What's New in Java 24

Introducing Java 24

Sander Mak

Software Developer & Java Champion

@sander_mak

Course Overview

Introducing Java 24

Stream Gatherers

Performance & Security Improvements

No preview features

Follow Along

Download JDK 24

jdk.java.net/24/

Java 24 release date: March 18 '25

Not a Long-term Supported release

Deprecations & Removals

Removal

Windows
32-bit x86

Deprecated for removal

Linux
32-bit x86

You can still use older versions

32-bit ARM unsupported

New APIs

java.util.concurrent

Before

```
waitFor(long timeout, TimeUnit unit)

process.waitFor(100, TimeUnit.MILLISECONDS)
```

java.time

Java 24

```
waitFor(Duration duration)

process.waitFor(Duration.ofMillis(100))
```

New APIs

Thread-safe!

Before

```
new StringReader("some string")

CharSequence cs = new StringBuilder();
new StringReader(cs.toString());
```

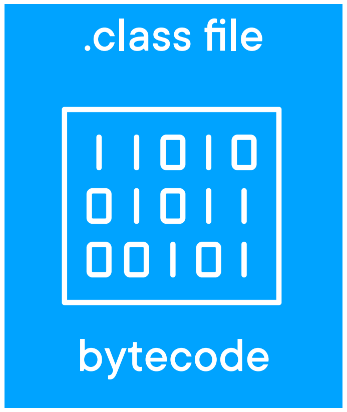
Java 24

```
Reader.of(CharSequence charSequence)

CharSequence cs = new StringBuilder();
Reader.of(cs);
```

Not thread-safe!

New APIs: The Class-File API

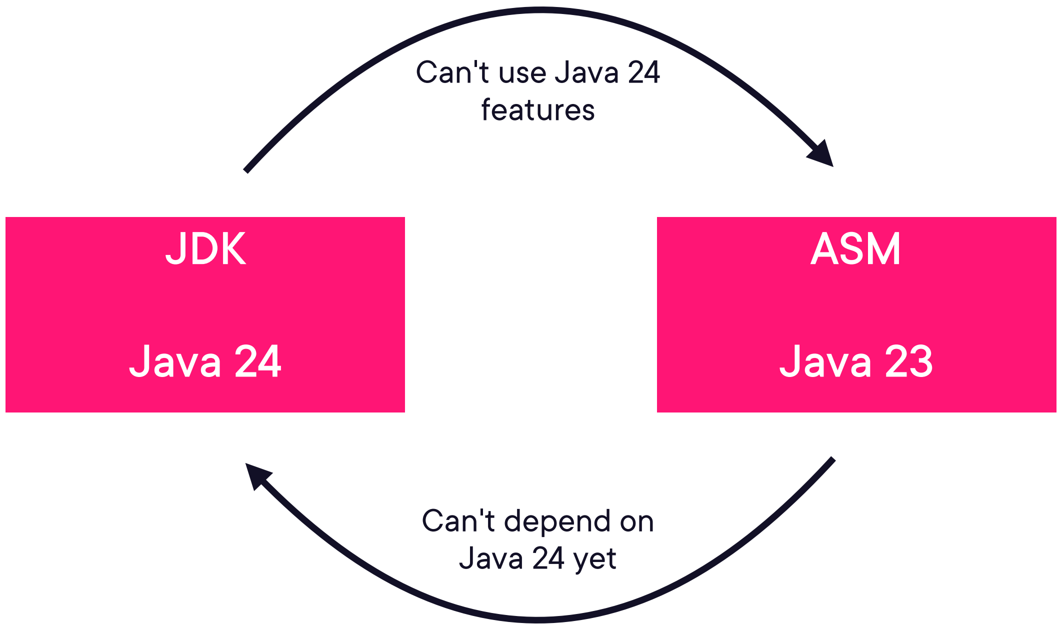


Parse

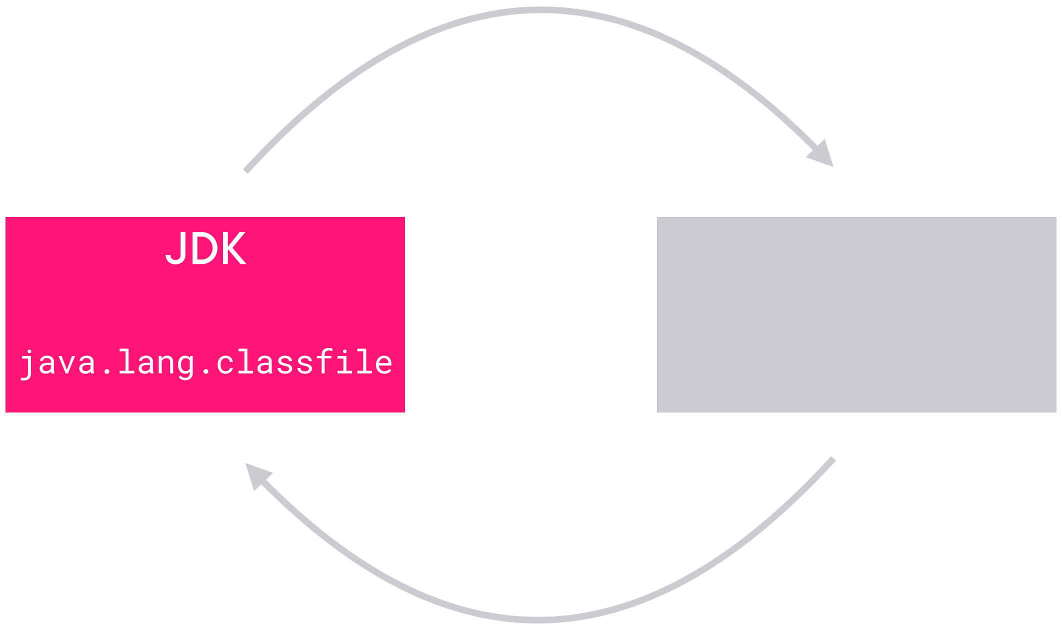
Generate

Transform

New APIs: The Class-File API

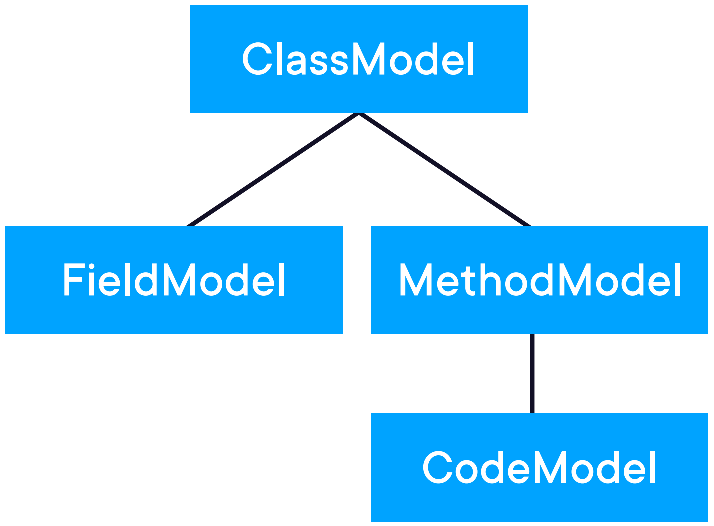


New APIs: The Class-File API



New APIs: The Class-File API

- Immutable & thread-safe
- Builder APIs
- Lazy parsing
- Modern API



Reading a Class File

```
ClassModel model =
  ClassFile.of().parse(path);

model.elementStream()
     .forEach(System.out::println);
```

```
AccessFlags[flags=33]
ClassFileVersion[majorVersion=68, ...]
Superclass[superclassEntry=java/lang/Object]
Interfaces[interfaces=]
FieldModel[fieldName=someField, fieldType=..]
MethodModel[methodName=<init>, methodType=..]
MethodModel[methodName=testMethod, ..]
MethodModel[methodName=regularMethod,..]
Attribute[name=SourceFile]
```

Using Pattern Matching

```
ClassModel model =
  ClassFile.of().parse(path);


model
   .elementStream()
   .map(elem ->
     switch (elem) {
          case MethodModel mm -> mm.methodName();
          case FieldModel fm -> fm.fieldName();
          case ClassFileElement cfe ->
                                  cfe.toString();
         })
    .toList();
```

Transforming and Writing Class Files

```java
// Contains `regularMethod` and `testMethod`
ClassModel model = ClassFile.of().parse(path);
```

Transforming and Writing Class Files

```
// Contains `regularMethod` and `testMethod`
ClassModel model = ClassFile.of().parse(path);


ClassFile.of().buildTo(Path.of("./NoTests.class"), ClassDesc.of("NoTests"), classBuilder -> {









         });
```

Transforming and Writing Class Files

```
// Contains `regularMethod` and `testMethod`
ClassModel model = ClassFile.of().parse(path);


ClassFile.of().buildTo(Path.of("./NoTests.class"), ClassDesc.of("NoTests"), classBuilder -> {








         });
```

Transforming and Writing Class Files

```
// Contains `regularMethod` and `testMethod`
ClassModel model = ClassFile.of().parse(path);


ClassFile.of().buildTo(Path.of("./NoTests.class"), ClassDesc.of("NoTests"), classBuilder -> {







         });
```

Transforming and Writing Class Files

```java
// Contains `regularMethod` and `testMethod`
ClassModel model = ClassFile.of().parse(path);


ClassFile.of().buildTo(Path.of("./NoTests.class"), ClassDesc.of("NoTests"), classBuilder -> {









         });
```

Transforming and Writing Class Files

```
// Contains `regularMethod` and `testMethod`
ClassModel model = ClassFile.of().parse(path);


ClassFile.of().buildTo(Path.of("./NoTests.class"), ClassDesc.of("NoTests"), classBuilder -> {

           for (ClassElement ce : model) {





                classBuilder.with(ce);
            }
         });
```

Transforming and Writing Class Files

```
// Contains `regularMethod` and `testMethod`
ClassModel model = ClassFile.of().parse(path);


ClassFile.of().buildTo(Path.of("./NoTests.class"), ClassDesc.of("NoTests"), classBuilder -> {

           for (ClassElement ce : model) {
                if (ce instanceof MethodModel mm
                        && mm.methodName().stringValue().startsWith("test")) {
                    continue;
                }

                classBuilder.with(ce);
            }
         });
```