Java versions

a migration strategy
by Albert Attard

/thoughtworks

Agenda

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

Code examples

All code examples can be downloaded from:

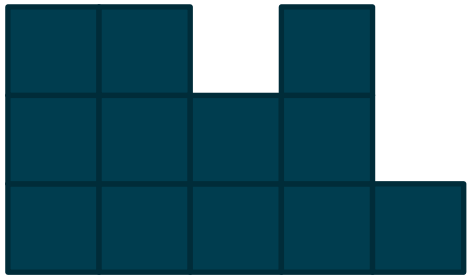
https://bit.ly/3bBcw4g

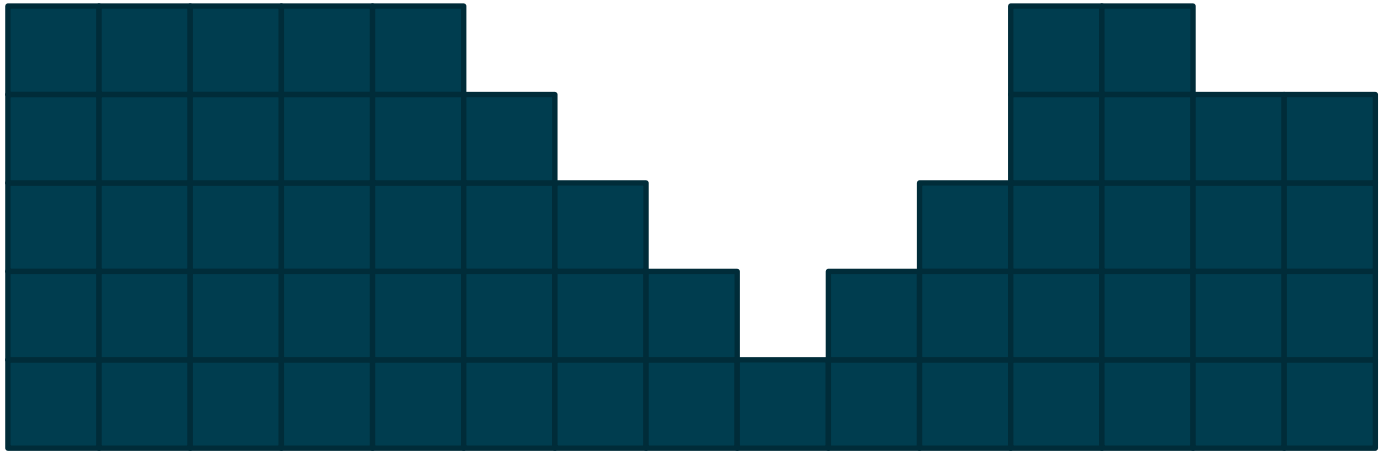
What's so unique to Java version migration?

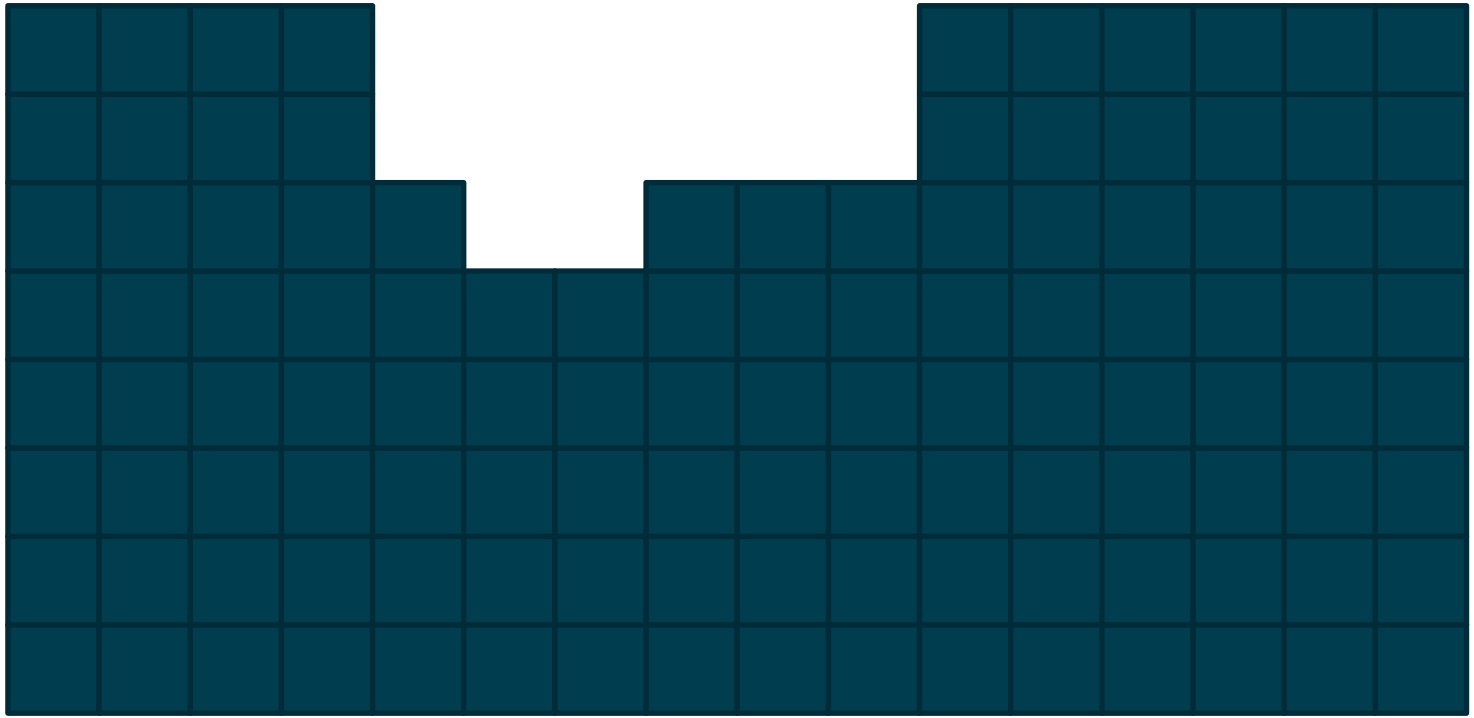
/thoughtworks

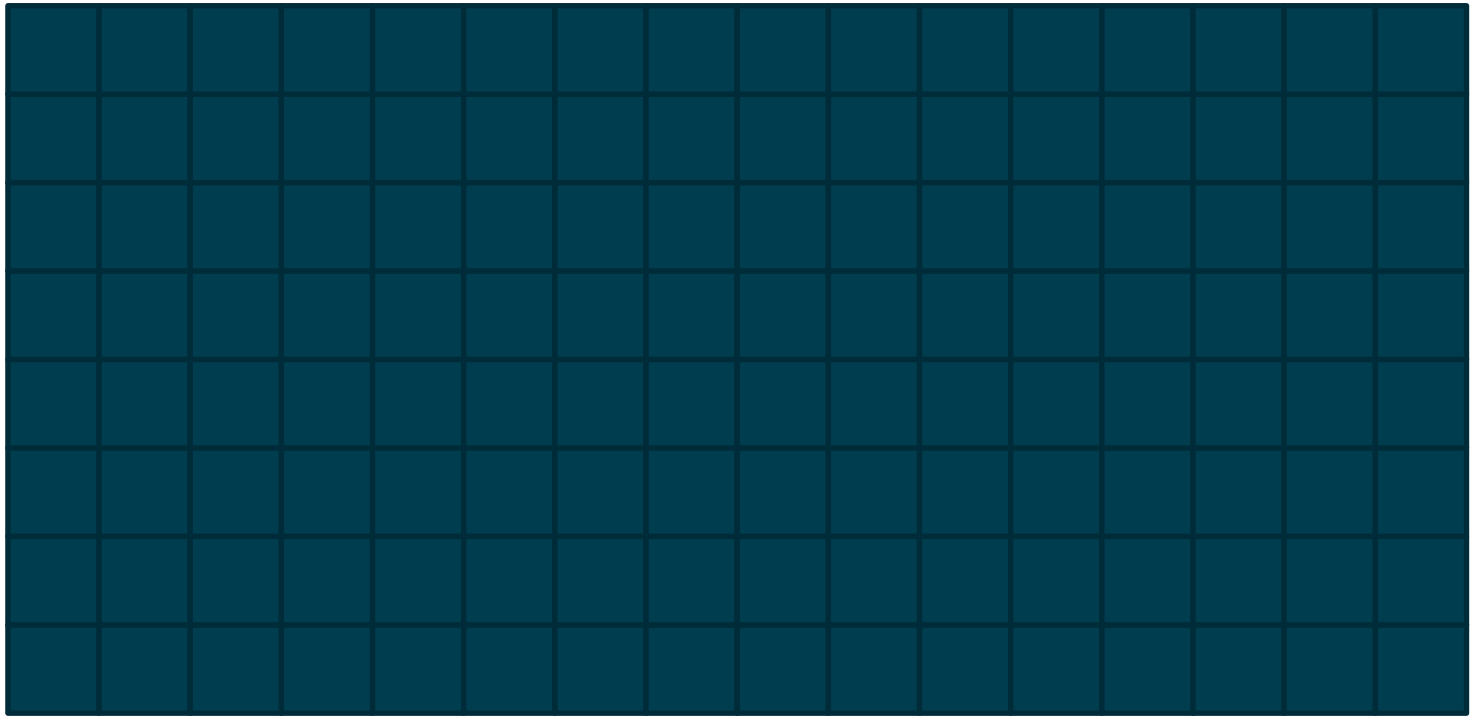
1 feature

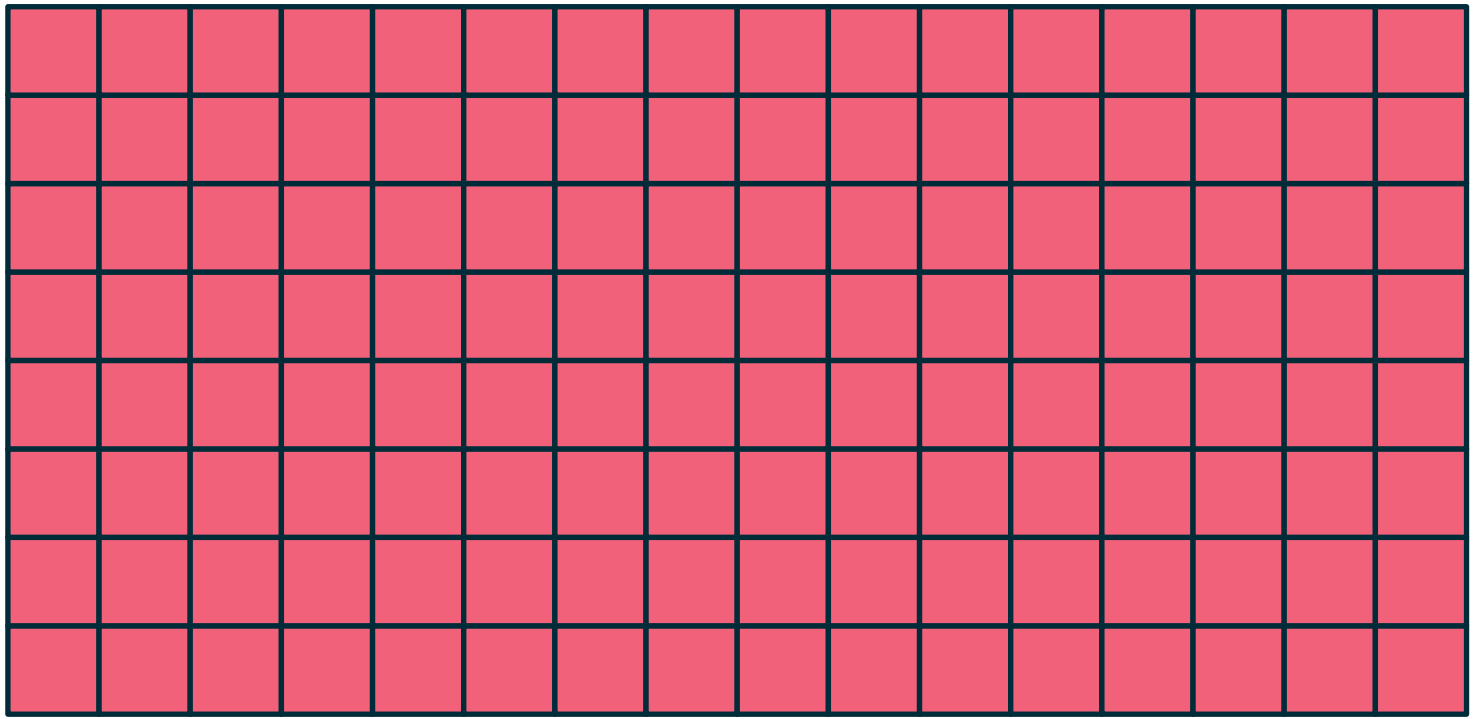
2 features











What can go wrong?

/thoughtworks

Is Java backwards compatible?

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

```
Terminal:  Local
➜  is-java-backwards-compatible git:(main) ✗ ./gradlew build

BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 up-to-date
➜  is-java-backwards-compatible git:(main) ✗
```

```java
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 { JavaToolchainSpec it ->
        languageVersion = JavaLanguageVersion.of(17)
    }
}
```

```
Terminal:  Local  +
Execution failed for task ':compileJava'.
> Compilation failed; see the compiler error output for details.

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug

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

BUILD FAILED in 1s
1 actionable task: 1 executed
➜  is-java-backwards-compatible git:(main) ✗
```

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

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

Terminal: Local

```
➜  is-java-backwards-compatible-lib git:(main) ✗ ./gradlew build

BUILD SUCCESSFUL in 4s
4 actionable tasks: 4 executed
➜  is-java-backwards-compatible-lib git:(main) ✗
```

```java
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 { JavaToolchainSpec it ->
        languageVersion = JavaLanguageVersion.of(17)
    }
}

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

Lombok 1.18.20, and above, support Java 17

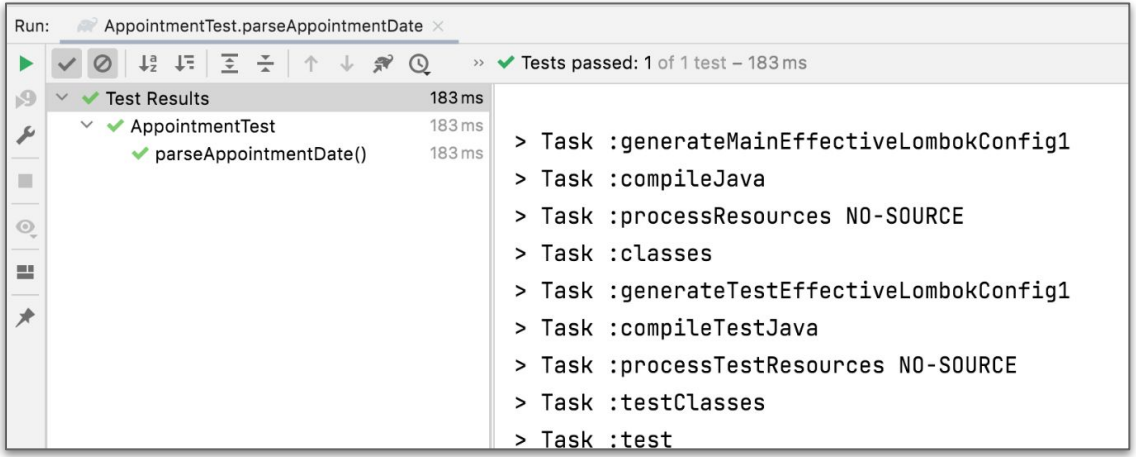
```
Terminal: Local
> java.lang.IllegalAccessError: class lombok.javac.apt.LombokProcessor (in unnamed module @0x1921798f) cannot access class com.sun.tools.javac.processing.JavacProcessingEnvironment (in module jdk.compiler) because module jdk.compiler does not export com.sun.tools.javac.processing to unnamed module @0x1921798f

* 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
2 actionable tasks: 2 executed
➜  is-java-backwards-compatible-lib git:(main) ×
```

A newer dependency version may not play well with other dependencies

```java
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;

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

```java
@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

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

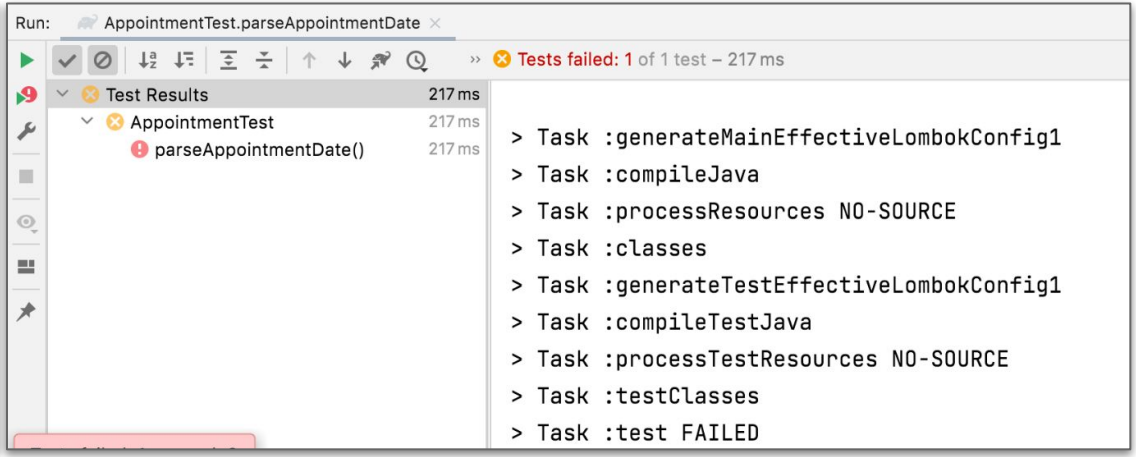
```
plugins {
    id "java"
    id "io.freefair.lombok" version "6.2.0"
}

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

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

repositories {
```



```java
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;

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

```java
@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

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

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



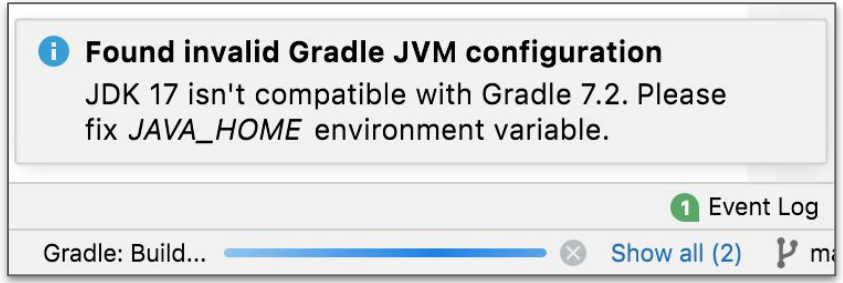
Gradle support for Java 17 is not yet official

Table 1. Java Compatibility

Java version	First Gradle version to support it
8	2.0
9	4.3
10	4.7
11	5.0
12	5.4
13	6.0
14	6.3
15	6.7
16	7.0

Gradle 7.2 seems to support Java 17, but this is not yet official

IntelliJ shows the following warning

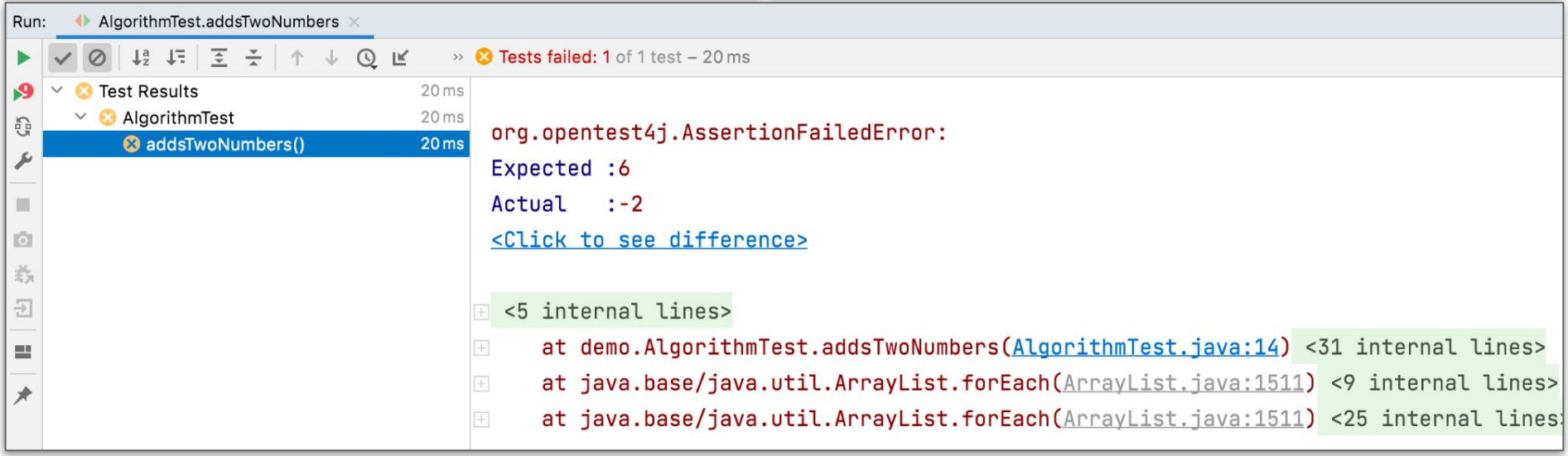


Are the tests really running?

```
kage demo;

lic 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);
}
```



```
➜  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 resource
```

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

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.013 s
[INFO] Finished at: 2021-11-02T08:53:51+01:00
[INFO] ------------------------------------------------------------------------
```

```xml
<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

```xml
<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

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

```java
@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

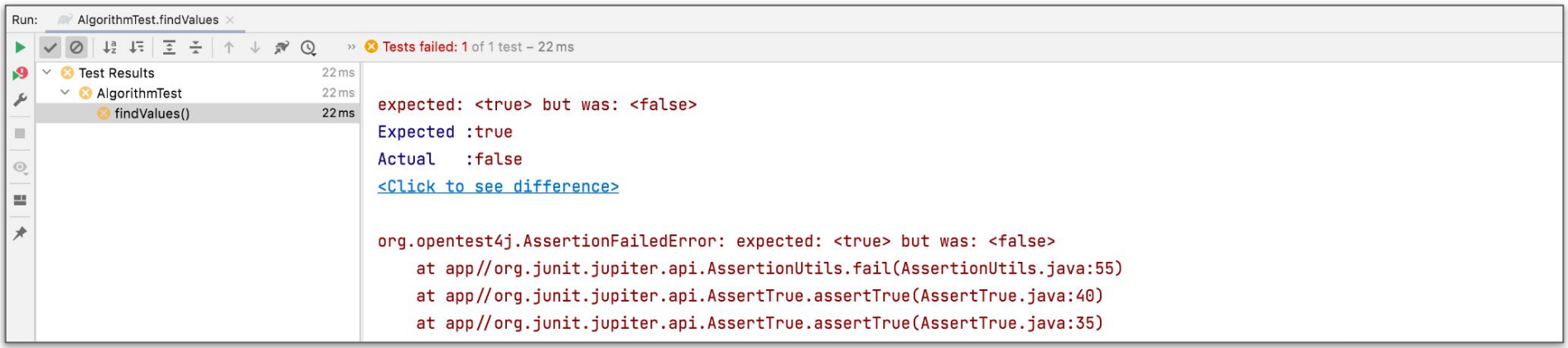
Element	Missed Instructions	Cov.	Missed Branches	Cov.
run(double, double, int, DoubleConsumer)		100 %		100 %
Algorithm()		100 %		n/a
Total	0 of 19	100 %	0 of 2	100 %

100% code coverage

```java
@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

```java
@Test
void varPlus1ShouldBeGreaterThanVar() {
    final double a = 1_000_000_000_000_000_1D;
    final double b = a + 1;

    assertTrue( condition: a < b);
}
```



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;
```

As fast to update as your slowest dependency

A migration strategy

/thoughtworks

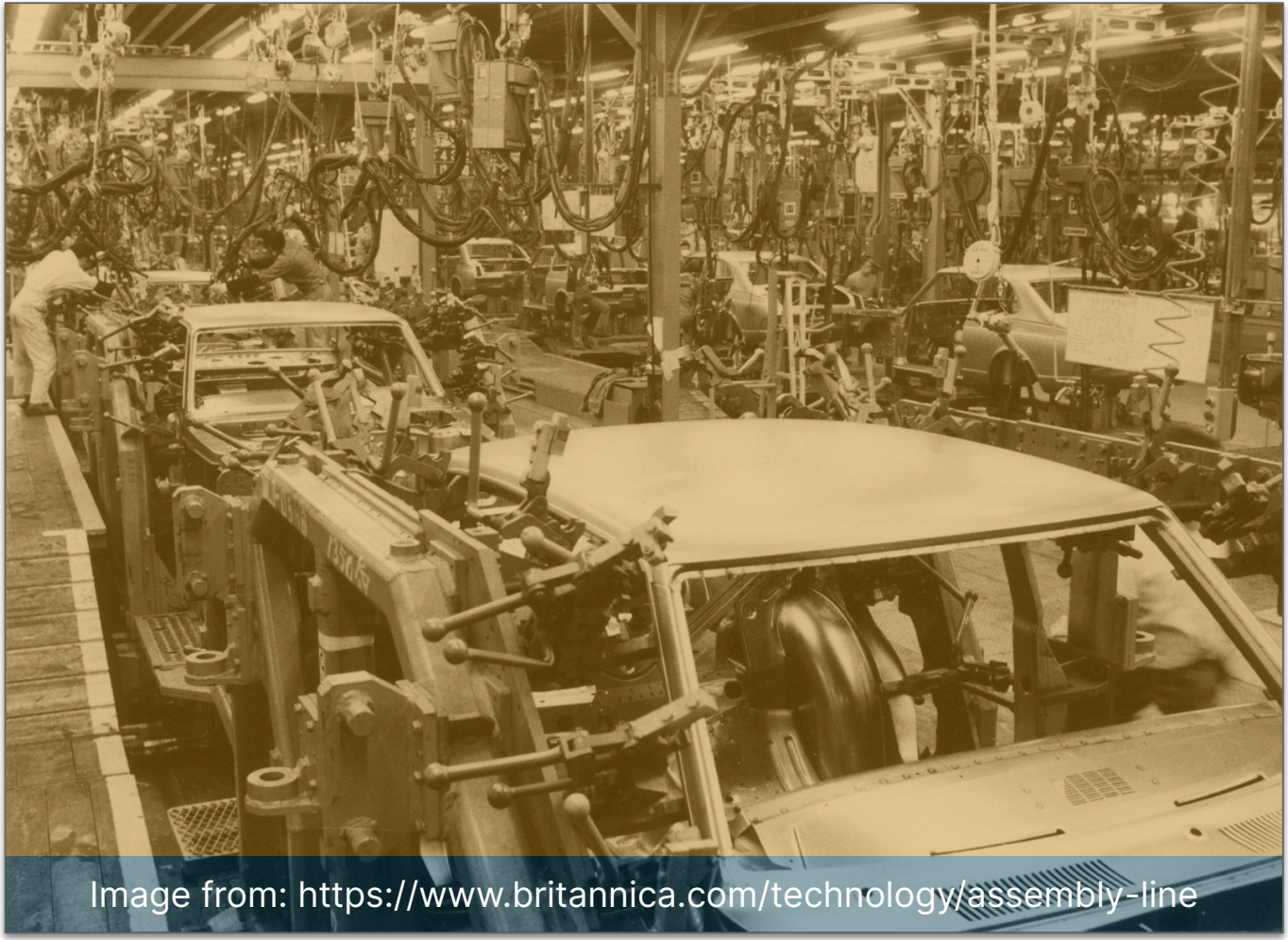


Image from: https://www.britannica.com/technology/assembly-line

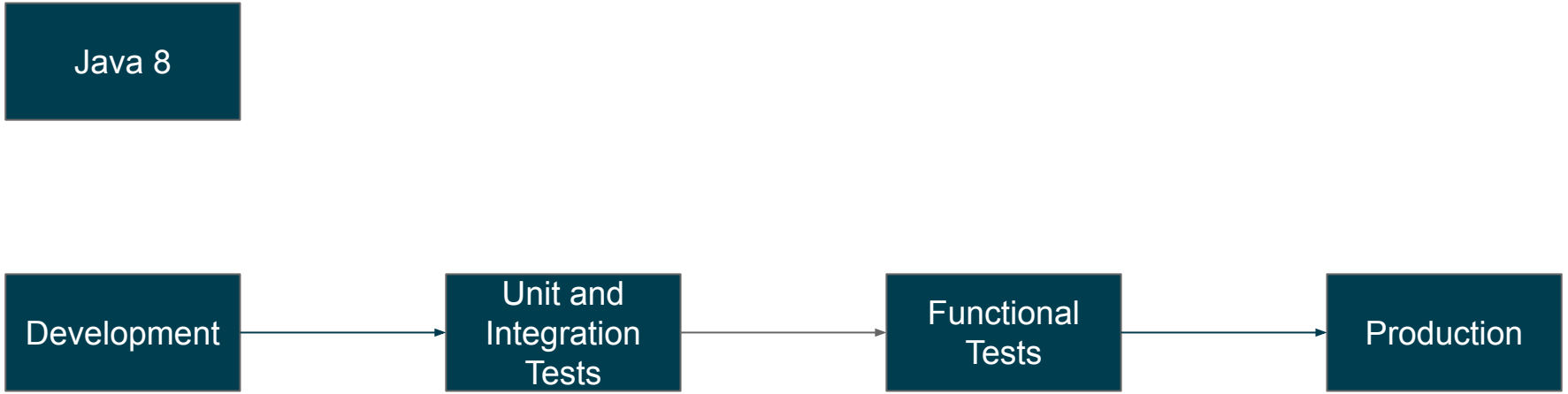
go

Jenkins

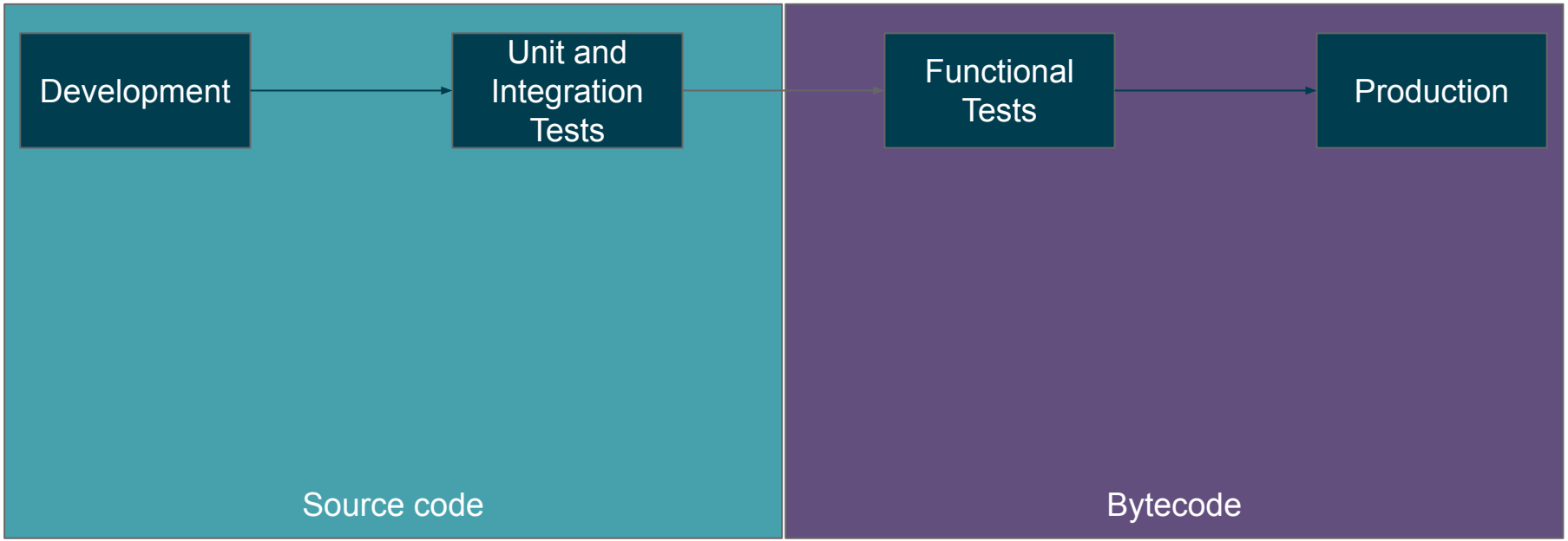
Travis CI

circleci

TeamCity



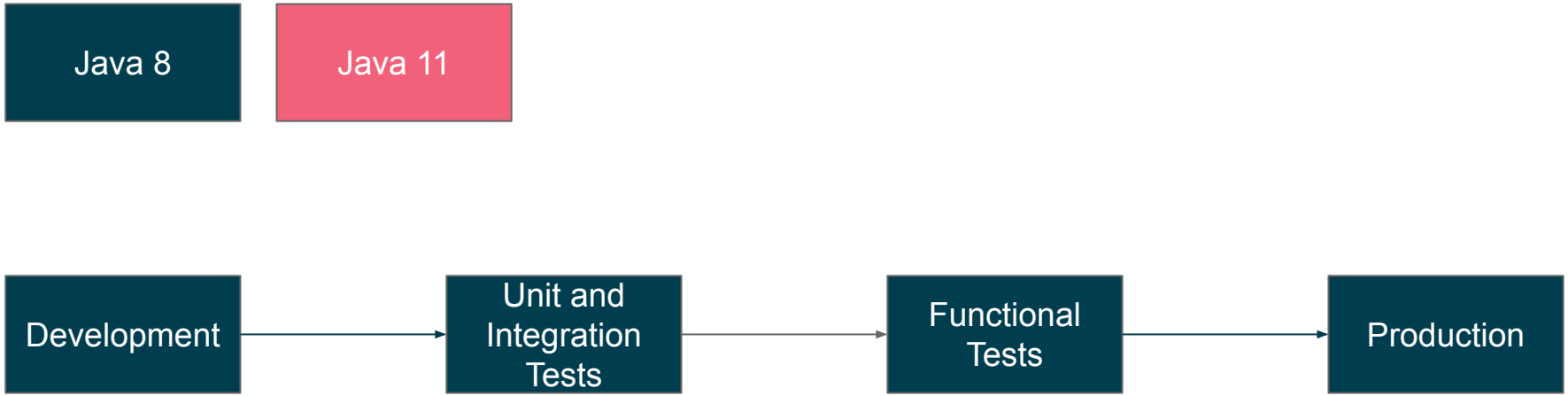
Java 8

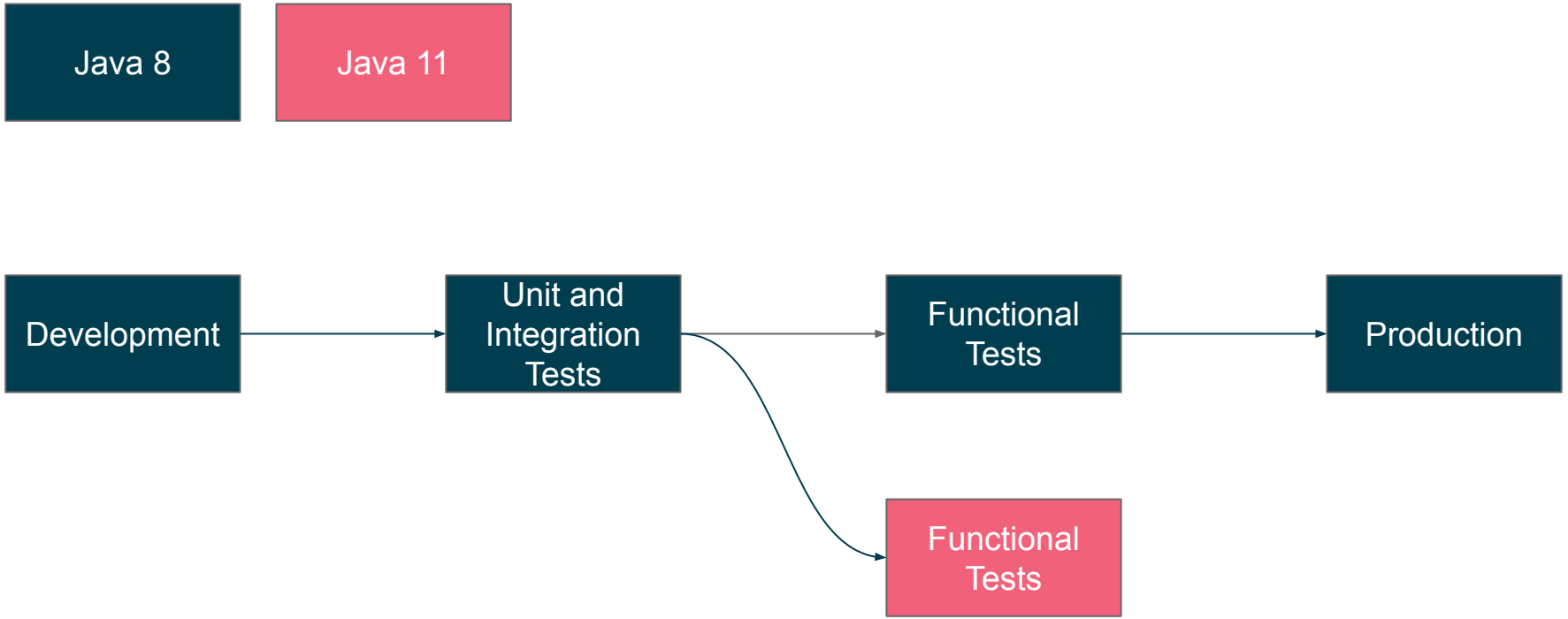


```
FROM adoptopenjdk:8u292-b10-jre-hotspot
WORKDIR /opt/app
COPY build/libs/app.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

The functional tests run against the container

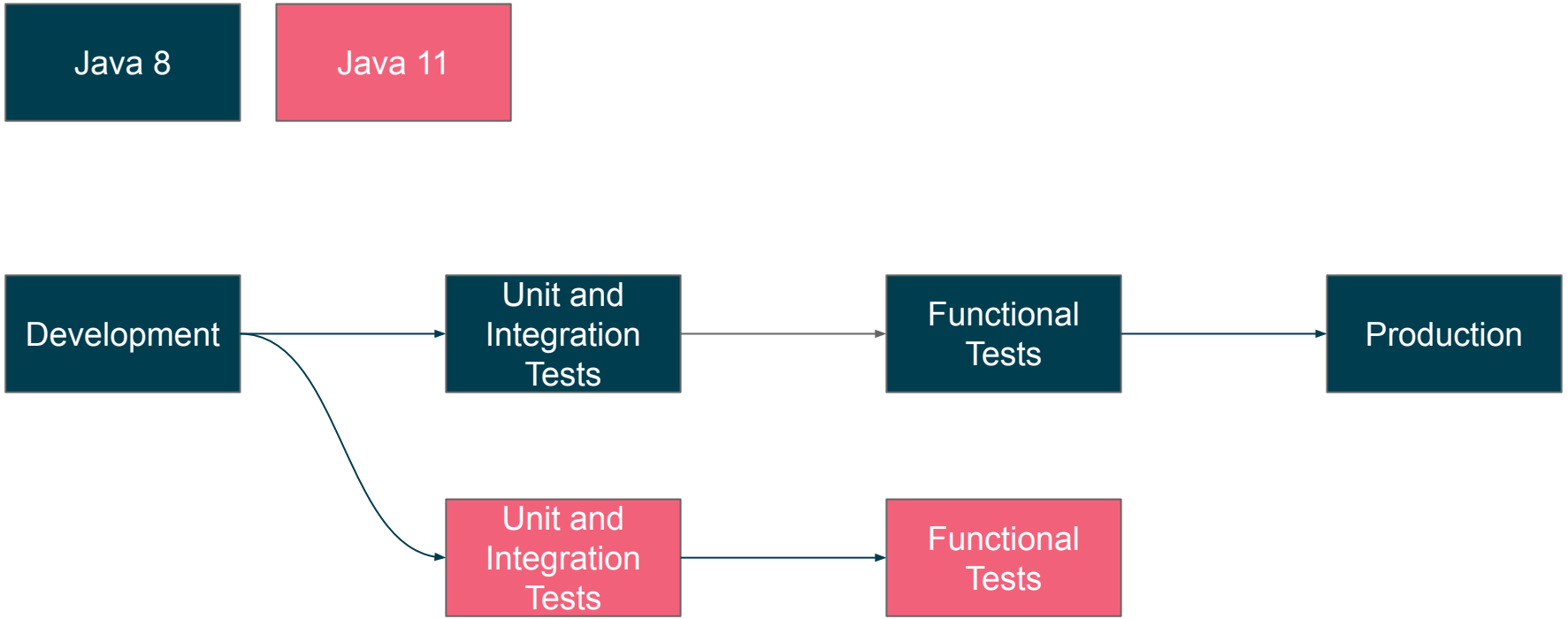
```java
@Container
private final GenericContainer<?> app = createAppContainer();

@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"));
}
```





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

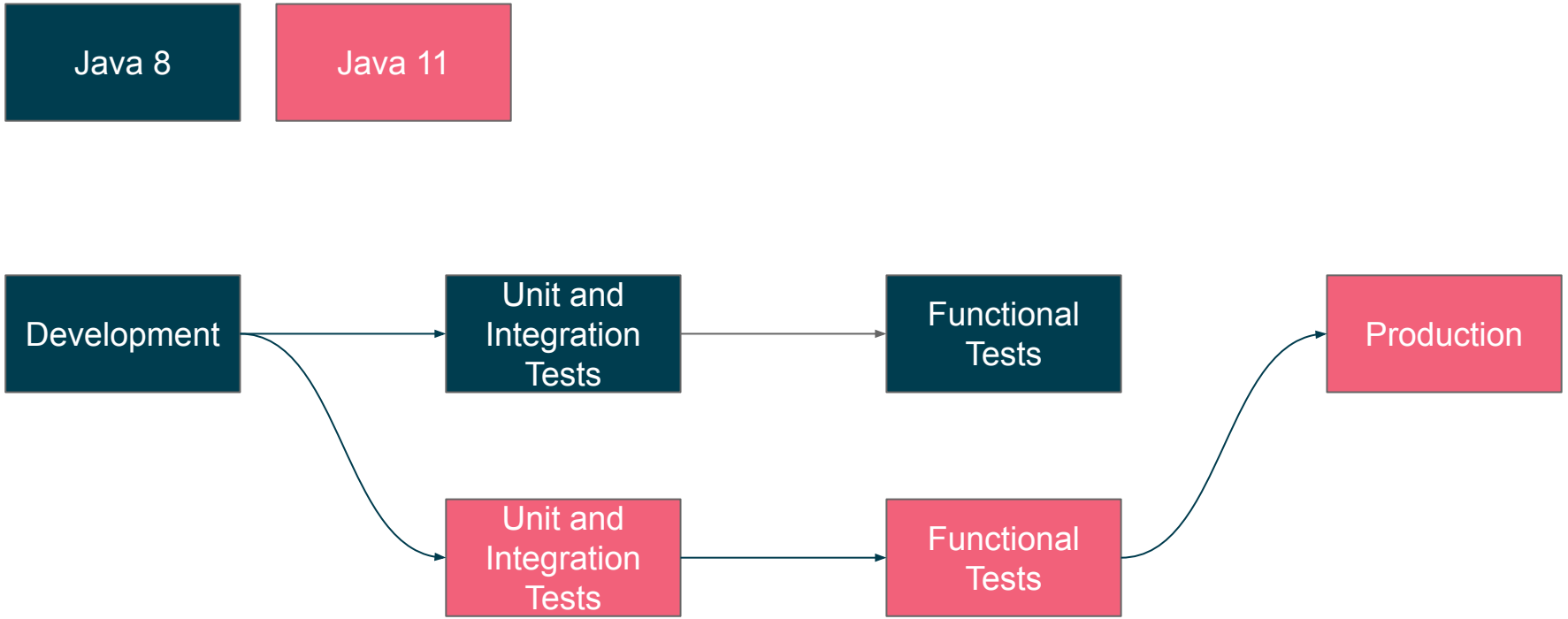
```
FROM adoptopenjdk/openjdk11:x86_64-alpine-jre-11.0.13_8
WORKDIR /opt/app
COPY build/libs/app.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

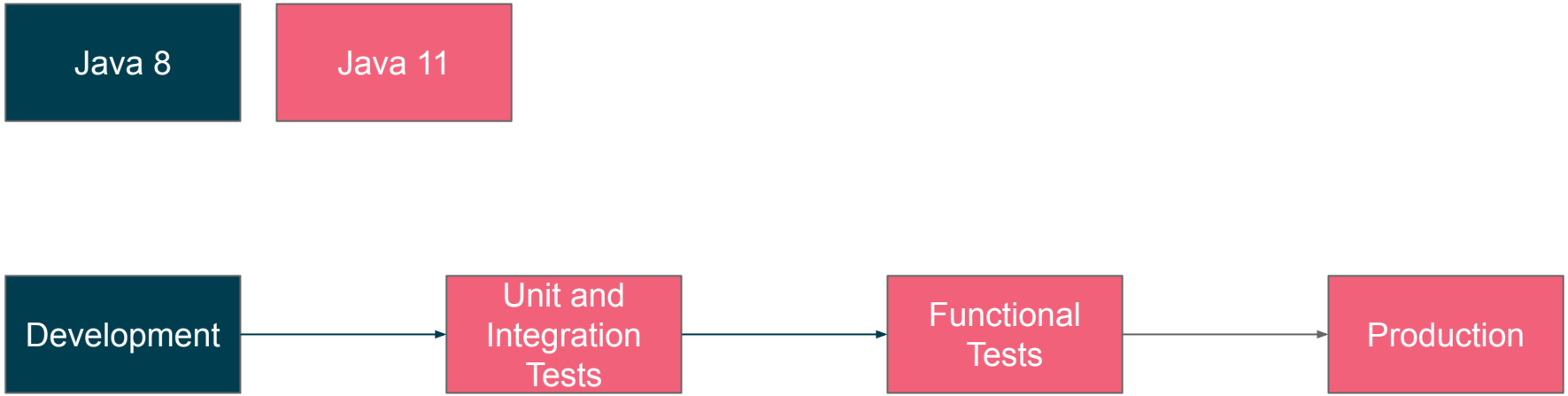


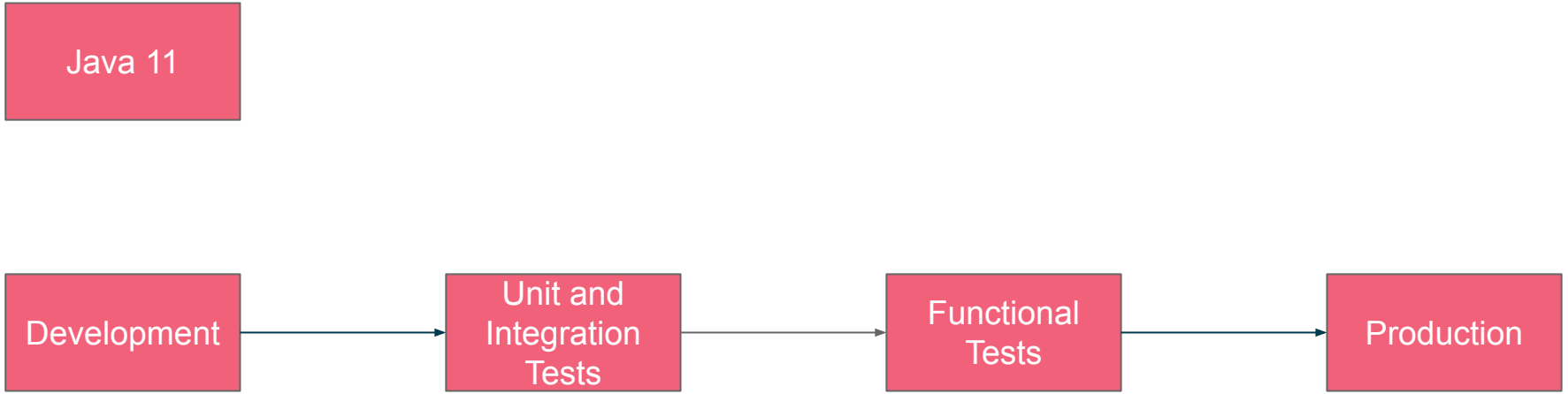
```
java {
    toolchain { JavaToolchainSpec it ->
        languageVersion = JavaLanguageVersion.of(8)
    }
}
```

```
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 {
    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

Thank you

Albert Attard
albert.attard@thoughtworks.com

/thoughtworks