.Activatable;
import java.rmi.activation.ActivationException;
import java.rmi.activation.ActivationID;
import java.rmi.server.RMIClientSocketFactory;
import java.rmi.server.RMIServerSocketFactory;
import java.util.Locale;

public class DefaultStringRemoteFunction extends Activatable implements StringRemoteFunction {
```

```
plugins {
    id "java"
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(11)
    }
}
```

```
Terminal: Local x + v
→ is-java-backwards-compatible-api git:(main) x ./gradlew build
Starting a Gradle Daemon, 2 busy Daemons could not be reused, use --status for details

BUILD SUCCESSFUL in 11s
2 actionable tasks: 2 executed
→ is-java-backwards-compatible-api git:(main) x █
```

Git TODO Problems Terminal Dependencies


```
package demo;
```

```
import java.rmi.MarshalledObject;
```

```
import java.rmi.RemoteException;
```

```
import java.rmi.activation.Activatable;
```

```
import java.rmi.activation.ActivationException;
```

```
import java.rmi.activation.ActivationID;
```

```
import java.rmi.server.RMIClientSocketFactory;
```

```
import java.rmi.server.RMIServerSocketFactory;
```

```
import java.util.Locale;
```

```
public class DefaultStringRemoteFunction extends Activatable implements StringRemoteFunction {
```

The `java.rmi.activation` package was removed in Java 15 as part of JEP 385

```
plugins {  
    id "java"  
}  
  
java {  
    toolchain {  
        languageVersion = JavaLanguageVersion.of(17)  
    }  
}
```

Terminal: Local x + v

> Compilation failed; see the compiler error output for details.

* Try:

Run with `--stacktrace` option to get the stack trace. Run with `--info` or `--debug` option to get more details.

* Get more help at <https://help.gradle.org>

BUILD FAILED in 1s

1 actionable task: 1 executed

→ `is-java-backwards-compatible-api` git:(main)

```
package demo;

import lombok.Builder;
import lombok.Data;

import java.time.OffsetDateTime;




@Data
@Builder(toBuilder = true)
public class Appointment {

    private OffsetDateTime date;
}
```

```
plugins {
    id "java"
    id "io.freefair.lombok" version "6.2.0"
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(11)
    }
}

lombok {
    getVersion().set("1.18.12")
}
```

Terminal: Local   

```
➔ is-java-backwards-compatible-api git:(main) ✕ ./gradlew build
Starting a Gradle Daemon, 3 busy Daemons could not be reused, use --status for details

BUILD SUCCESSFUL in 11s
2 actionable tasks: 1 executed, 1 up-to-date
➔ is-java-backwards-compatible-api git:(main) ✕
```

```

package demo;

import lombok.Builder;
import lombok.Data;

import java.time.OffsetDateTime;

@Data
@Builder(toBuilder = true)
public class Appointment {

    private OffsetDateTime date;
}

```

```

plugins {
    id "java"
    id "io.freefair.lombok" version "6.2.0"
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

lombok {
    getVersion().set("1.18.12")
}

```

Lombok 1.18.22, and
above, support Java 17

Terminal: Local x + v

> Compilation failed; see the compiler error output for details.

* Try:

Run with `--stacktrace` option to get the stack trace. Run with `--info` or `--debug` option to get more log output. Run with `--scan` to get full insights.

* Get more help at <https://help.gradle.org>

BUILD FAILED in 1s

1 actionable task: 1 executed

→ is-java-backwards-compatible-api git:(main) ✕

**A newer version of
a dependency
may not play well with
other dependencies**

```
import java.time.OffsetDateTime;
```

Jackson + Lombok

```
@Data
@Builder(toBuilder = true)
@JsonDeserialize(builder = Appointment.AppointmentBuilder.class)
public class Appointment {

    @JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ss'Z'")
    private OffsetDateTime date;

    @JsonPOJBuilder(withPrefix = "")
    public static class AppointmentBuilder { }

}
```

```
plugins {
    id "java"
    id "io.freefair.lombok" version "6.2.0"
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(11)
    }
}

lombok {
    getVersion().set("1.18.12")
}
```

@Test

```
void parseAppointmentDate() throws IOException {
    final String json = "{\"date\": \"2077-04-27T12:34:56Z\"}";

    final ObjectMapper mapper = new ObjectMapper()
        .registerModule(new JavaTimeModule());
    final Appointment appointment = mapper.readValue(json, Appointment.class);

    final LocalDateTime date = LocalDateTime.of(year: 2077, month: 4,
    final Appointment expected = Appointment.builder()
        .date(date.atOffset(ZoneOffset.UTC)).build();
    assertEquals(expected, appointment);
}
```

Run: AppointmentTest

Tests passed: 1 of 1 test – 255 ms

Test Results	255 ms
AppointmentTest	255 ms
parseAppointmentDate()	255 ms

Tests passed: 1

BUILD SUCCESSFUL in 6s

Tasks:

- > Task :generateMainEffectiveLombokConfig1
- > Task :compileJava
- > Task :processResources NO-SOURCE
- > Task :classes
- > Task :generateTestEffectiveLombokConfig1
- > Task :compileTestJava
- > Task :processTestResources NO-SOURCE
- > Task :testClasses
- > Task :test

Git Run TODO Problems Terminal Build Dependencies

```
import java.time.OffsetDateTime;

@Data
@Builder(toBuilder = true)
@JsonDeserialize(builder = Appointment.AppointmentBuilder.class)
public class Appointment {

    @JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ss'Z'")
    private OffsetDateTime date;

    @JsonPOJBuilder(withPrefix = "")
    public static class AppointmentBuilder { }
}
```

```
@Test
void parseAppointmentDate() throws IOException {

    final String json = "{\"date\": \"2077-04-27T12:34:56Z\"}";

    final ObjectMapper mapper = new ObjectMapper()
        .registerModule(new JavaTimeModule());
    final Appointment appointment = mapper.readValue(json, Appointment.class);

    final LocalDateTime date = LocalDateTime.of(year: 2077, month: 4,
    final Appointment expected = Appointment.builder()
        .date(date.atOffset(ZoneOffset.UTC)).build();
    assertEquals(expected, appointment);
}
```

```
plugins {
    id "java"
    id "io.freefair.lombok" version "6.2.0"
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(11)
    }
}

lombok {
    getVersion().set("1.18.16")
}
```

Run: AppointmentTest

Tests failed: 1 of 1 test – 201 ms

Test Results	201 ms
AppointmentTest	201 ms
parseAppointmentDate()	201 ms

Tests failed: 1, passed: 0

Cannot deserialize value of type `java.time.OffsetDateTime`
 at [Source: (String) "{"date": "2077-04-27T12:34:56Z"}"; li
 com.fasterxml.jackson.databind.exc.InvalidFormatException:
 at [Source: (String) "{"date": "2077-04-27T12:34:56Z"}"; li
 at app//com.fasterxml.jackson.databind.exc.InvalidForma
 at app//com.fasterxml.jackson.databind.DeserializationC
 at app//com.fasterxml.jackson.databind.DeserializationC
 at app//com.fasterxml.jackson.datatype.jsr310.deser.JSR
 at app//com.fasterxml.jackson.datatype.jsr310.deser.Ins

Git Run TODO Problems Terminal Build Dependencies

**Are the tests really
running?**

```
package demo;
```

```
public class Algorithm {  
    public int add(final int a, final int b) {  
        return a - b;  
    }  
}
```

This is an
intentional bug

```
@Test
```

```
void addsTwoNumbers() {  
    final Algorithm algorithm = new Algorithm();  
    final int result = algorithm.add(a: 2, b: 4);  
  
    assertEquals(expected: 6, result);  
}
```

Run: AlgorithmTest.addsTwoNumbers

Tests failed: 1 of 1 test – 20 ms

Test Results 20 ms
AlgorithmTest 20 ms
addsTwoNumbers() 20 ms

org.opentest4j.AssertionFailedError:
Expected :6
Actual :-2
[<Click to see difference>](#)

<5 internal lines>

at demo.AlgorithmTest.addsTwoNumbers(AlgorithmTest.java:14) <31 internal lines>

at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>

at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <25 internal lines>

Test run when
triggered from
the IDE


```
→ are-the-tests-running git:(main) ✕ ./mvnw test
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.example:are-the-tests-running >-----
[INFO] Building are-the-tests-running 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ are-the-tests-running
[WARNING] Using platform encoding (US-ASCII actually) to copy filtered resources, i.e. build
[INFO] Copying 0 resources
```

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

Actually, the test did not run!

```
[INFO] -----
[INFO] BUILD SUCCESS The test did not fail!
[INFO] -----
[INFO] Total time: 1.013 s
[INFO] Finished at: 2021-11-02T08:53:51+01:00
[INFO] -----
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.19.1</version>
    </plugin>
  </plugins>
</build>
```

JUnit 4 tests are executed
JUnit 5 tests are **NOT** executed

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.0</version>
    </plugin>
  </plugins>
</build>
```

JUnit 4 tests are executed
JUnit 5 tests are executed

**Code coverage
should not change**

```

public class Algorithm {

    public void run(final double from, final double until, final DoubleConsumer consumer) {
        for (double i = from; i < until; i++) {
            consumer.accept(i);
        }
    }
}

```

@Test

```

void run4Times() {
    final List<Double> values = new ArrayList<>();
    final DoubleConsumer consumer = values::add;

    final Algorithm algorithm = new Algorithm();
    algorithm.run(from: 1D, until: 5D, consumer);

    final List<Double> expected = List.of(1D, 2D, 3D, 4D);
    assertEquals(expected, values);
}

```

code-coverage > demo > Algorithm

Algorithm

Element	Missed Instructions	Cov.	Missed Branches	Cov.
run(double, double, int, DoubleConsumer)	<div><div></div></div>	100 %	<div><div></div></div>	100 %
Algorithm()	<div><div></div></div>	100 %		n/a
Total	0 of 19	100 %	0 of 2	100 %

100% code coverage

```

@Test
void runsForever() {
    final List<Double> values = new ArrayList<>();
    final DoubleConsumer consumer = values::add;

    final Algorithm algorithm = new Algorithm();
    algorithm.run( from: 1_000_000_000_000_000_1D, until: 1_000_000_000_000_000_5D, consumer);

    final List<Double> expected = List.of(1_000_000_000_000_000_1D,
        1_000_000_000_000_000_2D,
        1_000_000_000_000_000_3D,
        1_000_000_000_000_000_4D);
    assertEquals(expected, values);
}

```

Runs forever, or until it runs out of memory

@Test

```
void varPlus1ShouldBeGreaterThanVar() {  
    final double a = 1_000_000_000_000_000_1D;  
    final double b = a + 1;  
  
    assertTrue(condition: a < b);  
}
```

Run: AlgorithmTest.findValues x

Tests failed: 1 of 1 test - 22 ms

Test Results 22 ms

- AlgorithmTest 22 ms
 - findValues() 22 ms

expected: <true> but was: <false>
Expected :true
Actual :false
[<Click to see difference>](#)

org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
at app//org.junit.jupiter.api.AssertionUtils.fail(AssertionUtils.java:55)
at app//org.junit.jupiter.api.AssertTrue.assertTrue(AssertTrue.java:40)
at app//org.junit.jupiter.api.AssertTrue.assertTrue(AssertTrue.java:35)

Extra, Extra - Read All About It: Nearly All Binary Searches and Mergesorts are Broken

Friday, June 2, 2006

Posted by Joshua Bloch, Software Engineer

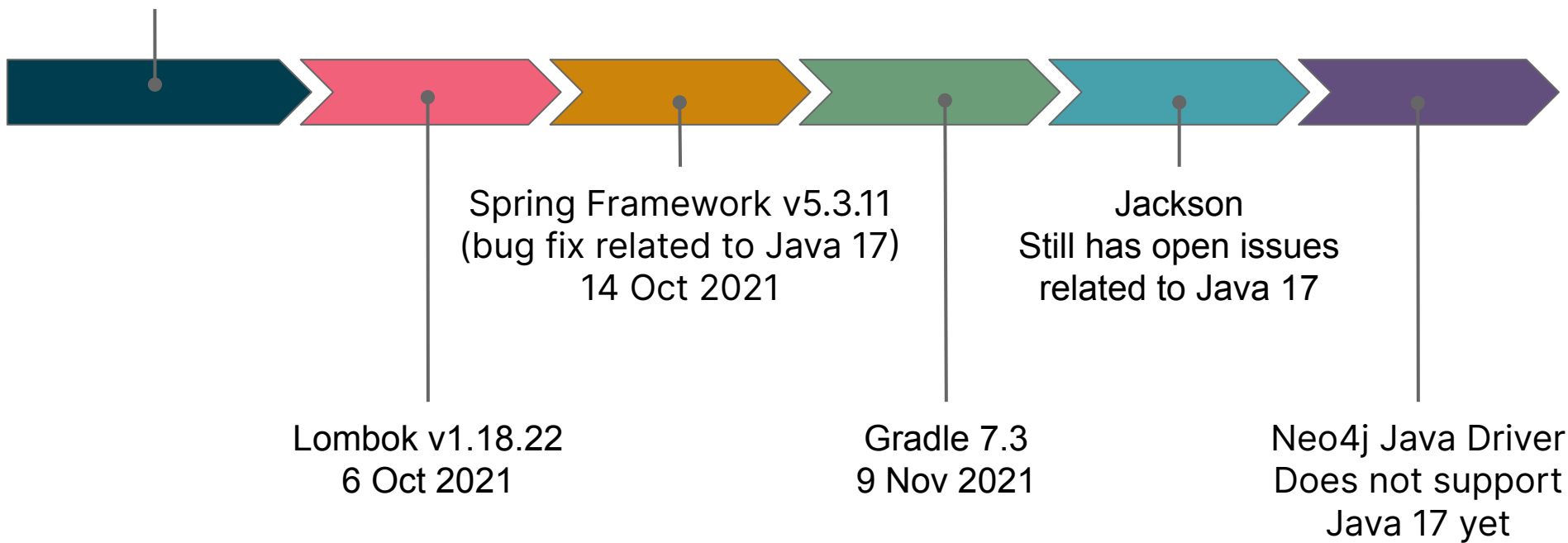
I remember vividly Jon Bentley's first Algorithms lecture at CMU, where he asked all of us incoming Ph.D. students to write a binary search, and then dissected one of our implementations in front of the class. Of course it was broken, as were most of our implementations. This made a real impression on me, as did the treatment of this material in his wonderful *Programming Pearls* (Addison-Wesley, 1986; Second Edition, 2000). The key lesson was to carefully consider the invariants in your programs.

The bug is in this line:

```
6:         int mid =(low + high) / 2;
```

**When updating,
you can only go as
fast as your slowest
dependency**

Java 17
14 Sep 2021



A migration strategy



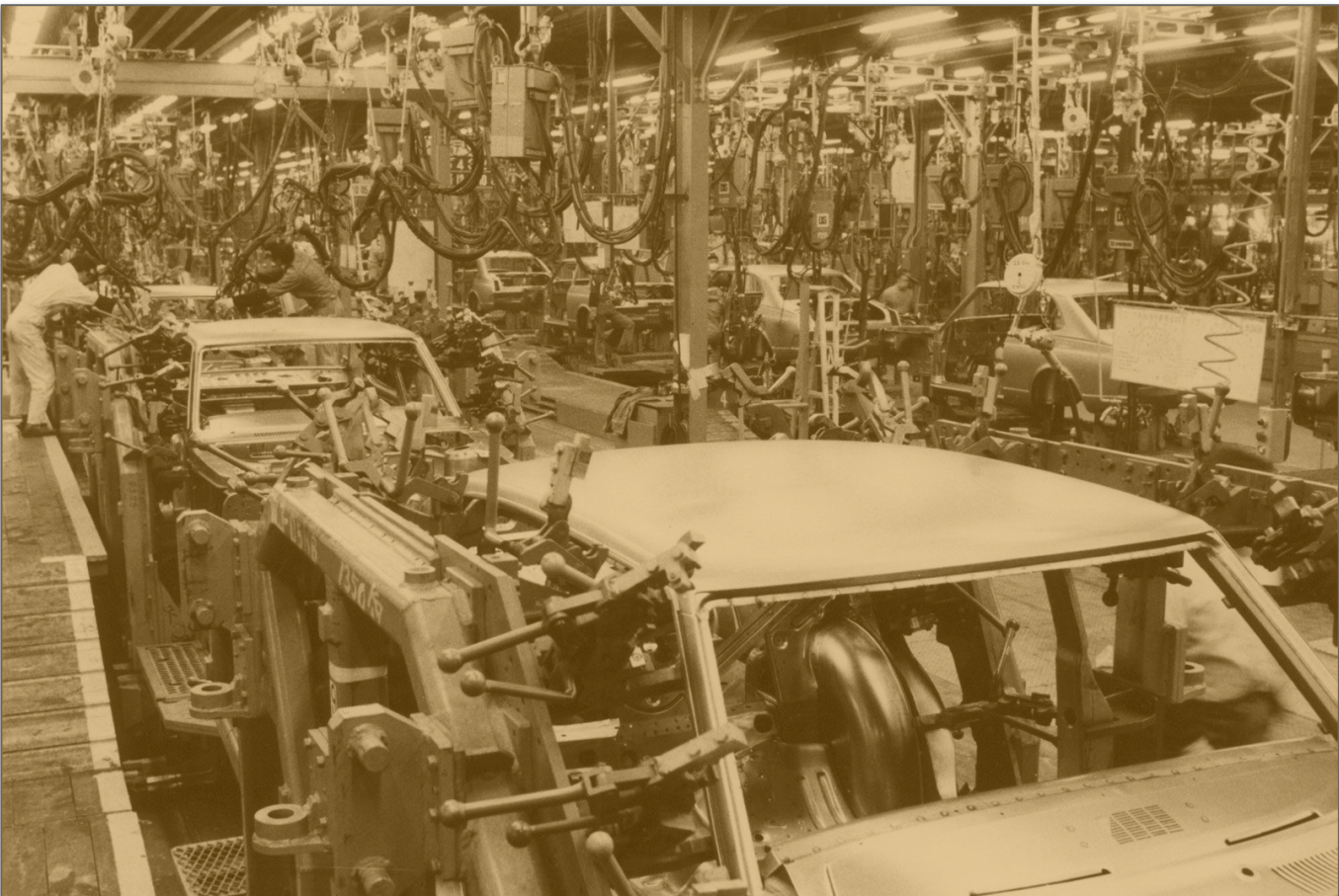


Image from: <https://www.britannica.com/technology/assembly-line>



Jenkins



Travis CI

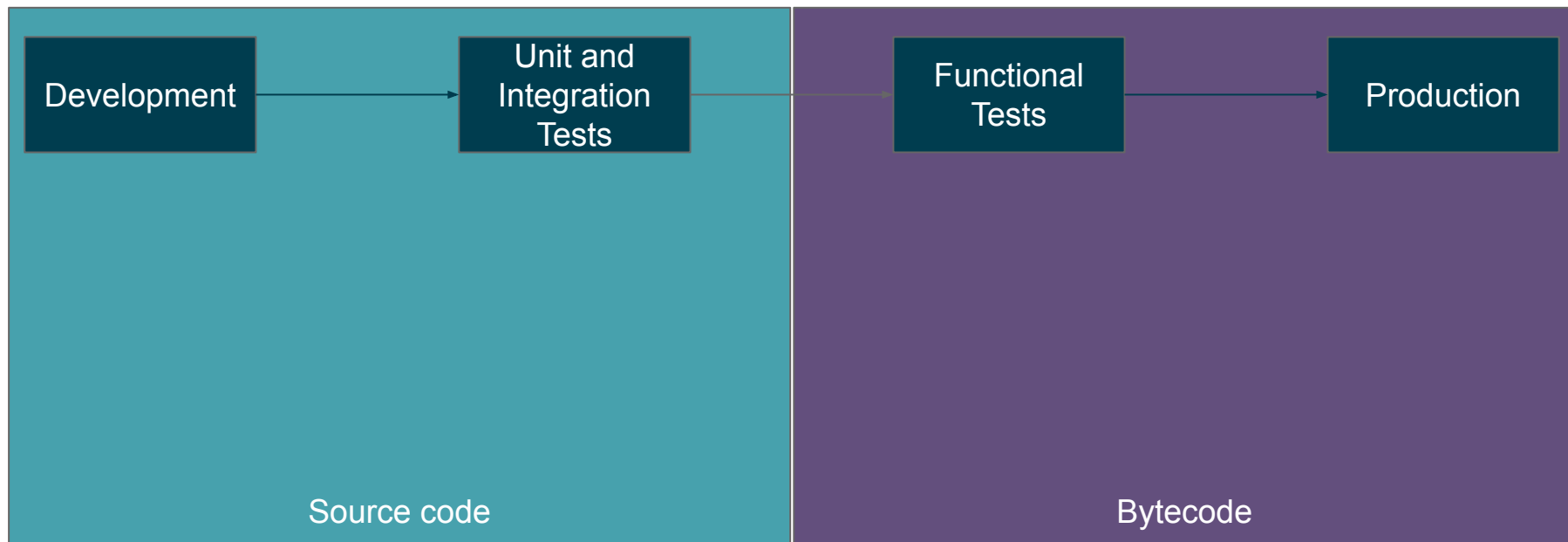


TeamCity

Java 8



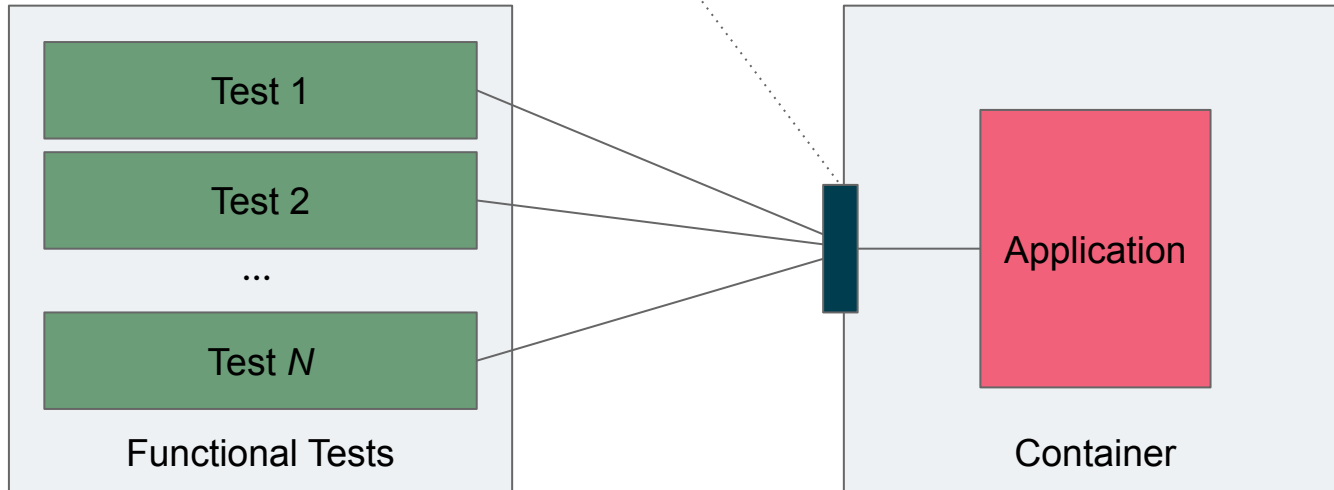
Java 8



```
FROM adoptopenjdk:8u292-b10-jre-hotspot  
WORKDIR /opt/app  
COPY build/libs/app.jar app.jar  
ENTRYPOINT ["java", "-jar", "app.jar"]
```

The functional tests run against the container

Ports exposed by
the container



REST/HTTP

@Container

```
private final GenericContainer<?> app = createAppContainer();
```

@Test

```
void returnStatusUp() {
```

```
    /* Given */
```

```
    assertTrue(app.isRunning());
```

```
    /* When */
```

```
    final ResponseEntity<HealthCheckResponse> response = actuatorHealth();
```

```
    /* Then */
```

```
    assertEquals(response.getStatusCode(), HttpStatus.OK);
```

```
    assertEquals(response.getBody(), HealthCheckResponse.of("UP"));
```

```
}
```


Java 8

Java 11

Development

Unit and
Integration
Tests

Functional
Tests

Production

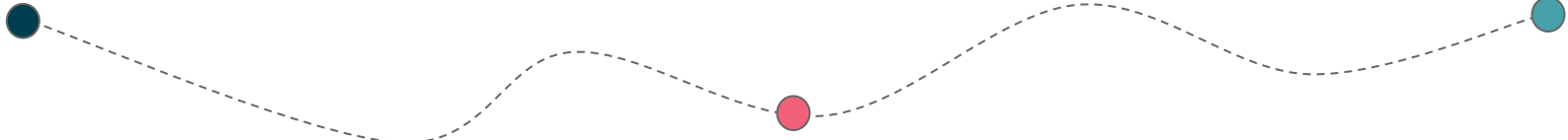
**Why not jumping
straight to Java 17?**

**We are
here!**

Java 8

Java 11

Java 17

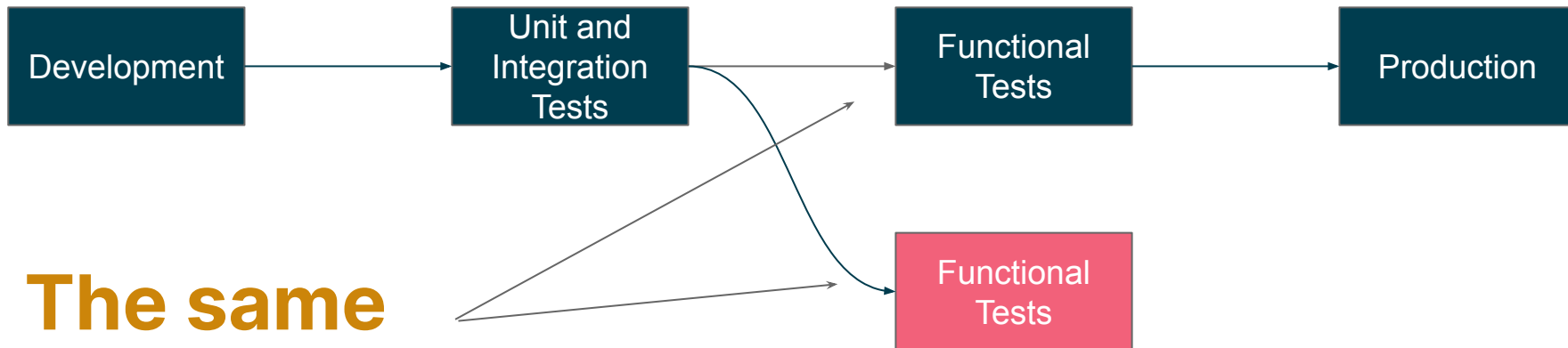


Changes required to move to Java 11

Changes required to move to Java 17

Java 8

Java 11



**The same
JAR file is used
in both cases**

```
FROM adoptopenjdk:8u292-b10-jre-hotspot  
WORKDIR /opt/app
```

```
COPY build/libs/app.jar app.jar
```

Same JAR File

```
ENTRYPOINT ["java", "-jar", "app.jar"]
```

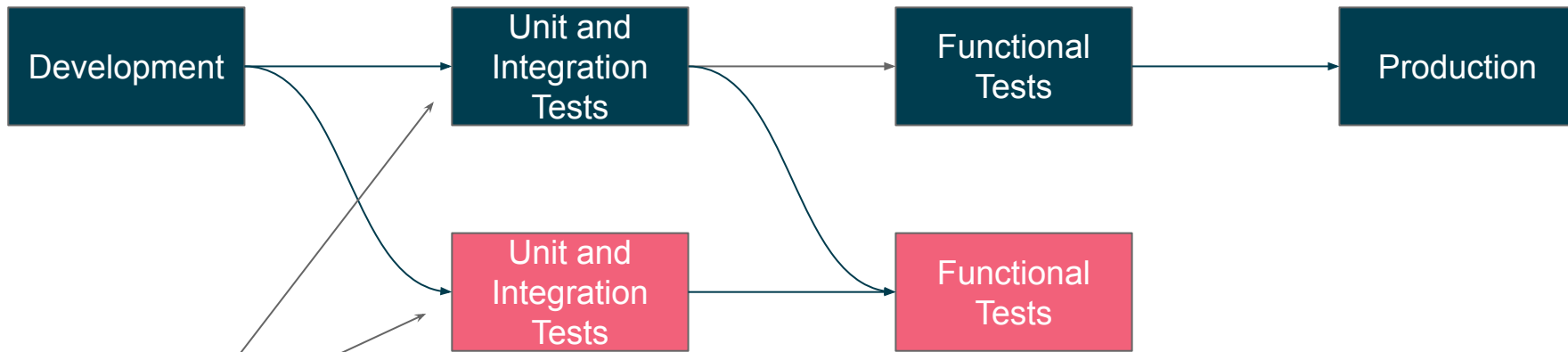
```
FROM adoptopenjdk/openjdk11:x86_64-alpine-jre-11.0.13_8  
WORKDIR /opt/app
```

```
COPY build/libs/app.jar app.jar
```

```
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Java 8

Java 11



**The same source files
are used in both cases**

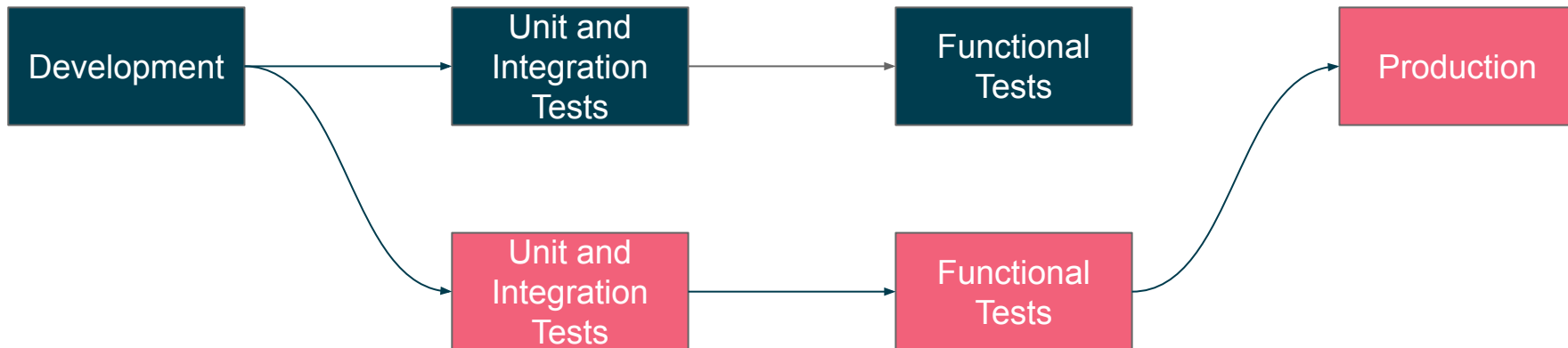
```
java {  
    toolchain { JavaToolchainSpec it ->  
        languageVersion = JavaLanguageVersion.of(8)  
    }  
}
```

```
java {  
    toolchain { JavaToolchainSpec it ->  
        boolean useNewJavaVersion = System.getProperty("app.new.java.version", "false")  
        int javaVersion = useNewJavaVersion ? 11 : 8  
        languageVersion = JavaLanguageVersion.of(javaVersion)  
    }  
}
```

```
$ ./gradlew build -Dapp.new.java.version=true
```

Java 8

Java 11



Java 8

Java 11

Development

Unit and
Integration
Tests

Functional
Tests

Production

**The source code
does not yet use
Java 11 features**

Java 11



Are we there yet?

```
java {  
  toolchain { JavaToolchainSpec it ->  
    languageVersion = JavaLanguageVersion.of(11)  
  }  
}
```

No need to have this
parameterized once the whole
pipeline is switched to the same
version of Java

Rinse & Repeat

Thank you

Albert Attard
albert.attard@thoughtworks.com

