Exploring New APIs: ProcessHandle, HttpClient and More

Sander Mak
FELLOW & SOFTWARE ARCHITECT
@Sander_Mak

ProcessHandle

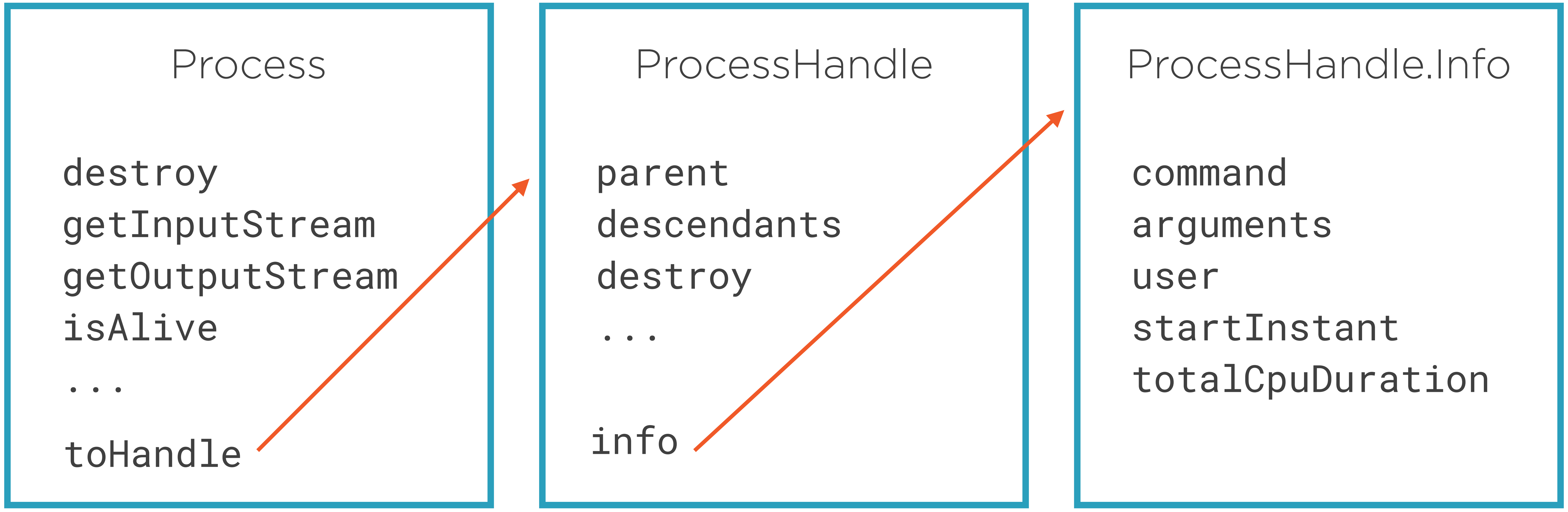
java.lang.Process

Represents native process created from Java

java.lang.ProcessHandle

Represent *any native process* on the operating system

Process and ProcessHandle



```
ProcessHandle.of(123)
```

Process and ProcessHandle

How to get your own process id (pid)

```
Long.parseLong(
  ManagementFactory
   .getRuntimeMXBean()
   .getName()
   .split("@")[0]
);
```

With Java 9

```
ProcessHandle.current().pid();
```

Process and ProcessHandle

If you're like me, the first thing you try to do with new APIs is to try and break stuff...

Kill your own process?

```
ProcessHandle.current().destroyForcibly();
```

java.lang.IllegalStateException: destroy of current process not allowed

Demo

ProcessHandle

Listing all system processes

Find and destroy a process

HttpClient

Replacement for java.net.HTTPURLConnection

Supports HTTP/2, WebSocket

Caveat: incubator module

Likely standard API in Java 10

Goal: easy to use in common cases, enough power for complex ones

HTTP/2 Highlights

Same as HTTP 1.1:

Request/response based

GET/POST/PUT/etc. methods

Differences:

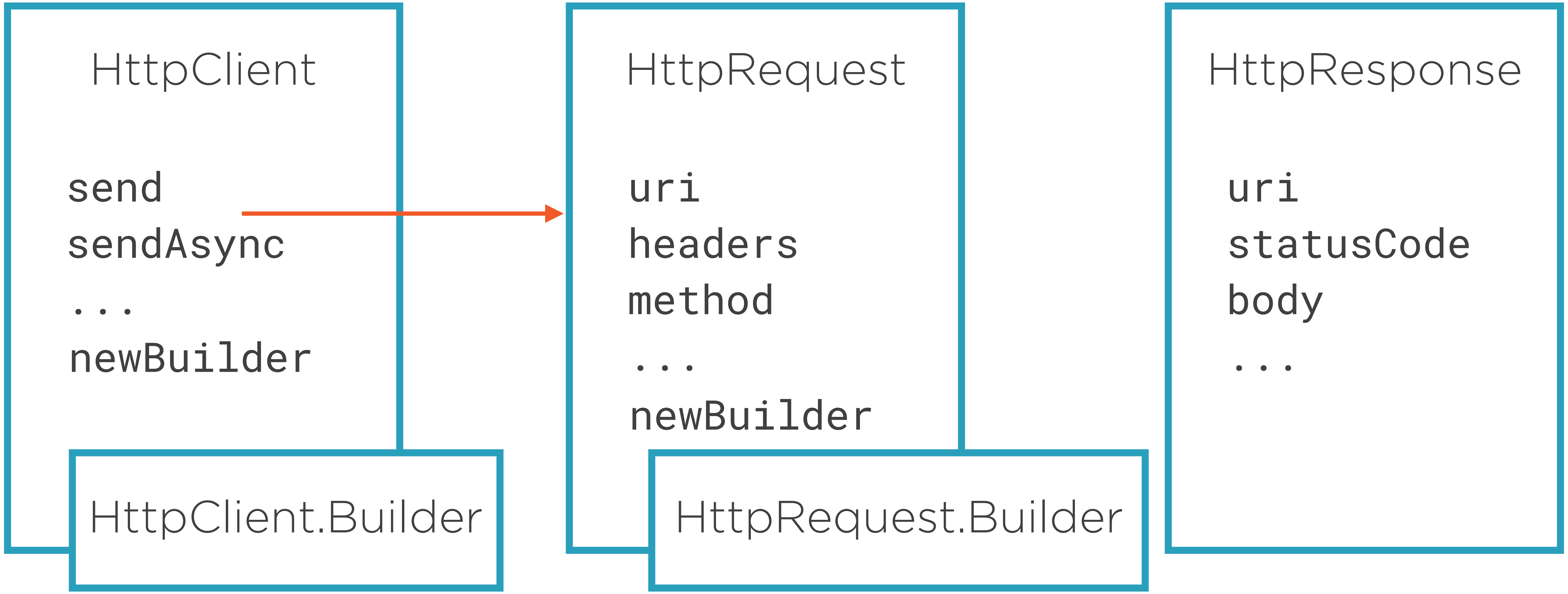
Binary protocol

Mandatory TLS

Multiplexing over single TCP connection (streams)

Server push capability

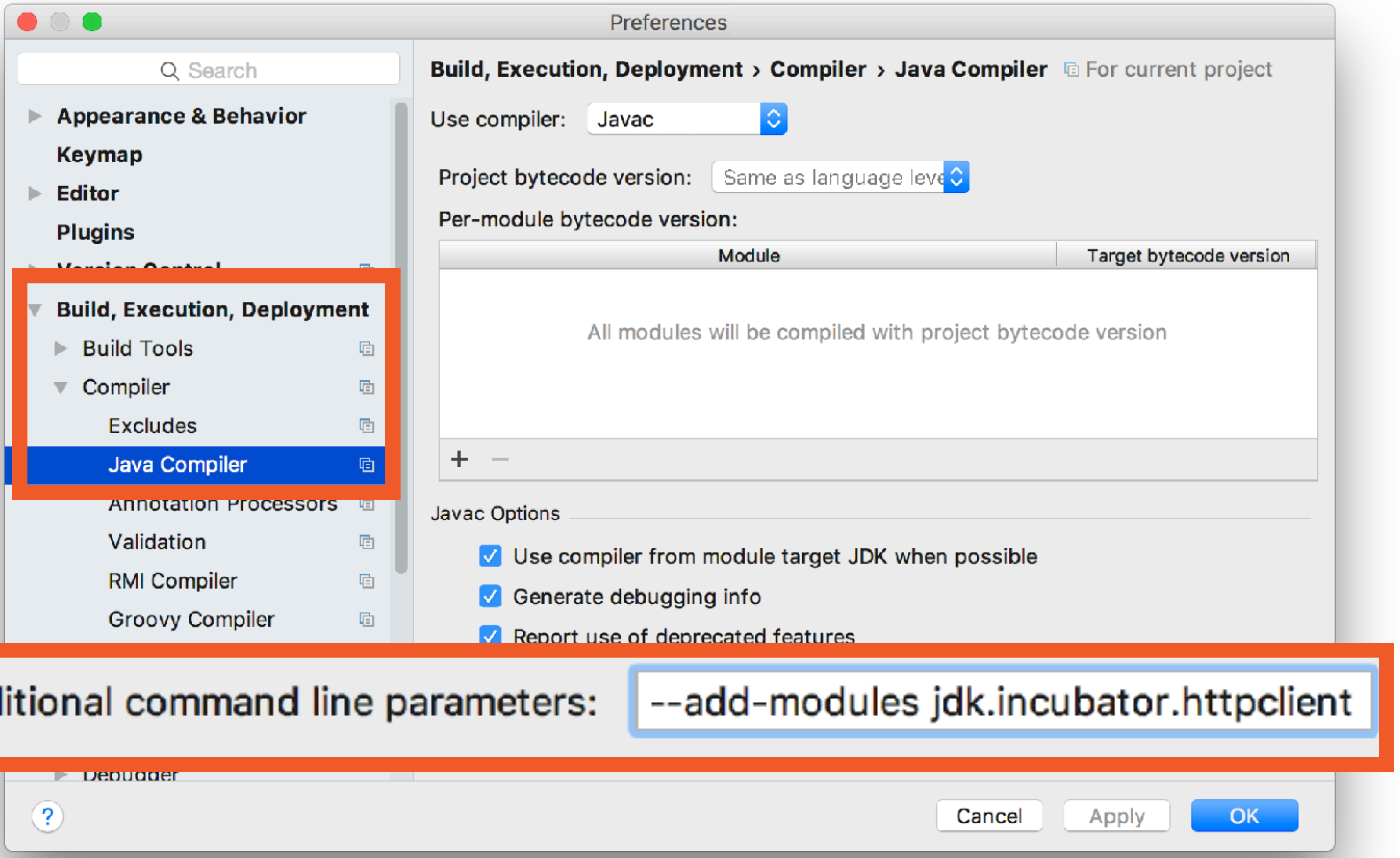
HttpClient: Important Types



Demo

HttpClient Demo

--add-modules jdk.incubator.httpclient



Use compiler: Javac

Project bytecode version: Same as language level

Per-module bytecode version:

Javac Options

- [✓] Use compiler from module target JDK when possible
- [✓] Generate debugging info
- [✓] Report use of deprecated features

Additional command line parameters: --add-modules jdk.incubator.httpclient

Reactive Streams

Stream data with support for backpressure

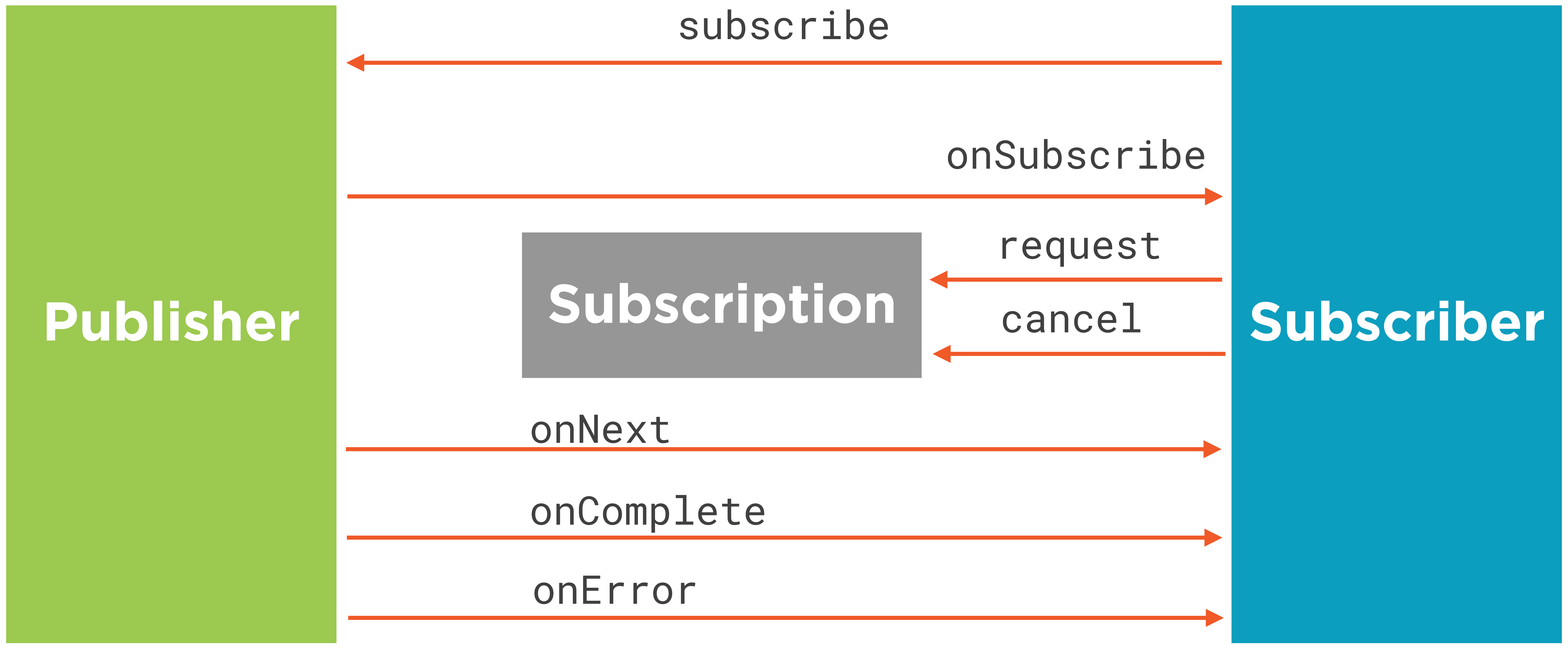
Vendor-neutral specification (http://www.reactive-streams.org)

Flow API: interfaces added to JDK

Interoperability for reactive projects like RxJava, Akka Streams

Not meant as an end-user API

Flow API


subscribe
onSubscribe
request
cancel
Publisher
Subscription
Subscriber
onNext
onComplete
onError

Adoption of java.util.concurrent.Flow

HttpClient implements Publisher/Subscriber interfaces

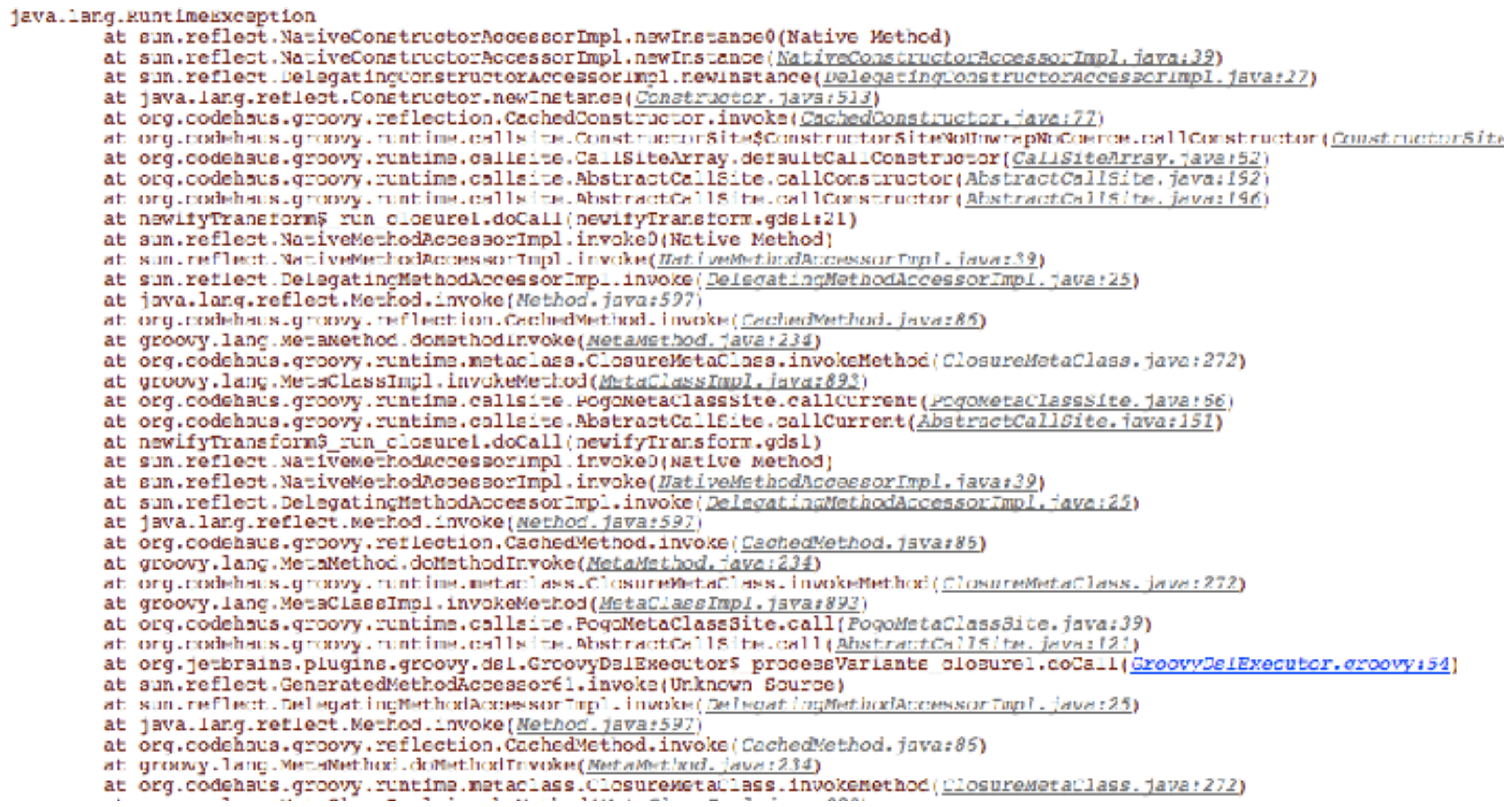
Following projects announced j.u.c.Flow support:

- **RxJava 2**
- **Spring 5**
- **Akka Streams**

Flow API: reactive streams interoperability

StackWalker

Stacktraces



```
java.lang.RuntimeException
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:39)
	at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:27)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:513)
	at org.codehaus.groovy.reflection.CachedConstructor.invoke(CachedConstructor.java:77)
	at org.codehaus.groovy.runtime.callsite.ConstructorSite$ConstructorSiteNoUnwrapNoCoerce.callConstructor(ConstructorSite
	at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCallConstructor(CallSiteArray.java:52)
	at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callConstructor(AbstractCallSite.java:192)
	at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callConstructor(AbstractCallSite.java:196)
	at newifyTransform$_run_closure1.doCall(newifyTransform.gdsl:21)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
	at java.lang.reflect.Method.invoke(Method.java:597)
	at org.codehaus.groovy.reflection.CachedMethod.invoke(CachedMethod.java:86)
	at groovy.lang.MetaMethod.doMethodInvoke(MetaMethod.java:234)
	at org.codehaus.groovy.runtime.metaclass.ClosureMetaClass.invokeMethod(ClosureMetaClass.java:272)
	at groovy.lang.MetaClassImpl.invokeMethod(MetaClassImpl.java:893)
	at org.codehaus.groovy.runtime.callsite.PogoMetaClassSite.callCurrent(PogoMetaClassSite.java:66)
	at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callCurrent(AbstractCallSite.java:151)
	at newifyTransform$_run_closure1.doCall(newifyTransform.gdsl)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
	at java.lang.reflect.Method.invoke(Method.java:597)
	at org.codehaus.groovy.reflection.CachedMethod.invoke(CachedMethod.java:85)
	at groovy.lang.MetaMethod.doMethodInvoke(MetaMethod.java:234)
	at org.codehaus.groovy.runtime.metaclass.ClosureMetaClass.invokeMethod(ClosureMetaClass.java:272)
	at groovy.lang.MetaClassImpl.invokeMethod(MetaClassImpl.java:893)
	at org.codehaus.groovy.runtime.callsite.PogoMetaClassSite.call(PogoMetaClassSite.java:39)
	at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call(AbstractCallSite.java:121)
	at org.jetbrains.plugins.groovy.dsl.GroovyDslExecutor$_processVariants_closure1.doCall(GroovyDslExecutor.groovy:54)
	at sun.reflect.GeneratedMethodAccessor61.invoke(Unknown Source)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
	at java.lang.reflect.Method.invoke(Method.java:597)
	at org.codehaus.groovy.reflection.CachedMethod.invoke(CachedMethod.java:85)
	at groovy.lang.MetaMethod.doMethodInvoke(MetaMethod.java:234)
	at org.codehaus.groovy.runtime.metaclass.ClosureMetaClass.invokeMethod(ClosureMetaClass.java:272)
```

StackWalker

Before Java 9

```
StackTraceElement[] stackTrace =
    new Throwable().getStackTrace();
```

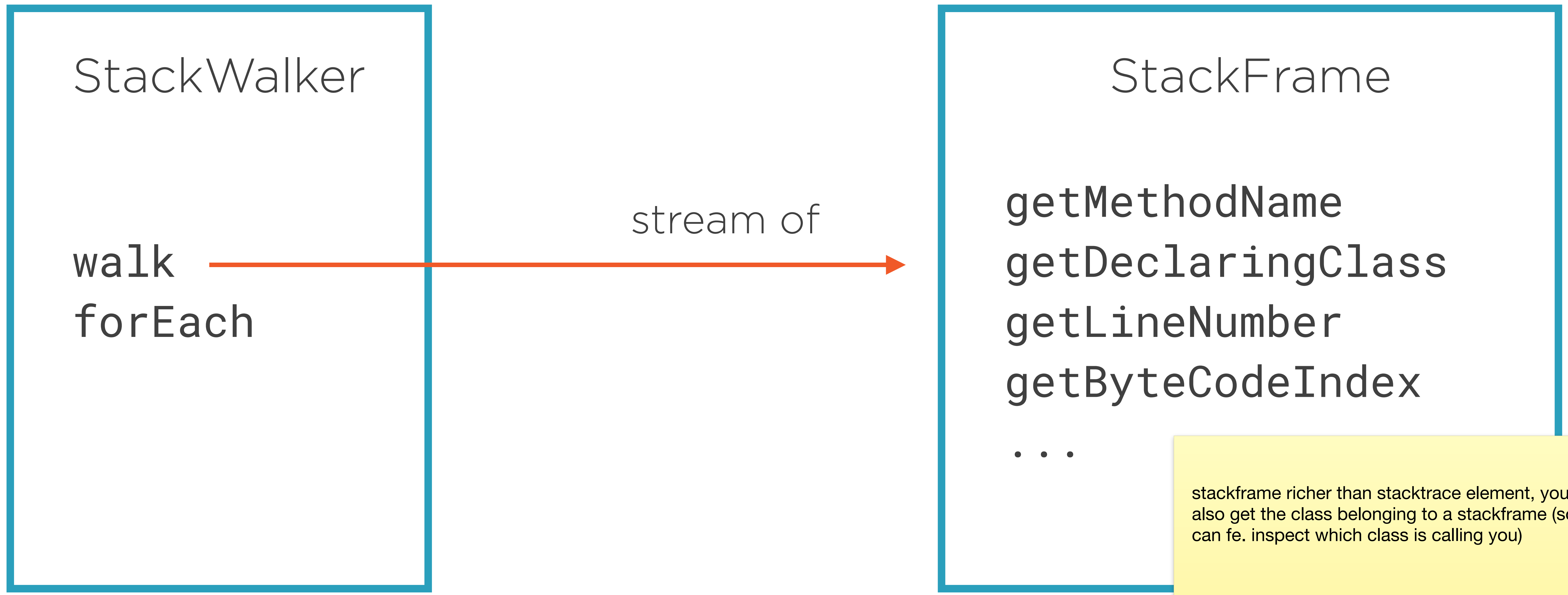
```
StackTraceElement[] stackTrace =
    Thread.getStackTrace();
```

Low performance

No guarantee all stack elements are returned

No partial handling possible

StackWalker



stackframe richer than stacktrace element, you can also get the class belonging to a stackframe (so you can fe. inspect which class is calling you)

StackWalker

Handling all StackFrames

```
StackWalker walker =
    StackWalker.getInstance();

walker.forEach(System.out::println);
```

```
StackWalkerDemo.method4(StackWalkerDemo.java:22)
StackWalkerDemo.method3(StackWalkerDemo.java:16)
StackWalkerDemo.method2(StackWalkerDemo.java:12)
StackWalkerDemo.method1(StackWalkerDemo.java:8)
StackWalkerDemo.main(StackWalkerDemo.java:4)
com.intellij.rt.execution.application.AppMainV2.main(AppMainV2.java:131)
```

StackWalker

Handling specific StackFrames

```
StackWalker walker =
    StackWalker.getInstance();

List<Integer> lines = walker.walk(stackStream ->
    stackStream
      .filter(f -> f.getMethodName().startsWith("m"))
      .map(StackFrame::getLineNumber)
      .collect(Collectors.toList())
);
```

java.lang.IllegalStateException when returning and using stream directly

Summary

Improved process handling

Incubating HttpClient

Reactive Streams interoperability

New stack inspection API