github.com/swankjesse/coroutines-party-tricks

Coroutines Party Tricks

Party Tricks?

Coroutines are neat

You can build powerful things with ’em

You can chop your head off too

I’ve got 10 party tricks!

TRICK #1

I/O

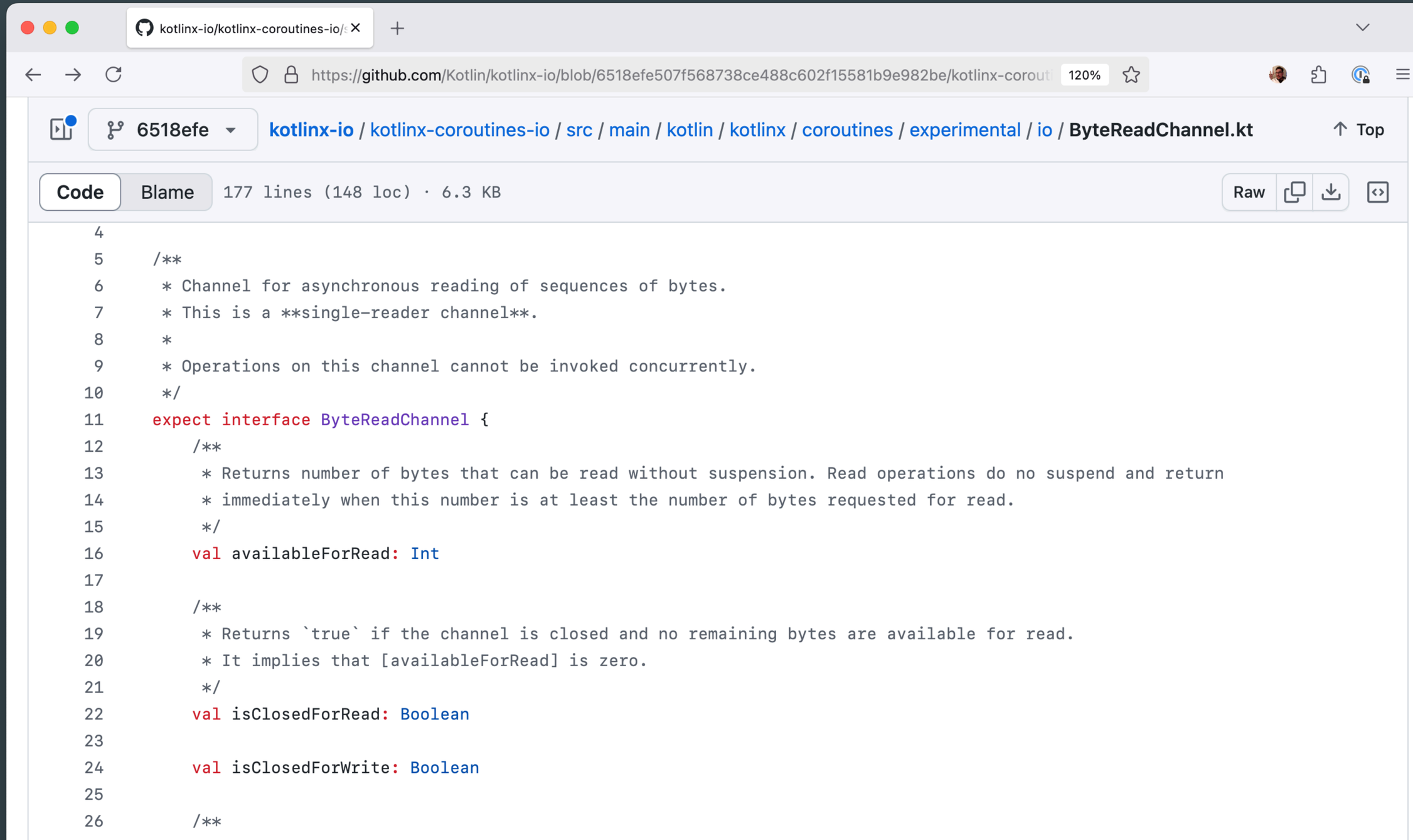
A Seductive Idea

I/O is a frequent source of blocking

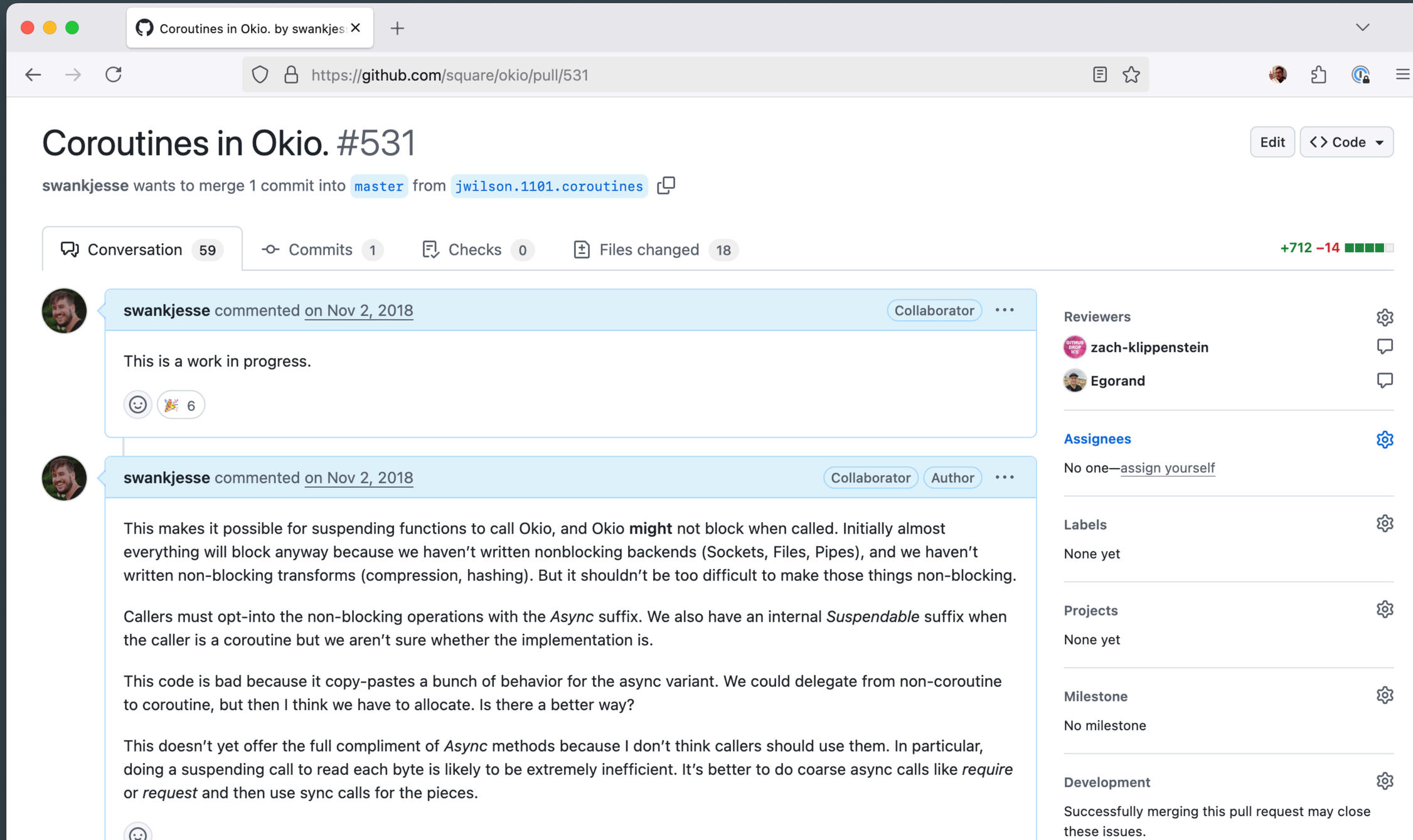
Blocking is bad

Suspending is good

Let’s do suspending I/O !




```
4
5   /**
6    * Channel for asynchronous reading of sequences of bytes.
7    * This is a **single-reader channel**.
8    *
9    * Operations on this channel cannot be invoked concurrently.
10   */
11  expect interface ByteReadChannel {
12      /**
13       * Returns number of bytes that can be read without suspension. Read operations do no suspend and return
14       * immediately when this number is at least the number of bytes requested for read.
15       */
16      val availableForRead: Int
17
18      /**
19       * Returns `true` if the channel is closed and no remaining bytes are available for read.
20       * It implies that [availableForRead] is zero.
21       */
22      val isClosedForRead: Boolean
23
24      val isClosedForWrite: Boolean
25
26      /**
```



Coroutines in Okio. #531

Edit Code

swankjesse wants to merge 1 commit into `master` from `jwilson.1101.coroutines`

Conversation 59 | Commits 1 | Checks 0 | Files changed 18 | +712 −14

swankjesse commented on Nov 2, 2018 — Collaborator

This is a work in progress.

6

swankjesse commented on Nov 2, 2018 — Collaborator, Author

This makes it possible for suspending functions to call Okio, and Okio **might** not block when called. Initially almost everything will block anyway because we haven't written nonblocking backends (Sockets, Files, Pipes), and we haven't written non-blocking transforms (compression, hashing). But it shouldn't be too difficult to make those things non-blocking.

Callers must opt-into the non-blocking operations with the *Async* suffix. We also have an internal *Suspendable* suffix when the caller is a coroutine but we aren't sure whether the implementation is.

This code is bad because it copy-pastes a bunch of behavior for the async variant. We could delegate from non-coroutine to coroutine, but then I think we have to allocate. Is there a better way?

This doesn't yet offer the full compliment of *Async* methods because I don't think callers should use them. In particular, doing a suspending call to read each byte is likely to be extremely inefficient. It's better to do coarse async calls like *require* or *request* and then use sync calls for the pieces.

Reviewers
zach-klippenstein
Egorand

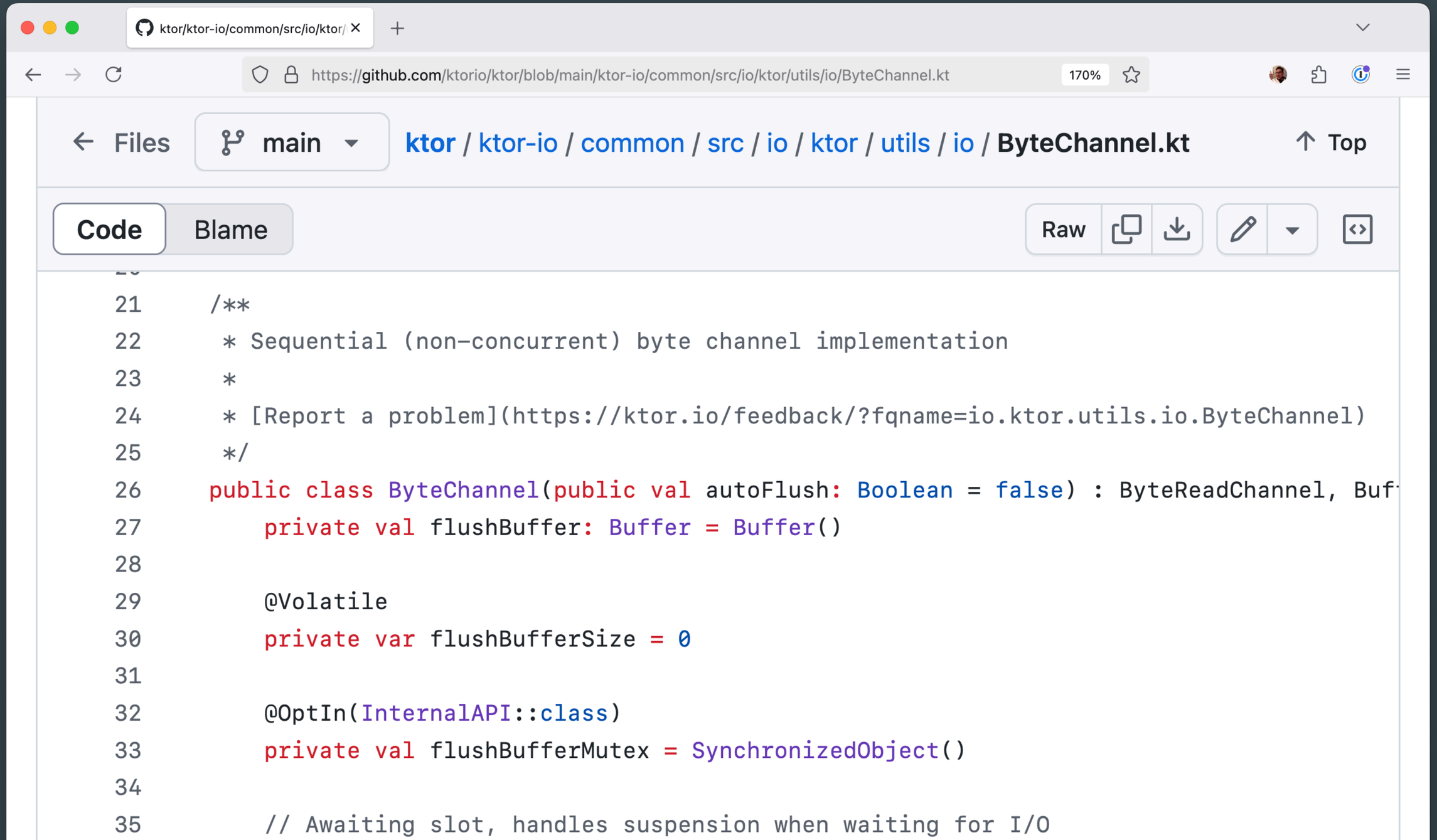
Assignees
No one—assign yourself

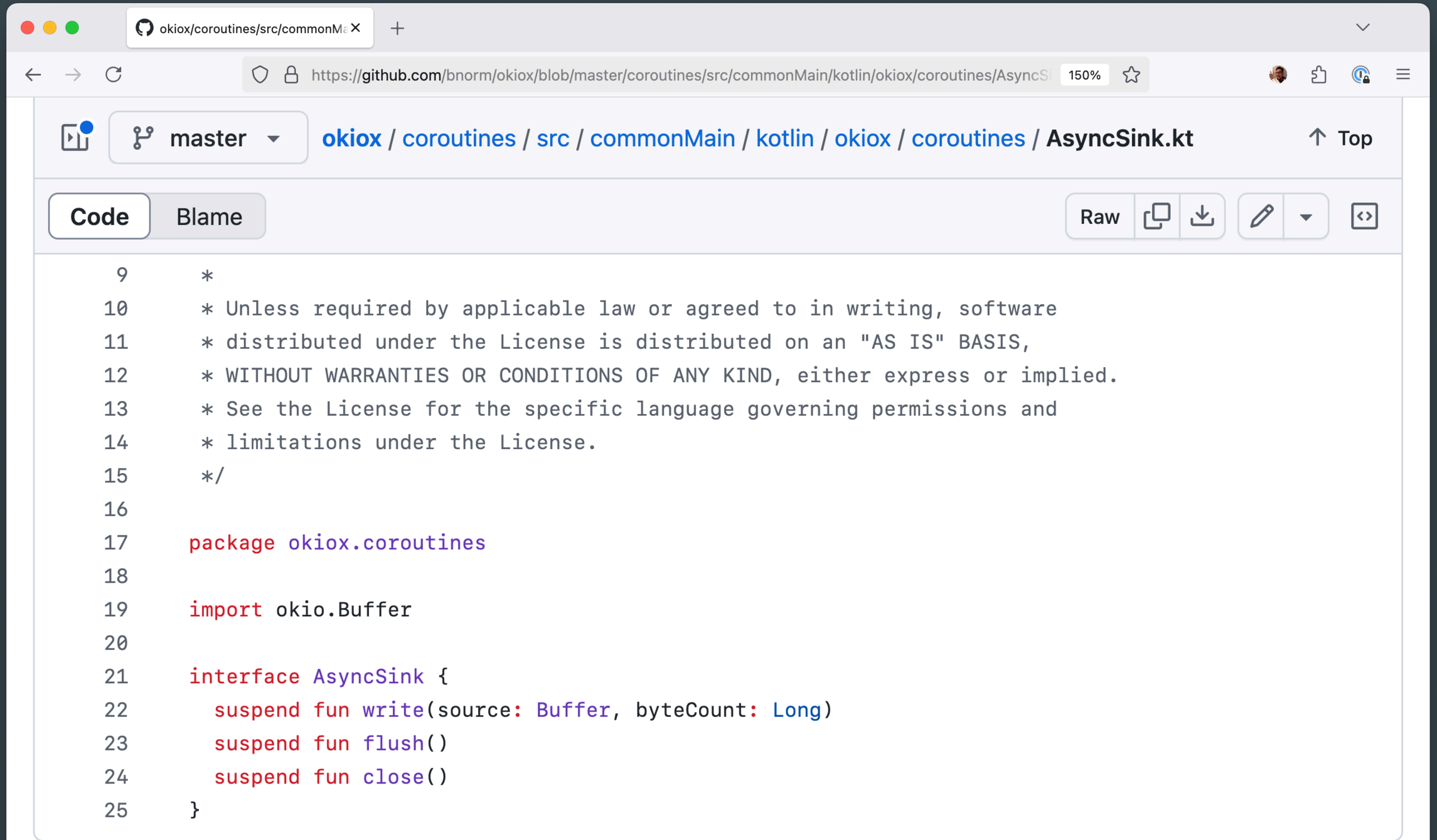
Labels
None yet

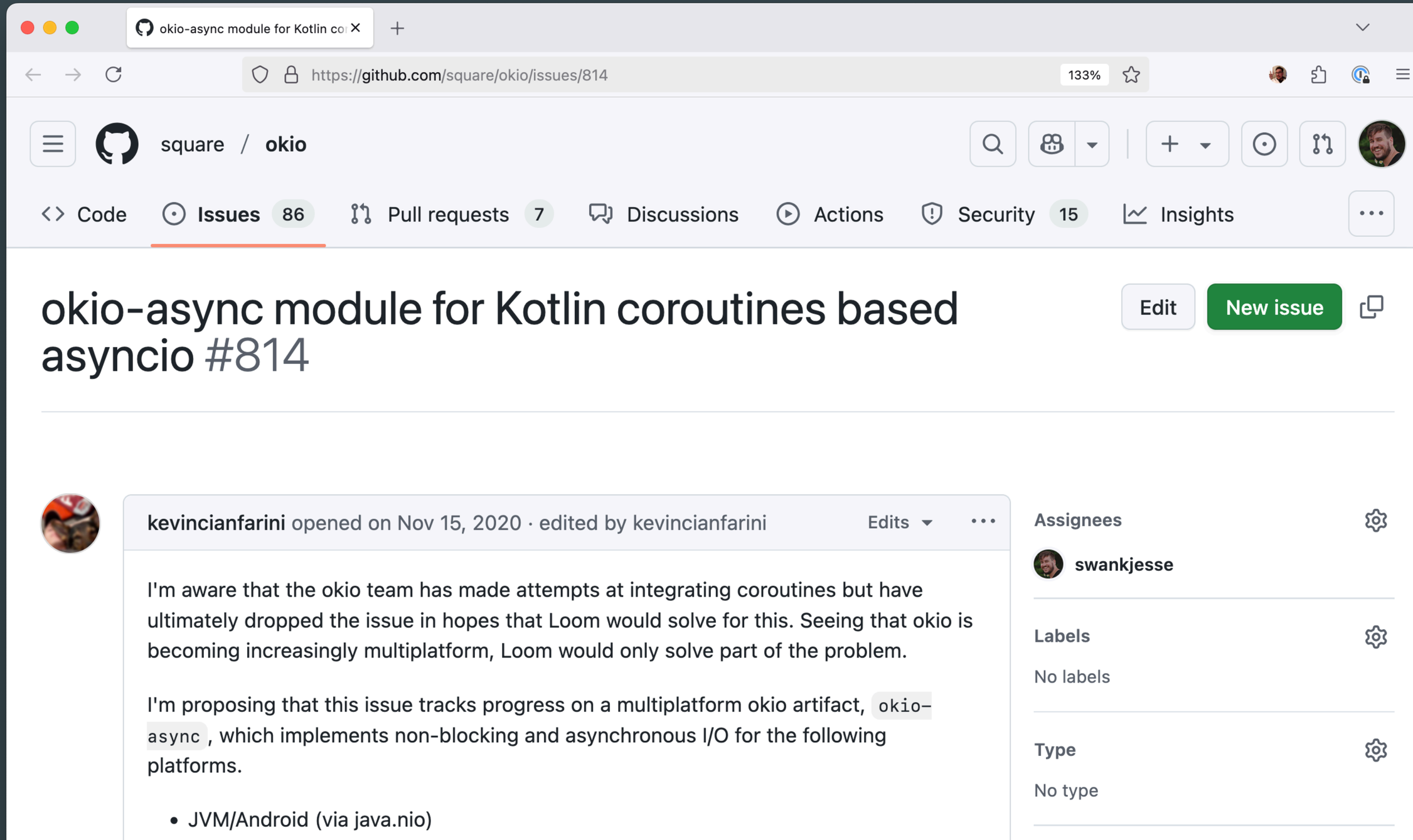
Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.


ktor/ktor-io/common/src/io/ktor/
https://github.com/ktorio/ktor/blob/main/ktor-io/common/src/io/ktor/utils/io/ByteChannel.kt
170%
Files
main
ktor / ktor-io / common / src / io / ktor / utils / io / ByteChannel.kt
Top
Code
Blame
Raw
21 /**
22 * Sequential (non-concurrent) byte channel implementation
23 *
24 * [Report a problem](https://ktor.io/feedback/?fqname=io.ktor.utils.io.ByteChannel)
25 */
26 public class ByteChannel(public val autoFlush: Boolean = false) : ByteReadChannel, Buf
27 private val flushBuffer: Buffer = Buffer()
28
29 @Volatile
30 private var flushBufferSize = 0
31
32 @OptIn(InternalAPI::class)
33 private val flushBufferMutex = SynchronizedObject()
34
35 // Awaiting slot, handles suspension when waiting for I/O


okiox/coroutines/src/commonM
https://github.com/bnorm/okiox/blob/master/coroutines/src/commonMain/kotlin/okiox/coroutines/AsyncSi
150%
master
okiox / coroutines / src / commonMain / kotlin / okiox / coroutines / AsyncSink.kt
Top
Code
Blame
Raw
9 *
10 * Unless required by applicable law or agreed to in writing, software
11 * distributed under the License is distributed on an "AS IS" BASIS,
12 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 * See the License for the specific language governing permissions and
14 * limitations under the License.
15 */
16
17 package okiox.coroutines
18
19 import okio.Buffer
20
21 interface AsyncSink {
22 suspend fun write(source: Buffer, byteCount: Long)
23 suspend fun flush()
24 suspend fun close()
25 }


okio-async module for Kotlin co
https://github.com/square/okio/issues/814
133%
square / okio
Code
Issues 86
Pull requests 7
Discussions
Actions
Security 15
Insights
okio-async module for Kotlin coroutines based asyncio #814
Edit
New issue
kevincianfarini opened on Nov 15, 2020 · edited by kevincianfarini
Edits
I'm aware that the okio team has made attempts at integrating coroutines but have ultimately dropped the issue in hopes that Loom would solve for this. Seeing that okio is becoming increasingly multiplatform, Loom would only solve part of the problem.
I'm proposing that this issue tracks progress on a multiplatform okio artifact, okio-async, which implements non-blocking and asynchronous I/O for the following platforms.
JVM/Android (via java.nio)
Assignees
swankjesse
Labels
No labels
Type
No type

```
interface BufferedSource {

  fun require(byteCount: Long)

  fun request(byteCount: Long): Boolean

  ...
}
```

```
interface BufferedSource {

  suspend fun require(byteCount: Long)

  suspend fun request(byteCount: Long): Boolean

  ...
}
```

Let’s Try

Change Okio & Moshi to `suspend` all the functions that could block

~300 functions in Okio

~200 functions in Moshi

Benchmark

```
@Benchmark
fun blocking() {
  runBlocking {
    val jsonReader = RegularJsonReader.of(regularJson.clone())
    jsonReader.readJsonValue()
  }
}

@Benchmark
fun suspending() {
  runBlocking {
    val jsonReader = SuspendingJsonReader.of(suspendingJson.clone())
    jsonReader.readJsonValue()
  }
}
```

https://zacsweers.github.io/json-serialization-benchmarking/

Benchmark

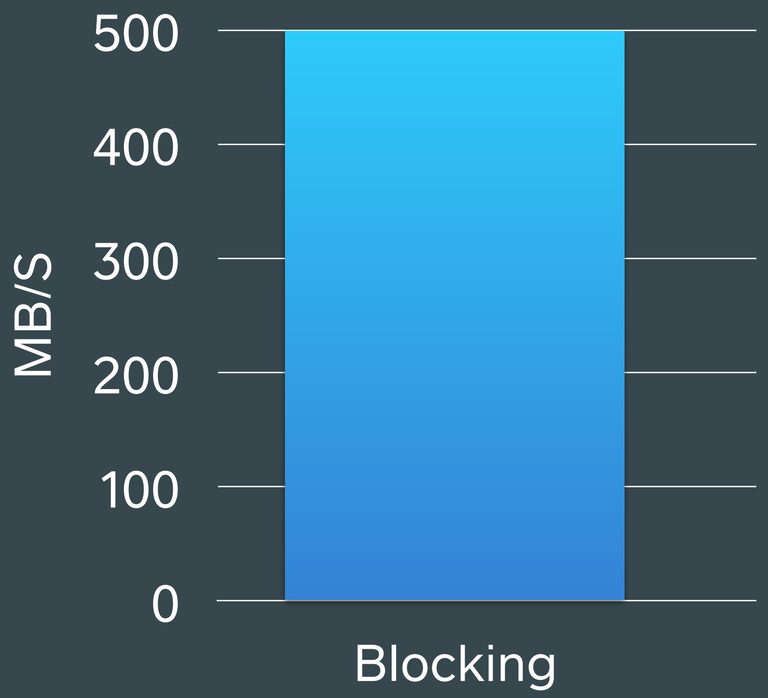
```
@Benchmark
fun blocking() {
  runBlocking {
    val jsonReader = RegularJsonReader.of(regularJson.clone())
    jsonReader.readJsonValue()
  }
}

@Benchmark
fun suspending() {
  runBlocking {
    val jsonReader = SuspendingJsonReader.of(suspendingJson.clone())
    jsonReader.readJsonValue()
  }
}
```

https://zacsweers.github.io/json-serialization-benchmarking/

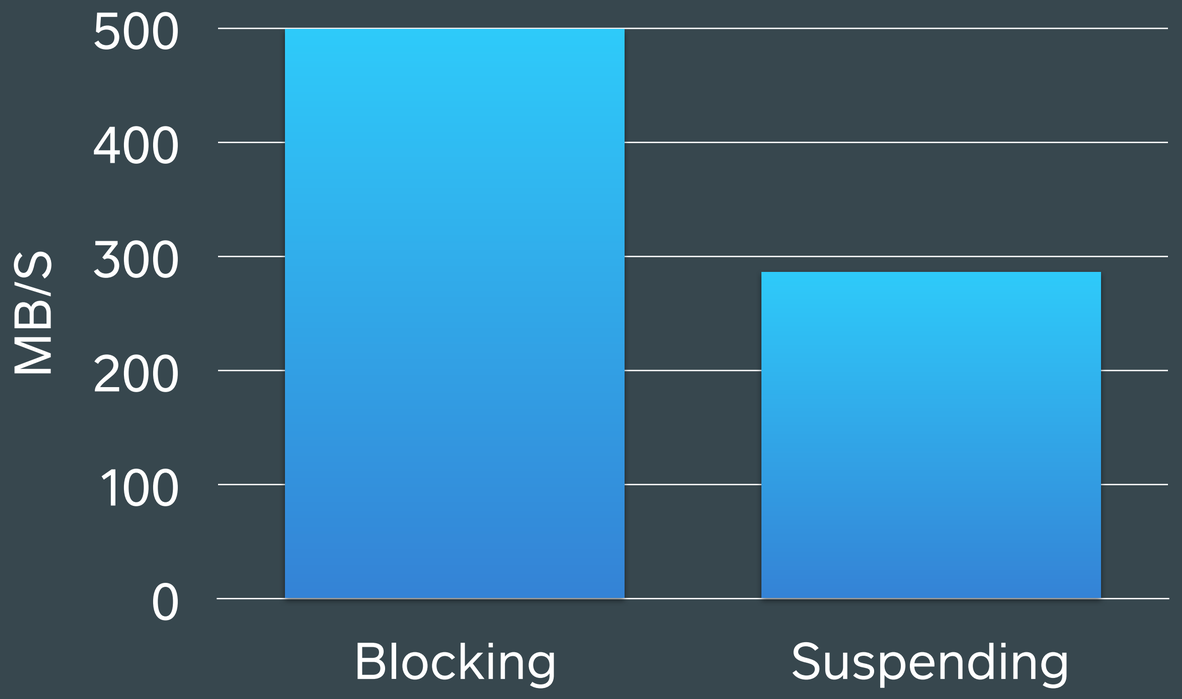
Benchmark

```
@Benchmark
fun blocking() {
  runBlocking {
    val jsonReader = RegularJsonReader.of(regularJson.clone())
    jsonReader.readJsonValue()
  }
}

@Benchmark
fun suspending() {
  runBlocking {
    val jsonReader = SuspendingJsonReader.of(suspendingJson.clone())
    jsonReader.readJsonValue()
  }
}
```

https://zacsweers.github.io/json-serialization-benchmarking/



(higher numbers are better)



(higher numbers are better)

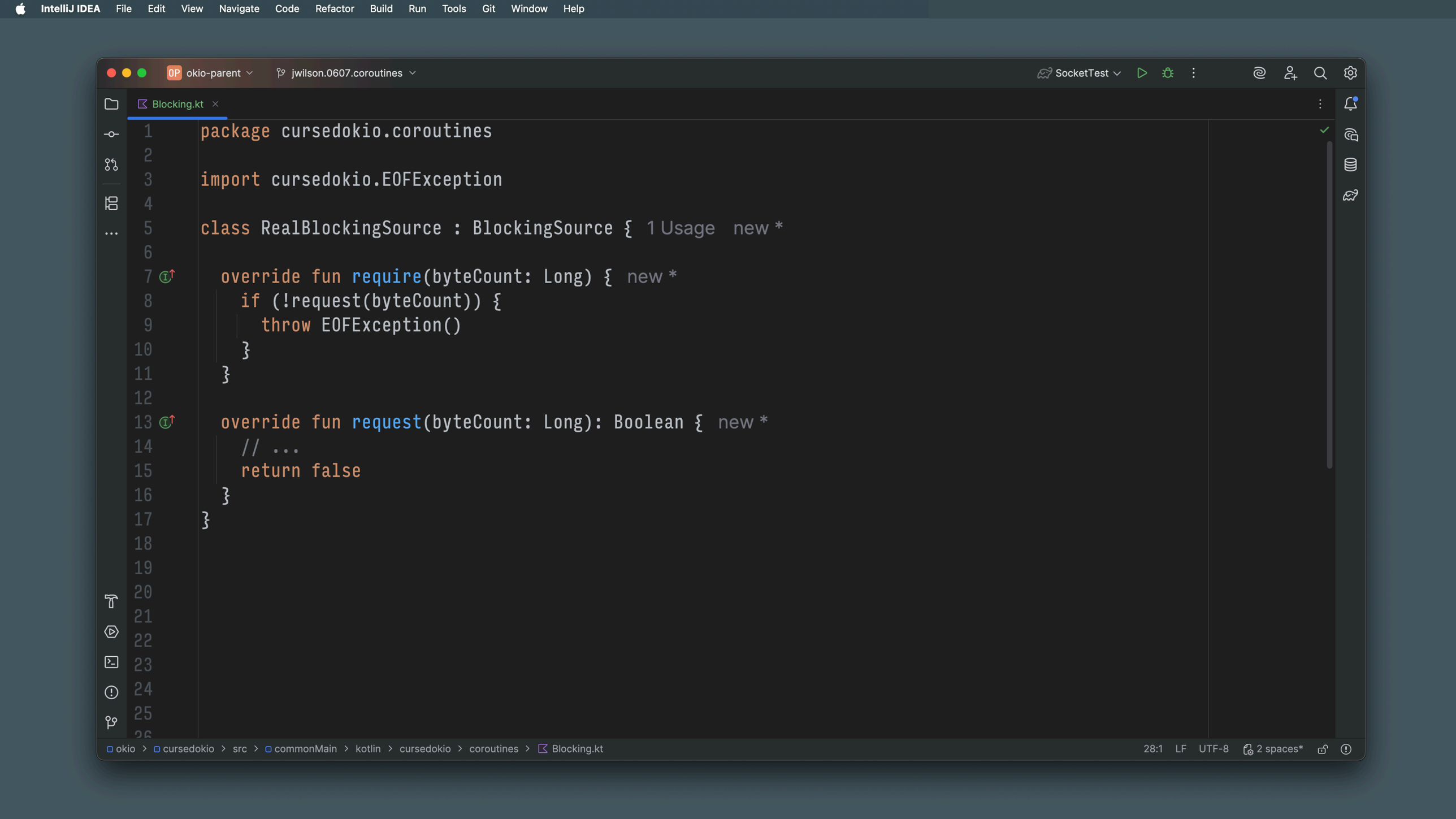
I added `suspend` about 500 times

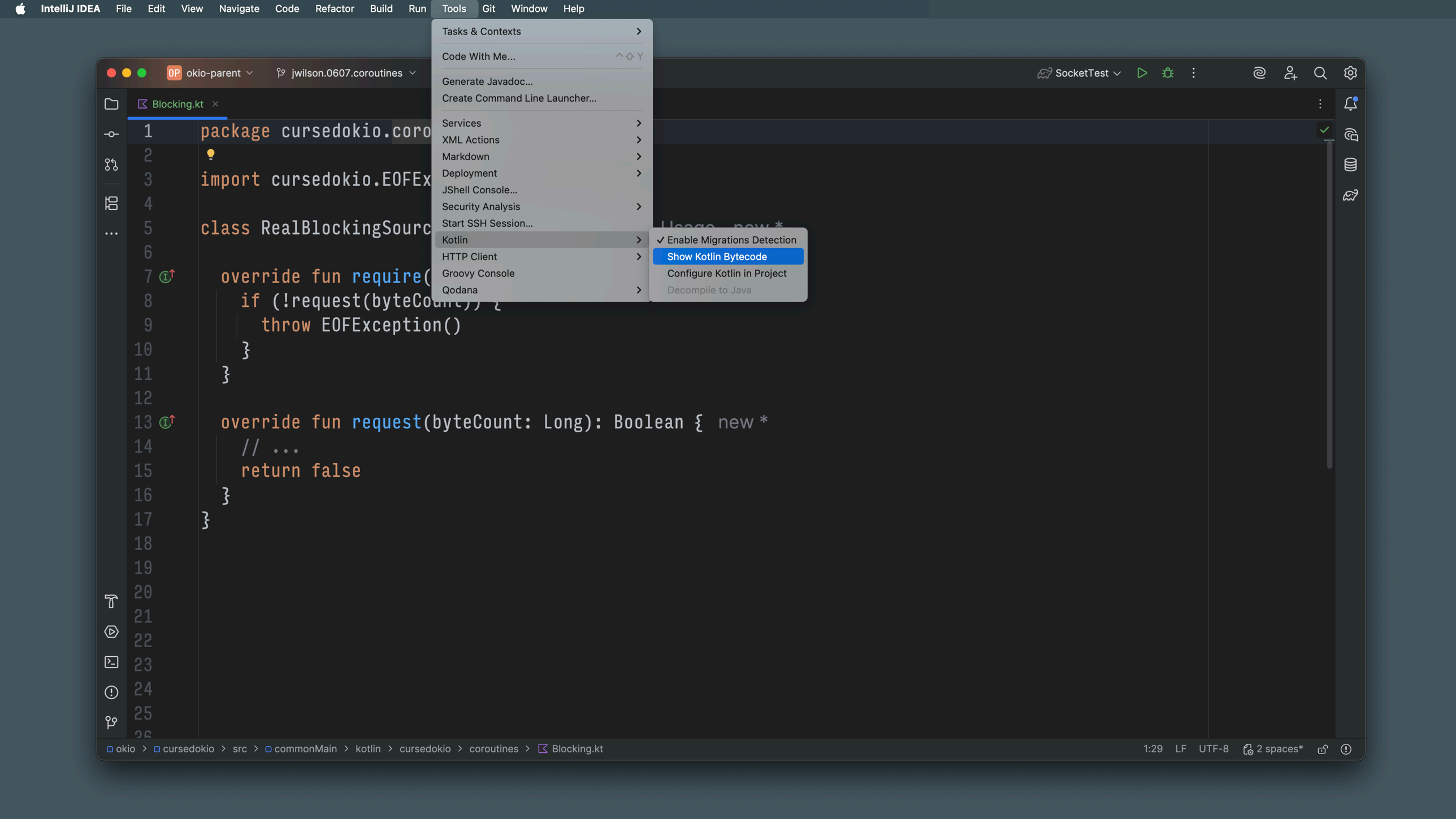
Parsing JSON got 73% slower

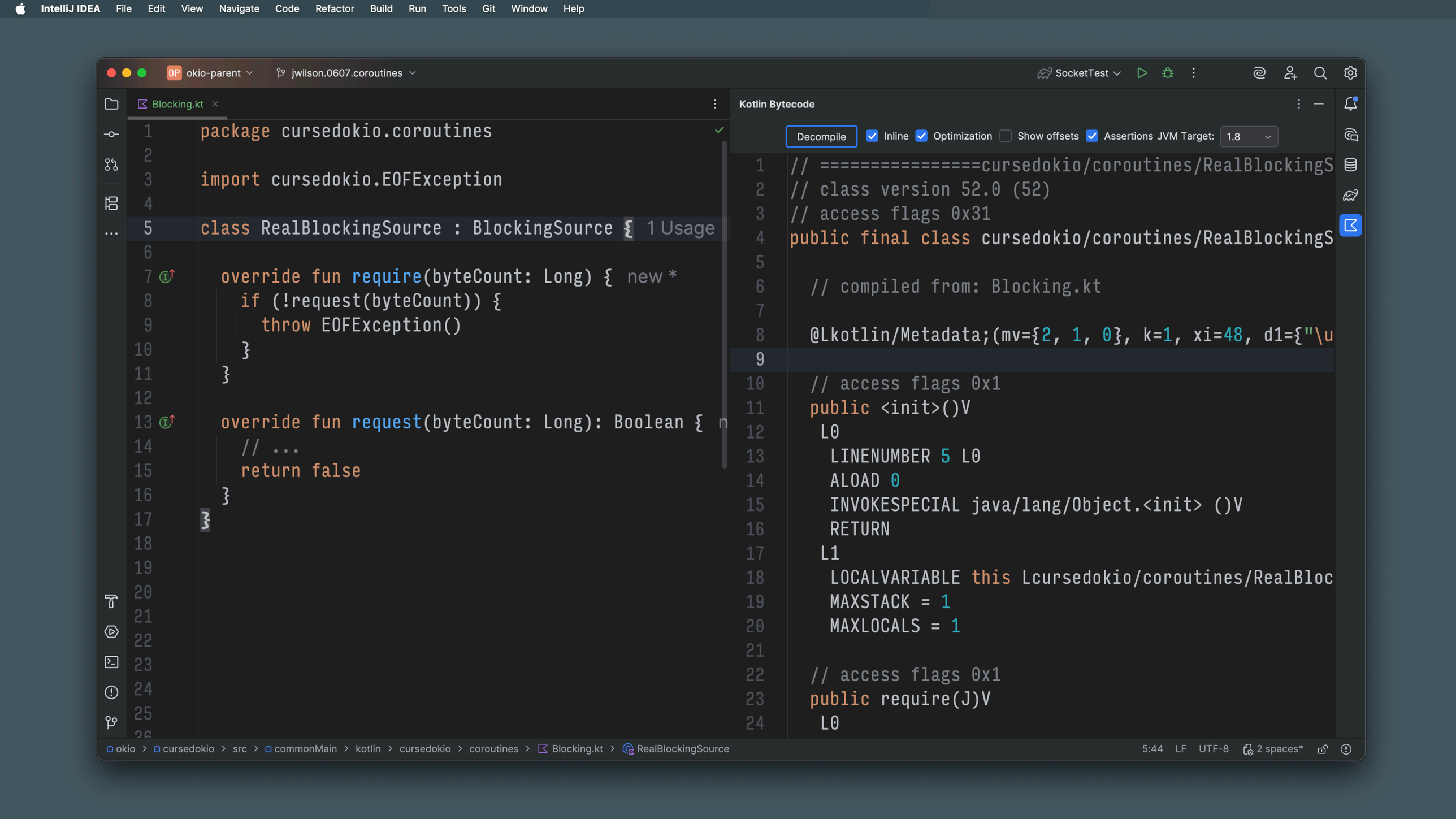
Why?!

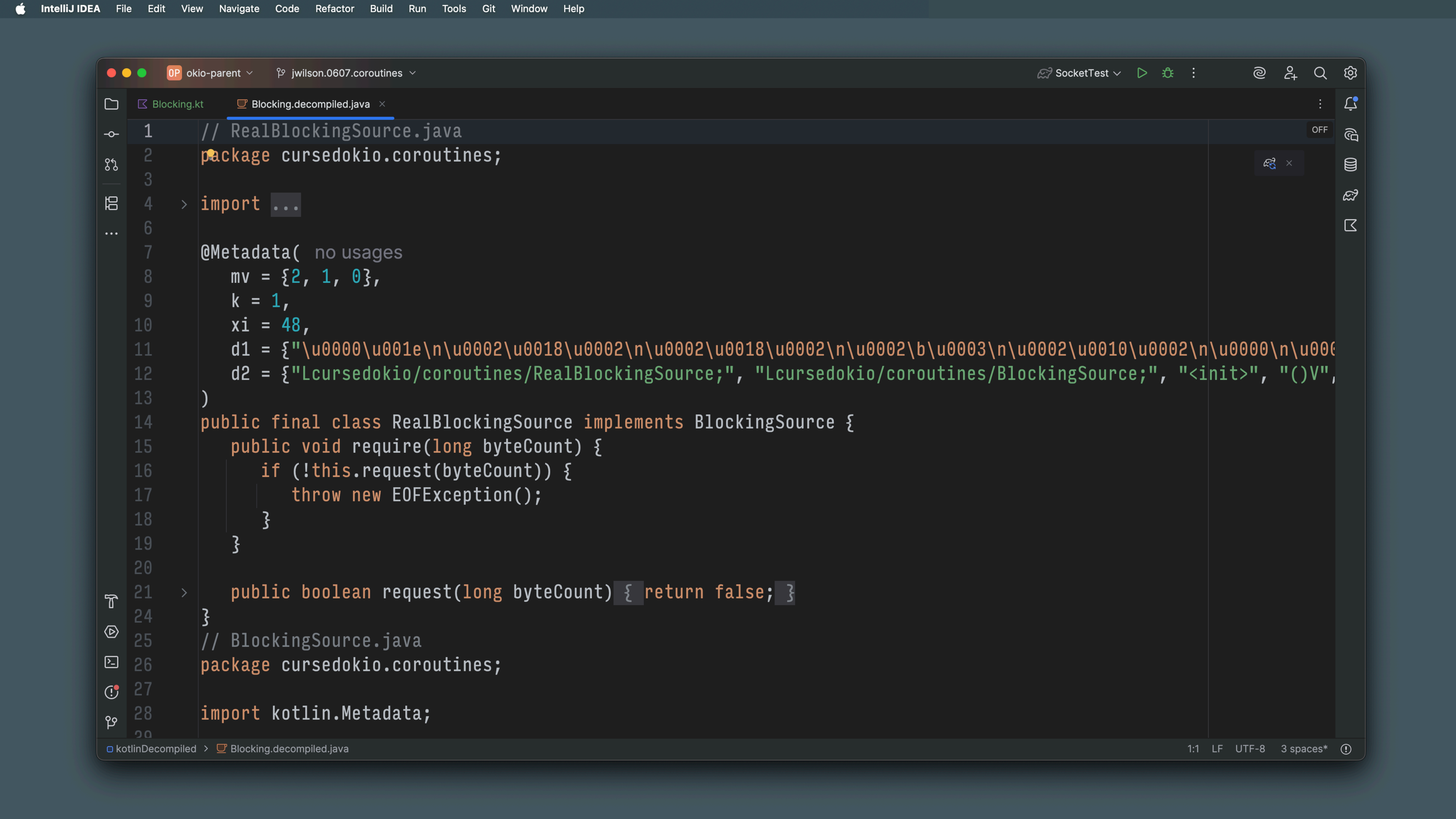
Let’s look at the bytecode...

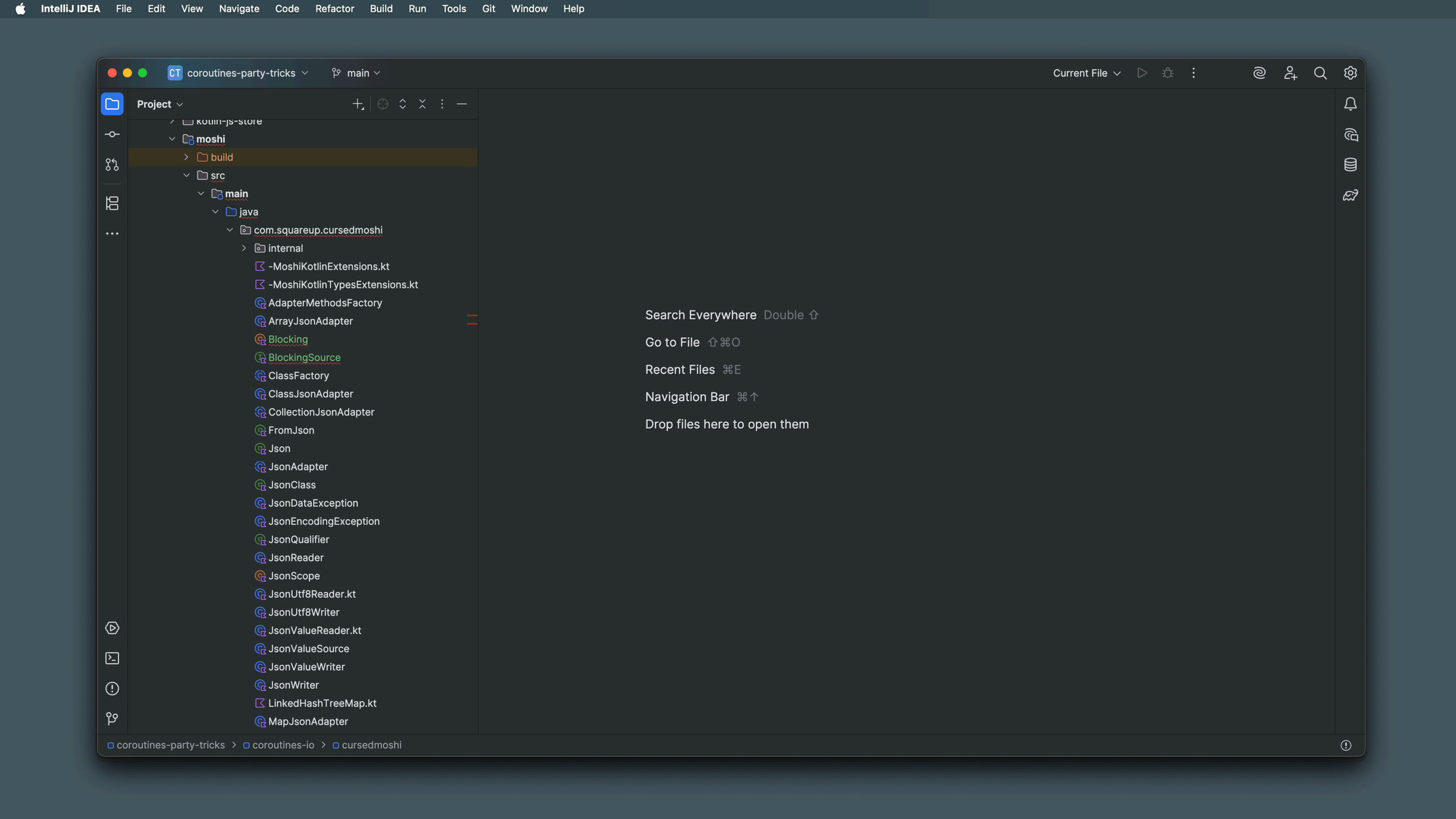
```
class RealBufferedSource : BufferedSource {

  override fun require(byteCount: Long) {
    if (!request(byteCount)) {
      throw EOFException()
    }
  }

  override fun request(byteCount: Long): Boolean {
    ...
  }


  ...
}
```

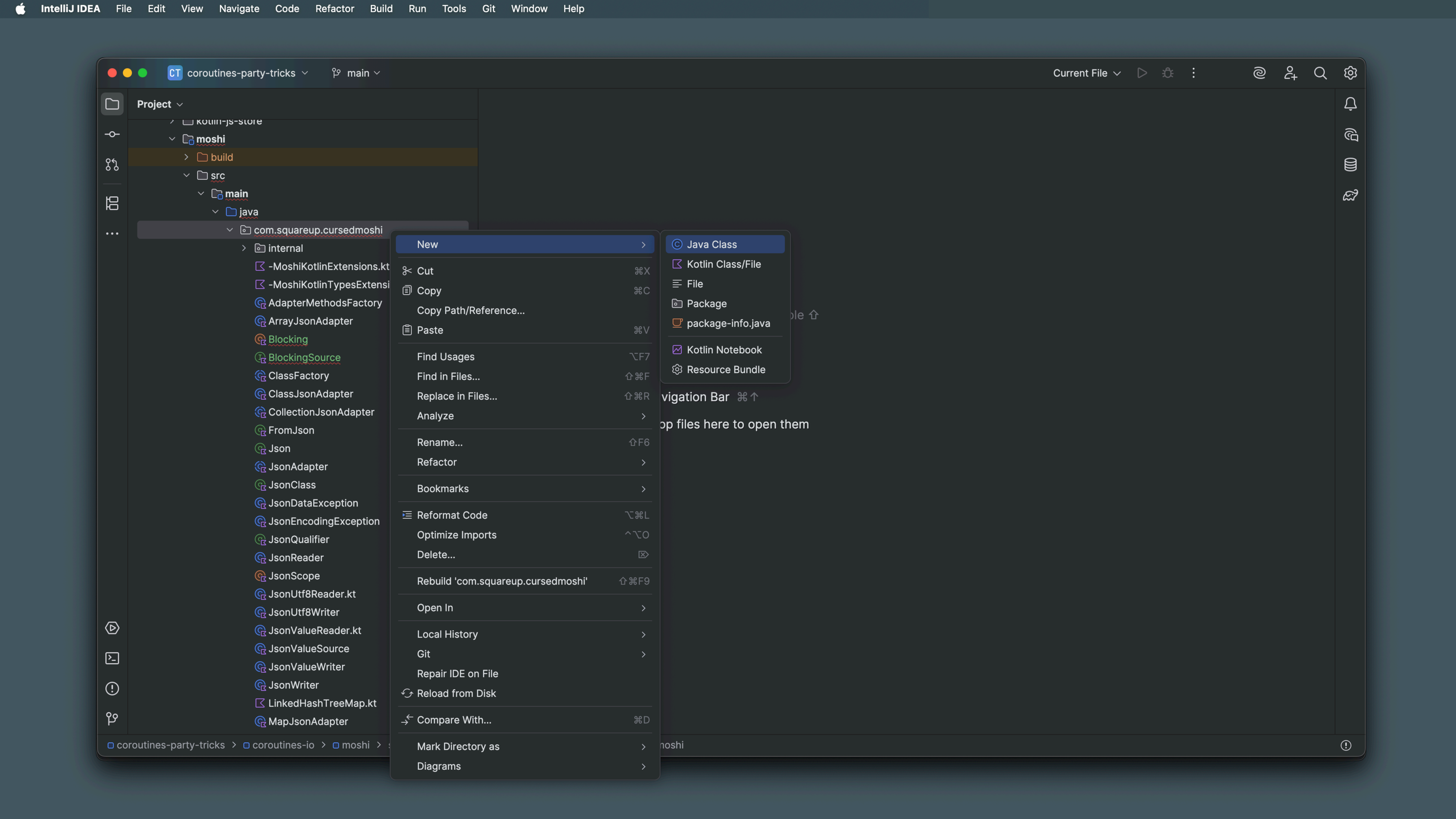

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
okio-parent
jwilson.0607.coroutines
SocketTest
Blocking.kt
package cursedokio.coroutines
import cursedokio.EOFException
class RealBlockingSource : BlockingSource { 1 Usage new *
override fun require(byteCount: Long) { new *
if (!request(byteCount)) {
throw EOFException()
}
}
override fun request(byteCount: Long): Boolean { new *
// ...
return false
}
}
okio > cursedokio > src > commonMain > kotlin > cursedokio > coroutines > Blocking.kt
28:1 LF UTF-8 2 spaces*

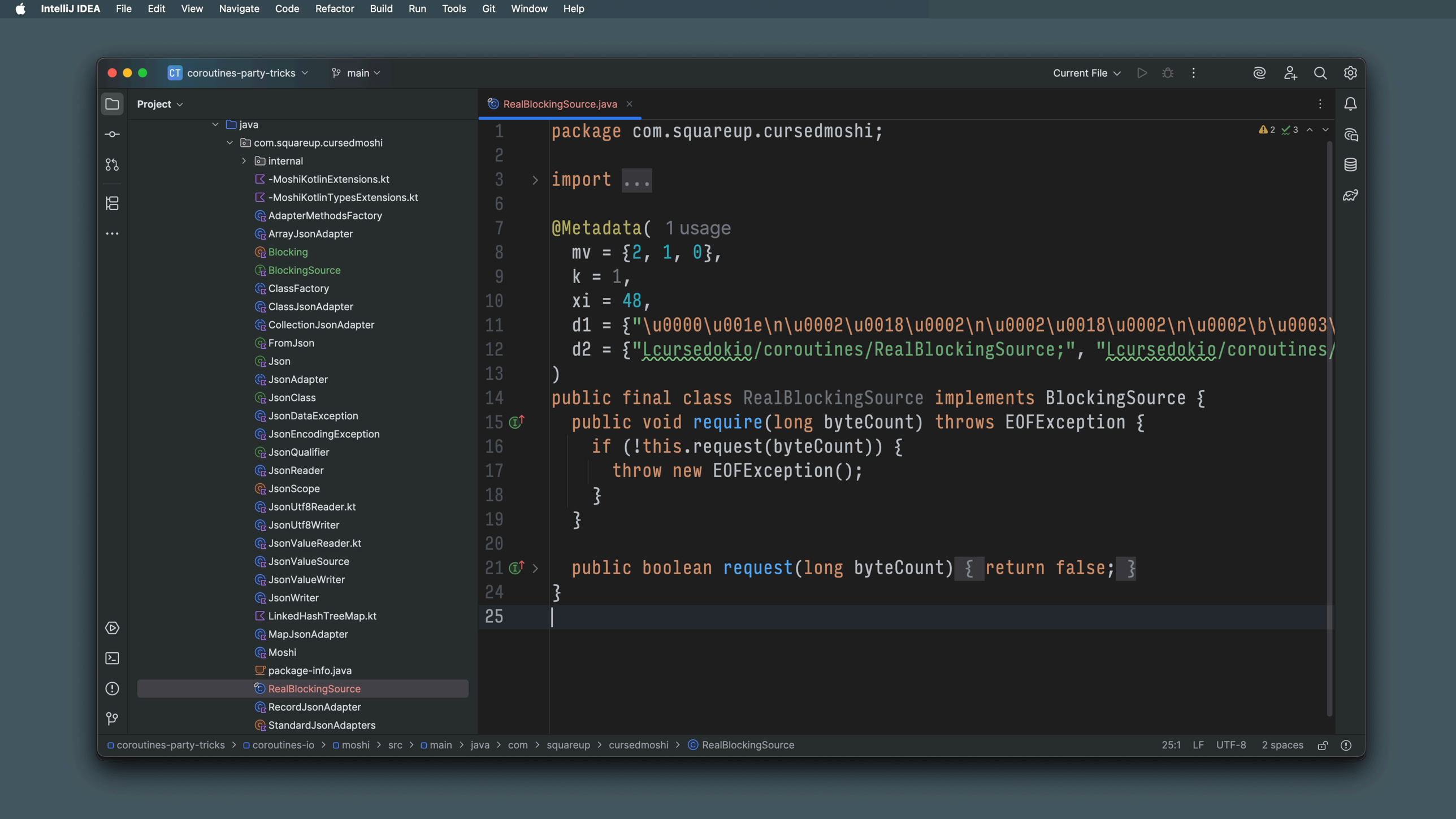

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Tasks & Contexts
Code With Me...
Generate Javadoc...
Create Command Line Launcher...
Services
XML Actions
Markdown
Deployment
JShell Console...
Security Analysis
Start SSH Session...
Kotlin
HTTP Client
Groovy Console
Qodana
Enable Migrations Detection
Show Kotlin Bytecode
Configure Kotlin in Project
Decompile to Java
OP okio-parent
jwilson.0607.coroutines
SocketTest
Blocking.kt
package cursedokio.coro
import cursedokio.EOFEx
class RealBlockingSourc
override fun require(
if (!request(byteCount)) {
throw EOFException()
}
}
override fun request(byteCount: Long): Boolean { new *
// ...
return false
}
}
okio > cursedokio > src > commonMain > kotlin > cursedokio > coroutines > Blocking.kt
1:29 LF UTF-8 2 spaces*

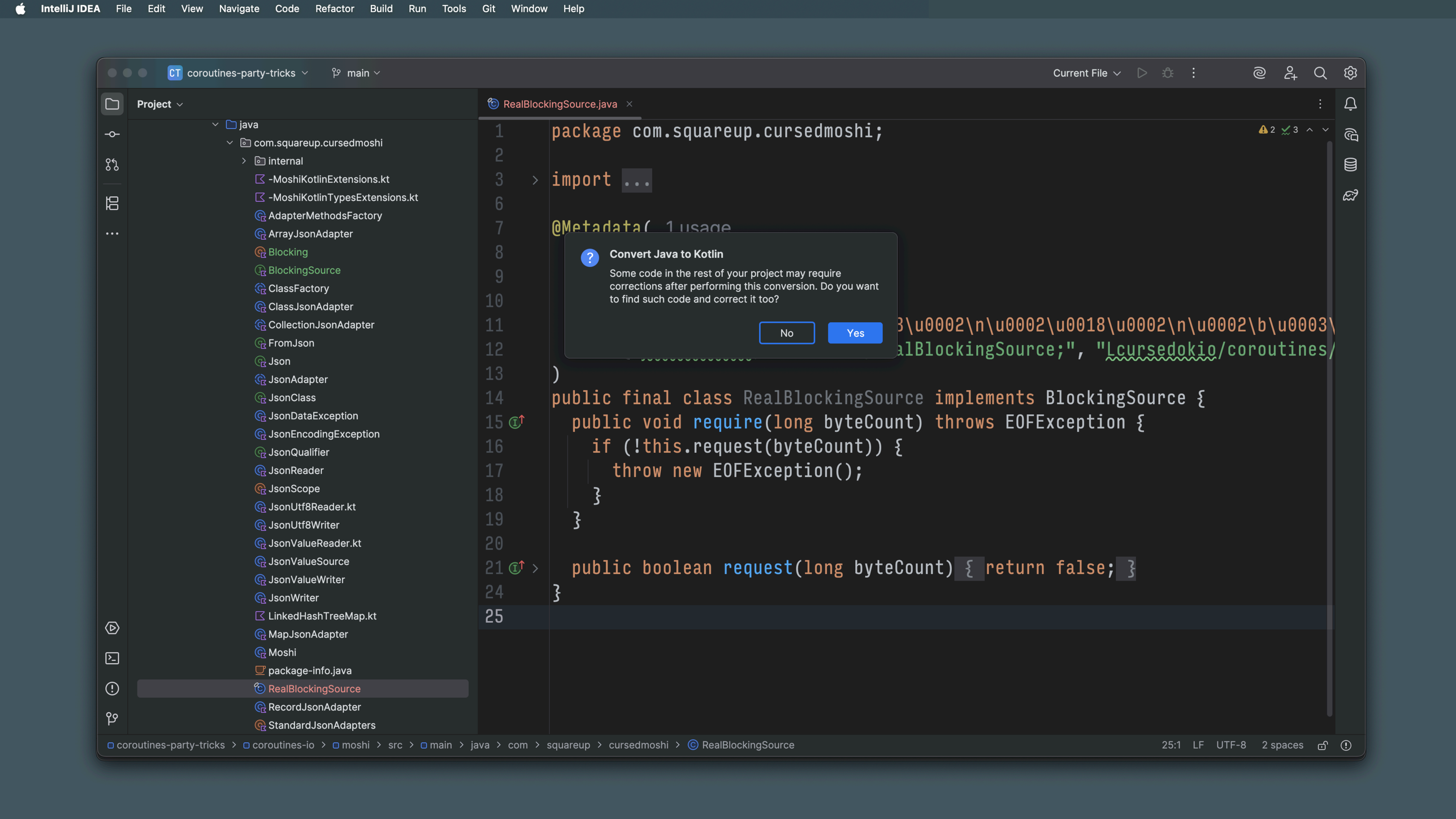

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
okio-parent
jwilson.0607.coroutines
SocketTest
Blocking.kt
package cursedokio.coroutines
import cursedokio.EOFException
class RealBlockingSource : BlockingSource { 1 Usage
override fun require(byteCount: Long) { new *
if (!request(byteCount)) {
throw EOFException()
}
}
override fun request(byteCount: Long): Boolean { n
// ...
return false
}
}
Kotlin Bytecode
Decompile
Inline
Optimization
Show offsets
Assertions
JVM Target:
1.8
// ================cursedokio/coroutines/RealBlockingS
// class version 52.0 (52)
// access flags 0x31
public final class cursedokio/coroutines/RealBlockingS
// compiled from: Blocking.kt
@Lkotlin/Metadata;(mv={2, 1, 0}, k=1, xi=48, d1={"\u
// access flags 0x1
public <init>()V
L0
LINENUMBER 5 L0
ALOAD 0
INVOKESPECIAL java/lang/Object.<init> ()V
RETURN
L1
LOCALVARIABLE this Lcursedokio/coroutines/RealBloc
MAXSTACK = 1
MAXLOCALS = 1
// access flags 0x1
public require(J)V
L0
okio > cursedokio > src > commonMain > kotlin > cursedokio > coroutines > Blocking.kt > RealBlockingSource
5:44 LF UTF-8 2 spaces*

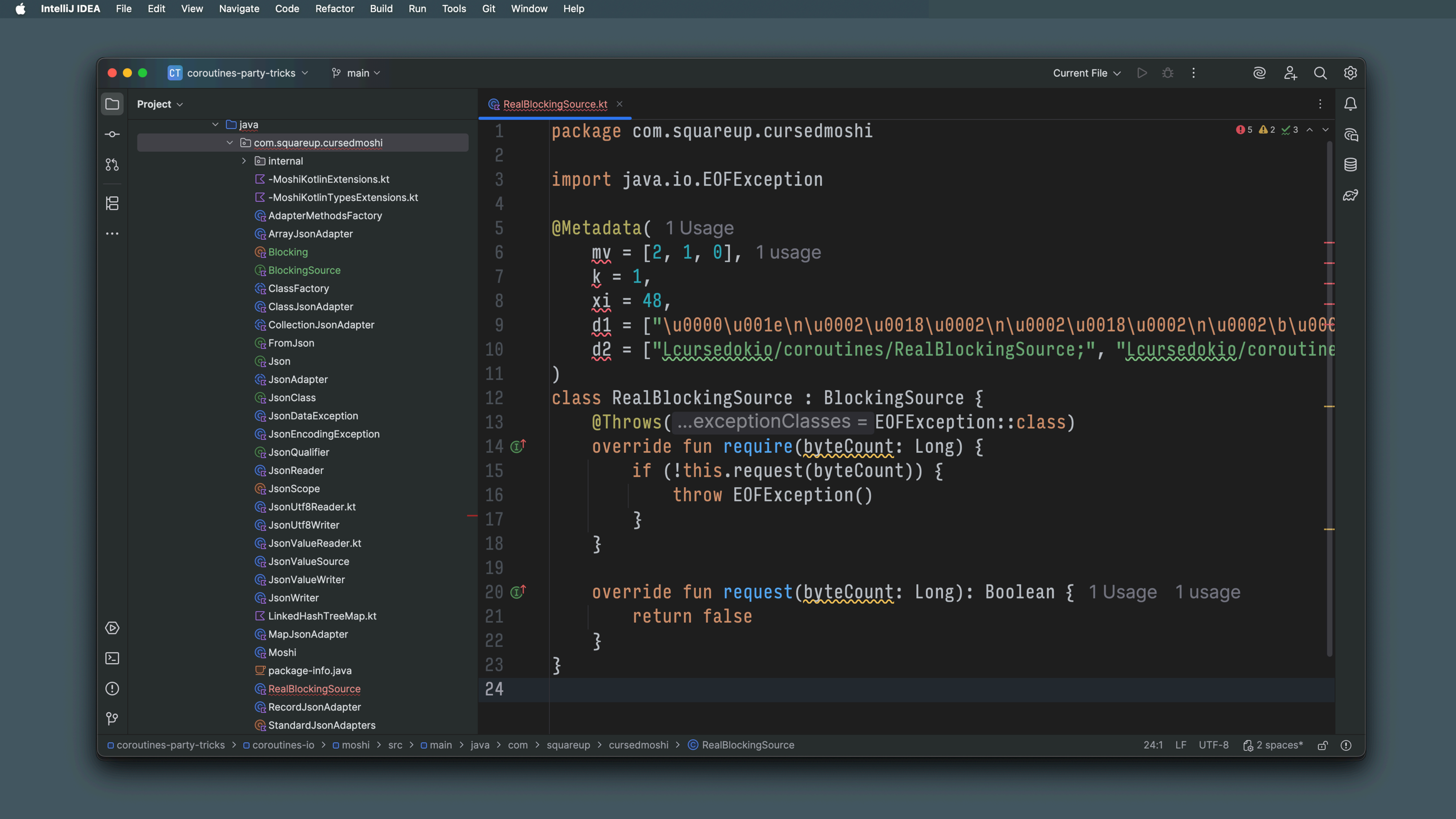

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
okio-parent
jwilson.0607.coroutines
SocketTest
Blocking.kt
Blocking.decompiled.java
OFF
// RealBlockingSource.java
package cursedokio.coroutines;
import ...
@Metadata(no usages
mv = {2, 1, 0},
k = 1,
xi = 48,
d1 = {"\u0000\u001e\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\b\u0003\n\u0002\u0010\u0002\n\u0000\n\u000
d2 = {"Lcursedokio/coroutines/RealBlockingSource;", "Lcursedokio/coroutines/BlockingSource;", "<init>", "()V",
)
public final class RealBlockingSource implements BlockingSource {
public void require(long byteCount) {
if (!this.request(byteCount)) {
throw new EOFException();
}
}
public boolean request(long byteCount) { return false; }
}
// BlockingSource.java
package cursedokio.coroutines;
import kotlin.Metadata;
kotlinDecompiled
Blocking.decompiled.java
1:1 LF UTF-8 3 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
moshi
build
src
main
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Search Everywhere Double ⇧
Go to File ⇧⌘O
Recent Files ⌘E
Navigation Bar ⌘↑
Drop files here to open them
coroutines-party-tricks
coroutines-io
cursedmoshi


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
moshi
build
src
main
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensi
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
New
Cut ⌘X
Copy ⌘C
Copy Path/Reference...
Paste ⌘V
Find Usages ⌥F7
Find in Files... ⇧⌘F
Replace in Files... ⇧⌘R
Analyze
Rename... ⇧F6
Refactor
Bookmarks
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Delete...
Rebuild 'com.squareup.cursedmoshi' ⇧⌘F9
Open In
Local History
Git
Repair IDE on File
Reload from Disk
Compare With... ⌘D
Mark Directory as
Diagrams
Java Class
Kotlin Class/File
File
Package
package-info.java
Kotlin Notebook
Resource Bundle
vigation Bar ⌘↑
op files here to open them
coroutines-party-tricks
coroutines-io
moshi


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource
RecordJsonAdapter
StandardJsonAdapters
RealBlockingSource.java
package com.squareup.cursedmoshi;
import ...
@Metadata(1 usage
mv = {2, 1, 0},
k = 1,
xi = 48,
d1 = {"\u0000\u001e\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\b\u0003\
d2 = {"Lcursedokio/coroutines/RealBlockingSource;", "Lcursedokio/coroutines/
)
public final class RealBlockingSource implements BlockingSource {
public void require(long byteCount) throws EOFException {
if (!this.request(byteCount)) {
throw new EOFException();
}
}
public boolean request(long byteCount) { return false; }
}
coroutines-party-tricks > coroutines-io > moshi > src > main > java > com > squareup > cursedmoshi > RealBlockingSource
25:1 LF UTF-8 2 spaces


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource
RecordJsonAdapter
StandardJsonAdapters
RealBlockingSource.java
package com.squareup.cursedmoshi;
import ...
@Metadata(1 usage
Convert Java to Kotlin
Some code in the rest of your project may require corrections after performing this conversion. Do you want to find such code and correct it too?
No
Yes
alBlockingSource;", "Lcursedokio/coroutines/
)
public final class RealBlockingSource implements BlockingSource {
public void require(long byteCount) throws EOFException {
if (!this.request(byteCount)) {
throw new EOFException();
}
}
public boolean request(long byteCount) { return false; }
}
coroutines-party-tricks › coroutines-io › moshi › src › main › java › com › squareup › cursedmoshi › RealBlockingSource
25:1 LF UTF-8 2 spaces


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource
RecordJsonAdapter
StandardJsonAdapters
RealBlockingSource.kt
package com.squareup.cursedmoshi
import java.io.EOFException
@Metadata(1 Usage
mv = [2, 1, 0], 1 usage
k = 1,
xi = 48,
d1 = ["\u0000\u001e\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\b\u00€
d2 = ["Lcursedokio/coroutines/RealBlockingSource;", "Lcursedokio/coroutine
)
class RealBlockingSource : BlockingSource {
@Throws(...exceptionClasses = EOFException::class)
override fun require(byteCount: Long) {
if (!this.request(byteCount)) {
throw EOFException()
}
}
override fun request(byteCount: Long): Boolean { 1 Usage 1 usage
return false
}
}
5 2 3
coroutines-party-tricks > coroutines-io > moshi > src > main > java > com > squareup > cursedmoshi > RealBlockingSource
24:1 LF UTF-8 2 spaces*


IntelliK IKEA
package com.squareup.cursedmoshi
import java.io.EOFException
@Metadata(1 Usage
mv = [2, 1, 0], 1 usage
throw EOFException()
override fun request(byteCount: Long): Boolean { 1 Usage 1 usage
return false
RealBlockingSource.kt
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdap
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingExcep
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource
RecordJsonAdapter
StandardJsonAdapters
coroutines-party-tricks
main
Current File
coroutines-party-tricks › coroutines-io › moshi › src › main › java › com › squareup › cursedmoshi › RealBlockingSource
20:1 (167 chars, 7 line breaks) LF UTF-8 2 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdap
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingExcep
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource
RecordJsonAdapter
StandardJsonAdapters
RealBlockingSource.kt
package com.squareup.cursedmoshi
import java.io.EOFException
@Metadata(1 Usage
mv = [2, 1, 0], 1 usage
Keynote
throw EOFException()
override fun request(byteCount: Long): Boolean { 1 Usage 1 usage
return false
coroutines-party-tricks > coroutines-io > moshi > src > main > java > com > squareup > cursedmoshi > RealBlockingSource
20:1 (167 chars, 7 line breaks) LF UTF-8 2 spaces*

Keynote File Edit Insert Slide Format Arrange View Play Window Help

Coroutines Party Tricks — Edited

View Zoom Add Slide Play Table Chart Text Shape Media Comment Share Format Animate Document

1 Coroutines Party Tricks

2

23

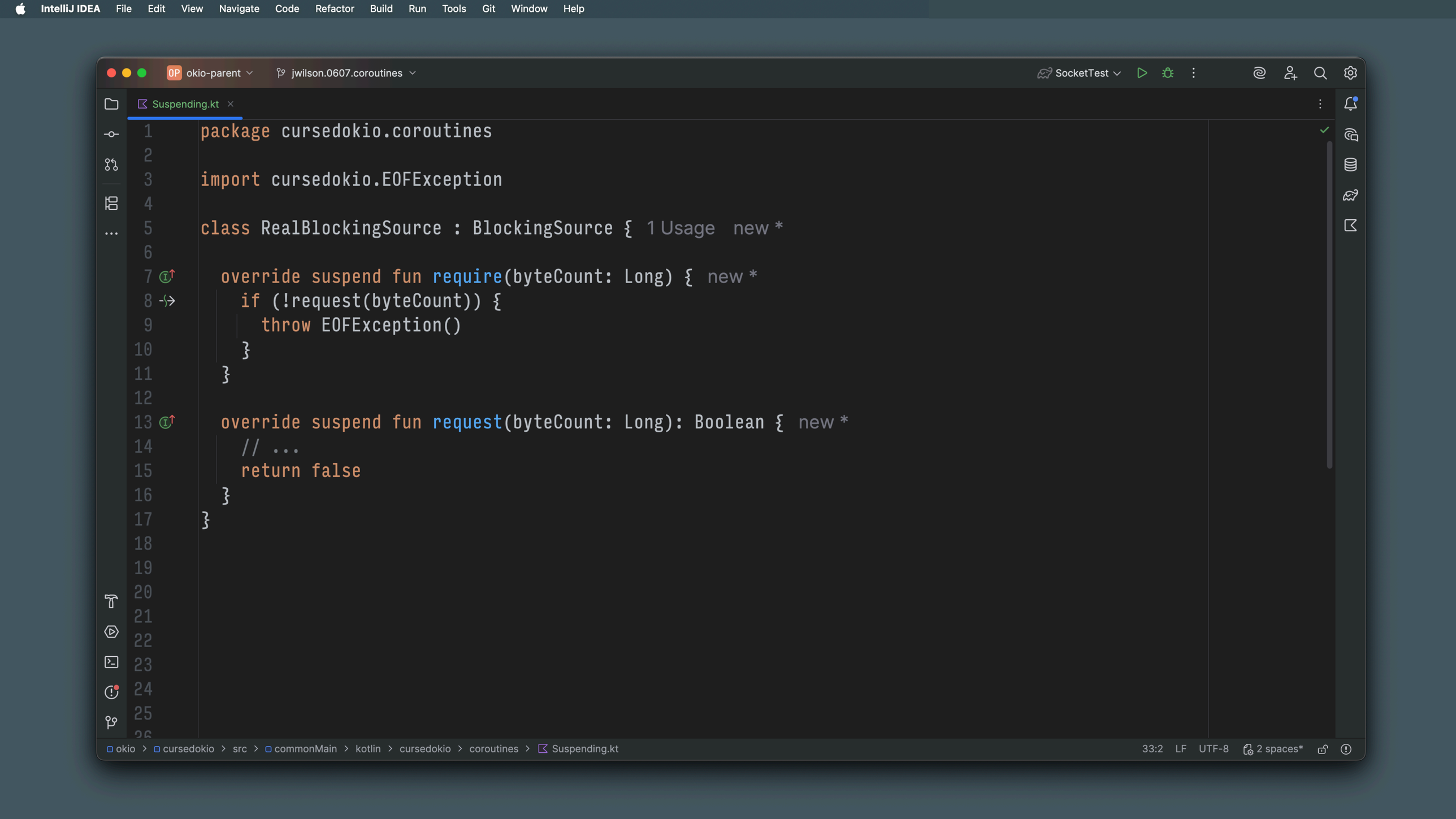
24

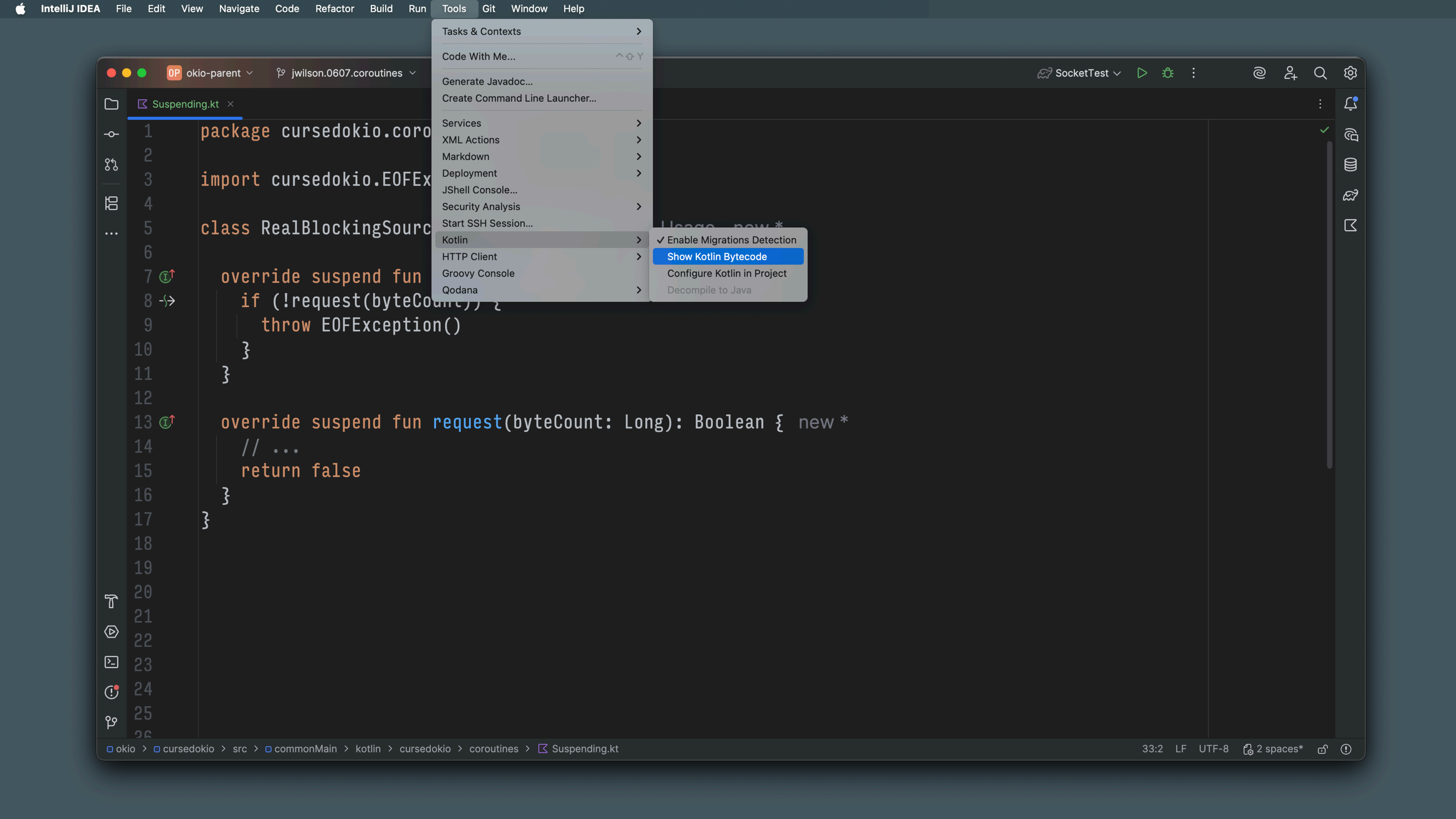
25 Thanks

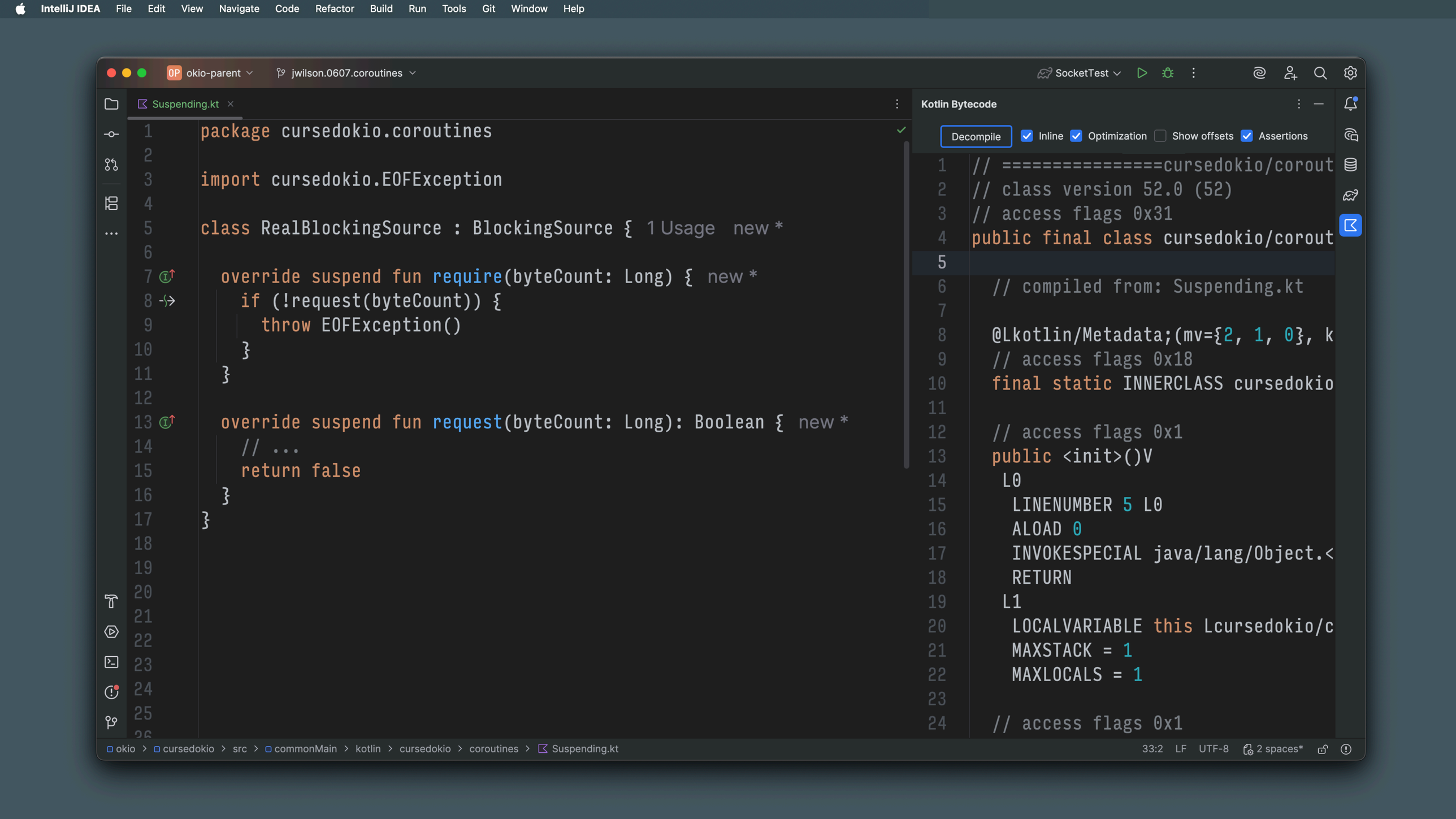
```
class RealBufferedSource : BufferedSource {

  override fun require(byteCount: Long) {
    if (!request(byteCount)) {
      throw EOFException()
    }
  }

  override fun request(byteCount: Long): Boolean {
    ...
  }

  ...
}
```

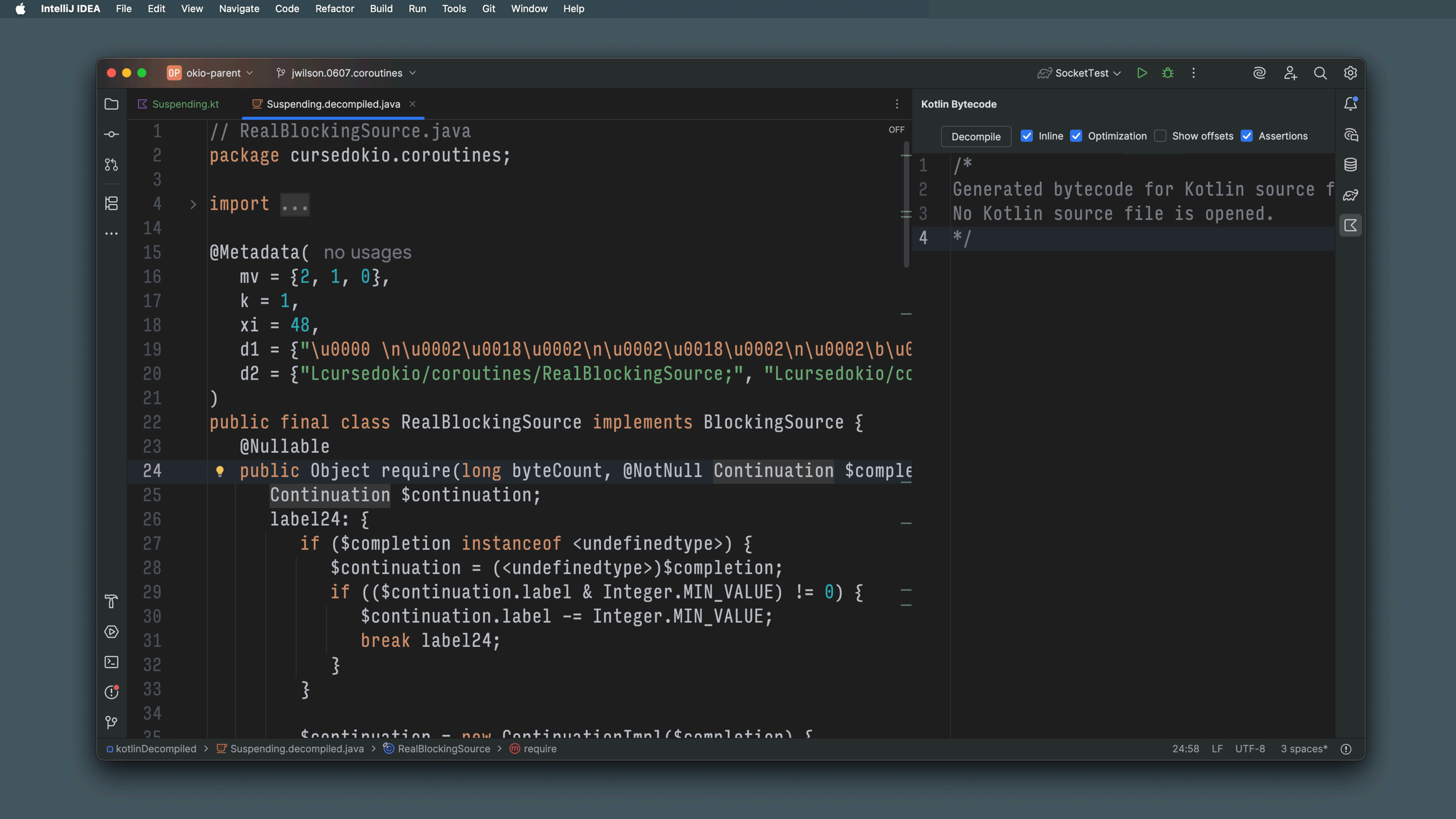
```
class RealBufferedSource : BufferedSource {

  override fun require(byteCount: Long) {
    if (!request(byteCount)) {
      throw EOFException()
    }
  }

  override fun request(byteCount: Long): Boolean {
    ...
  }


  ...
}
```

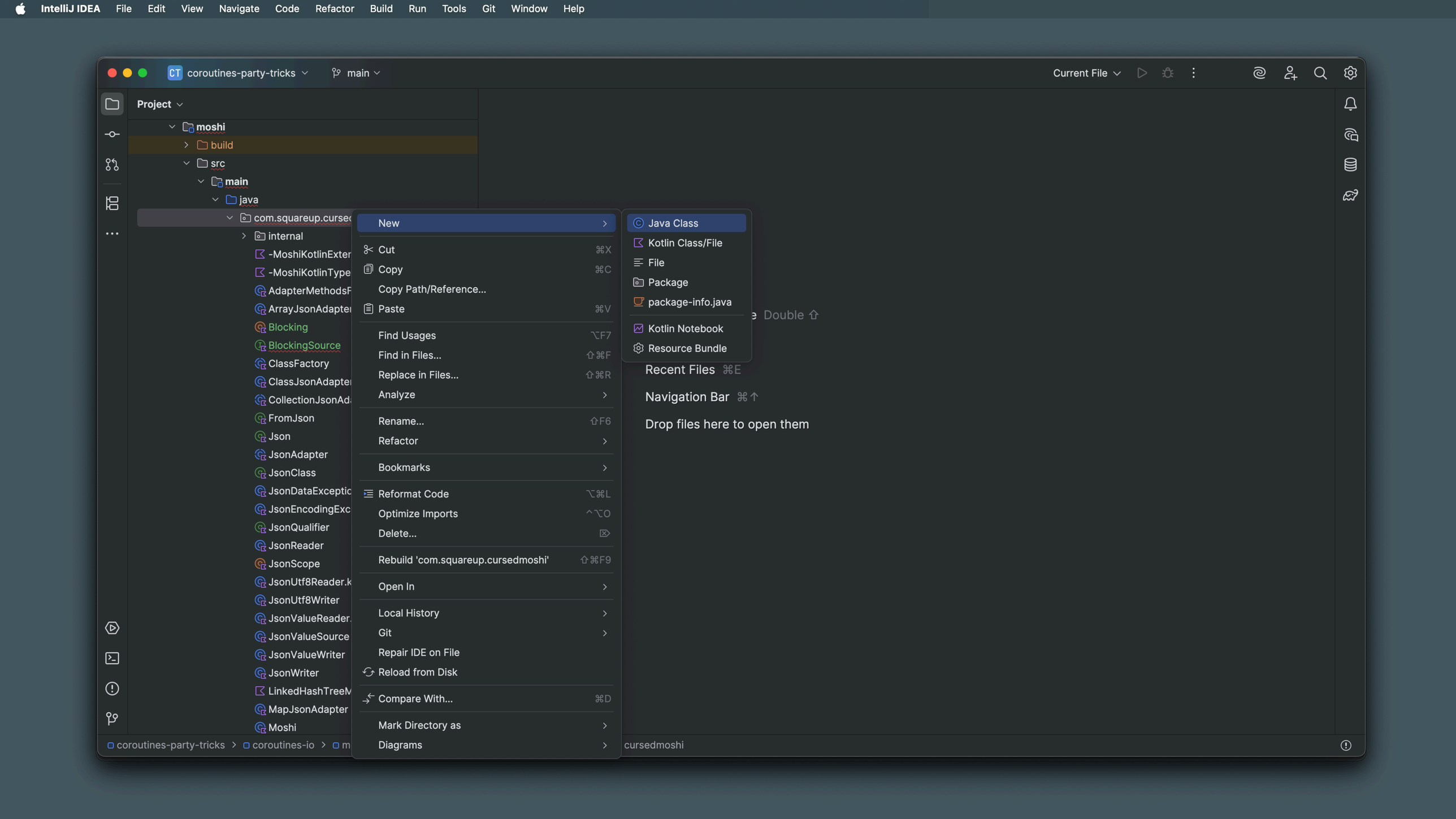
```
class RealBufferedSource : BufferedSource {

  override suspend fun require(byteCount: Long) {
    if (!request(byteCount)) {
      throw EOFException()
    }
  }

  override suspend fun request(byteCount: Long): Boolean {
    ...
  }


  ...
}
```

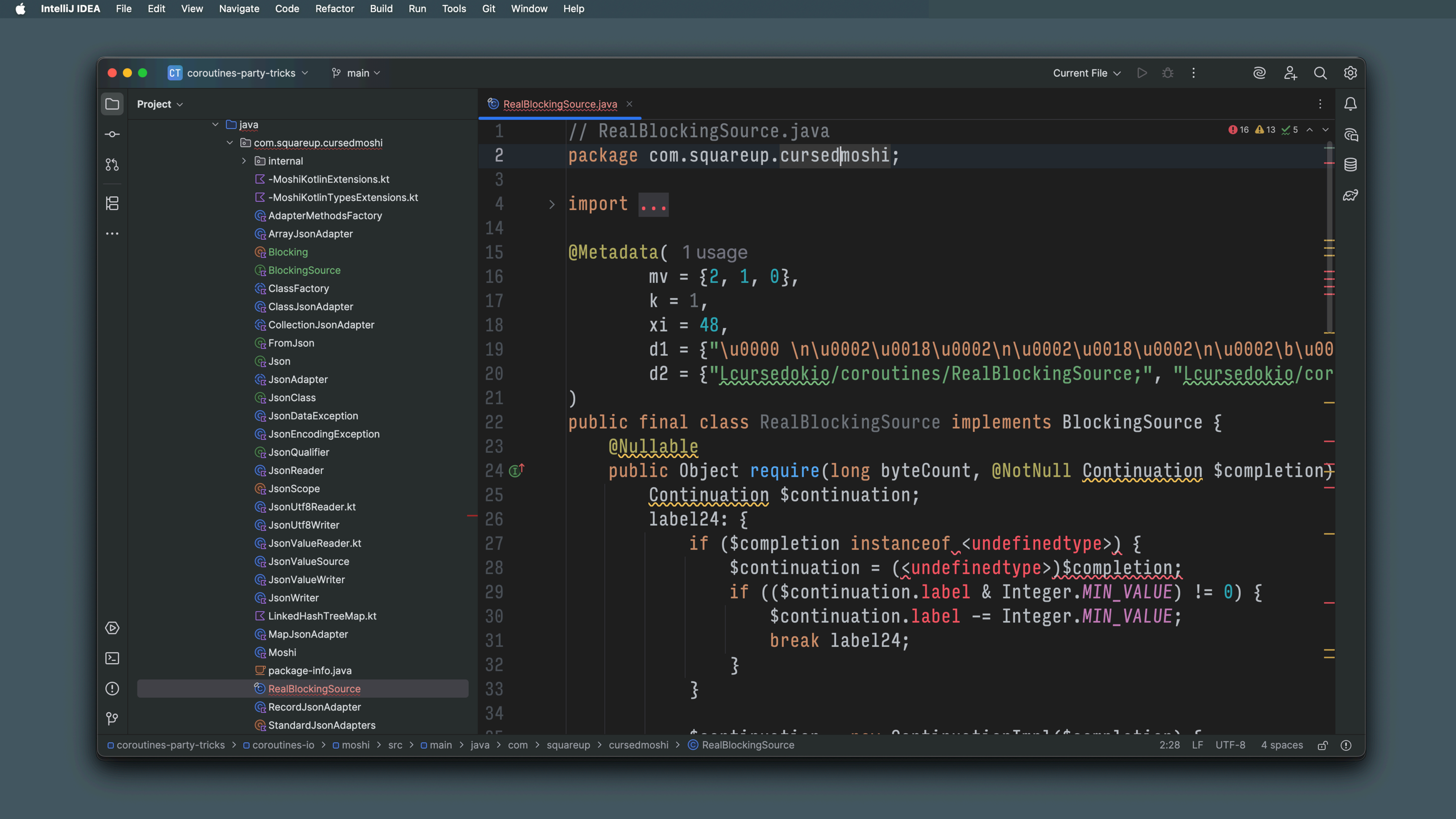

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
okio-parent
jwilson.0607.coroutines
SocketTest
Suspending.kt
package cursedokio.coroutines
import cursedokio.EOFException
class RealBlockingSource : BlockingSource { 1 Usage new *
override suspend fun require(byteCount: Long) { new *
if (!request(byteCount)) {
throw EOFException()
}
}
override suspend fun request(byteCount: Long): Boolean { new *
// ...
return false
}
}
okio > cursedokio > src > commonMain > kotlin > cursedokio > coroutines > Suspending.kt
33:2 LF UTF-8 2 spaces*

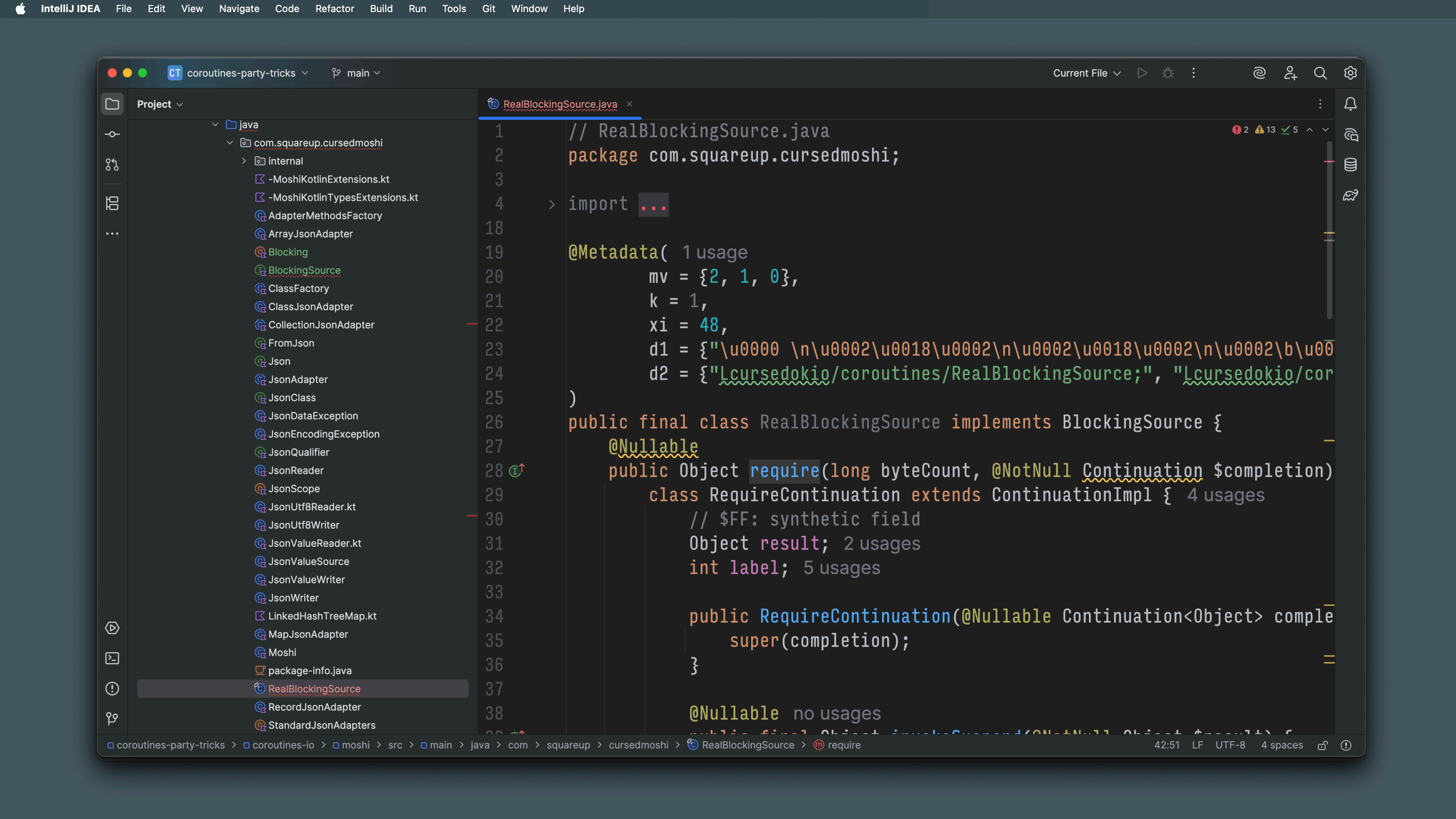

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Tasks & Contexts
Code With Me...
Generate Javadoc...
Create Command Line Launcher...
Services
XML Actions
Markdown
Deployment
JShell Console...
Security Analysis
Start SSH Session...
Kotlin
HTTP Client
Groovy Console
Qodana
Enable Migrations Detection
Show Kotlin Bytecode
Configure Kotlin in Project
Decompile to Java
okio-parent
jwilson.0607.coroutines
SocketTest
Suspending.kt
package cursedokio.coro
import cursedokio.EOFEx
class RealBlockingSourc
override suspend fun
if (!request(byteCount)) {
throw EOFException()
}
}
override suspend fun request(byteCount: Long): Boolean { new *
// ...
return false
}
}
okio › cursedokio › src › commonMain › kotlin › cursedokio › coroutines › Suspending.kt
33:2 LF UTF-8 2 spaces*

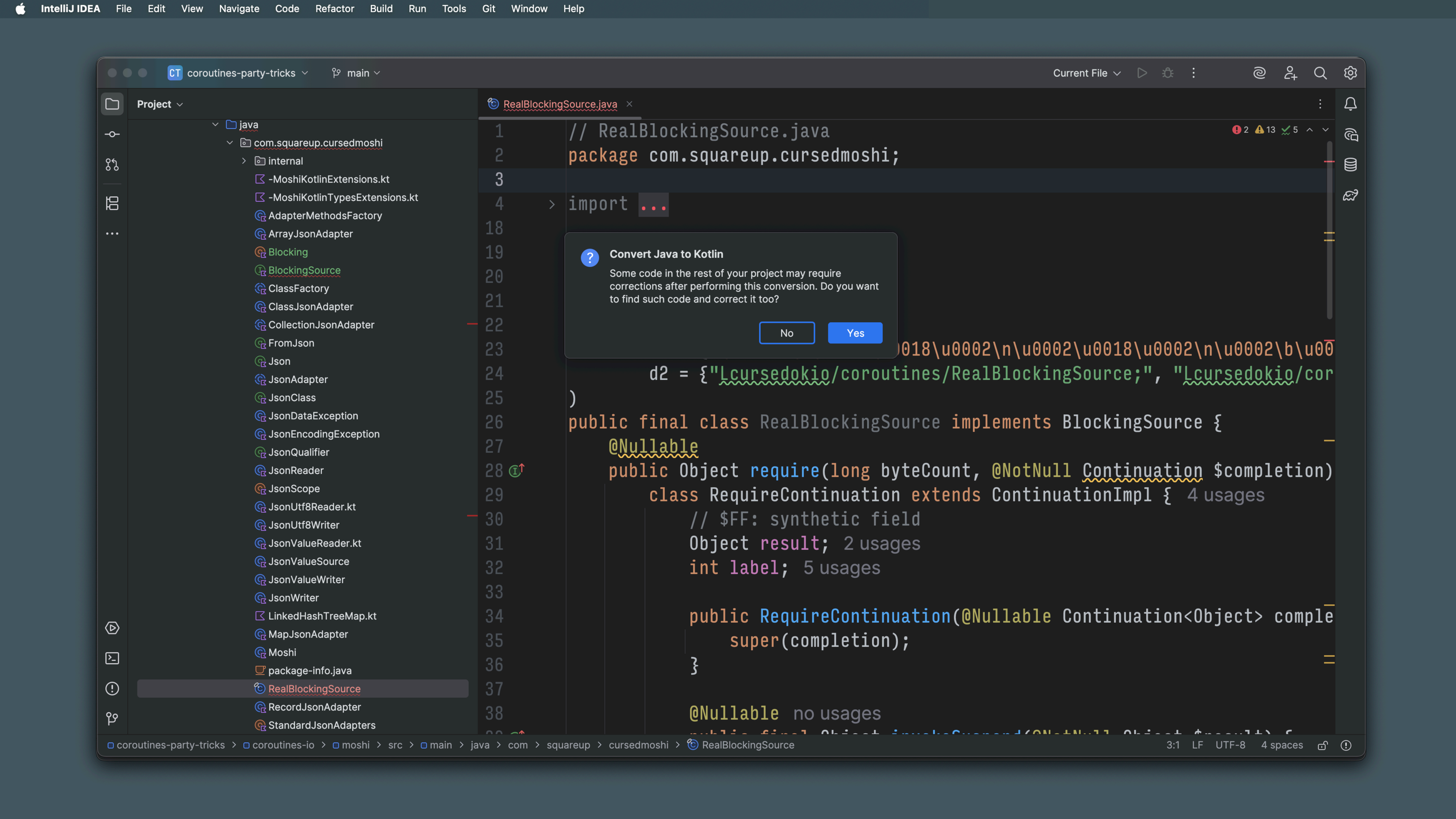

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
okio-parent
jwilson.0607.coroutines
SocketTest
Suspending.kt
package cursedokio.coroutines
import cursedokio.EOFException
class RealBlockingSource : BlockingSource { 1 Usage new *
override suspend fun require(byteCount: Long) { new *
if (!request(byteCount)) {
throw EOFException()
}
}
override suspend fun request(byteCount: Long): Boolean { new *
// ...
return false
}
}
Kotlin Bytecode
Decompile
Inline
Optimization
Show offsets
Assertions
// ================cursedokio/corout
// class version 52.0 (52)
// access flags 0x31
public final class cursedokio/corout
// compiled from: Suspending.kt
@Lkotlin/Metadata;(mv={2, 1, 0}, k
// access flags 0x18
final static INNERCLASS cursedokio
// access flags 0x1
public <init>()V
L0
LINENUMBER 5 L0
ALOAD 0
INVOKESPECIAL java/lang/Object.<
RETURN
L1
LOCALVARIABLE this Lcursedokio/c
MAXSTACK = 1
MAXLOCALS = 1
// access flags 0x1
okio > cursedokio > src > commonMain > kotlin > cursedokio > coroutines > Suspending.kt
33:2 LF UTF-8 2 spaces*

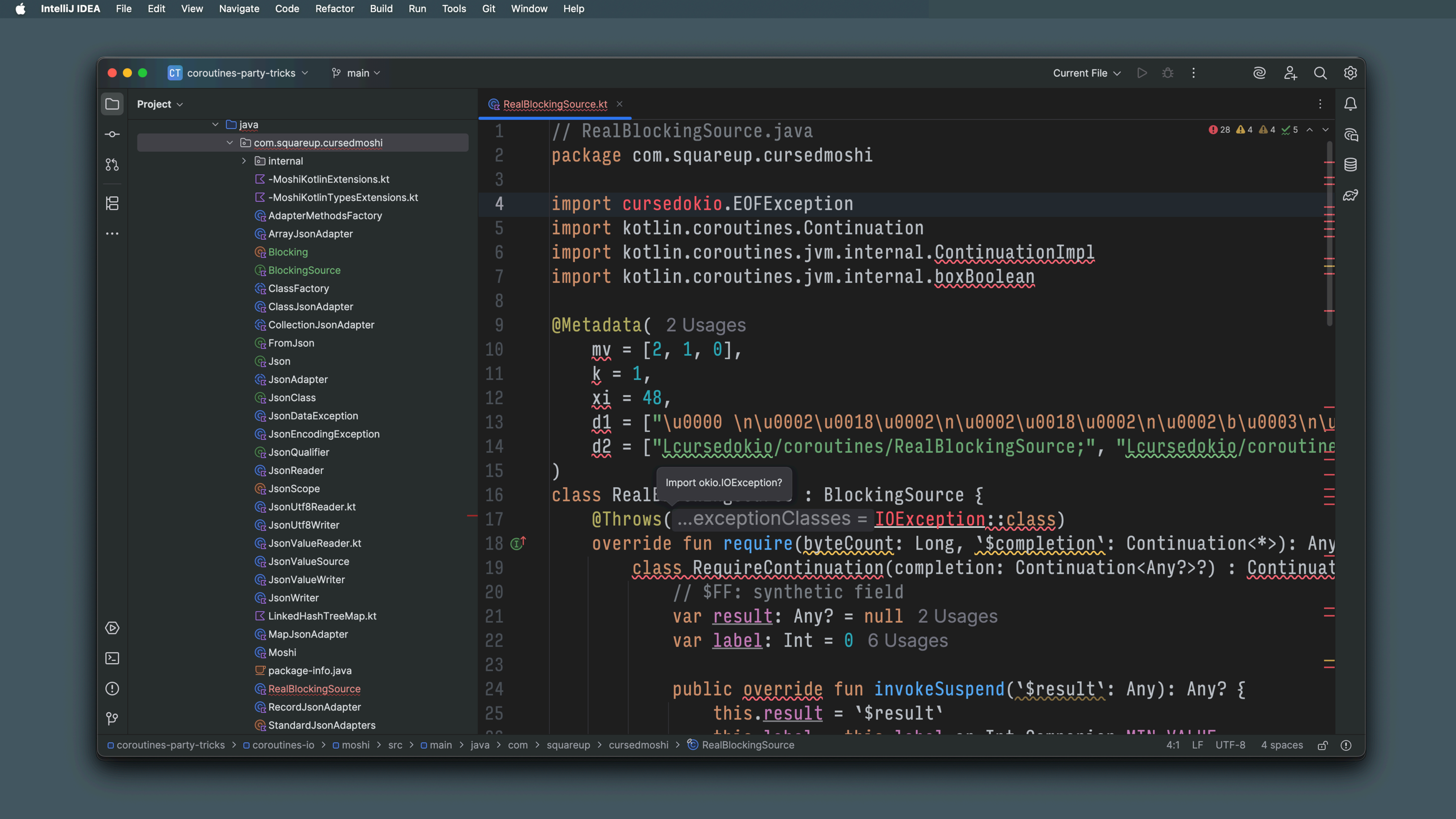

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
okio-parent
jwilson.0607.coroutines
SocketTest
Suspending.kt
Suspending.decompiled.java
OFF
// RealBlockingSource.java
package cursedokio.coroutines;
import ...
@Metadata(no usages
mv = {2, 1, 0},
k = 1,
xi = 48,
d1 = {"\u0000 \n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\b\u0
d2 = {"Lcursedokio/coroutines/RealBlockingSource;", "Lcursedokio/cc
)
public final class RealBlockingSource implements BlockingSource {
@Nullable
public Object require(long byteCount, @NotNull Continuation $comple
Continuation $continuation;
label24: {
if ($completion instanceof <undefinedtype>) {
$continuation = (<undefinedtype>)$completion;
if (($continuation.label & Integer.MIN_VALUE) != 0) {
$continuation.label -= Integer.MIN_VALUE;
break label24;
}
}
Kotlin Bytecode
Decompile
Inline
Optimization
Show offsets
Assertions
/*
Generated bytecode for Kotlin source f
No Kotlin source file is opened.
*/
kotlinDecompiled
Suspending.decompiled.java
RealBlockingSource
require
24:58
LF
UTF-8
3 spaces*

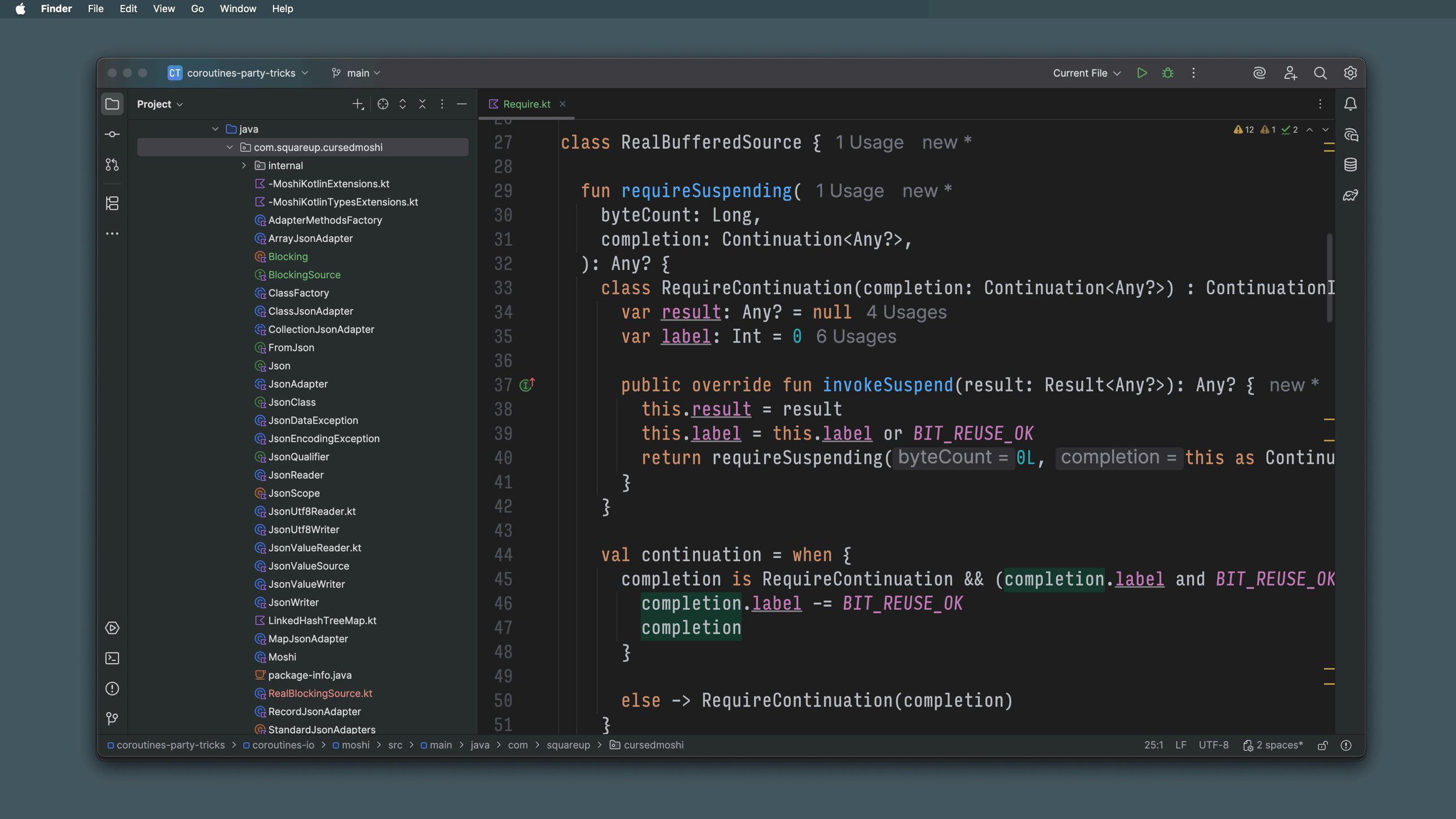

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
moshi
build
src
main
java
com.squareup.cursed
internal
-MoshiKotlinExter
-MoshiKotlinType
AdapterMethodsF
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAd
FromJson
Json
JsonAdapter
JsonClass
JsonDataExceptic
JsonEncodingExc
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.k
JsonUtf8Writer
JsonValueReader
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeM
MapJsonAdapter
Moshi
New
Cut ⌘X
Copy ⌘C
Copy Path/Reference...
Paste ⌘V
Find Usages ⌥F7
Find in Files... ⇧⌘F
Replace in Files... ⇧⌘R
Analyze
Rename... ⇧F6
Refactor
Bookmarks
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Delete...
Rebuild 'com.squareup.cursedmoshi' ⇧⌘F9
Open In
Local History
Git
Repair IDE on File
Reload from Disk
Compare With... ⌘D
Mark Directory as
Diagrams
Java Class
Kotlin Class/File
File
Package
package-info.java
Kotlin Notebook
Resource Bundle
Double ⇧
Recent Files ⌘E
Navigation Bar ⌘↑
Drop files here to open them
coroutines-party-tricks
coroutines-io
cursedmoshi

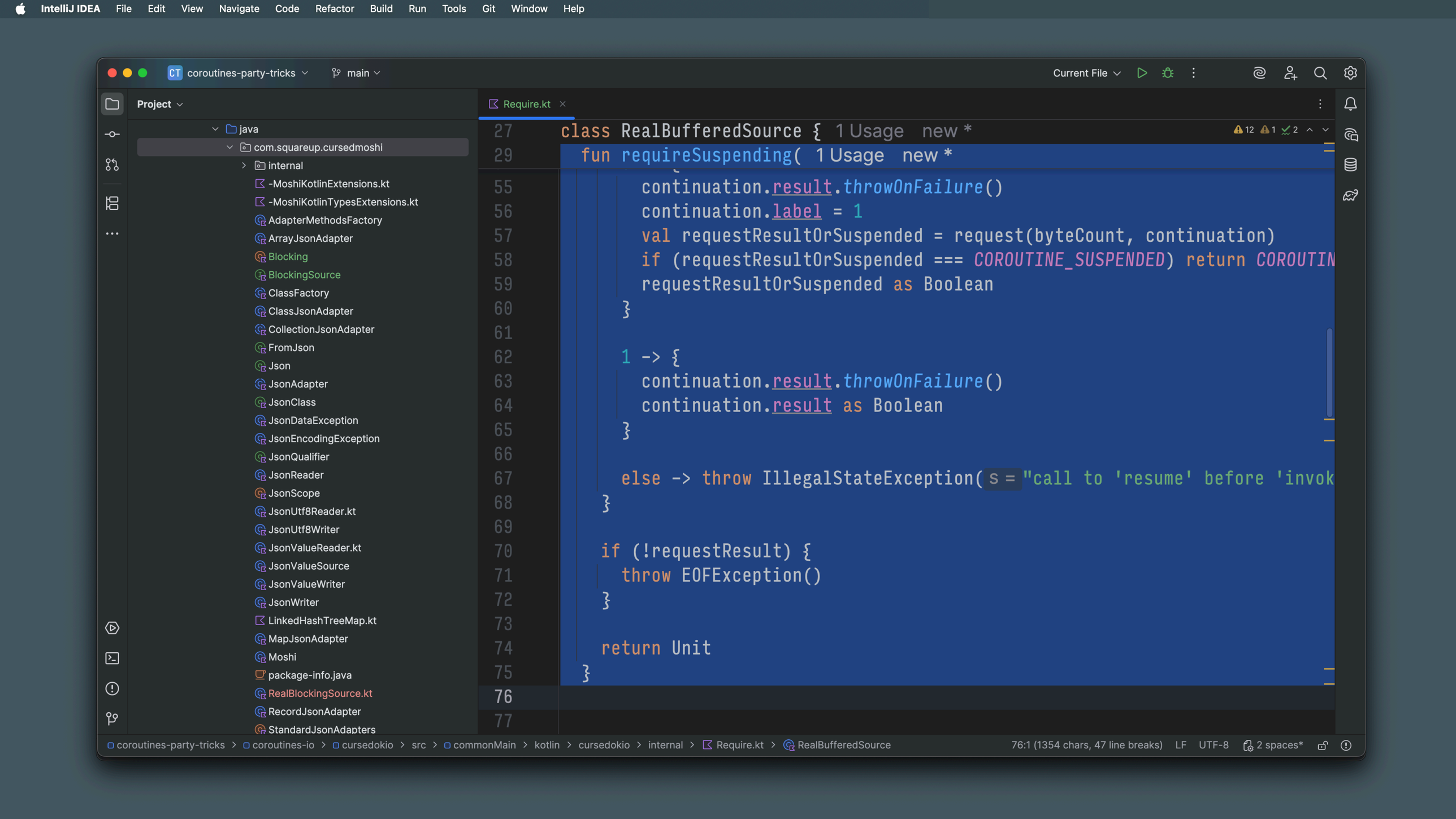

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource
RecordJsonAdapter
StandardJsonAdapters
RealBlockingSource.java
// RealBlockingSource.java
package com.squareup.cursedmoshi;
import ...
@Metadata(1 usage
mv = {2, 1, 0},
k = 1,
xi = 48,
d1 = {"\u0000 \n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\b\u00
d2 = {"Lcursedokio/coroutines/RealBlockingSource;", "Lcursedokio/cor
)
public final class RealBlockingSource implements BlockingSource {
@Nullable
public Object require(long byteCount, @NotNull Continuation $completion)
Continuation $continuation;
label24: {
if ($completion instanceof <undefinedtype>) {
$continuation = (<undefinedtype>)$completion;
if (($continuation.label & Integer.MIN_VALUE) != 0) {
$continuation.label -= Integer.MIN_VALUE;
break label24;
}
}
16 13 5
coroutines-party-tricks > coroutines-io > moshi > src > main > java > com > squareup > cursedmoshi > RealBlockingSource
2:28 LF UTF-8 4 spaces

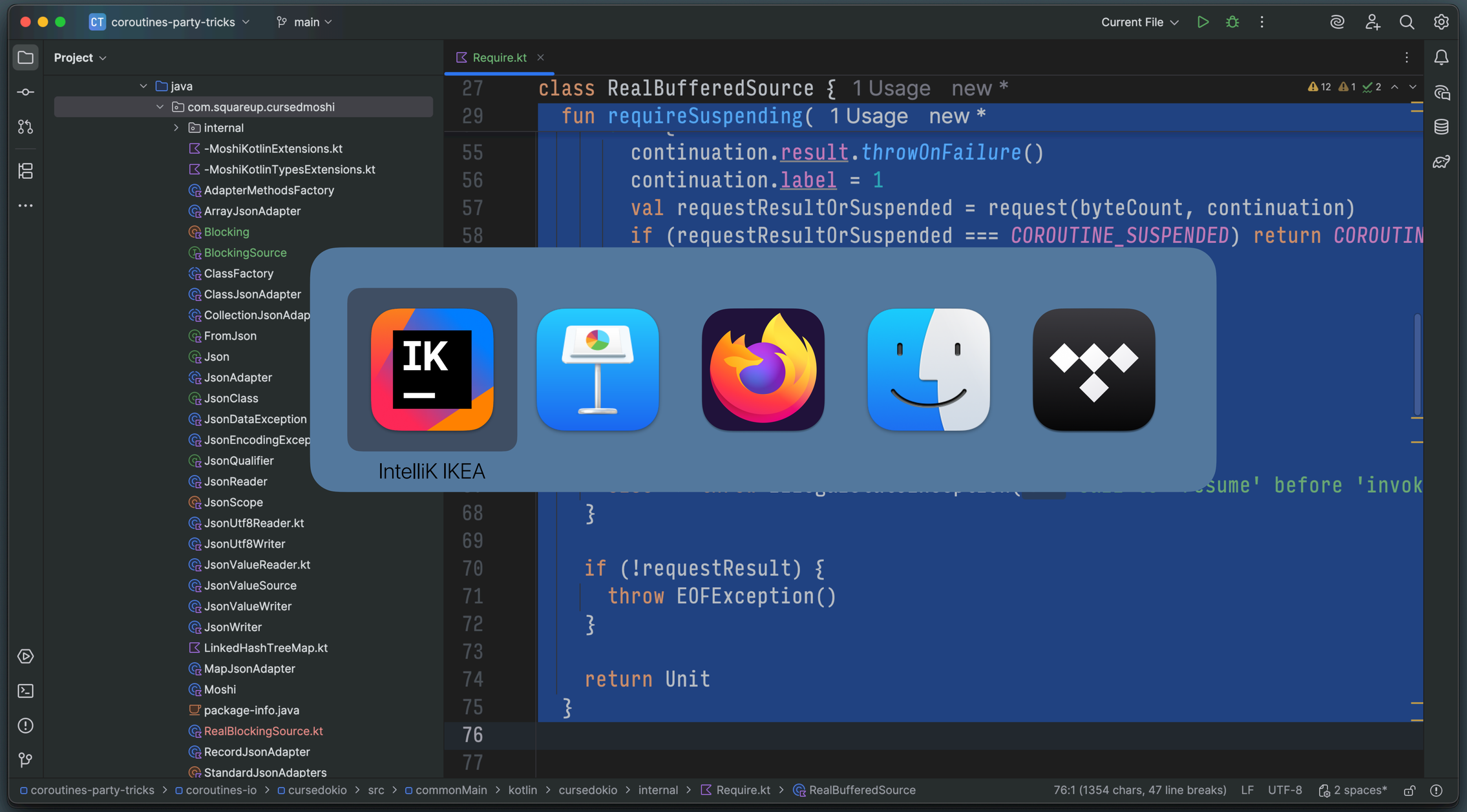

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource
RecordJsonAdapter
StandardJsonAdapters
RealBlockingSource.java
// RealBlockingSource.java
package com.squareup.cursedmoshi;
import ...
@Metadata(1 usage
mv = {2, 1, 0},
k = 1,
xi = 48,
d1 = {"\u0000 \n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\b\u00
d2 = {"Lcursedokio/coroutines/RealBlockingSource;", "Lcursedokio/cor
)
public final class RealBlockingSource implements BlockingSource {
@Nullable
public Object require(long byteCount, @NotNull Continuation $completion)
class RequireContinuation extends ContinuationImpl { 4 usages
// $FF: synthetic field
Object result; 2 usages
int label; 5 usages
public RequireContinuation(@Nullable Continuation<Object> comple
super(completion);
}
@Nullable no usages
2 13 5
coroutines-party-tricks > coroutines-io > moshi > src > main > java > com > squareup > cursedmoshi > RealBlockingSource > require
42:51 LF UTF-8 4 spaces


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource
RecordJsonAdapter
StandardJsonAdapters
RealBlockingSource.java
// RealBlockingSource.java
package com.squareup.cursedmoshi;
import ...
Convert Java to Kotlin
Some code in the rest of your project may require corrections after performing this conversion. Do you want to find such code and correct it too?
No
Yes
d2 = {"Lcursedokio/coroutines/RealBlockingSource;", "Lcursedokio/cor
)
public final class RealBlockingSource implements BlockingSource {
@Nullable
public Object require(long byteCount, @NotNull Continuation $completion)
class RequireContinuation extends ContinuationImpl { 4 usages
// $FF: synthetic field
Object result; 2 usages
int label; 5 usages
public RequireContinuation(@Nullable Continuation<Object> comple
super(completion);
}
@Nullable no usages
coroutines-party-tricks > coroutines-io > moshi > src > main > java > com > squareup > cursedmoshi > RealBlockingSource
3:1 LF UTF-8 4 spaces


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource
RecordJsonAdapter
StandardJsonAdapters
RealBlockingSource.kt
// RealBlockingSource.java
package com.squareup.cursedmoshi
import cursedokio.EOFException
import kotlin.coroutines.Continuation
import kotlin.coroutines.jvm.internal.ContinuationImpl
import kotlin.coroutines.jvm.internal.boxBoolean
@Metadata(2 Usages
mv = [2, 1, 0],
k = 1,
xi = 48,
d1 = ["\u0000 \n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\b\u0003\n\u
d2 = ["Lcursedokio/coroutines/RealBlockingSource;", "Lcursedokio/coroutine
)
Import okio.IOException?
class Real : BlockingSource {
@Throws(...exceptionClasses = IOException::class)
override fun require(byteCount: Long, $completion: Continuation<*>): Any
class RequireContinuation(completion: Continuation<Any?>?) : Continuat
// $FF: synthetic field
var result: Any? = null 2 Usages
var label: Int = 0 6 Usages
public override fun invokeSuspend($result: Any): Any? {
this.result = $result
28 4 4 5
coroutines-party-tricks > coroutines-io > moshi > src > main > java > com > squareup > cursedmoshi > RealBlockingSource
4:1 LF UTF-8 4 spaces


Finder File Edit View Go Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource.kt
RecordJsonAdapter
StandardJsonAdapters
Require.kt
class RealBufferedSource { 1 Usage new *
fun requireSuspending(1 Usage new *
byteCount: Long,
completion: Continuation<Any?>,
): Any? {
class RequireContinuation(completion: Continuation<Any?>) : ContinuationI
var result: Any? = null 4 Usages
var label: Int = 0 6 Usages
public override fun invokeSuspend(result: Result<Any?>): Any? { new *
this.result = result
this.label = this.label or BIT_REUSE_OK
return requireSuspending(byteCount = 0L, completion = this as Continu
}
}
val continuation = when {
completion is RequireContinuation && (completion.label and BIT_REUSE_OK
completion.label -= BIT_REUSE_OK
completion
}
else -> RequireContinuation(completion)
}
coroutines-party-tricks > coroutines-io > moshi > src > main > java > com > squareup > cursedmoshi
25:1 LF UTF-8 2 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdapter
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingException
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource.kt
RecordJsonAdapter
StandardJsonAdapters
Require.kt
class RealBufferedSource { 1 Usage new *
fun requireSuspending(1 Usage new *
continuation.result.throwOnFailure()
continuation.label = 1
val requestResultOrSuspended = request(byteCount, continuation)
if (requestResultOrSuspended === COROUTINE_SUSPENDED) return COROUTIN
requestResultOrSuspended as Boolean
}
1 -> {
continuation.result.throwOnFailure()
continuation.result as Boolean
}
else -> throw IllegalStateException(s = "call to 'resume' before 'invok
}
if (!requestResult) {
throw EOFException()
}
return Unit
}
12 1 2
coroutines-party-tricks > coroutines-io > cursedokio > src > commonMain > kotlin > cursedokio > internal > Require.kt > RealBufferedSource
76:1 (1354 chars, 47 line breaks) LF UTF-8 2 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
coroutines-party-tricks
main
Current File
Project
java
com.squareup.cursedmoshi
internal
-MoshiKotlinExtensions.kt
-MoshiKotlinTypesExtensions.kt
AdapterMethodsFactory
ArrayJsonAdapter
Blocking
BlockingSource
ClassFactory
ClassJsonAdapter
CollectionJsonAdap
FromJson
Json
JsonAdapter
JsonClass
JsonDataException
JsonEncodingExcep
JsonQualifier
JsonReader
JsonScope
JsonUtf8Reader.kt
JsonUtf8Writer
JsonValueReader.kt
JsonValueSource
JsonValueWriter
JsonWriter
LinkedHashTreeMap.kt
MapJsonAdapter
Moshi
package-info.java
RealBlockingSource.kt
RecordJsonAdapter
StandardJsonAdapters
Require.kt
class RealBufferedSource { 1 Usage new *
fun requireSuspending(1 Usage new *
continuation.result.throwOnFailure()
continuation.label = 1
val requestResultOrSuspended = request(byteCount, continuation)
if (requestResultOrSuspended === COROUTINE_SUSPENDED) return COROUTIN
sume' before 'invok
}
if (!requestResult) {
throw EOFException()
}
return Unit
}
IntelliK IKEA
coroutines-party-tricks > coroutines-io > cursedokio > src > commonMain > kotlin > cursedokio > internal > Require.kt > RealBufferedSource
76:1 (1354 chars, 47 line breaks) LF UTF-8 2 spaces*

Keynote File Edit Insert Slide Format Arrange View Play Window Help

Coroutines Party Tricks — Edited

View Zoom Add Slide Play Table Chart Text Shape Media Comment Share Format Animate Document

1 Coroutines Party Tricks

2

32

33

56

57 Thanks

```
class RealBufferedSource : BufferedSource {

  override suspend fun require(byteCount: Long) {
    if (!request(byteCount)) {
      throw EOFException()
    }
  }

  override suspend fun request(byteCount: Long): Boolean {
    ...
  }

  ...
}
```

Coroutines Party Tricks — Edited

View Zoom 61% Add Slide Play Table Chart Text Shape Media Comment Share Format Animate Document

1

2

32

33

56

57

58

```
fun requireSuspending(
  byteCount: Long,
  completion: Continuation<Any?>,
): Any? {
  class RequireContinuation(completion: Continuation<Any?>) : ContinuationImpl(completion) {
    var result: Any? = null
    var label: Int = 0

    public override fun invokeSuspend(result: Result<Any?>): Any? {
      this.result = result
      this.label = this.label or BIT_REUSE_OK
      return requireSuspending(0L, this as Continuation<Any?>)
    }
  }

  val continuation = when {
    completion is RequireContinuation && (completion.label and BIT_REUSE_OK) != 0 -> {
      completion.label -= BIT_REUSE_OK
      completion
    }

    else -> RequireContinuation(completion)
  }

  val requestResult = when (continuation.label) {
    0 -> {
      continuation.result.throwOnFailure()
      continuation.label = 1
      val requestResultOrSuspended = request(byteCount, continuation)
      if (requestResultOrSuspended === COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
      requestResultOrSuspended as Boolean
    }

    1 -> {
      continuation.result.throwOnFailure()
      continuation.result as Boolean
    }

    else -> throw IllegalStateException("call to 'resume' before 'invoke' with coroutine")
  }

  if (!requestResult) {
    throw EOFException()
  }

  return Unit
}
```

```
fun requireSuspending(
  byteCount: Long,
  completion: Continuation<Any?>,
): Any? {
  class RequireContinuation(completion: Continuation<Any?>) : ContinuationImpl(completion) {
    var result: Any? = null
    var label: Int = 0

    public override fun invokeSuspend(result: Result<Any?>): Any? {
      this.result = result
      this.label = this.label or BIT_REUSE_OK
      return requireSuspending(0L, this as Continuation<Any?>)
    }
  }

  val continuation = when {
    completion is RequireContinuation && (completion.label and BIT_REUSE_OK) != 0 -> {
      completion.label -= BIT_REUSE_OK
      completion
    }

    else -> RequireContinuation(completion)
  }

  val requestResult = when (continuation.label) {
    0 -> {
      continuation.result.throwOnFailure()
      continuation.label = 1
      val requestResultOrSuspended = request(byteCount, continuation)
      if (requestResultOrSuspended === COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
      requestResultOrSuspended as Boolean
    }

    1 -> {
      continuation.result.throwOnFailure()
      continuation.result as Boolean
    }

    else -> throw IllegalStateException("call to 'resume' before 'invoke' with coroutine")
  }

  if (!requestResult) {
    throw EOFException()
  }

  return Unit
}
```

```
fun requireSuspending(
  byteCount: Long,
  completion: Continuation<Any?>,
): Any? {
  class RequireContinuation(completion: Continuation<Any?>) : ContinuationImpl(completion) {
    var result: Any? = null
    var label: Int = 0

    public override fun invokeSuspend(result: Result<Any?>): Any? {
      this.result = result
      this.label = this.label or BIT_REUSE_OK
      return requireSuspending(0L, this as Continuation<Any?>)
    }
  }

  val continuation = when {
    completion is RequireContinuation && (completion.label and BIT_REUSE_OK) != 0 -> {
      completion.label -= BIT_REUSE_OK
      completion
    }

    else -> RequireContinuation(completion)
  }
```

```
fun requireSuspending(
  byteCount: Long,
  completion: Continuation<Any?>,
): Any? {
  class RequireContinuation(completion: Continuation<Any?>) : ContinuationImpl(completion) {
    var result: Any? = null
    var label: Int = 0

    public override fun invokeSuspend(result: Result<Any?>): Any? {
      this.result = result
      this.label = this.label or BIT_REUSE_OK
      return requireSuspending(0L, this as Continuation<Any?>)
    }
  }

  val continuation = when {
    completion is RequireContinuation && (completion.label and BIT_REUSE_OK) != 0 -> {
      completion.label -= BIT_REUSE_OK
      completion
    }

    else -> RequireContinuation(completion)
  }
```

```
fun requireSuspending(
  byteCount: Long,
  completion: Continuation<Any?>,
): Any? {
  class RequireContinuation(completion: Continuation<Any?>) : ContinuationImpl(completion) {
    var result: Any? = null
    var label: Int = 0

    public override fun invokeSuspend(result: Result<Any?>): Any? {
      this.result = result
      this.label = this.label or BIT_REUSE_OK
      return requireSuspending(0L, this as Continuation<Any?>)
    }
  }

  val continuation = when {
    completion is RequireContinuation && (completion.label and BIT_REUSE_OK) != 0 -> {
      completion.label -= BIT_REUSE_OK
      completion
    }

    else -> RequireContinuation(completion)
  }
```

```
  var label: Int = 0

  public override fun invokeSuspend(result: Result<Any?>): Any? {
    this.result = result
    this.label = this.label or BIT_REUSE_OK
    return requireSuspending(0L, this as Continuation<Any?>)
  }
}

val continuation = when {
  completion is RequireContinuation && (completion.label and BIT_REUSE_OK) != 0 -> {
    completion.label -= BIT_REUSE_OK
    completion
  }

  else -> RequireContinuation(completion)
}

val requestResult = when (continuation.label) {
  0 -> {
    continuation.result.throwOnFailure()
    continuation.label = 1
    val requestResultOrSuspended = request(byteCount, continuation)
```

```
    }

    else -> RequireContinuation(completion)
  }

  val requestResult = when (continuation.label) {
    0 -> {
      continuation.result.throwOnFailure()
      continuation.label = 1
      val requestResultOrSuspended = request(byteCount, continuation)
      if (requestResultOrSuspended === COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
      requestResultOrSuspended as Boolean
    }

    1 -> {
      continuation.result.throwOnFailure()
      continuation.result as Boolean
    }

    else -> throw IllegalStateException("call to 'resume' before 'invoke' with coroutine")
  }

  if (!requestResult) {
    throw EOFException()
```

```
    }

    else -> RequireContinuation(completion)
  }

  val requestResult = when (continuation.label) {
    0 -> {
      continuation.result.throwOnFailure()
      continuation.label = 1
      val requestResultOrSuspended = request(byteCount, continuation)
      if (requestResultOrSuspended === COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
      requestResultOrSuspended as Boolean
    }

    1 -> {
      continuation.result.throwOnFailure()
      continuation.result as Boolean
    }

    else -> throw IllegalStateException("call to 'resume' before 'invoke' with coroutine")
  }

  if (!requestResult) {
    throw EOFException()
```

```
  val requestResult = when (continuation.label) {
    0 -> {
      continuation.result.throwOnFailure()
      continuation.label = 1
      val requestResultOrSuspended = request(byteCount, continuation)
      if (requestResultOrSuspended === COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
      requestResultOrSuspended as Boolean
    }

    1 -> {
      continuation.result.throwOnFailure()
      continuation.result as Boolean
    }

    else -> throw IllegalStateException("call to 'resume' before 'invoke' with coroutine")
  }

  if (!requestResult) {
    throw EOFException()
  }

  return Unit
}
```

```
  val requestResult = when (continuation.label) {
    0 -> {
      continuation.result.throwOnFailure()
      continuation.label = 1
      val requestResultOrSuspended = request(byteCount, continuation)
      if (requestResultOrSuspended === COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
      requestResultOrSuspended as Boolean
    }

    1 -> {
      continuation.result.throwOnFailure()
      continuation.result as Boolean
    }

    else -> throw IllegalStateException("call to 'resume' before 'invoke' with coroutine")
  }

  if (!requestResult) {
    throw EOFException()
  }

  return Unit
}
```

```
fun requireSuspending(
  byteCount: Long,
  completion: Continuation<Any?>,
): Any? {
  class RequireContinuation(completion: Continuation<Any?>) : ContinuationImpl(completion) {
    var result: Any? = null
    var label: Int = 0

    public override fun invokeSuspend(result: Result<Any?>): Any? {
      this.result = result
      this.label = this.label or BIT_REUSE_OK
      return requireSuspending(0L, this as Continuation<Any?>)
    }
  }

  val continuation = when {
    completion is RequireContinuation && (completion.label and BIT_REUSE_OK) != 0 -> {
      completion.label -= BIT_REUSE_OK
      completion
    }

    else -> RequireContinuation(completion)
  }

  val requestResult = when (continuation.label) {
    0 -> {
      continuation.result.throwOnFailure()
      continuation.label = 1
      val requestResultOrSuspended = request(byteCount, continuation)
      if (requestResultOrSuspended === COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
      requestResultOrSuspended as Boolean
    }

    1 -> {
      continuation.result.throwOnFailure()
      continuation.result as Boolean
    }

    else -> throw IllegalStateException("call to 'resume' before 'invoke' with coroutine")
  }

  if (!requestResult) {
    throw EOFException()
  }

  return Unit
}
```

Coroutines and I/O

Suspend functions do more

But they also do more

Advice

Suspend is fast actually

Use it everywhere

Low-level I/O code is weird and extremely performance-sensitive

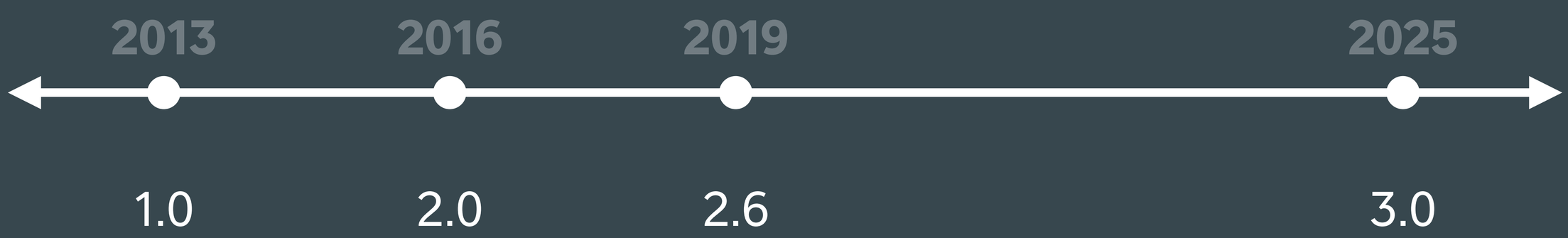
TRICK #2

Dynamic Proxies

Retrofit

```
public interface GitHubService {

  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(
    @Path("user") String user
  );

}
```

Older than Kotlin!



2.6 added a `compileOnly` dependency on Kotlin + coroutines!

Java users don’t depend on Kotlin

```
package java.lang.reflect;

public class Proxy {
  public static Object newProxyInstance(
    ClassLoader loader,
    Class<?>[] interfaces,
    InvocationHandler invocationHandler
  ) throws IllegalArgumentException;
}

public interface InvocationHandler {
  public Object invoke(
    Object proxy,
    Method method,
    Object[] args
  ) throws Throwable;
}
```

```
package java.lang.reflect;

public class Proxy {
  public static Object newProxyInstance(
    ClassLoader loader,
    Class<?>[] interfaces,
    InvocationHandler invocationHandler
  ) throws IllegalArgumentException;
}

public interface InvocationHandler {
  public Object invoke(
    Object proxy,
    Method method,
    Object[] args
  ) throws Throwable;
}
```

```
interface GitHubService {

  @GET("users/{user}/repos")
  fun listRepos(
    @Path("user") user: String
  ): Call<List<Repo>>

}
```

```
interface GitHubService {

  @GET("users/{user}/repos")
  suspend fun listRepos(
    @Path("user") user: String
  ): Call<List<Repo>>

}
```

```
static class SuspendingInvocationHandler implements InvocationHandler {

  @Override
  public Object invoke(
    Object proxy,
    Method method,
    Object[] args
  ) throws Throwable {

    Call<?> call = new OkHttpCall<>(args, ...);

    Continuation<?> continuation = (Continuation<?>) args[args.length - 1];

    ...

    return KotlinExtensions.await(call, continuation);
  }
}
```

```
static class SuspendingInvocationHandler implements InvocationHandler {

  @Override
  public Object invoke(
    Object proxy,
    Method method,
    Object[] args
  ) throws Throwable {

    Call<?> call = new OkHttpCall<>(args, ...);

    Continuation<?> continuation = (Continuation<?>) args[args.length - 1];

    ...

    return KotlinExtensions.await(call, continuation);
  }
}
```

```
static class SuspendingInvocationHandler implements InvocationHandler {

  @Override
  public Object invoke(
    Object proxy,
    Method method,
    Object[] args
  ) throws Throwable {

    Call<?> call = new OkHttpCall<>(args, ...);

    Continuation<?> continuation = (Continuation<?>) args[args.length - 1];

    ...

    return KotlinExtensions.await(call, continuation);
  }
}
```

```
static class SuspendingInvocationHandler implements InvocationHandler {

  @Override
  public Object invoke(
    Object proxy,
    Method method,
    Object[] args
  ) throws Throwable {

    Call<?> call = new OkHttpCall<>(args, ...);

    Continuation<?> continuation = (Continuation<?>) args[args.length - 1];

    ...

    return KotlinExtensions.await(call, continuation);
  }
}
```

```
static class SuspendingInvocationHandler implements InvocationHandler {

  @Override
  public Object invoke(
    Object proxy,
    Method method,
    Object[] args
  ) throws Throwable {

    Call<?> call = new OkHttpCall<>(args, ...);

    Continuation<?> continuation = (Continuation<?>) args[args.length - 1];

    ...

    return KotlinExtensions.await(call, continuation);
  }
}
```

Java + Coroutines

Java signatures have an additional argument, a `Continuation`

Receive it in a dynamic proxy

Pass it when calling Kotlin from Java

Gotchas

Kotlin coroutines and Java don’t really interop

Java wraps exceptions with `UndeclaredThrowableException`

Work around that by forcing a suspend!

Advice

Suspending in Retrofit means we don’t need to in Okio!

TRICK #3

Cancel, Safely

```
interface HttpCall {
  fun cancel()
  fun execute(): Response
  fun enqueue(callback: HttpCallback)
}

interface HttpCallback {
  fun onResponse(response: Response)
  fun onFailure(e: IOException)
}

suspend fun executeAsync(httpCall: HttpCall): Response {
  // how?
}
```

ATTEMPT 1

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 1

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 1

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 1

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 1

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 1

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 1

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

Canceling the coroutine does nothing

ATTEMPT 1

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 2

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCancellableCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 2

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCancellableCoroutine { continuation ->
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 2

```
suspend fun executeAsync(httpCall: HttpCall): Respon
  return suspendCancellableCoroutine { continua ion
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Respons
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

Canceling the coroutine
doesn’t cancel the HTTP call

ATTEMPT 3

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCancellableCoroutine { continuation ->
    continuation.invokeOnCancellation {
      httpCall.cancel()
    }
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 3

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCancellableCoroutine { continuation ->
    continuation.invokeOnCancellation {
      httpCall.cancel()
    }
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

ATTEMPT 3

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCancellableCoroutine { continuation ->
    continuation.invokeOnCancellation {
      httpCall.cancel()
    }
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response)
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

The response isn't closed if it's canceled after `onResponse`

ATTEMPT 4

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCancellableCoroutine { continuation ->
    continuation.invokeOnCancellation {
      httpCall.cancel()
    }
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response) { cause, value, context ->
            response.close()
          }
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

```
suspend fun executeAsync(httpCall: HttpCall): Response {
  return suspendCancellableCoroutine { continuation ->
    continuation.invokeOnCancellation {
      httpCall.cancel()
    }
    httpCall.enqueue(
      object : HttpCallback {
        override fun onResponse(response: Response) {
          continuation.resume(response) { cause, value, context ->
            response.close()
          }
        }

        override fun onFailure(e: IOException) {
          continuation.resumeWithException(e)
        }
      }
    )
  }
}
```

Cancel works & nothing leaks!

Advice

Use `suspendCancelableCoroutine()` and `invokeOnCancellation()`

Be careful passing `Closable` things to `Continuation.resume()`

TRICK #4

Stacks

Let’s do some math

3 = 3

3+3 = 6

3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3 = 72

Let’s write some code

```
fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

Let’s write some code

```
fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

Let’s write some code

```
fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

Let’s write some code

```
fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

Let’s write some code

```
fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

Let’s test

```
class EvaluateTest {
  @Test
  fun happyPath() {
    assertThat(evaluate("3")).isEqualTo(3)
    assertThat(evaluate("3+3+3")).isEqualTo(9)
  }

  @Test
  fun veryRecursive() {
    assertThat(evaluate("1+".repeat(1234) + "0")).isEqualTo(1234)
    assertThat(evaluate("1+".repeat(12345) + "0")).isEqualTo(12345)
  }
}
```

Let’s test

```
class EvaluateTest {
  @Test
  fun happyPath() {
    assertThat(evaluate("3")).isEqualTo(3)
    assertThat(evaluate("3+3+3")).isEqualTo(9)
  }

  @Test
  fun veryRecursive() {
    assertThat(evaluate("1+".repeat(1234) + "0")).isEqualTo(1234)
    assertThat(evaluate("1+".repeat(12345) + "0")).isEqualTo(12345)
  }
}
```

Let’s test

```
class EvaluateTest {
  @Test
  fun happyPath() {
    assertThat(evaluate("3")).isEqualTo(3)
    assertThat(evaluate("3+3+3")).isEqualTo(9)
  }

  @Test
  fun veryRecursive() {
    assertThat(evaluate("1+".repeat(1234) + "0")).isEqualTo(1234)
    assertThat(evaluate("1+".repeat(12345) + "0")).isEqualTo(12345)
  }
}
```

Let’s be sad

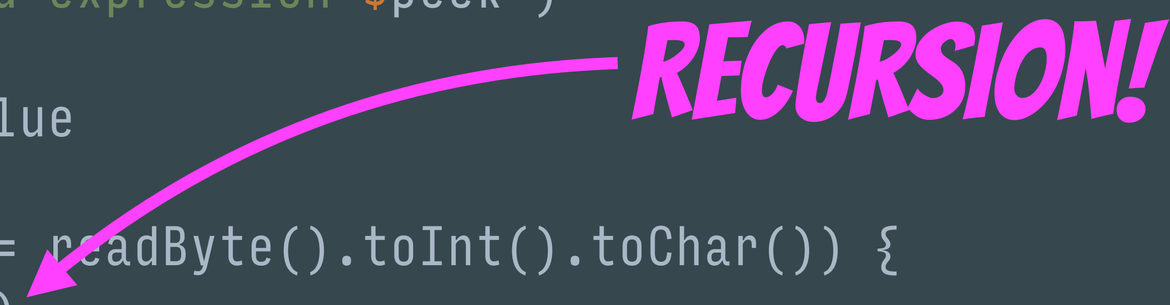
```
java.lang.StackOverflowError
  at okio.Buffer.getByte(Buffer.kt:714)
  at EvaluateKt.evaluate(Evaluate.kt:6)
  at EvaluateKt.evaluate(Evaluate.kt:13)
  at EvaluateKt.evaluate(Evaluate.kt:13)
  at EvaluateKt.evaluate(Evaluate.kt:13)
  at EvaluateKt.evaluate(Evaluate.kt:13)
   <1,016 folded frames>
```

Recursion!

```
fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

Recursion!

```
fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```



Recursion!

```
fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

RECURSION!

RECURSION!

Let’s fix the code

But how?

What’s this talk about again?

Let’s coroutines?

```
fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

Let’s coroutines?

```
fun suspend Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

Nope

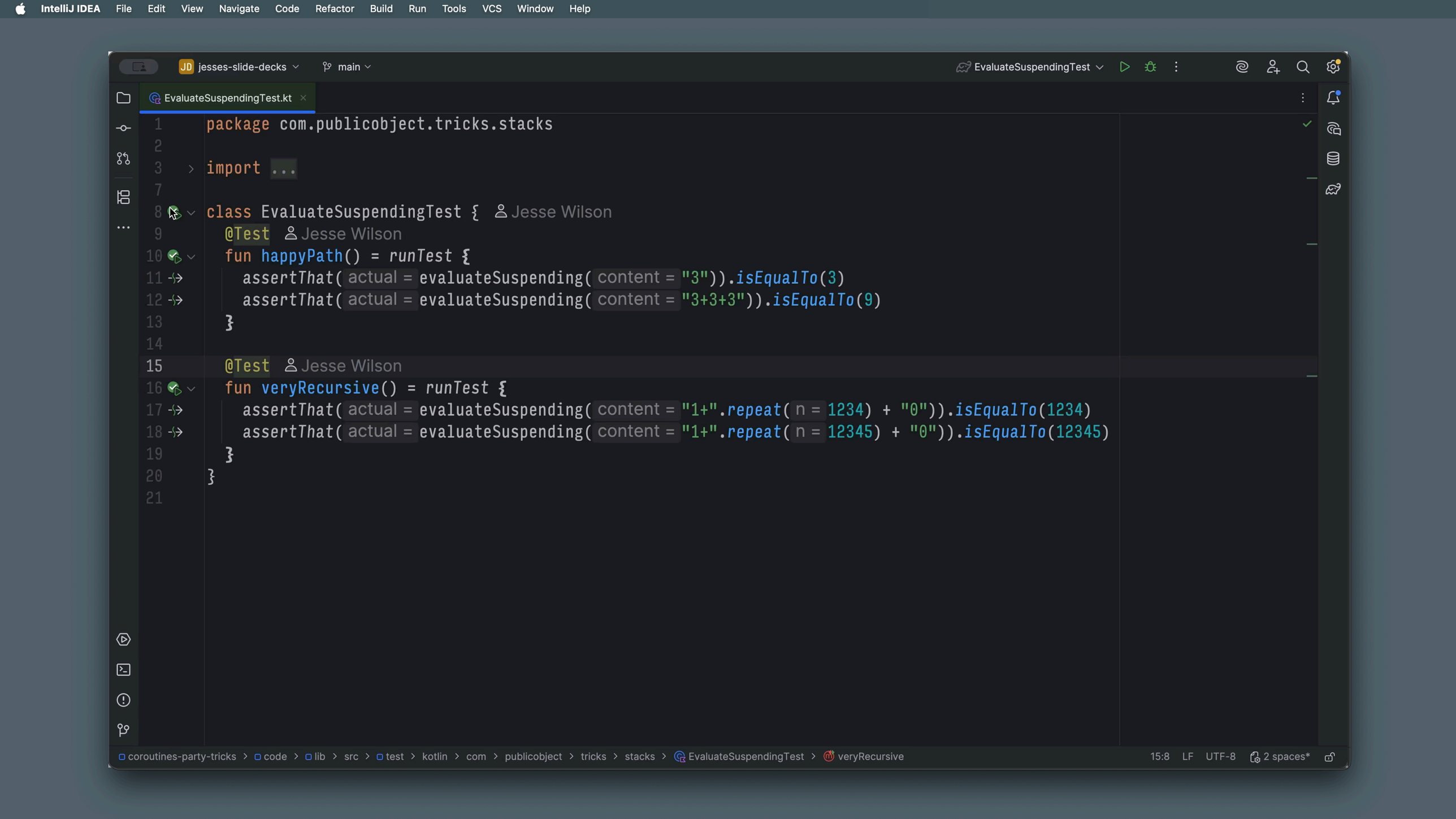
```
java.lang.StackOverflowError
  at kotlin.coroutines.jvm.internal.ContinuationImpl.<init>(ContinuationImpl.kt:102)
  at EvaluateKt$evaluate$1.<init>(EvaluateSuspending.kt)
  at EvaluateKt.evaluate(Evaluate.kt)
  at EvaluateKt.evaluate(Evaluate.kt:13)
  at EvaluateKt.evaluate(Evaluate.kt:13)
  at EvaluateKt.evaluate(Evaluate.kt:13)
  at EvaluateKt.evaluate(Evaluate.kt:13)
   <1,016 folded frames>
```

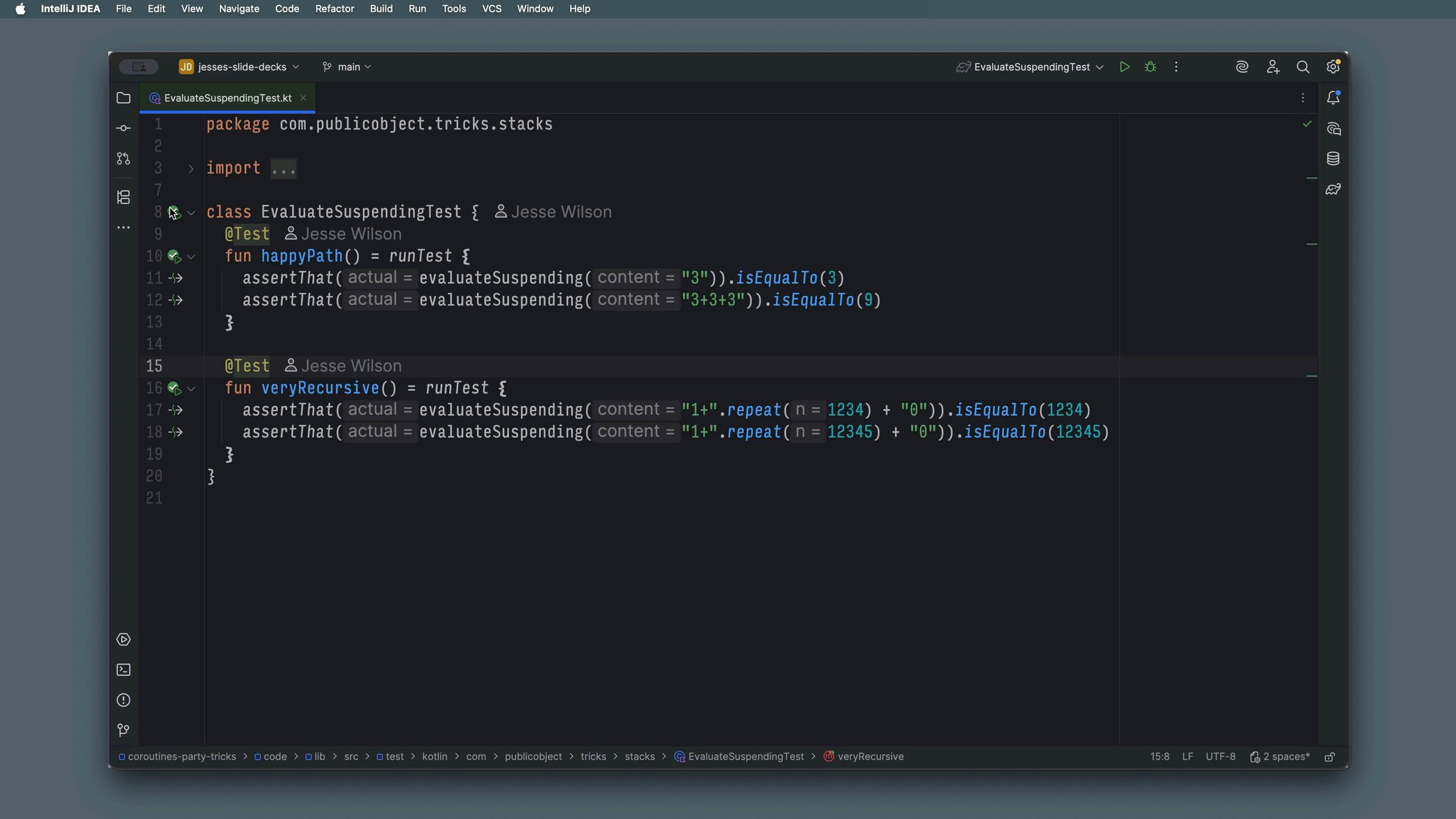
Let’s coroutines... harder?

```
suspend fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```

Let’s coroutines... harder?

```
suspend fun Buffer.evaluate(): Long {
  val value = when (val peek = this[0].toInt().toChar()) {
    in '0'..'9' -> readDecimalLong()
    else -> error("unexpected expression $peek")
  }
  if (exhausted()) return value

  yield()

  return when (val operator = readByte().toInt().toChar()) {
    '+' -> value + evaluate()
    '*' -> value * evaluate()
    else -> error("unexpected expression $operator")
  }
}
```


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
jesses-slide-decks
main
EvaluateSuspendingTest
EvaluateSuspendingTest.kt
package com.publicobject.tricks.stacks
import ...
class EvaluateSuspendingTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluateSuspending(content = "3")).isEqualTo(3)
assertThat(actual = evaluateSuspending(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluateSuspending(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluateSuspending(content = "1+".repeat(n = 12345) + "0")).isEqualTo(12345)
}
}
coroutines-party-tricks › code › lib › src › test › kotlin › com › publicobject › tricks › stacks › EvaluateSuspendingTest › veryRecursive
15:8 LF UTF-8 2 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
jesses-slide-decks
main
EvaluateSuspendingTest
EvaluateSuspendingTest.kt
package com.publicobject.tricks.stacks
import ...
class EvaluateSuspendingTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluateSuspending(content = "3")).isEqualTo(3)
assertThat(actual = evaluateSuspending(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluateSuspending(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluateSuspending(content = "1+".repeat(n = 12345) + "0")).isEqualTo(12345)
}
}
coroutines-party-tricks › code › lib › src › test › kotlin › com › publicobject › tricks › stacks › EvaluateSuspendingTest › veryRecursive
15:8 LF UTF-8 2 spaces*

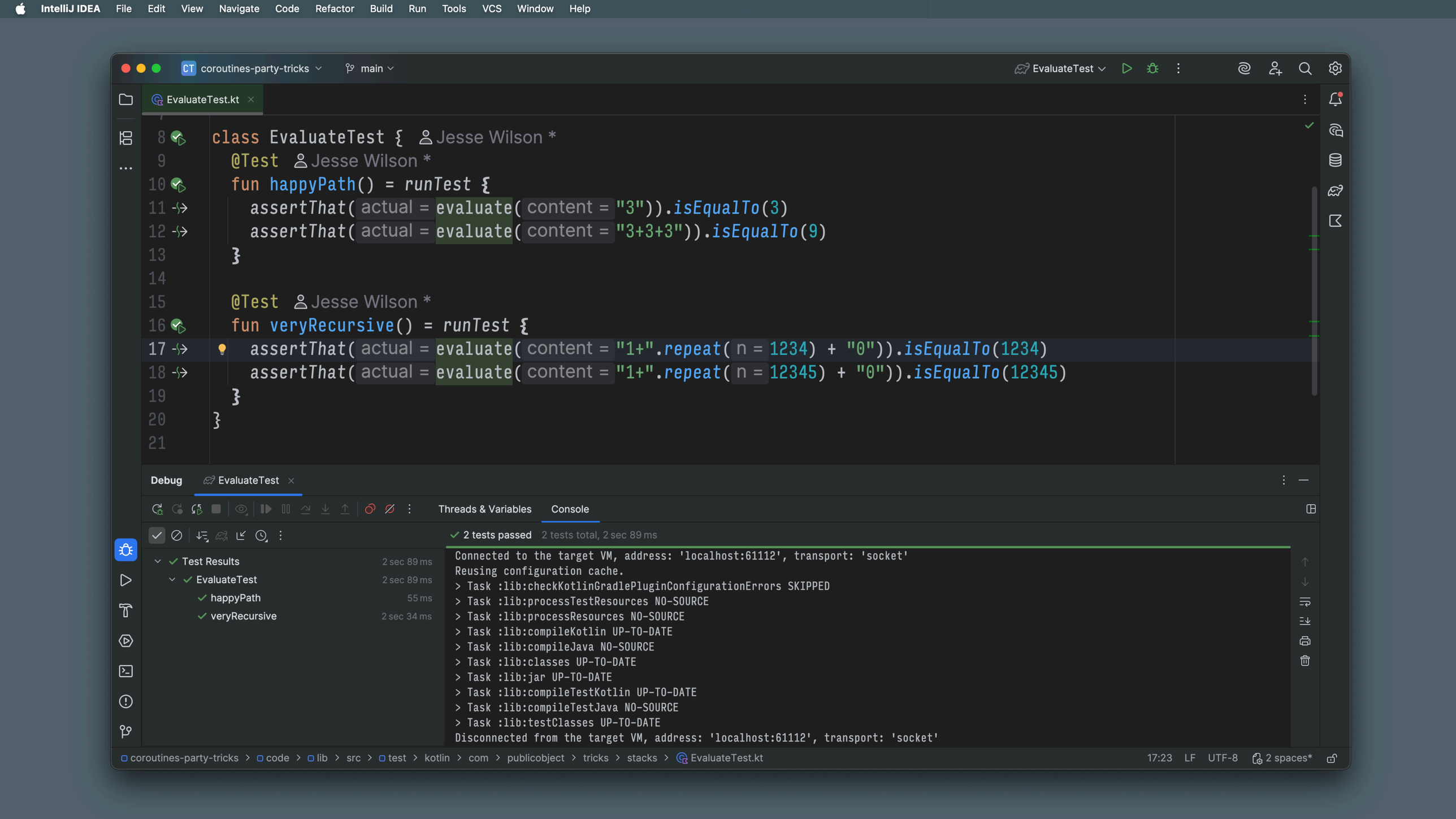
Why Does It Work?

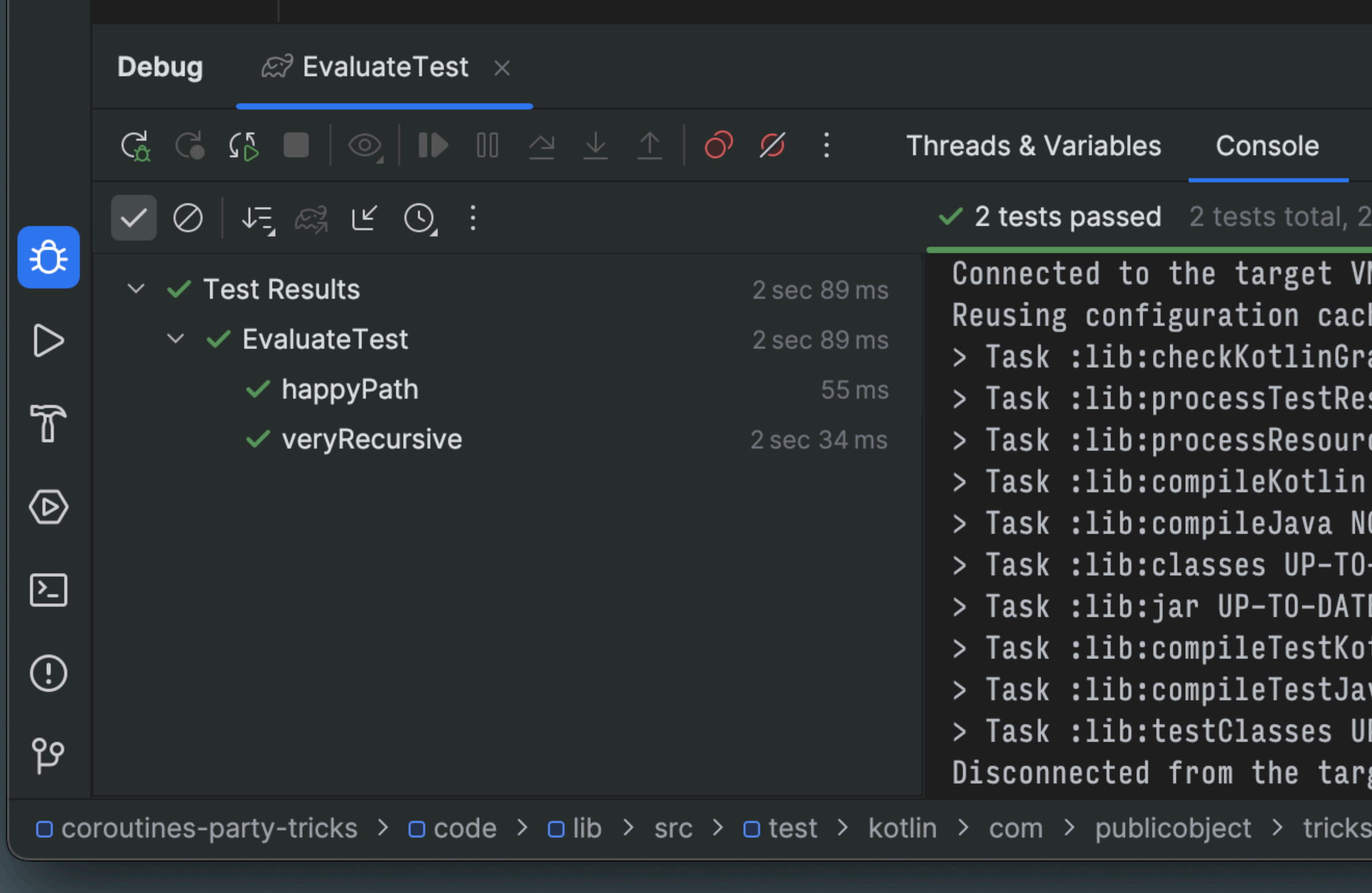
Calling a function pushes a frame on the stack

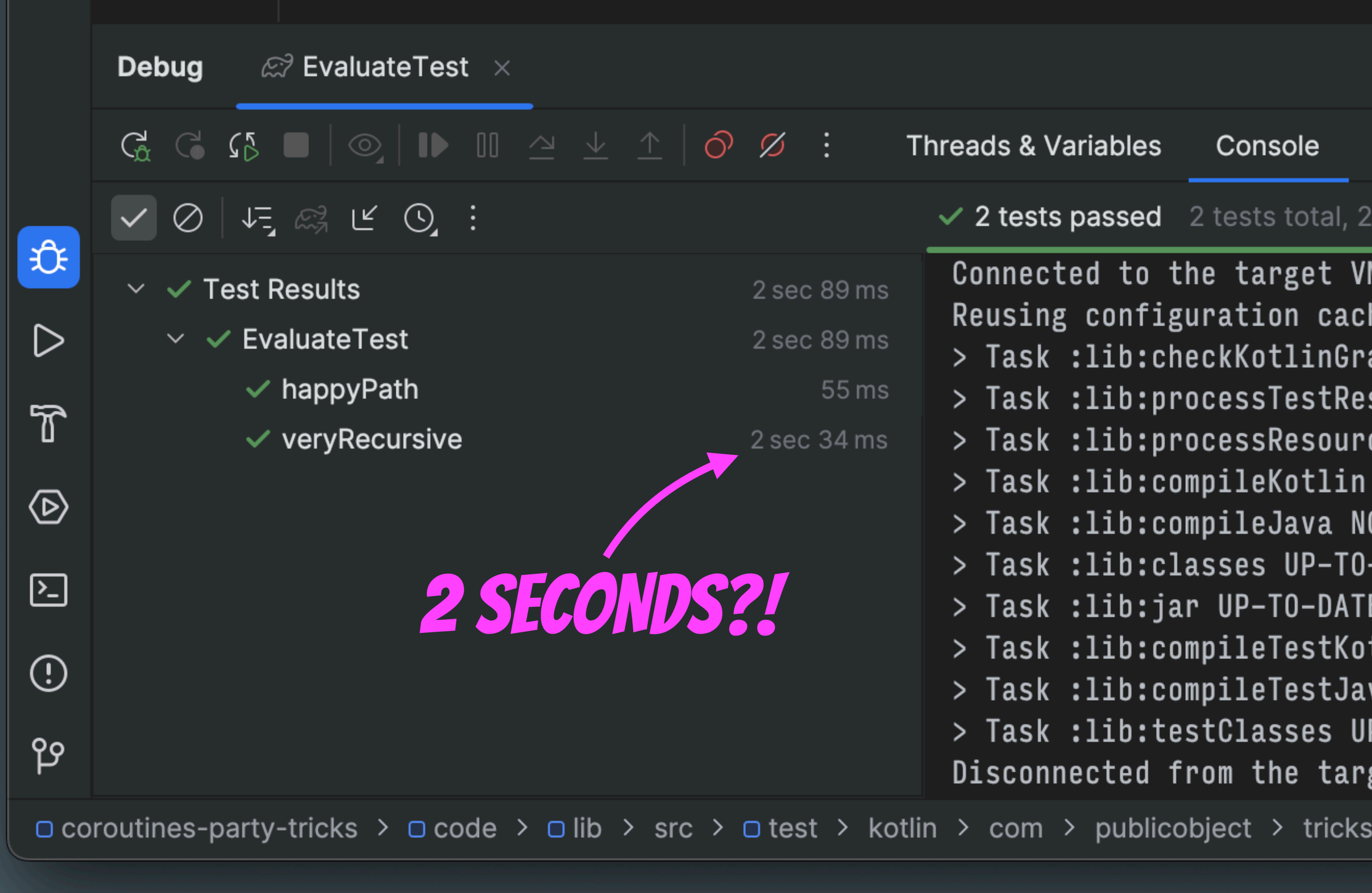
- This applies to regular functions and `suspend` functions!
- Calling 12,345 functions causes a `StackOverflowError`

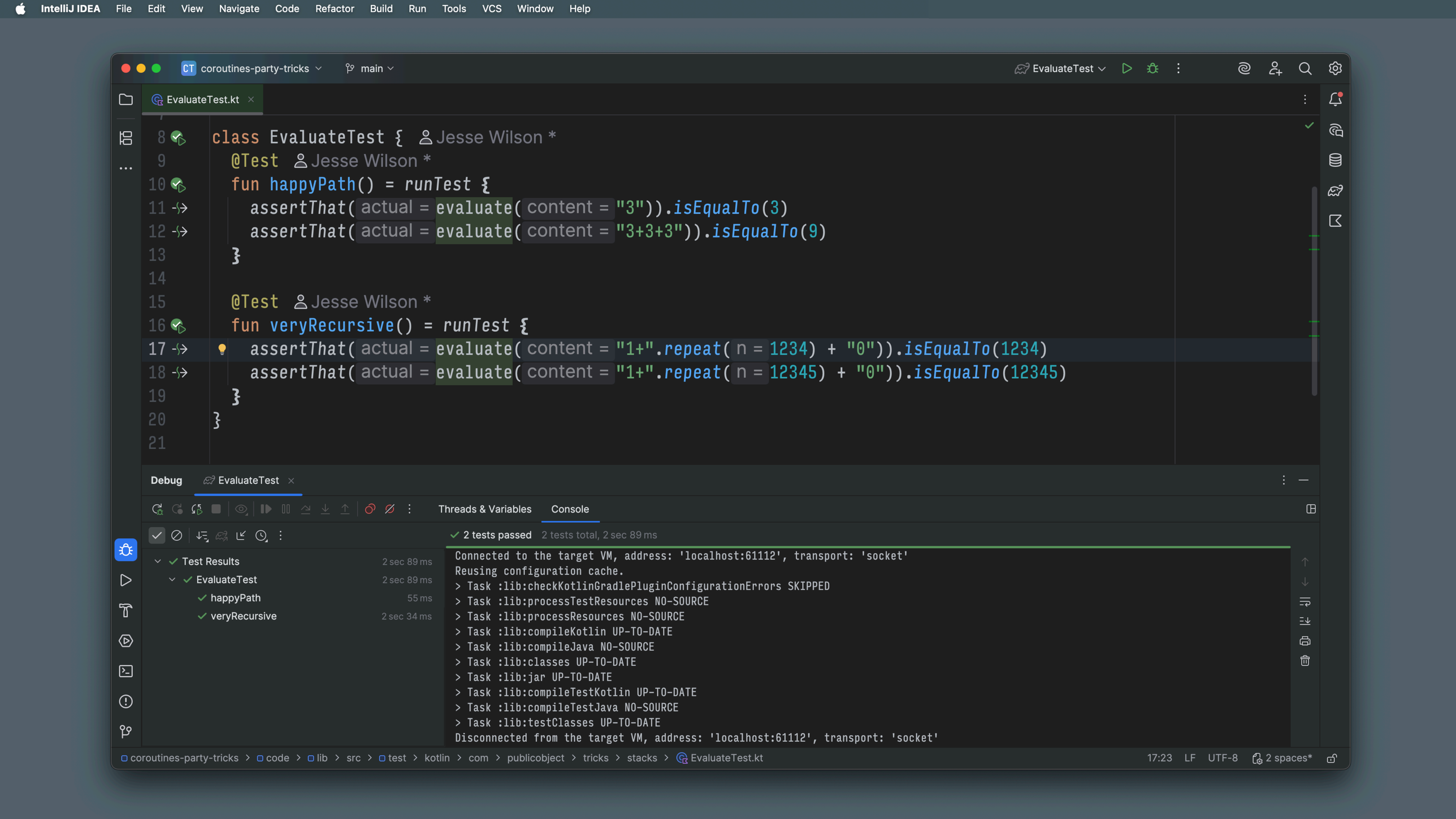
But `yield()` clears the stack!

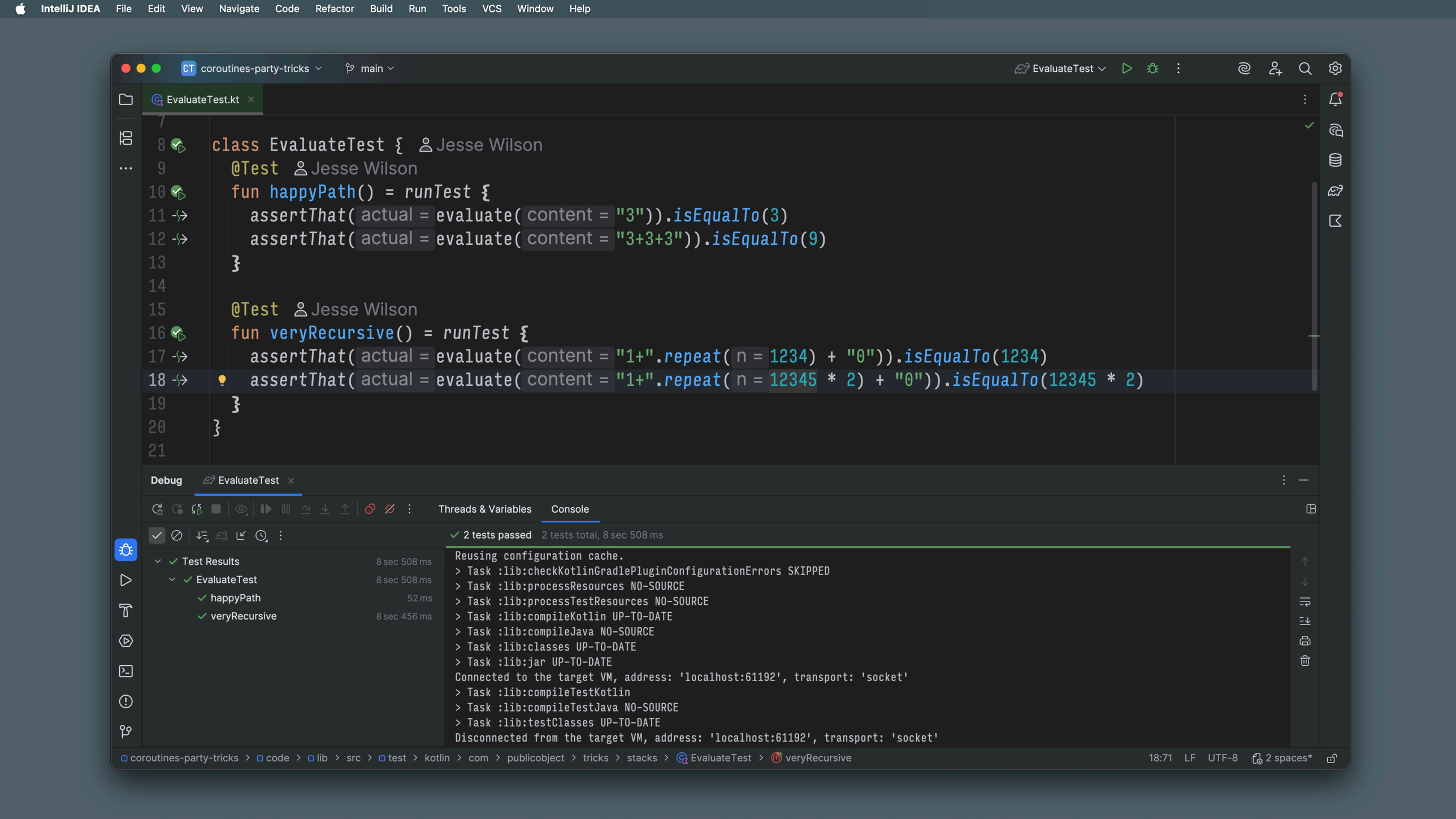
(Each stack frame is in memory, just not on the stack)

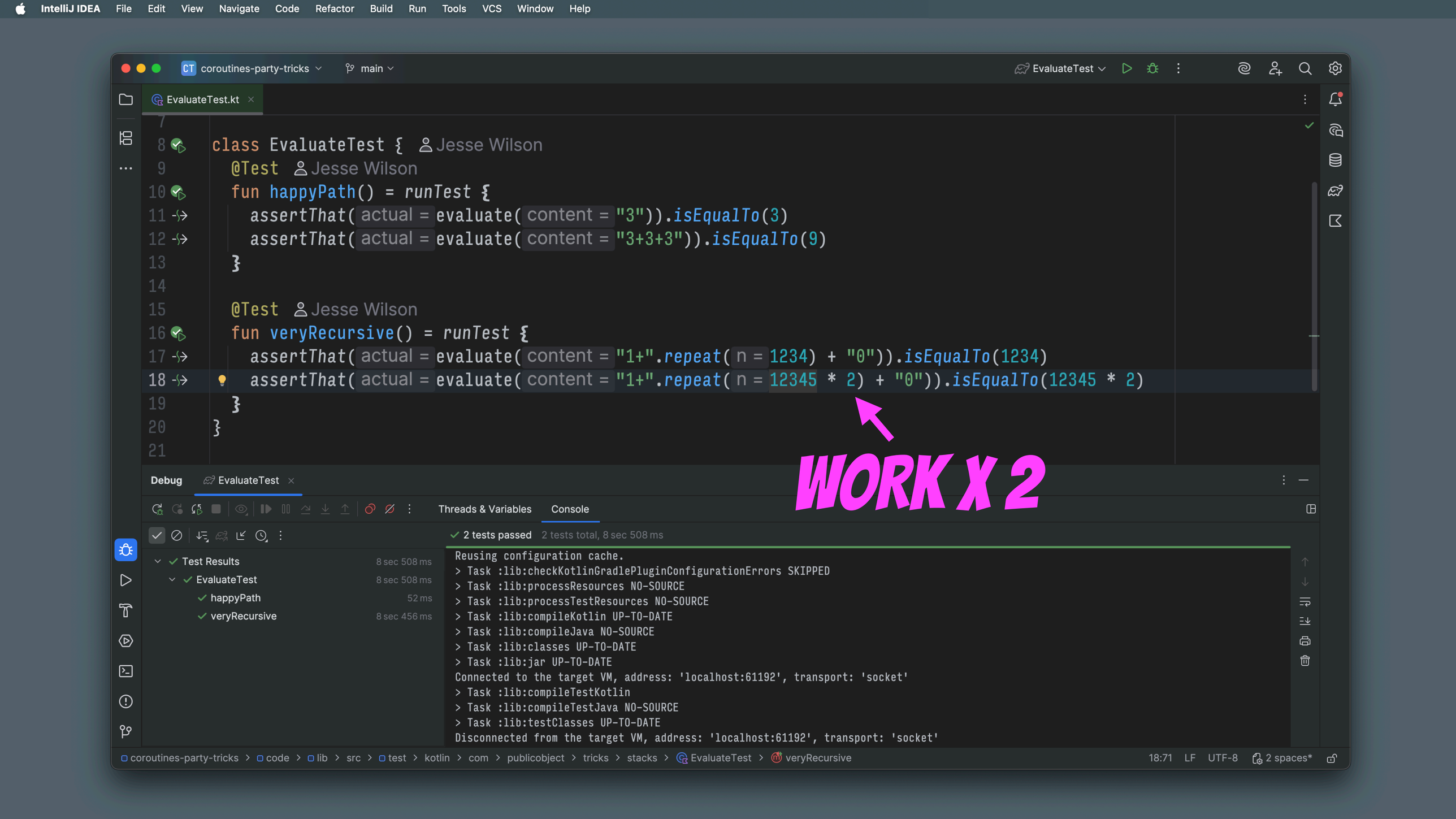

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson *
@Test Jesse Wilson *
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson *
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345) + "0")).isEqualTo(12345)
}
}
Debug
EvaluateTest
Threads & Variables
Console
2 tests passed 2 tests total, 2 sec 89 ms
Test Results 2 sec 89 ms
EvaluateTest 2 sec 89 ms
happyPath 55 ms
veryRecursive 2 sec 34 ms
Connected to the target VM, address: 'localhost:61112', transport: 'socket'
Reusing configuration cache.
> Task :lib:checkKotlinGradlePluginConfigurationErrors SKIPPED
> Task :lib:processTestResources NO-SOURCE
> Task :lib:processResources NO-SOURCE
> Task :lib:compileKotlin UP-TO-DATE
> Task :lib:compileJava NO-SOURCE
> Task :lib:classes UP-TO-DATE
> Task :lib:jar UP-TO-DATE
> Task :lib:compileTestKotlin UP-TO-DATE
> Task :lib:compileTestJava NO-SOURCE
> Task :lib:testClasses UP-TO-DATE
Disconnected from the target VM, address: 'localhost:61112', transport: 'socket'
coroutines-party-tricks > code > lib > src > test > kotlin > com > publicobject > tricks > stacks > EvaluateTest.kt
17:23 LF UTF-8 2 spaces*

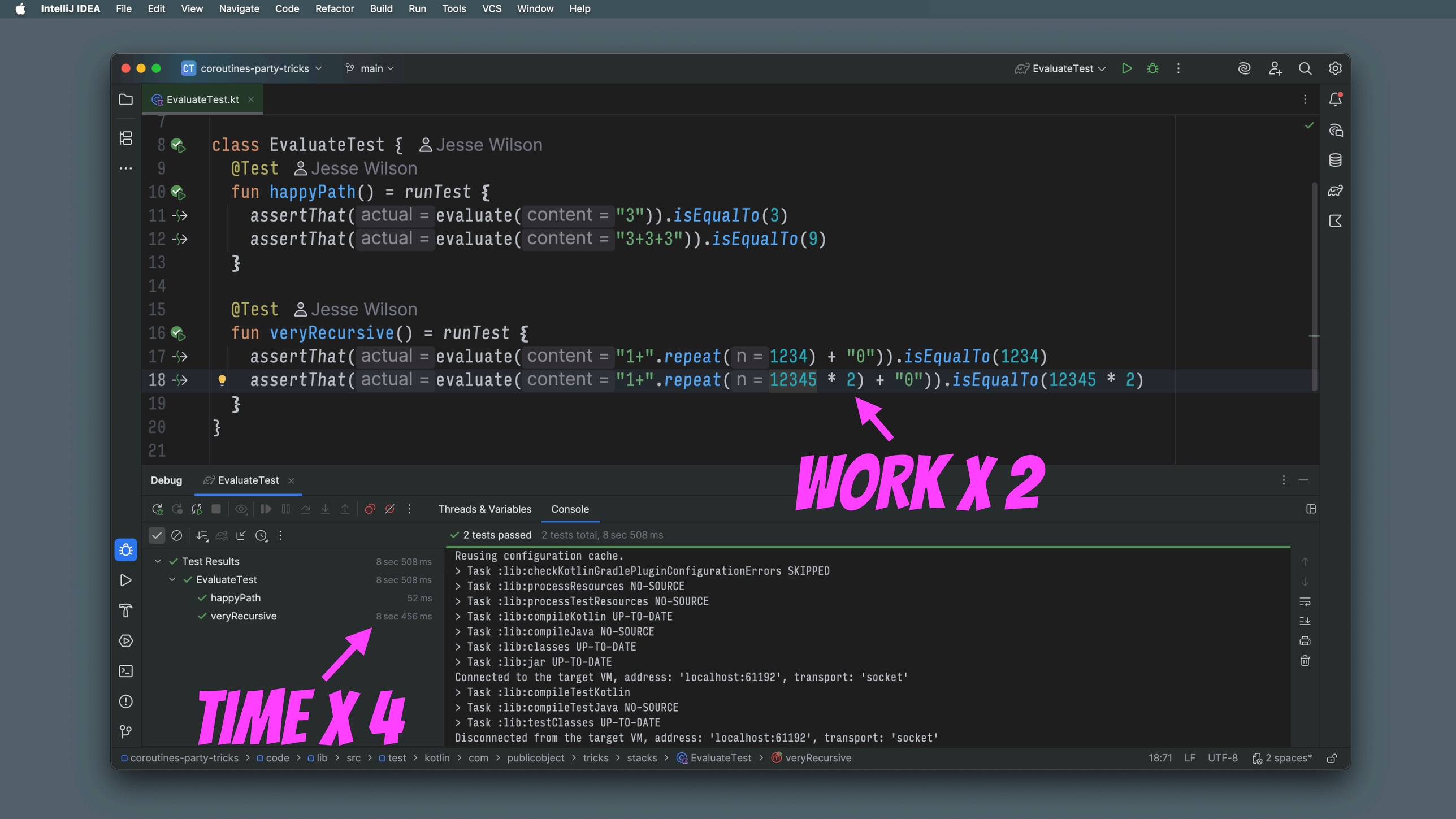

Debug
EvaluateTest
Threads & Variables
Console
2 tests passed
2 tests total, 2
Test Results 2 sec 89 ms
EvaluateTest 2 sec 89 ms
happyPath 55 ms
veryRecursive 2 sec 34 ms
Connected to the target VM
Reusing configuration cach
> Task :lib:checkKotlinGra
> Task :lib:processTestRes
> Task :lib:processResourc
> Task :lib:compileKotlin
> Task :lib:compileJava NO
> Task :lib:classes UP-TO-
> Task :lib:jar UP-TO-DATE
> Task :lib:compileTestKot
> Task :lib:compileTestJav
> Task :lib:testClasses UP
Disconnected from the targ
coroutines-party-tricks
code
lib
src
test
kotlin
com
publicobject
tricks

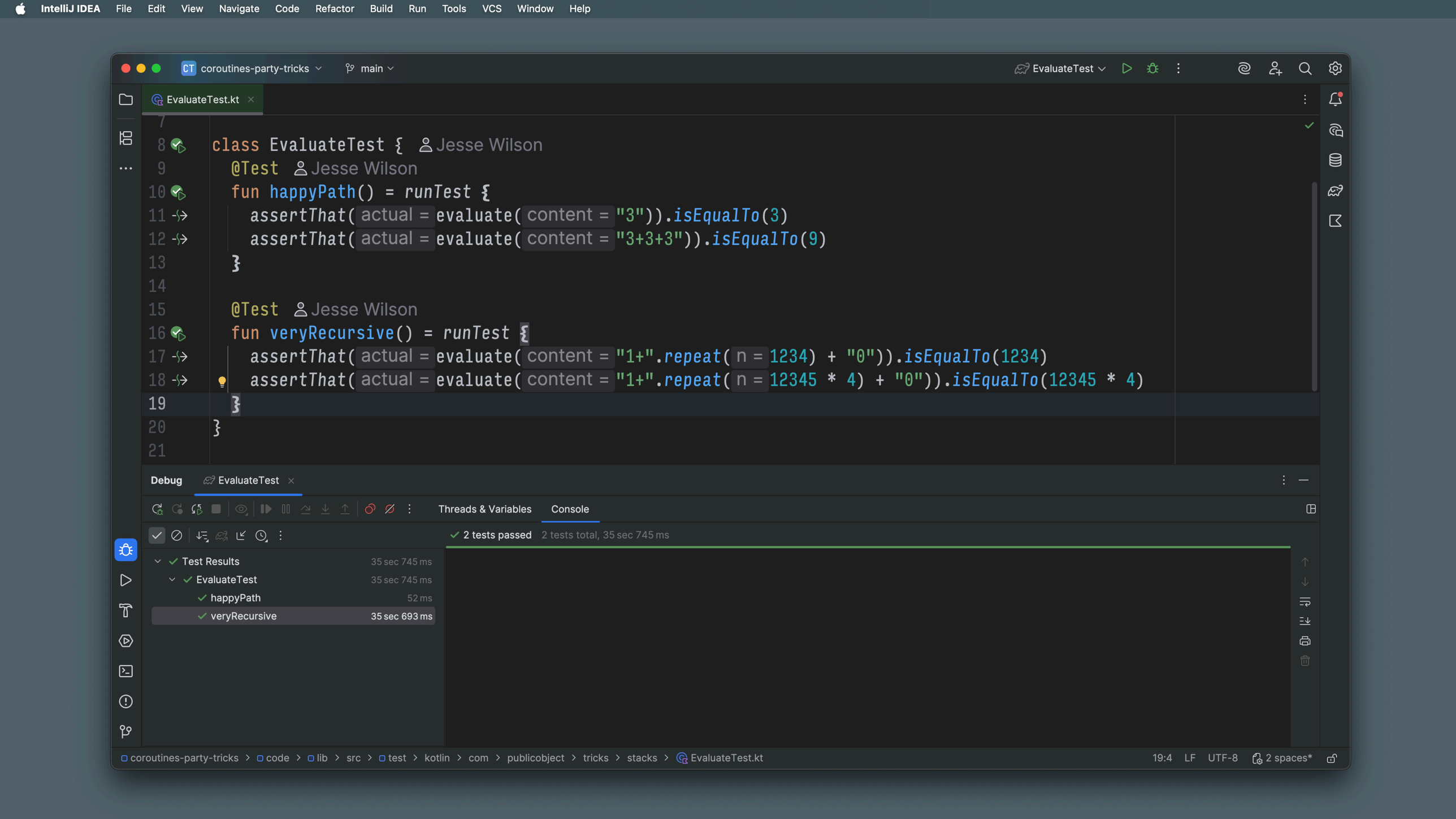

Debug
EvaluateTest
Threads & Variables
Console
2 tests passed
2 tests total, 2
Test Results 2 sec 89 ms
EvaluateTest 2 sec 89 ms
happyPath 55 ms
veryRecursive 2 sec 34 ms
2 SECONDS?!
Connected to the target VM
Reusing configuration cach
> Task :lib:checkKotlinGra
> Task :lib:processTestRes
> Task :lib:processResourc
> Task :lib:compileKotlin
> Task :lib:compileJava NO
> Task :lib:classes UP-TO-
> Task :lib:jar UP-TO-DATE
> Task :lib:compileTestKot
> Task :lib:compileTestJav
> Task :lib:testClasses UP
Disconnected from the targ
coroutines-party-tricks
code
lib
src
test
kotlin
com
publicobject
tricks

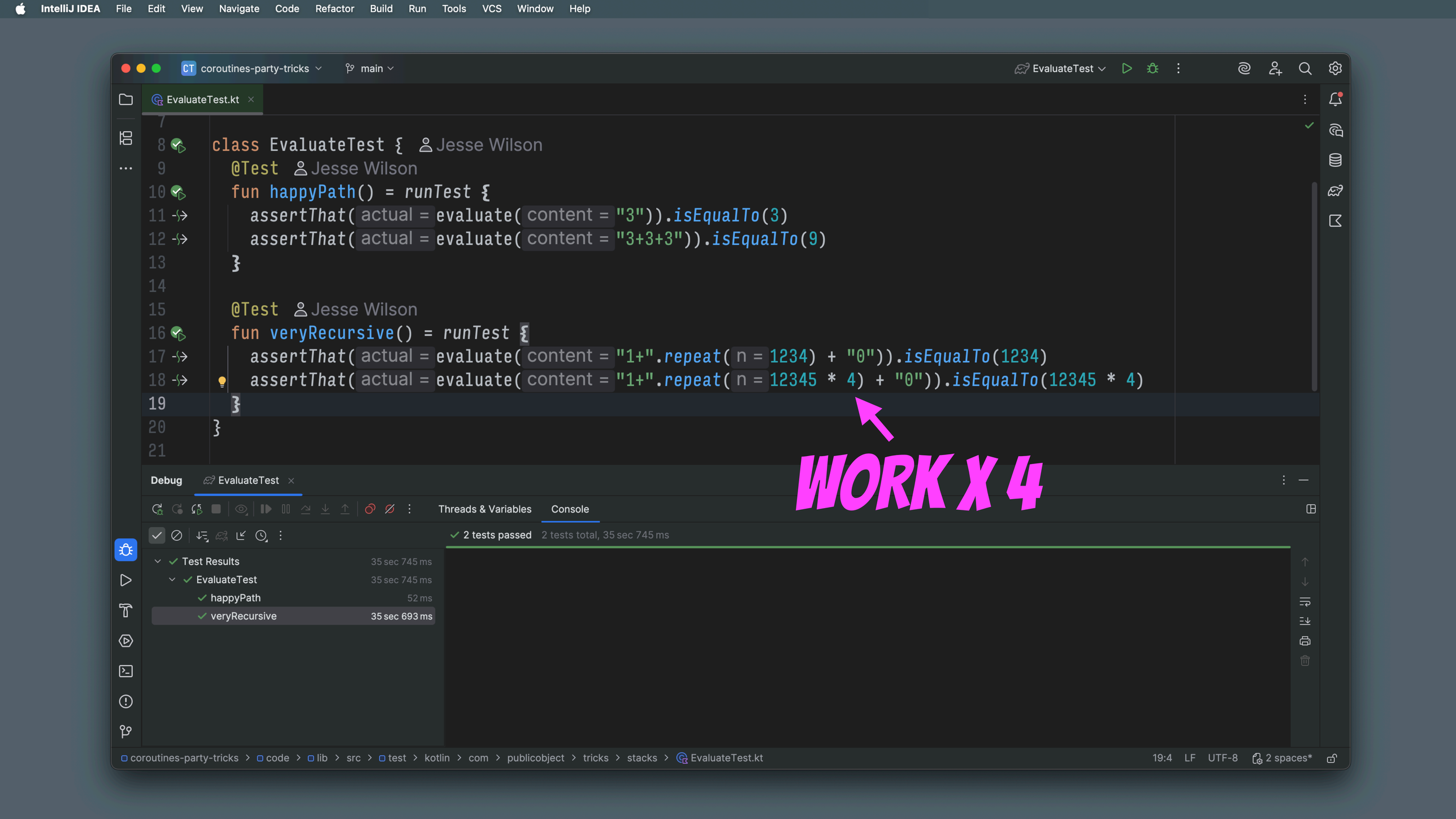

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson *
@Test Jesse Wilson *
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson *
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345) + "0")).isEqualTo(12345)
}
}
Debug
EvaluateTest
Threads & Variables
Console
2 tests passed 2 tests total, 2 sec 89 ms
Test Results 2 sec 89 ms
EvaluateTest 2 sec 89 ms
happyPath 55 ms
veryRecursive 2 sec 34 ms
Connected to the target VM, address: 'localhost:61112', transport: 'socket'
Reusing configuration cache.
> Task :lib:checkKotlinGradlePluginConfigurationErrors SKIPPED
> Task :lib:processTestResources NO-SOURCE
> Task :lib:processResources NO-SOURCE
> Task :lib:compileKotlin UP-TO-DATE
> Task :lib:compileJava NO-SOURCE
> Task :lib:classes UP-TO-DATE
> Task :lib:jar UP-TO-DATE
> Task :lib:compileTestKotlin UP-TO-DATE
> Task :lib:compileTestJava NO-SOURCE
> Task :lib:testClasses UP-TO-DATE
Disconnected from the target VM, address: 'localhost:61112', transport: 'socket'
coroutines-party-tricks > code > lib > src > test > kotlin > com > publicobject > tricks > stacks > EvaluateTest.kt
17:23 LF UTF-8 2 spaces*

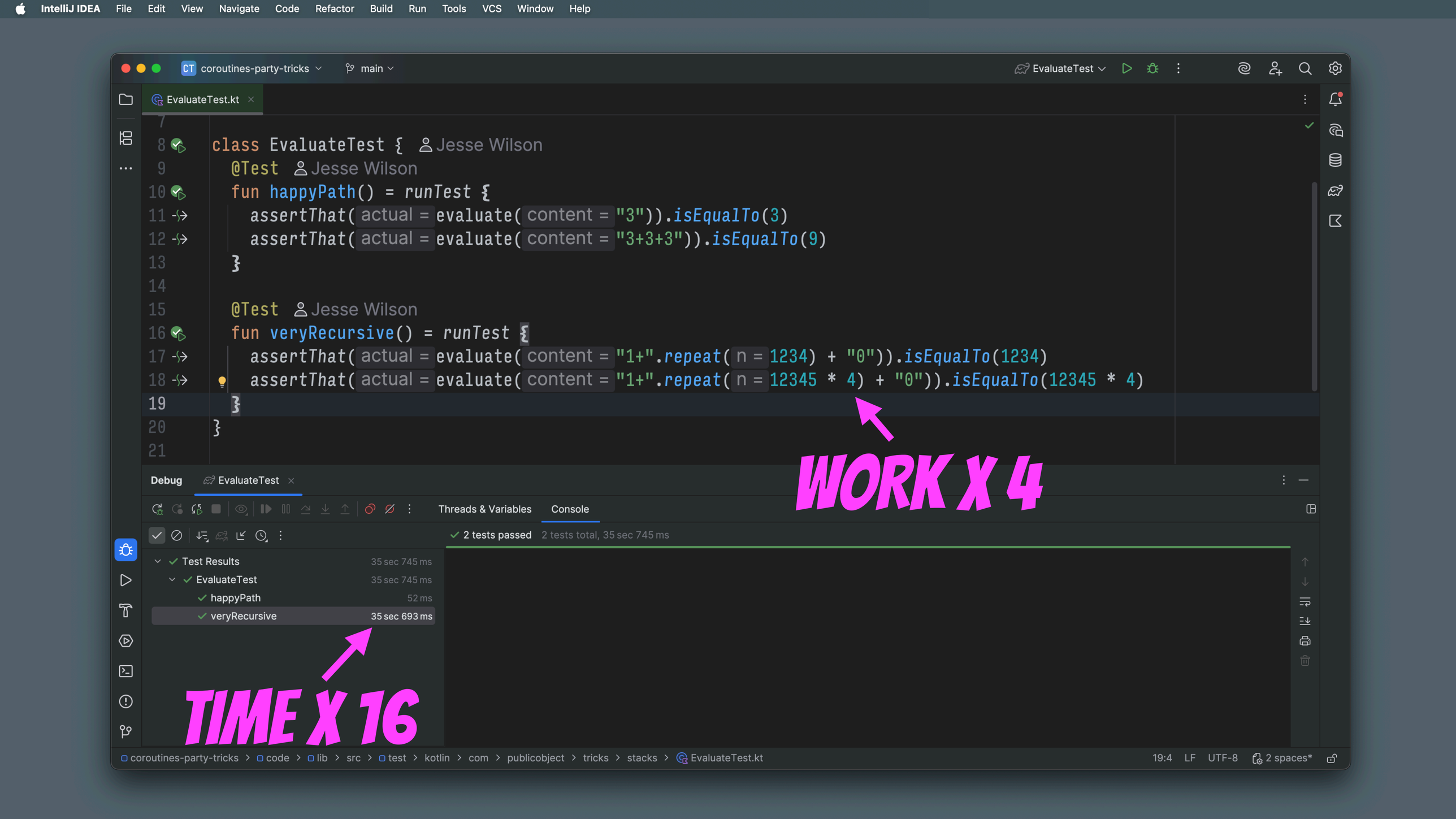

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345 * 2) + "0")).isEqualTo(12345 * 2)
}
}
Debug
EvaluateTest
Threads & Variables
Console
2 tests passed 2 tests total, 8 sec 508 ms
Test Results 8 sec 508 ms
EvaluateTest 8 sec 508 ms
happyPath 52 ms
veryRecursive 8 sec 456 ms
Reusing configuration cache.
> Task :lib:checkKotlinGradlePluginConfigurationErrors SKIPPED
> Task :lib:processResources NO-SOURCE
> Task :lib:processTestResources NO-SOURCE
> Task :lib:compileKotlin UP-TO-DATE
> Task :lib:compileJava NO-SOURCE
> Task :lib:classes UP-TO-DATE
> Task :lib:jar UP-TO-DATE
Connected to the target VM, address: 'localhost:61192', transport: 'socket'
> Task :lib:compileTestKotlin
> Task :lib:compileTestJava NO-SOURCE
> Task :lib:testClasses UP-TO-DATE
Disconnected from the target VM, address: 'localhost:61192', transport: 'socket'
coroutines-party-tricks > code > lib > src > test > kotlin > com > publicobject > tricks > stacks > EvaluateTest > veryRecursive
18:71 LF UTF-8 2 spaces*

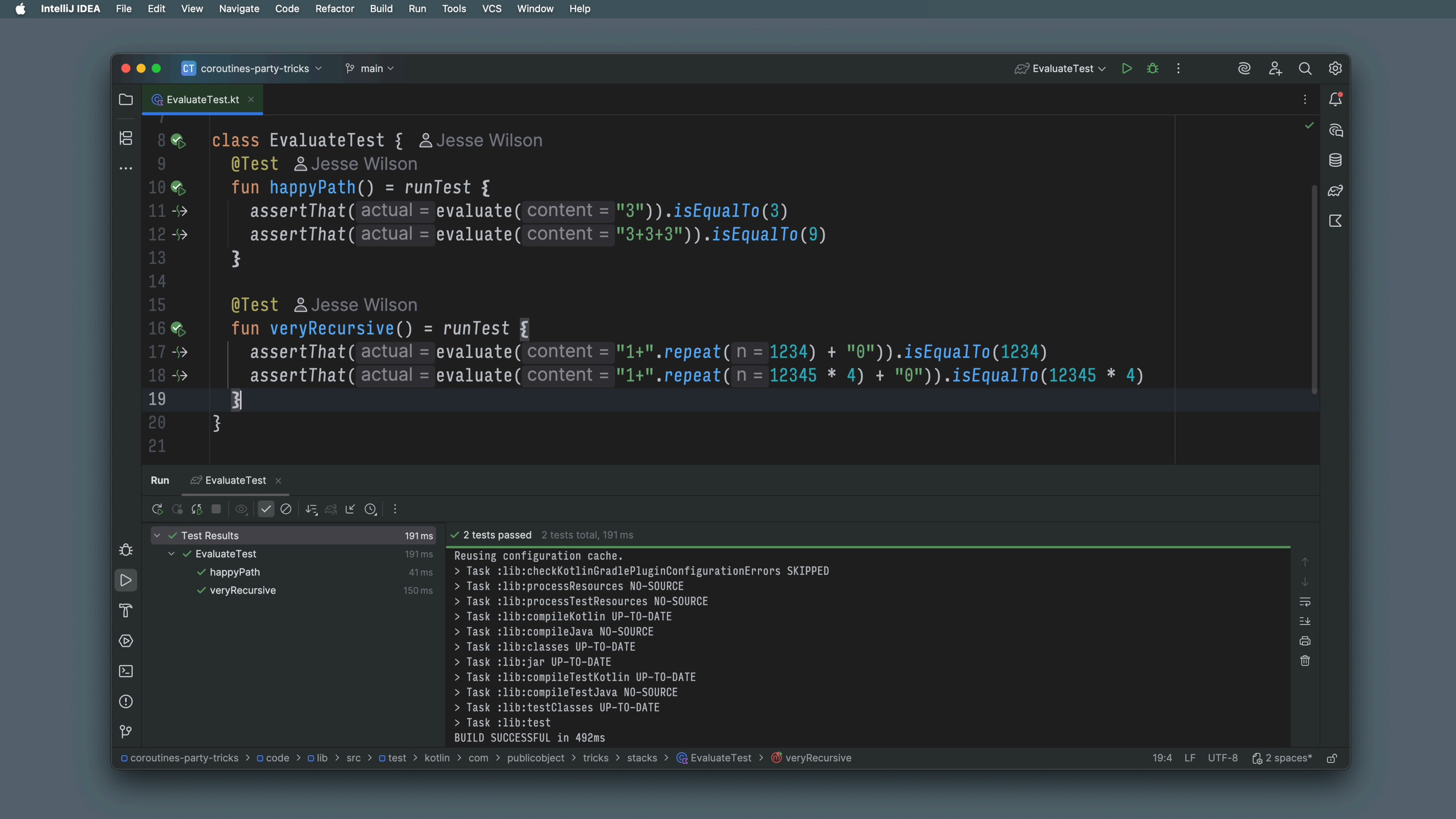

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345 * 2) + "0")).isEqualTo(12345 * 2)
}
}
WORK X 2
Debug
EvaluateTest
Threads & Variables
Console
2 tests passed 2 tests total, 8 sec 508 ms
Test Results 8 sec 508 ms
EvaluateTest 8 sec 508 ms
happyPath 52 ms
veryRecursive 8 sec 456 ms
Reusing configuration cache.
> Task :lib:checkKotlinGradlePluginConfigurationErrors SKIPPED
> Task :lib:processResources NO-SOURCE
> Task :lib:processTestResources NO-SOURCE
> Task :lib:compileKotlin UP-TO-DATE
> Task :lib:compileJava NO-SOURCE
> Task :lib:classes UP-TO-DATE
> Task :lib:jar UP-TO-DATE
Connected to the target VM, address: 'localhost:61192', transport: 'socket'
> Task :lib:compileTestKotlin
> Task :lib:compileTestJava NO-SOURCE
> Task :lib:testClasses UP-TO-DATE
Disconnected from the target VM, address: 'localhost:61192', transport: 'socket'
coroutines-party-tricks > code > lib > src > test > kotlin > com > publicobject > tricks > stacks > EvaluateTest > veryRecursive
18:71 LF UTF-8 2 spaces*

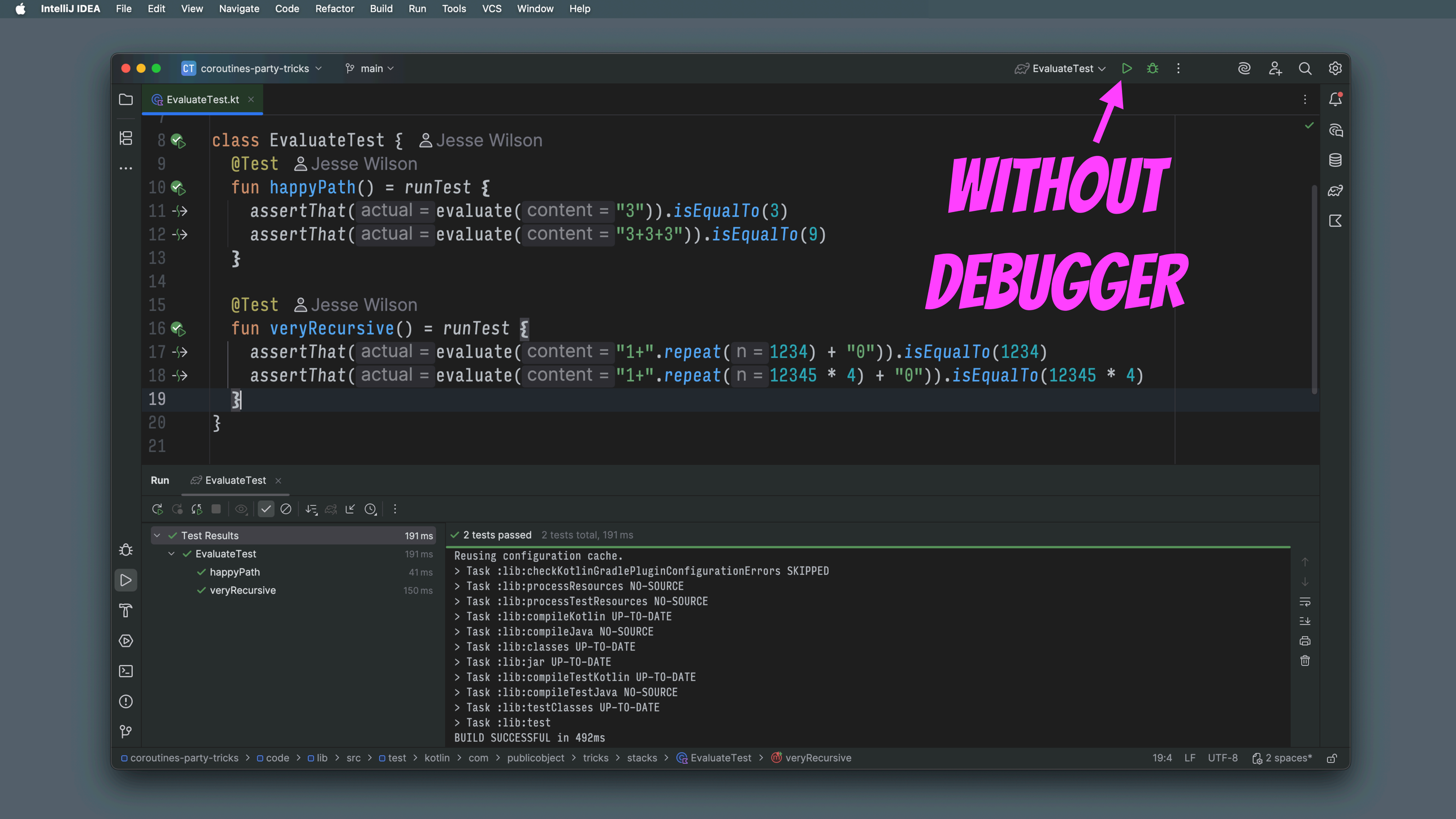

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345 * 2) + "0")).isEqualTo(12345 * 2)
}
}
WORK X 2
Debug
EvaluateTest
Threads & Variables
Console
2 tests passed 2 tests total, 8 sec 508 ms
Test Results 8 sec 508 ms
EvaluateTest 8 sec 508 ms
happyPath 52 ms
veryRecursive 8 sec 456 ms
TIME X 4
Reusing configuration cache.
> Task :lib:checkKotlinGradlePluginConfigurationErrors SKIPPED
> Task :lib:processResources NO-SOURCE
> Task :lib:processTestResources NO-SOURCE
> Task :lib:compileKotlin UP-TO-DATE
> Task :lib:compileJava NO-SOURCE
> Task :lib:classes UP-TO-DATE
> Task :lib:jar UP-TO-DATE
Connected to the target VM, address: 'localhost:61192', transport: 'socket'
> Task :lib:compileTestKotlin
> Task :lib:compileTestJava NO-SOURCE
> Task :lib:testClasses UP-TO-DATE
Disconnected from the target VM, address: 'localhost:61192', transport: 'socket'
coroutines-party-tricks > code > lib > src > test > kotlin > com > publicobject > tricks > stacks > EvaluateTest > veryRecursive
18:71 LF UTF-8 2 spaces*

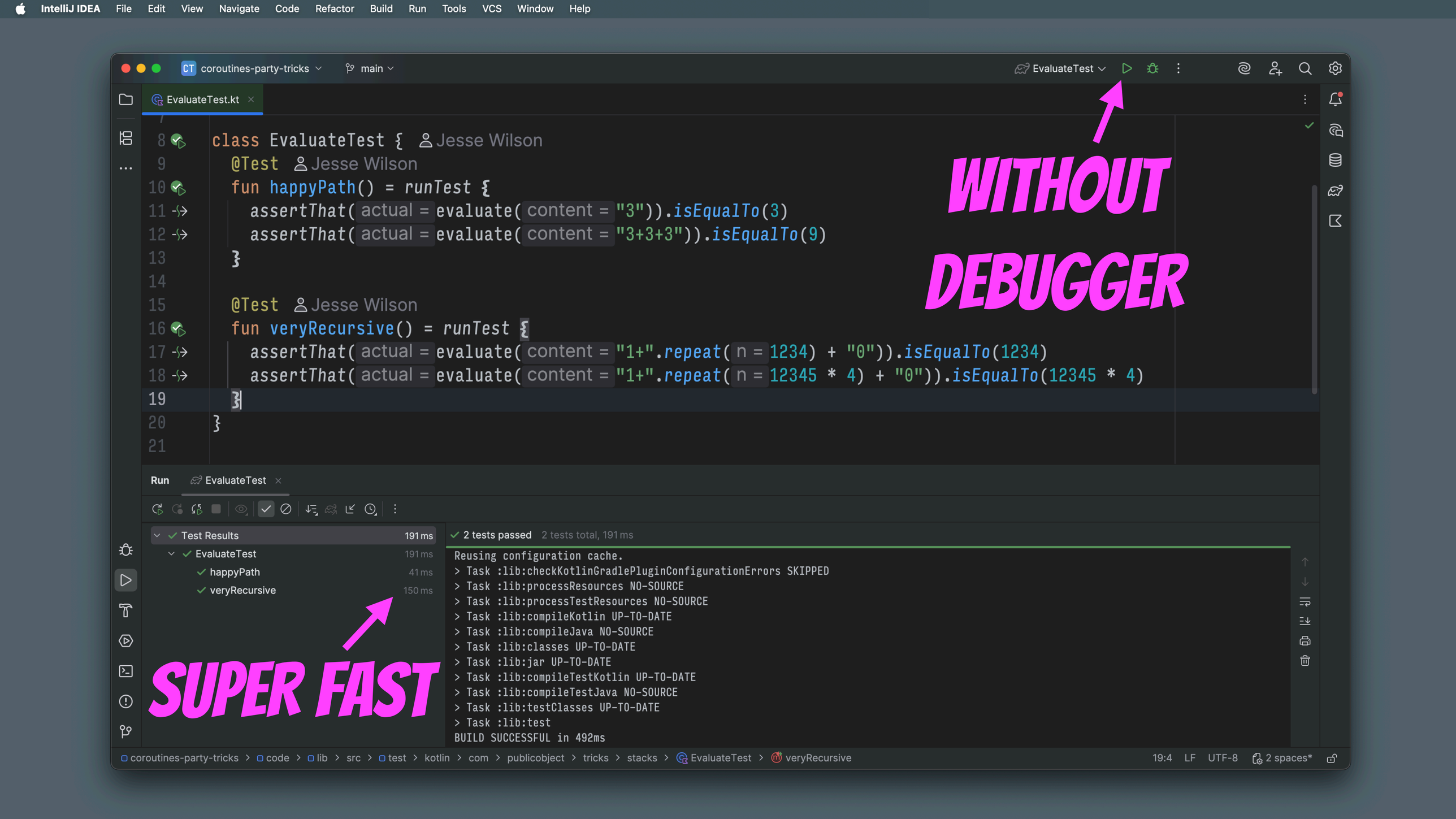

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345 * 4) + "0")).isEqualTo(12345 * 4)
}
}
Debug
EvaluateTest
Threads & Variables
Console
2 tests passed 2 tests total, 35 sec 745 ms
Test Results 35 sec 745 ms
EvaluateTest 35 sec 745 ms
happyPath 52 ms
veryRecursive 35 sec 693 ms
coroutines-party-tricks > code > lib > src > test > kotlin > com > publicobject > tricks > stacks > EvaluateTest.kt
19:4 LF UTF-8 2 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345 * 4) + "0")).isEqualTo(12345 * 4)
}
}
WORK X 4
Debug
EvaluateTest
Threads & Variables
Console
2 tests passed
2 tests total, 35 sec 745 ms
Test Results 35 sec 745 ms
EvaluateTest 35 sec 745 ms
happyPath 52 ms
veryRecursive 35 sec 693 ms
coroutines-party-tricks › code › lib › src › test › kotlin › com › publicobject › tricks › stacks › EvaluateTest.kt
19:4 LF UTF-8 2 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345 * 4) + "0")).isEqualTo(12345 * 4)
}
}
WORK X 4
Debug
EvaluateTest
Threads & Variables
Console
2 tests passed
2 tests total, 35 sec 745 ms
Test Results 35 sec 745 ms
EvaluateTest 35 sec 745 ms
happyPath 52 ms
veryRecursive 35 sec 693 ms
TIME X 16
coroutines-party-tricks › code › lib › src › test › kotlin › com › publicobject › tricks › stacks › EvaluateTest.kt
19:4 LF UTF-8 2 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345 * 4) + "0")).isEqualTo(12345 * 4)
}
}
Run
EvaluateTest
Test Results 191 ms
EvaluateTest 191 ms
happyPath 41 ms
veryRecursive 150 ms
2 tests passed 2 tests total, 191 ms
Reusing configuration cache.
> Task :lib:checkKotlinGradlePluginConfigurationErrors SKIPPED
> Task :lib:processResources NO-SOURCE
> Task :lib:processTestResources NO-SOURCE
> Task :lib:compileKotlin UP-TO-DATE
> Task :lib:compileJava NO-SOURCE
> Task :lib:classes UP-TO-DATE
> Task :lib:jar UP-TO-DATE
> Task :lib:compileTestKotlin UP-TO-DATE
> Task :lib:compileTestJava NO-SOURCE
> Task :lib:testClasses UP-TO-DATE
> Task :lib:test
BUILD SUCCESSFUL in 492ms
coroutines-party-tricks › code › lib › src › test › kotlin › com › publicobject › tricks › stacks › EvaluateTest › veryRecursive
19:4 LF UTF-8 2 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345 * 4) + "0")).isEqualTo(12345 * 4)
}
}
WITHOUT DEBUGGER
Run
EvaluateTest
Test Results 191 ms
EvaluateTest 191 ms
happyPath 41 ms
veryRecursive 150 ms
2 tests passed 2 tests total, 191 ms
Reusing configuration cache.
> Task :lib:checkKotlinGradlePluginConfigurationErrors SKIPPED
> Task :lib:processResources NO-SOURCE
> Task :lib:processTestResources NO-SOURCE
> Task :lib:compileKotlin UP-TO-DATE
> Task :lib:compileJava NO-SOURCE
> Task :lib:classes UP-TO-DATE
> Task :lib:jar UP-TO-DATE
> Task :lib:compileTestKotlin UP-TO-DATE
> Task :lib:compileTestJava NO-SOURCE
> Task :lib:testClasses UP-TO-DATE
> Task :lib:test
BUILD SUCCESSFUL in 492ms
coroutines-party-tricks › code › lib › src › test › kotlin › com › publicobject › tricks › stacks › EvaluateTest › veryRecursive
19:4 LF UTF-8 2 spaces*


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
coroutines-party-tricks
main
EvaluateTest
EvaluateTest.kt
class EvaluateTest { Jesse Wilson
@Test Jesse Wilson
fun happyPath() = runTest {
assertThat(actual = evaluate(content = "3")).isEqualTo(3)
assertThat(actual = evaluate(content = "3+3+3")).isEqualTo(9)
}
@Test Jesse Wilson
fun veryRecursive() = runTest {
assertThat(actual = evaluate(content = "1+".repeat(n = 1234) + "0")).isEqualTo(1234)
assertThat(actual = evaluate(content = "1+".repeat(n = 12345 * 4) + "0")).isEqualTo(12345 * 4)
}
}
WITHOUT DEBUGGER
Run
EvaluateTest
Test Results 191 ms
EvaluateTest 191 ms
happyPath 41 ms
veryRecursive 150 ms
SUPER FAST
2 tests passed 2 tests total, 191 ms
Reusing configuration cache.
> Task :lib:checkKotlinGradlePluginConfigurationErrors SKIPPED
> Task :lib:processResources NO-SOURCE
> Task :lib:processTestResources NO-SOURCE
> Task :lib:compileKotlin UP-TO-DATE
> Task :lib:compileJava NO-SOURCE
> Task :lib:classes UP-TO-DATE
> Task :lib:jar UP-TO-DATE
> Task :lib:compileTestKotlin UP-TO-DATE
> Task :lib:compileTestJava NO-SOURCE
> Task :lib:testClasses UP-TO-DATE
> Task :lib:test
BUILD SUCCESSFUL in 492ms
coroutines-party-tricks › code › lib › src › test › kotlin › com › publicobject › tricks › stacks › EvaluateTest › veryRecursive
19:4 LF UTF-8 2 spaces*

Quadratic Execution Time?

IntelliJ synthesizes stack traces for coroutines

- Only in the debugger
- Only when it’s launched from IntelliJ

This is a great feature

Absolutely not a performance problem in practice

Advice

Don’t use `yield()` to get smaller stack frames

You will go to programmer jail

Check out `DeepRecursiveFunction` if you need that

TRICK #5

Sequences

Let’s Scan the FileSystem

Goals:

- **Recursive** - return all the paths in a subtree, ‘depth-first’
- **Streaming** - the file system could be huge! We should return some results before we’ve visited the whole thing
- **Simple** - both as a user and as implementor

Java's API

```
@Test
fun scanFileSystem() {
  for (path in Files.walk(Paths.get("."))) {
    println(path)
  }
}
```

Java's Implementation

```
class FileTreeWalker implements Closeable {
  private final boolean followLinks;
  private final LinkOption[] linkOptions;
  private final int maxDepth;
  private final ArrayDeque<DirectoryNode> stack = new ArrayDeque<>();
  private boolean closed;

  /**
   * The element on the walking stack corresponding to a directory node.
   */
  private static class DirectoryNode {
    private final Path dir;
    private final Object key;
    private final DirectoryStream<Path> stream;
    private final Iterator<Path> iterator;
    private boolean skipped;

    DirectoryNode(Path dir, Object key, DirectoryStream<Path> stream) {
```

Java's Implementation

```
   */
  boolean isOpen() {
    return !closed;
  }

  /**
   * Closes/pops all directories on the stack.
   */
  @Override
  public void close() {
    if (!closed) {
      while (!stack.isEmpty()) {
        pop();
      }
      closed = true;
    }
  }
}
```

Java's is complex

Lots of features:

- Symlinks
- Streaming contents of directories

But also lots of computer science:

- Needs to keep state for the position in each parent directory
- Uses an `ArrayDeque<DirectoryNode>`

Sequence API

```
@Test
fun sequence() {
  for (path in myFileSystem.listRecursively(".".toPath())) {
    println(path)
  }
}
```

Sequence Implementation

```
interface FileSystem {

  fun listOrNull(dir: Path): List<Path>?

  // ...



  fun listRecursively(dir: Path): Sequence<Path> {
    return sequence {
      for (child in listOrNull(dir)!!) {
        collectRecursively(child)
      }
    }
  }

  private suspend fun SequenceScope<Path>.collectRecursively(path: Path) {
    yield(path)
```

Sequence Implementation

```
fun listRecursively(dir: Path): Sequence<Path> {
  return sequence {
    for (child in listOrNull(dir)!!) {
      collectRecursively(child)
    }
  }
}

private suspend fun SequenceScope<Path>.collectRecursively(path: Path) {
  yield(path)
  val children = listOrNull(path) ?: return
  for (child in children) {
    collectRecursively(child)
  }
}
```

Sequence Implementation

```
fun listRecursively(dir: Path): Sequence<Path> {
  return sequence {
    for (child in listOrNull(dir)!!) {
      collectRecursively(child)
    }
  }
}

private suspend fun SequenceScope<Path>.collectRecursively(path: Path) {
  yield(path)
  val children = listOrNull(path) ?: return
  for (child in children) {
    collectRecursively(child)
  }
}
```

Sequence Implementation

```
fun listRecursively(dir: Path): Sequence<Path> {
  return sequence {
    for (child in listOrNull(dir)!!) {
      collectRecursively(child)
    }
  }
}

private suspend fun SequenceScope<Path>.collectRecursively(path: Path) {
  yield(path)
  val children = listOrNull(path) ?: return
  for (child in children) {
    collectRecursively(child)
  }
}
}
```

Sequence Implementation

```
fun listRecursively(dir: Path): Sequence<Path> {
  return sequence {
    for (child in listOrNull(dir)!!) {
      collectRecursively(child)
    }
  }
}

private suspend fun SequenceScope<Path>.collectRecursively(path: Path) {
  yield(path)
  val children = listOrNull(path) ?: return
  for (child in children) {
    collectRecursively(child)
  }
}
```

Sequence Implementation

```
fun listRecursively(dir: Path): Sequence<Path> {
  return sequence {
    for (child in listOrNull(dir)!!) {
      collectRecursively(child)
    }
  }
}

private suspend fun SequenceScope<Path>.collectRecursively(path: Path) {
  yield(path)
  val children = listOrNull(path) ?: return
  for (child in children) {
    collectRecursively(child)
  }
}
```

Sequence Implementation

```
fun listRecursively(dir: Path): Sequence<Path> {
  return sequence {
    for (child in listOrNull(dir)!!) {
      collectRecursively(child)
    }
  }
}

private suspend fun SequenceScope<Path>.collectRecursively(path: Path) {
  yield(path)
  val children = listOrNull(path) ?: return
  for (child in children) {
    collectRecursively(child)
  }
}
```

That’s it?!

We use the call stack to track where we are in the traversal

Kotlin’s `sequence {}` API manages that call stack between yielded values

It’s Restricted!

```kotlin
@RestrictsSuspension
@SinceKotlin("1.3")
public abstract class SequenceScope<in T> {

  public abstract suspend fun yield(value: T)

  public abstract suspend fun yieldAll(iterator: Iterator<T>)

  public suspend fun yieldAll(elements: Iterable<T>)

  public suspend fun yieldAll(sequence: Sequence<T>)

}
```

Advice

Do use `SequenceScope.yield()` to implement generators

You will not go to programmer jail

TRICK #6

A Small Fast Map

I Love Hash Maps

```
val map = HashMap<String, String>()
map["M"] = "Mushrooms"
map["H"] = "Ham"
println(map["M"])
```

Easy put & get API

Scales from a couple elements to millions

Hash Table Refresher!

Kotlin's `mapOf()` and `mutableMapOf()` functions return implementations of the hash table data structure

It's general-purpose: it'll efficiently store 5 elements or 5,000,000

Kotlin's implementations are different on different platforms

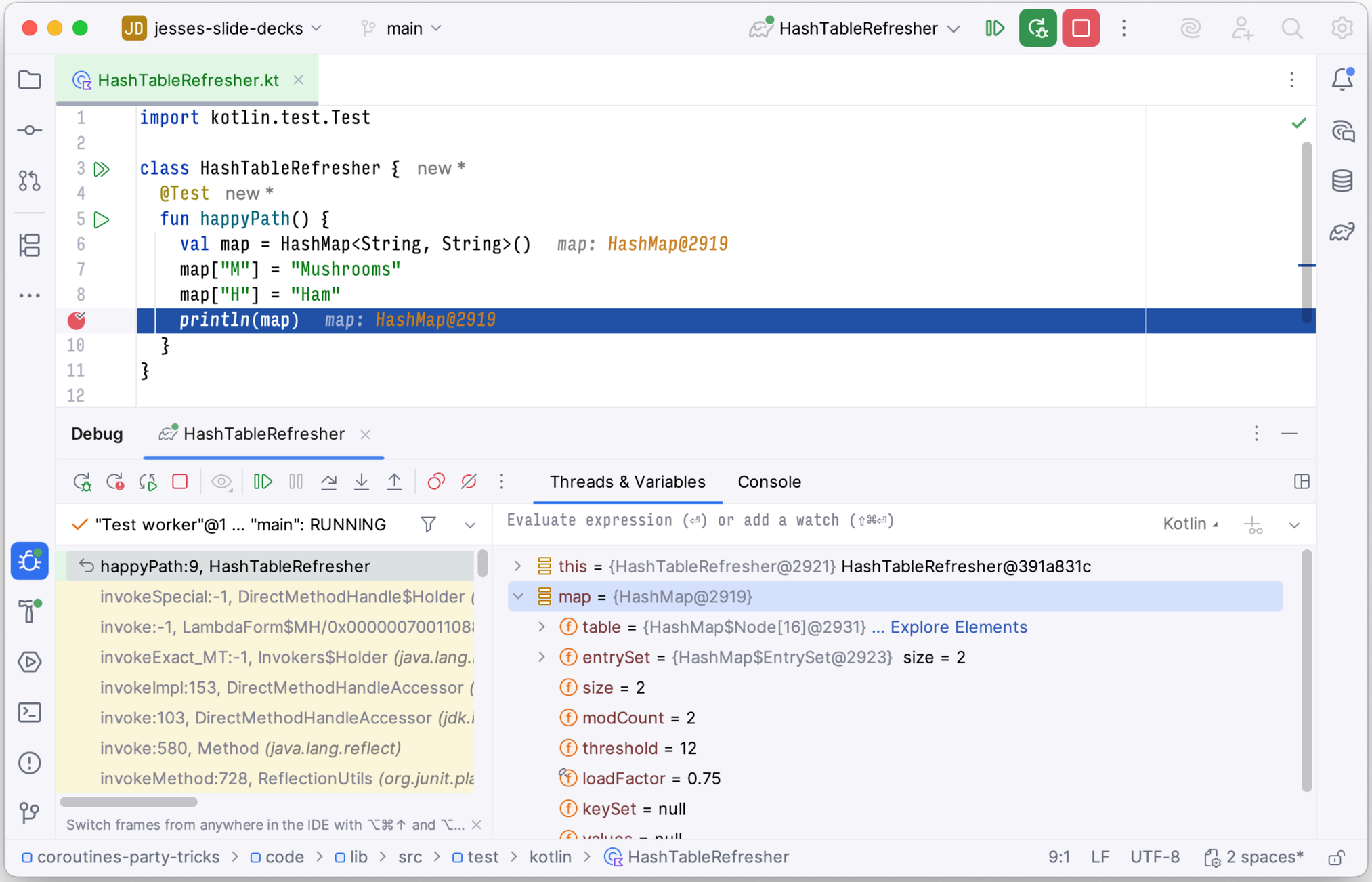
The Computer Science

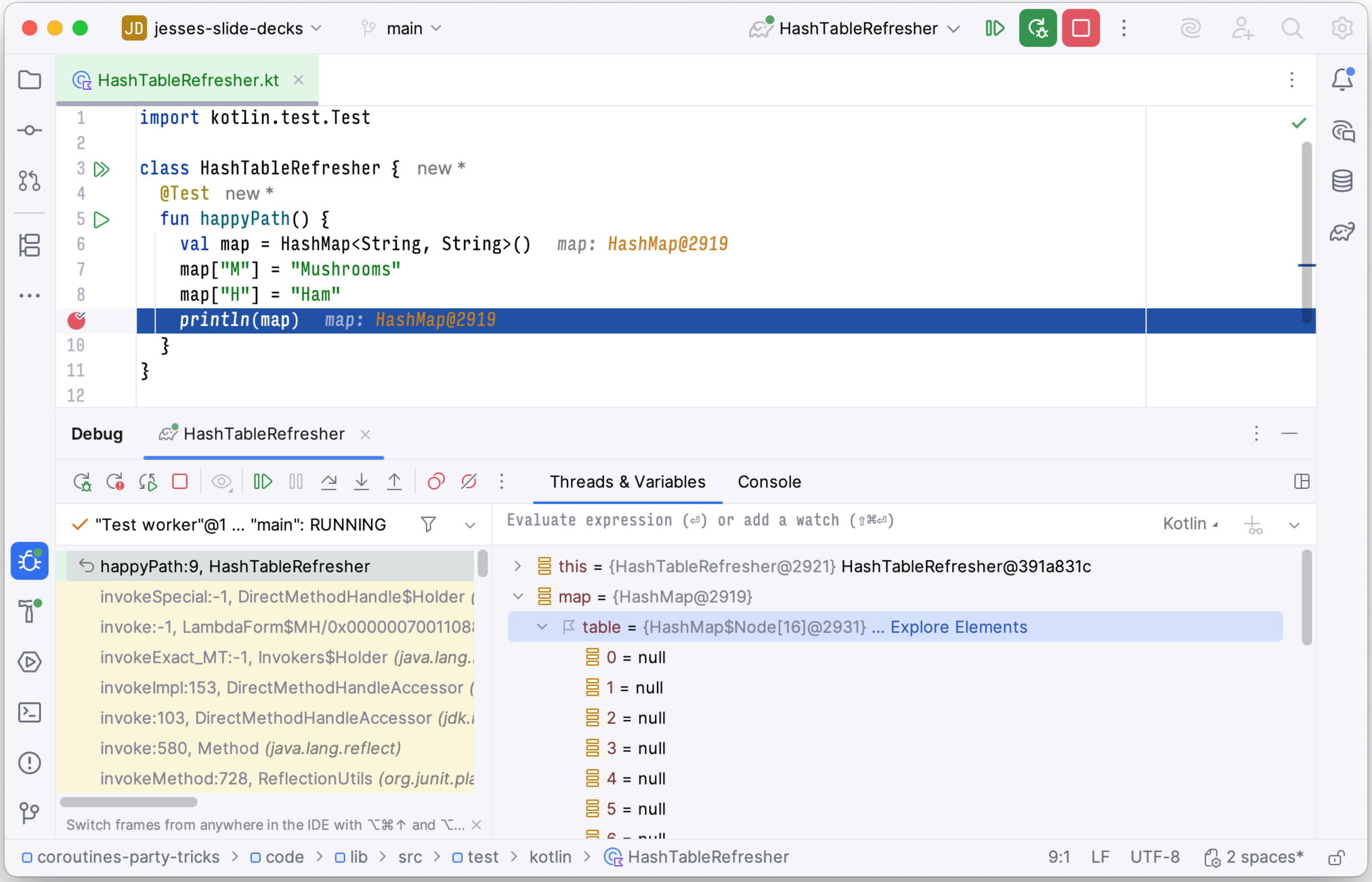
Every value has a `hashCode()` function that returns an `Int`

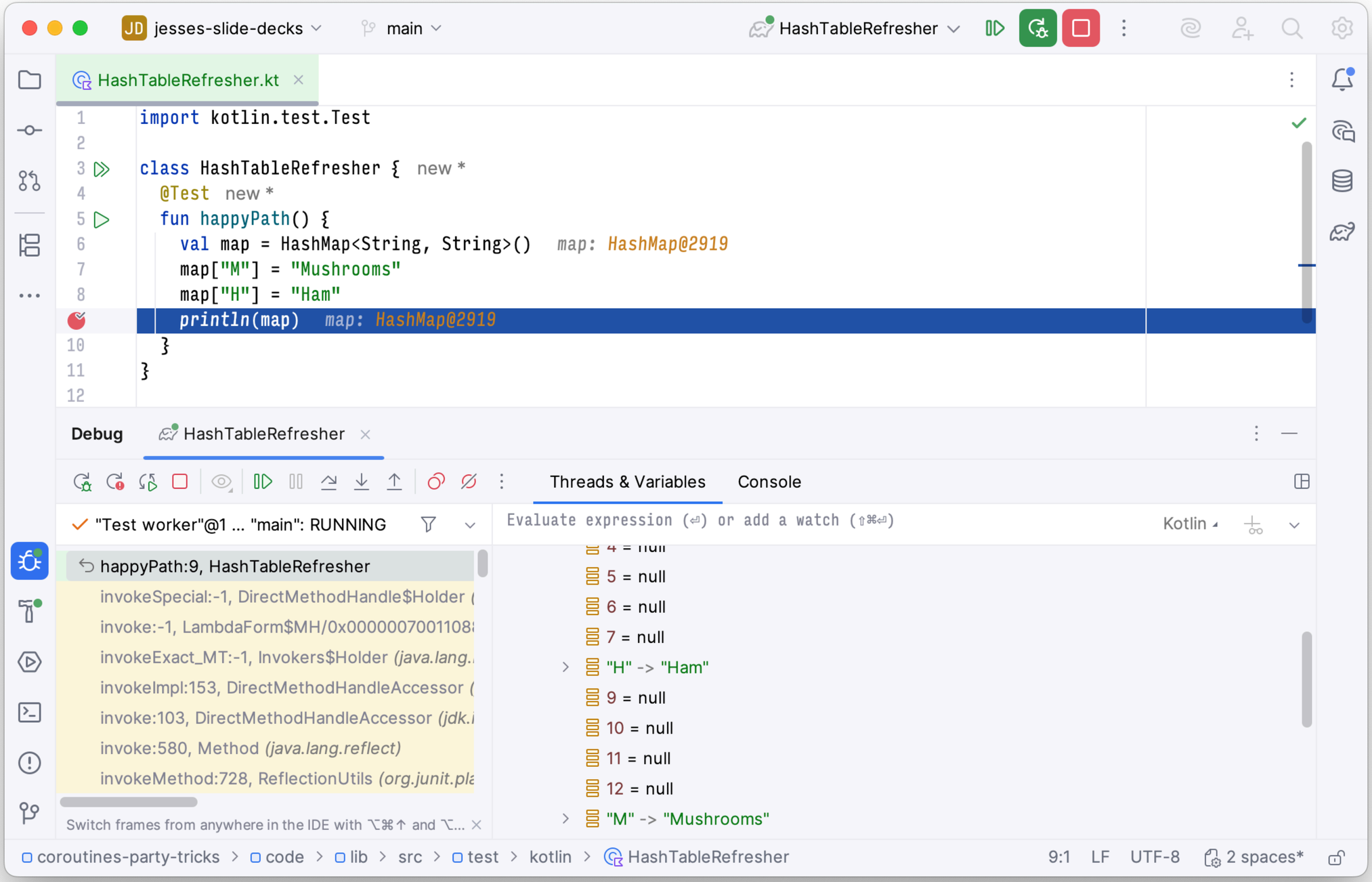
- If two values are equal, they **must** return the same `Int`
- If two values are different, they **should** return different `Int`s

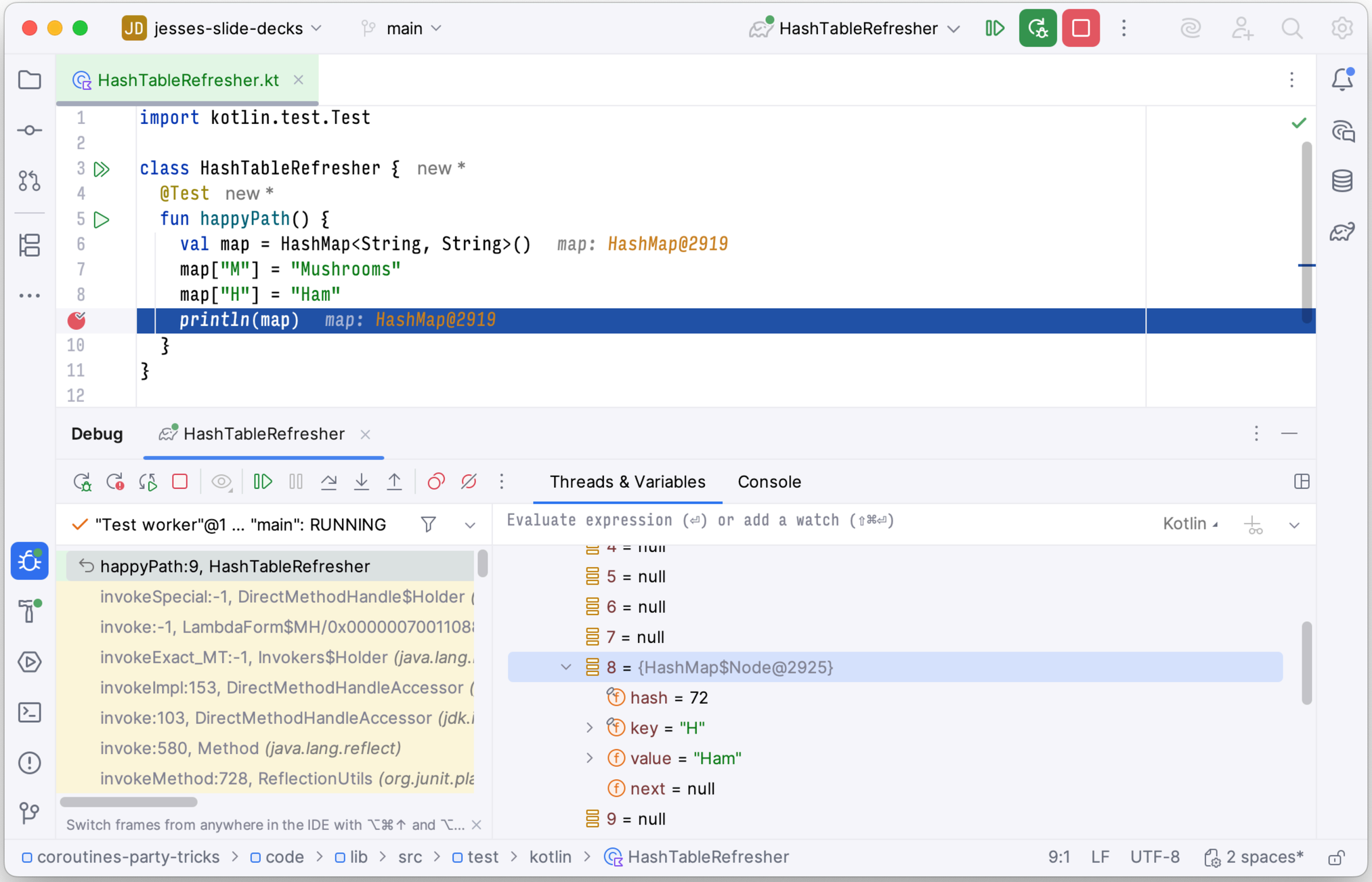
Use the `Int` to pick an index in the hash array

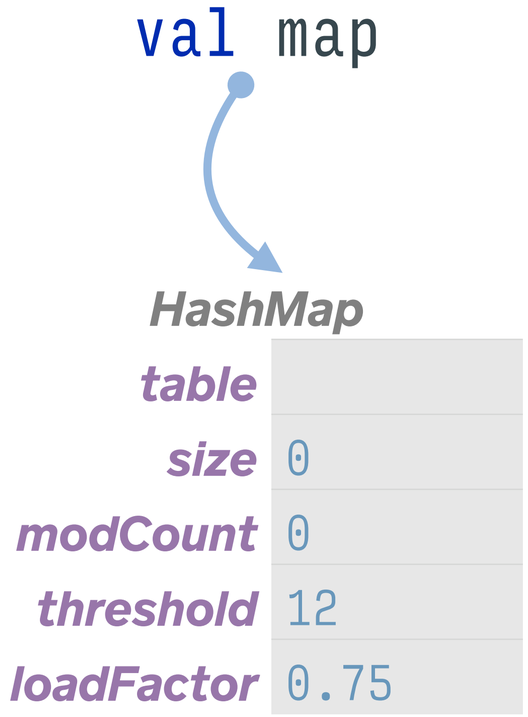
Lookups are fast ’cause we’ll often find our item immediately


jesses-slide-decks
main
HashTableRefresher
HashTableRefresher.kt
1 import kotlin.test.Test
2
3 class HashTableRefresher { new *
4 @Test new *
5 fun happyPath() {
6 val map = HashMap<String, String>() map: HashMap@2919
7 map["M"] = "Mushrooms"
8 map["H"] = "Ham"
9 println(map) map: HashMap@2919
10 }
11 }
12
Debug
HashTableRefresher
Threads & Variables
Console
"Test worker"@1 ... "main": RUNNING
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
Kotlin
happyPath:9, HashTableRefresher
invokeSpecial:-1, DirectMethodHandle$Holder
invoke:-1, LambdaForm$MH/0x0000007001108
invokeExact_MT:-1, Invokers$Holder (java.lang.
invokeImpl:153, DirectMethodHandleAccessor
invoke:103, DirectMethodHandleAccessor (jdk.i
invoke:580, Method (java.lang.reflect)
invokeMethod:728, ReflectionUtils (org.junit.pla
Switch frames from anywhere in the IDE with ⌥⌘↑ and ⌥...
this = {HashTableRefresher@2921} HashTableRefresher@391a831c
map = {HashMap@2919}
table = {HashMap$Node[16]@2931} ... Explore Elements
entrySet = {HashMap$EntrySet@2923} size = 2
size = 2
modCount = 2
threshold = 12
loadFactor = 0.75
keySet = null
coroutines-party-tricks
code
lib
src
test
kotlin
HashTableRefresher
9:1
LF
UTF-8
2 spaces*

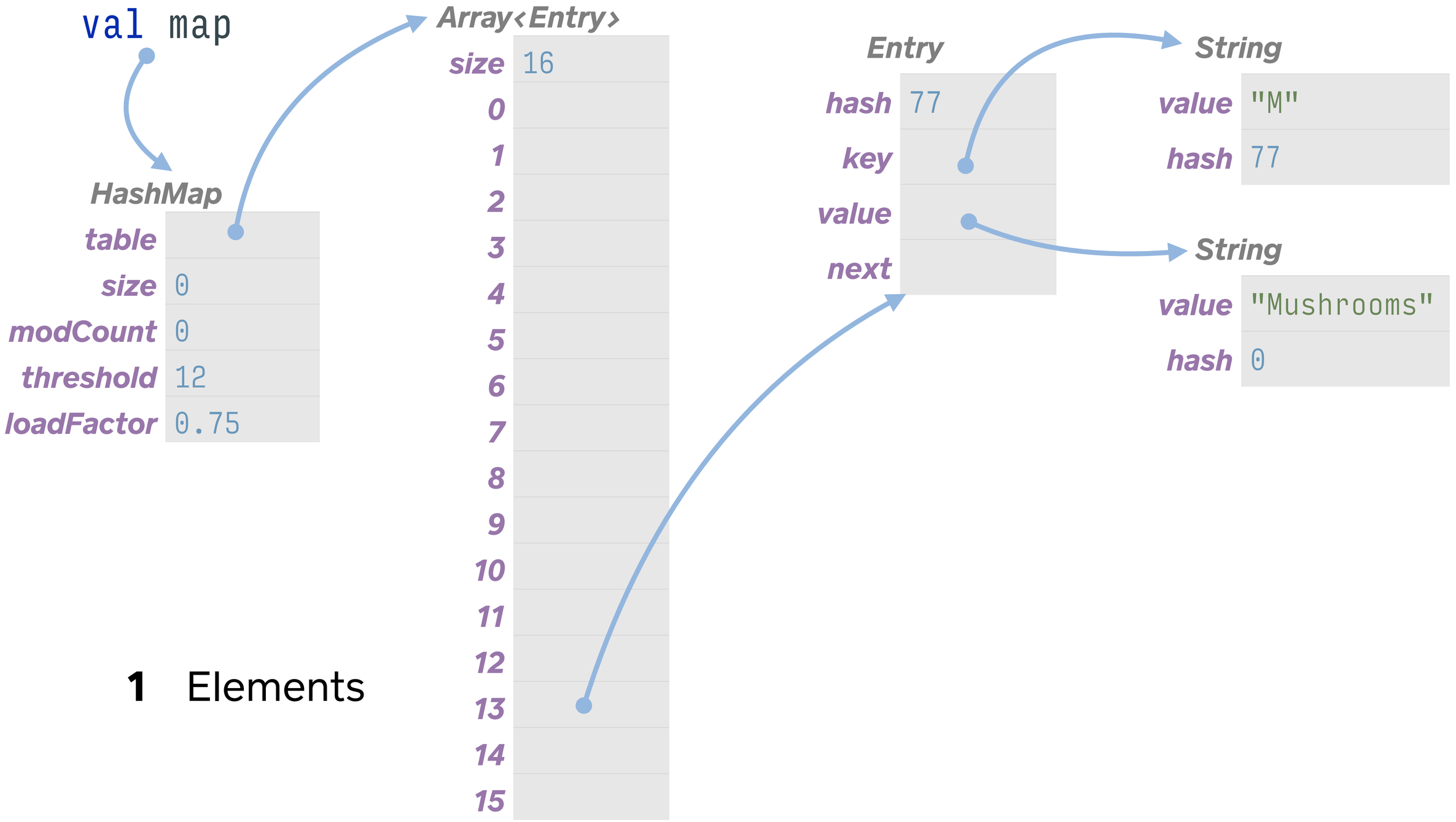

jesses-slide-decks
main
HashTableRefresher
HashTableRefresher.kt
1 import kotlin.test.Test
2
3 class HashTableRefresher { new *
4 @Test new *
5 fun happyPath() {
6 val map = HashMap<String, String>() map: HashMap@2919
7 map["M"] = "Mushrooms"
8 map["H"] = "Ham"
9 println(map) map: HashMap@2919
10 }
11 }
12
Debug
HashTableRefresher
Threads & Variables
Console
"Test worker"@1 ... "main": RUNNING
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
Kotlin
happyPath:9, HashTableRefresher
invokeSpecial:-1, DirectMethodHandle$Holder
invoke:-1, LambdaForm$MH/0x0000007001108
invokeExact_MT:-1, Invokers$Holder (java.lang.
invokeImpl:153, DirectMethodHandleAccessor
invoke:103, DirectMethodHandleAccessor (jdk.i
invoke:580, Method (java.lang.reflect)
invokeMethod:728, ReflectionUtils (org.junit.pla
Switch frames from anywhere in the IDE with ⌥⌘↑ and ⌥...
this = {HashTableRefresher@2921} HashTableRefresher@391a831c
map = {HashMap@2919}
table = {HashMap$Node[16]@2931} ... Explore Elements
0 = null
1 = null
2 = null
3 = null
4 = null
5 = null
coroutines-party-tricks
code
lib
src
test
kotlin
HashTableRefresher
9:1
LF
UTF-8
2 spaces*


jesses-slide-decks
main
HashTableRefresher
HashTableRefresher.kt
import kotlin.test.Test
class HashTableRefresher { new *
@Test new *
fun happyPath() {
val map = HashMap<String, String>() map: HashMap@2919
map["M"] = "Mushrooms"
map["H"] = "Ham"
println(map) map: HashMap@2919
}
}
Debug
HashTableRefresher
Threads & Variables
Console
"Test worker"@1 ... "main": RUNNING
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
Kotlin
happyPath:9, HashTableRefresher
invokeSpecial:-1, DirectMethodHandle$Holder
invoke:-1, LambdaForm$MH/0x0000007001108
invokeExact_MT:-1, Invokers$Holder (java.lang.
invokeImpl:153, DirectMethodHandleAccessor
invoke:103, DirectMethodHandleAccessor (jdk.i
invoke:580, Method (java.lang.reflect)
invokeMethod:728, ReflectionUtils (org.junit.pla
5 = null
6 = null
7 = null
"H" -> "Ham"
9 = null
10 = null
11 = null
12 = null
"M" -> "Mushrooms"
Switch frames from anywhere in the IDE with ⌥⌘↑ and ⌥...
coroutines-party-tricks
code
lib
src
test
kotlin
HashTableRefresher
9:1
LF
UTF-8
2 spaces*

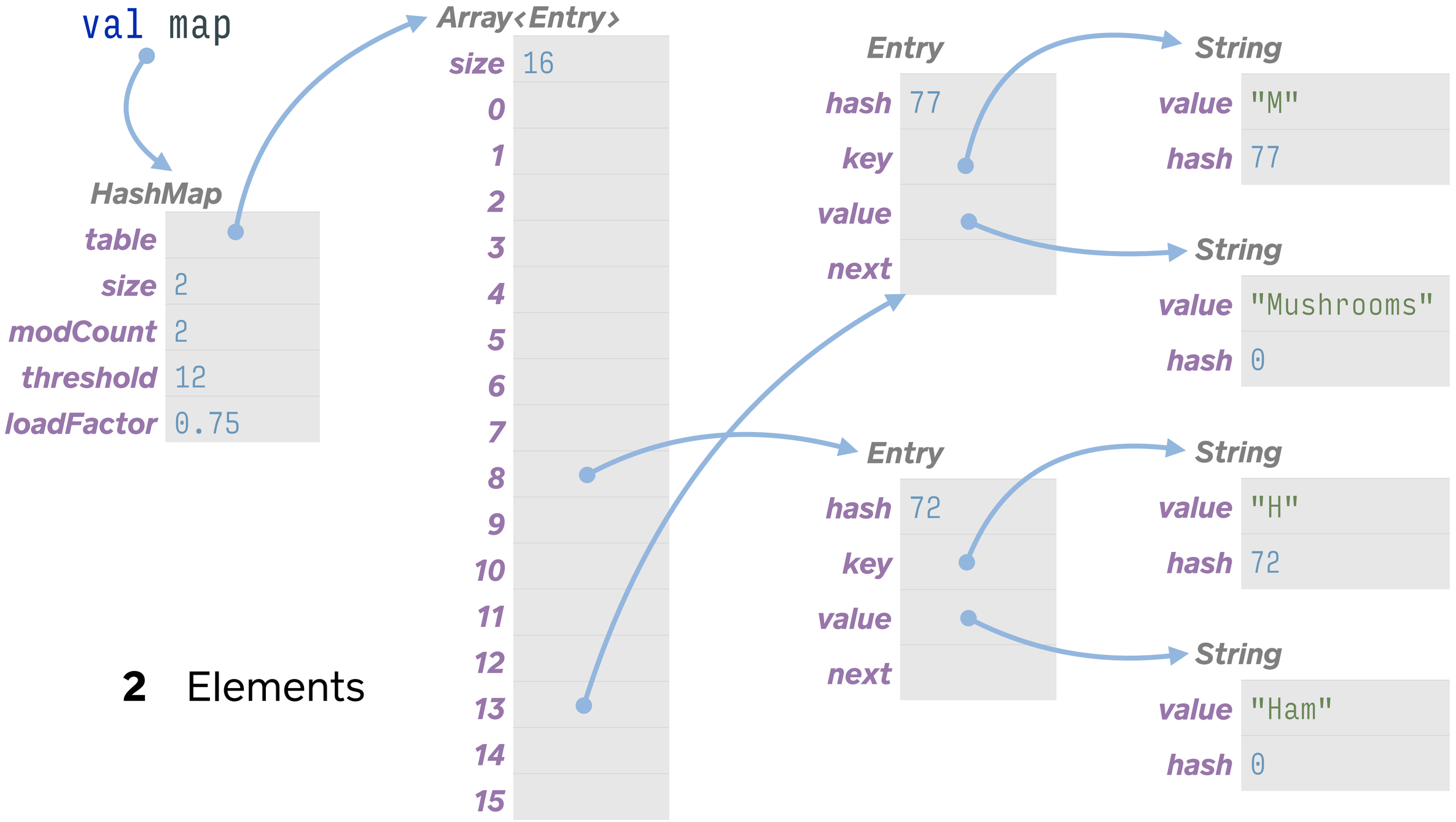

jesses-slide-decks
main
HashTableRefresher
HashTableRefresher.kt
1 import kotlin.test.Test
2
3 class HashTableRefresher { new *
4 @Test new *
5 fun happyPath() {
6 val map = HashMap<String, String>() map: HashMap@2919
7 map["M"] = "Mushrooms"
8 map["H"] = "Ham"
9 println(map) map: HashMap@2919
10 }
11 }
12
Debug
HashTableRefresher
Threads & Variables
Console
"Test worker"@1 ... "main": RUNNING
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
Kotlin
happyPath:9, HashTableRefresher
invokeSpecial:-1, DirectMethodHandle$Holder
invoke:-1, LambdaForm$MH/0x0000007001108
invokeExact_MT:-1, Invokers$Holder (java.lang.
invokeImpl:153, DirectMethodHandleAccessor
invoke:103, DirectMethodHandleAccessor (jdk.i
invoke:580, Method (java.lang.reflect)
invokeMethod:728, ReflectionUtils (org.junit.pla
Switch frames from anywhere in the IDE with ⌥⌘↑ and ⌥...
5 = null
6 = null
7 = null
8 = {HashMap$Node@2925}
hash = 72
key = "H"
value = "Ham"
next = null
9 = null
coroutines-party-tricks
code
lib
src
test
kotlin
HashTableRefresher
9:1
LF
UTF-8
2 spaces*


val map
HashMap
table
size 0
modCount 0
threshold 12
loadFactor 0.75

0 Elements


val map
HashMap
table
size 0
modCount 0
threshold 12
loadFactor 0.75
Array<Entry>
size 16
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Entry
hash 77
key
value
next
String
value "M"
hash 77
String
value "Mushrooms"
hash 0

1 Elements


val map
HashMap
table
size 2
modCount 2
threshold 12
loadFactor 0.75
Array<Entry>
size 16
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Entry
hash 77
key
value
next
String
value "M"
hash 77
String
value "Mushrooms"
hash 0
Entry
hash 72
key
value
next
String
value "H"
hash 72
String
value "Ham"
hash 0

2 Elements

Memory Overhead

The `HashMap` object

The `Array<Entry>` that's between 33% and 167% larger than *N*

N `Entry` objects

Classes like `String` have extra fields to cache hash codes

Compute Overhead

Each `get()` accesses 4 objects (map, entries array, entry, key)

Computing the hash code needs to traverse the entire key object

- There's a reason this is cached!

Checking `equals()` needs to traverse two objects

Extra garbage collection!

Hash Tables are Great

They are fast actually

Use them everywhere

Coroutines' implementation details are weird and extremely performance-sensitive

CoroutineContext is Map-like

```
interface CoroutineContext {
  operator fun plus(context: CoroutineContext): CoroutineContext

  operator fun <E : Element> get(key: Key<E>): E?
}
```

Uses `plus()` instead of `put()`

Returns a new instance rather than mutating

Really Map-Like

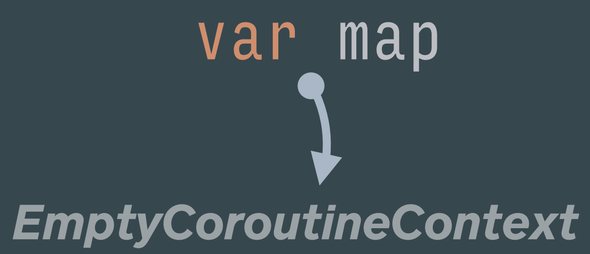
```
fun doMapStuff() {
  var map: CoroutineContext = EmptyCoroutineContext
  map += Topping(Topping.Key("M"), "Mushrooms")
  map += Topping(Topping.Key("H"), "Ham")
  println(map[Topping.Key("M")])
}

data class Topping(
  override val key: Key,
  val name: String
) : CoroutineContext.Element {
  data class Key(val letter: String): CoroutineContext.Key<Topping>
}
```

3 Kinds

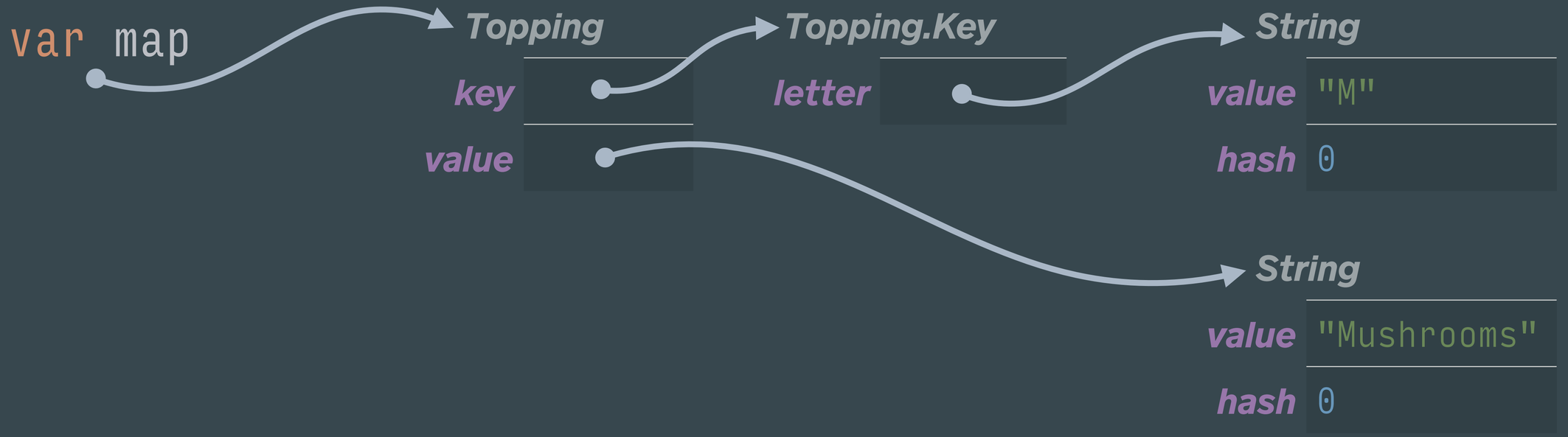
0 `EmptyCoroutineContext`

1 `CoroutineContext.Element`
Includes `CoroutineDispatcher`, `CoroutineName`, `CoroutineExceptionHandler`, `Job`, etc.

1 + N `CombinedContext`



0 Elements


var map
Topping
key
value
Topping.Key
letter
String
value
"M"
hash
0
String
value
"Mushrooms"
hash
0

1 Elements


var map
CombinedContext
left
element
Topping
key
value
Topping.Key
letter
String
value
"M"
hash
0
String
value
"Mushrooms"
hash
0
Topping
key
value
Topping.Key
letter
String
value
"H"
hash
0
String
value
"Ham"
hash
0

2 Elements


var map
CombinedContext
left
element
CombinedContext
left
element
Topping
key
value
Topping.Key
letter
key
value
Topping
key
value
Topping.Key
letter
String
value "P"
hash 0
String
value "Pineapple"
hash 0

3 Elements

It’s a Map! It’s a Linked List!

Looking up a key in `CoroutineContext` takes *N* steps

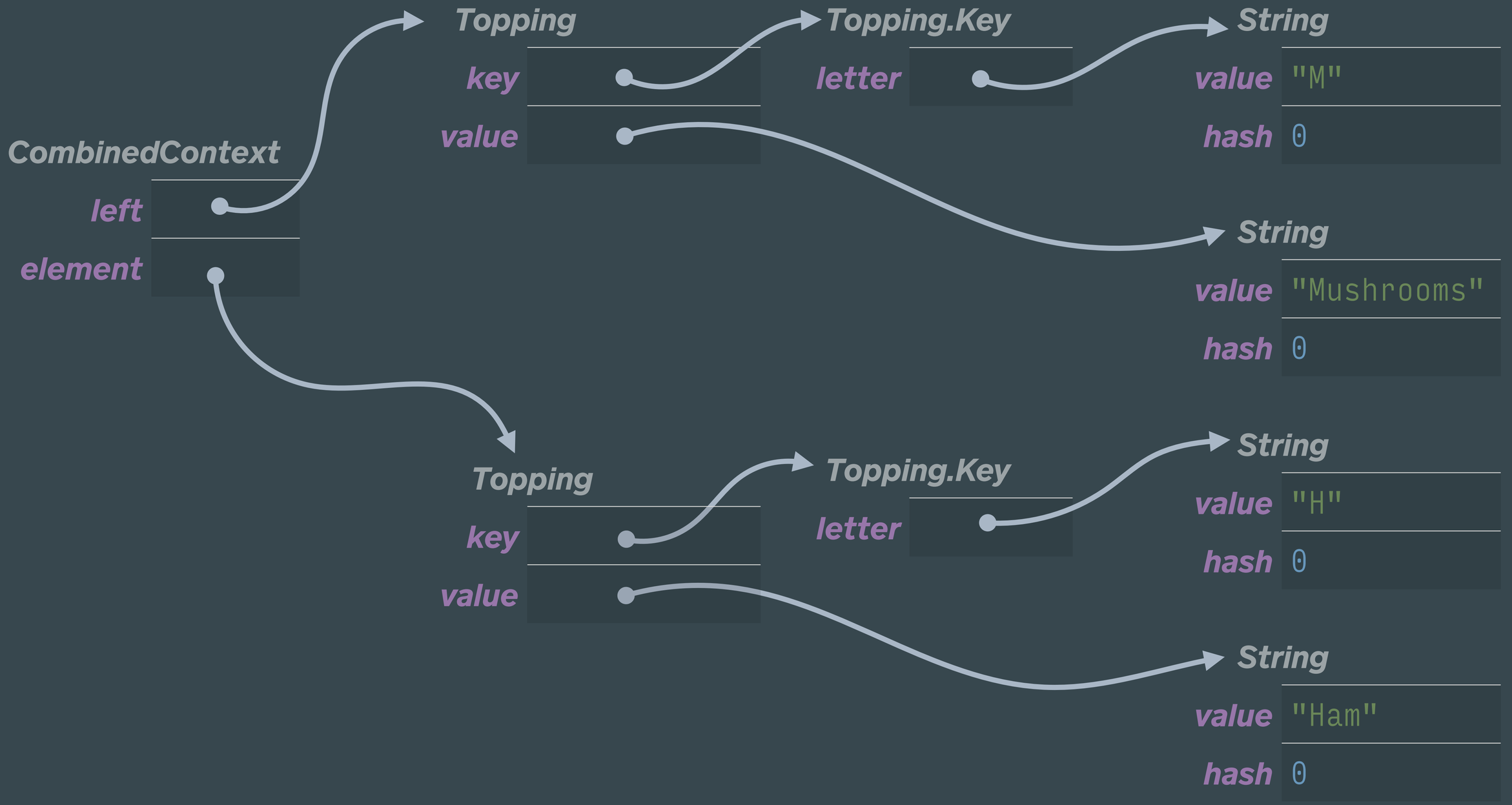
Thankfully, *N* is very small in practice

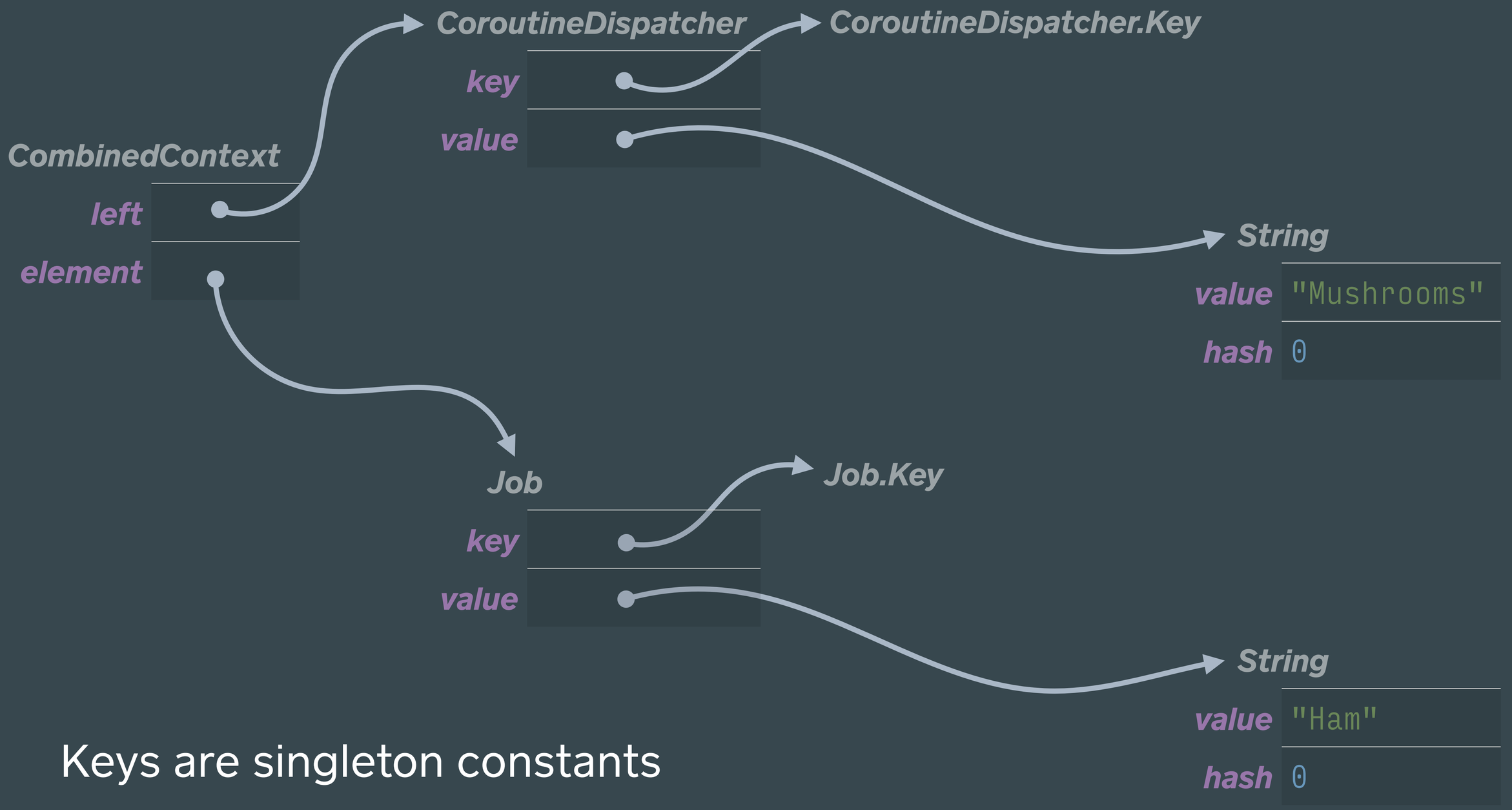
Three more optimizations!

Keys are singleton constants

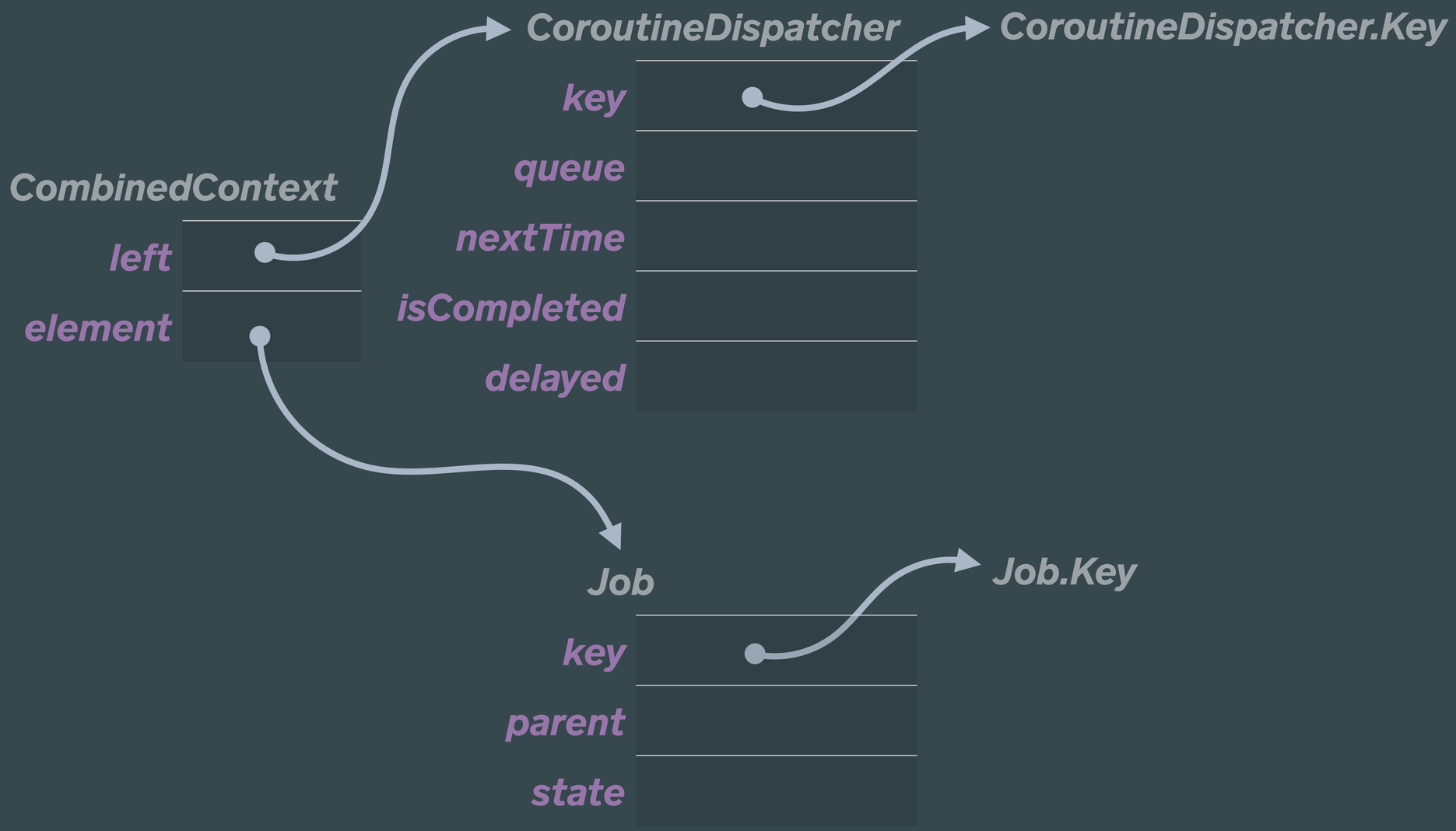
The entry and the value is the same object

The `CoroutineDispatcher` is always the head of the linked list

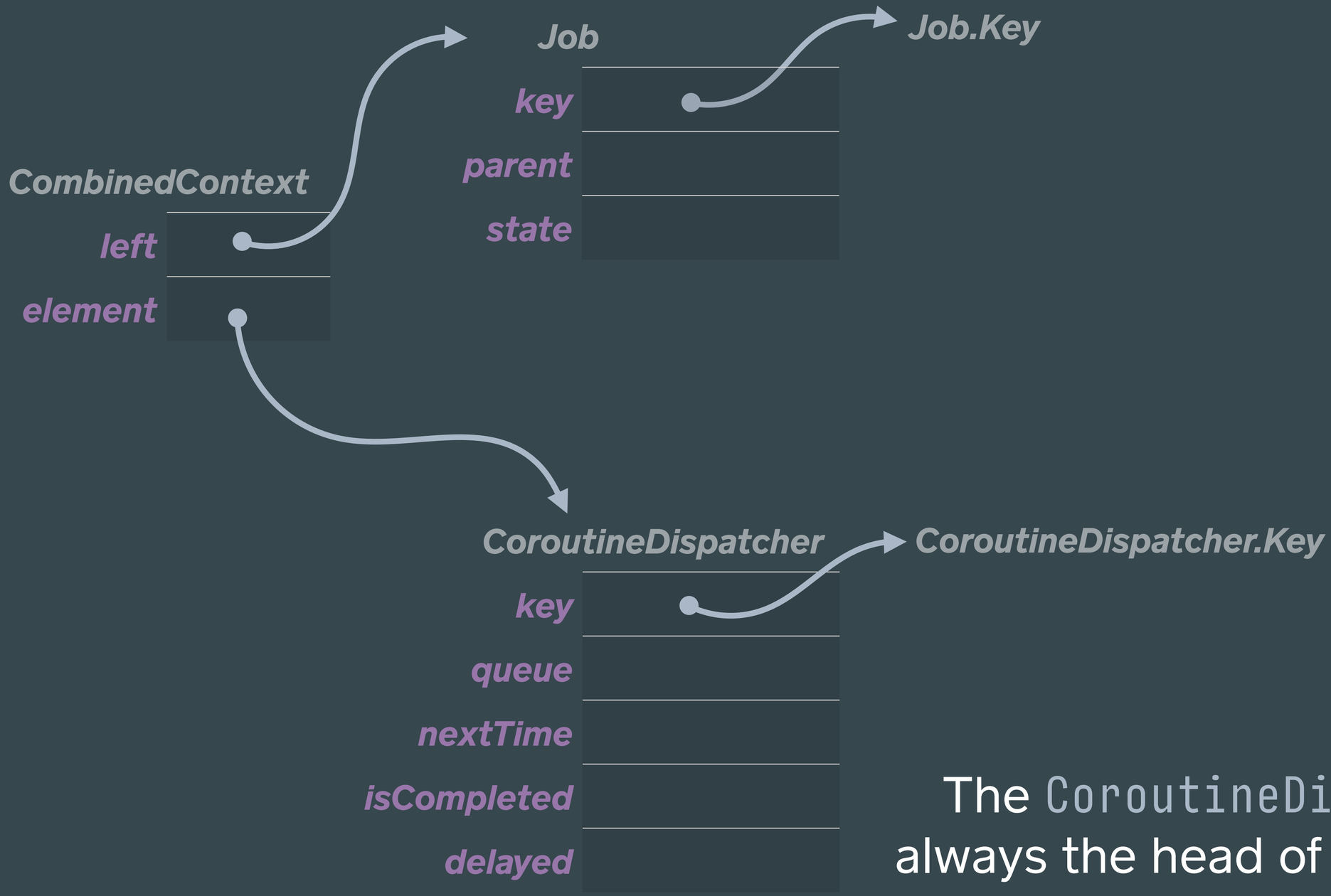

CombinedContext
left
element
Topping
key
value
Topping.Key
letter
String
value
"M"
hash
0
String
value
"Mushrooms"
hash
0
Topping
key
value
Topping.Key
letter
String
value
"H"
hash
0
String
value
"Ham"
hash
0



Keys are singleton constants



The entry and the value is the same object



The `CoroutineDispatcher` is always the head of the linked list

Memory Overhead

N - 1 `CombinedContext` objects

Compute Overhead

Each `get()` accesses up to $N \times 2$ objects

So?

`CoroutineContext` is a thoughtful & fast

It’s awesome to be a Kotlin developer

TRICK #7

Fail Gracefully

Try/Catch is Rad

We expect some operations to sometimes fail

Use try/catch for safe & easy error handling

Exception Handlers are Also Rad

Operations also fail in ways we don’t expect

Add a `CoroutineExceptionHandler` to your `CoroutineContext`

It’ll get called if the coroutine throws an exception that nobody catches

100% Conformance is Too Hard

We might not register our `CoroutineExceptionHandler` everywhere

- Perhaps we forgot!
- Perhaps it’s 3rd party code

How Java does it

Java's `Thread` has two properties:

- `uncaughtExceptionHandler`
- `defaultUncaughtExceptionHandler`, as a fallback

What about Kotlin/Multiplatform?

Sadly nothing

Unless...

```
/**
 * Installs a global coroutine exception handler using an internal API.
 */
@Suppress("INVISIBLE_MEMBER", "INVISIBLE_REFERENCE")
internal fun registerUncaughtExceptionHandler(
  coroutineExceptionHandler: CoroutineExceptionHandler,
) {
  kotlinx.coroutines.internal.ensurePlatformExceptionHandlerLoaded(
    object : CoroutineExceptionHandler by coroutineExceptionHandler {
      override fun handleException(context: CoroutineContext, exception: Throwable) {
        coroutineExceptionHandler.handleException(context, exception)
        throw kotlinx.coroutines.internal.ExceptionSuccessfullyProcessed
      }
    },
  )
}
```

Advice

Using internal APIs feels bad

- It makes it dangerous to update libraries!
- Vote this up: github.com/Kotlin/kotlinx.coroutines/issues/3978

But hard crashes feel **very** bad

TRICK #8

Navigation

Your Server is a Function

We pretend a remote API is a function we can call!

- Take a request object
- Encode it as bytes & transmit it
- Await response
- Decode it and return it

Not a perfect abstraction!

Your User is a Function

We pretend a person is a function we can call!

- Take a decision
- Encode it as pixels & display it
- Await clicks, taps, and keystrokes
- Model it and return it

Not a perfect abstraction!

Here’s a Pizza REST API

```
POST http://localhost:80/api/order-pizza
Content-Type: application/json
Accept: application/json

{
  "address": {"line_1": "Building 293, Brooklyn NY"},
  "pizzas": [{"size": "medium", "toppings": ["pineapple", "onions"]}]
}

HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": "CPT1001",
  "status": "pending",
  "eta": "2025-05026T14:03:00Z"
}
```

Here’s a Pizza REST API

```
POST http://localhost:80/api/order-pizza
Content-Type: application/json
Accept: application/json

{
  "address": {"line_1": "Building 293, Brooklyn NY"},
  "pizzas": [{"size": "medium", "toppings": ["pineapple", "onions"]}]
}

HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": "CPT1001",
  "status": "pending",
  "eta": "2025-05026T14:03:00Z"
}
```

Here's a Pizza REST API

```
POST http://localhost:80/api/order-pizza
Content-Type: application/json
Accept: application/json

{
  "address": {"line_1": "Building 293, Brooklyn NY"},
  "pizzas": [{"size": "medium", "toppings": ["pineapple", "onions"]}]
}

HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": "CPT1001",
  "status": "pending",
  "eta": "2025-05026T14:03:00Z"
}
```

PARAMETER

RETURN VALUE

Call the server like a function

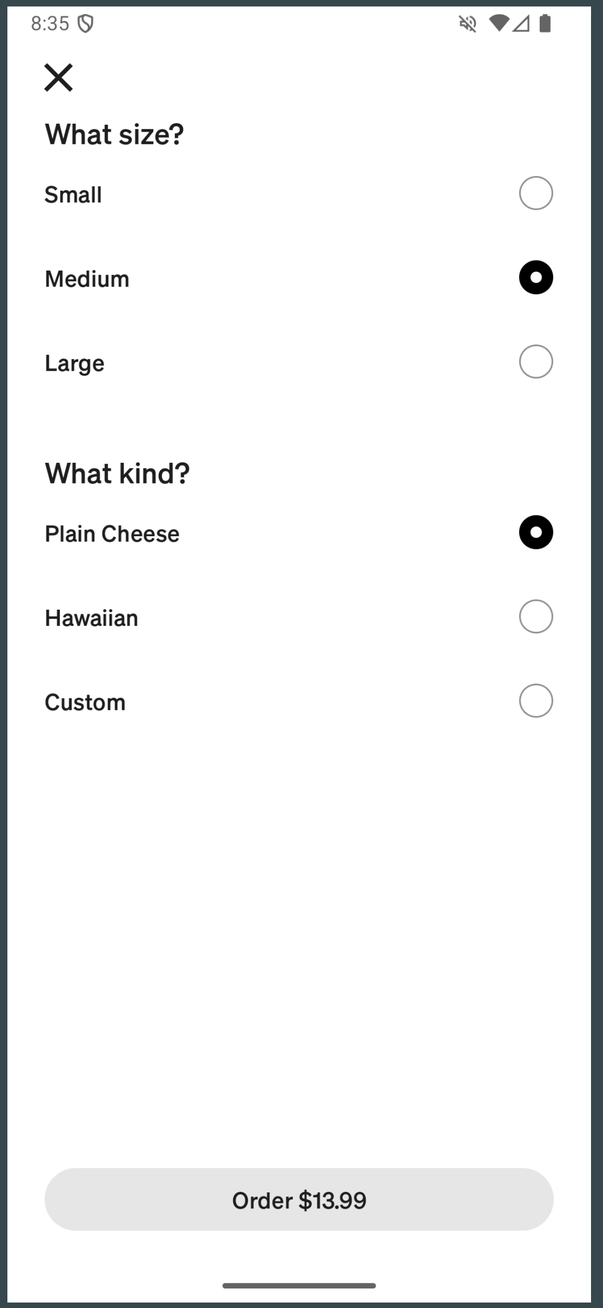
```
val orderResponse = pizzaApi.order(
  OrderRequest(
    Address("Building 293, Brooklyn NY"),
    listOf(pizza),
  )
)
```

The `order()` function suspends while the server does its thing

It returns the server’s decision

Here's a Pizza UI

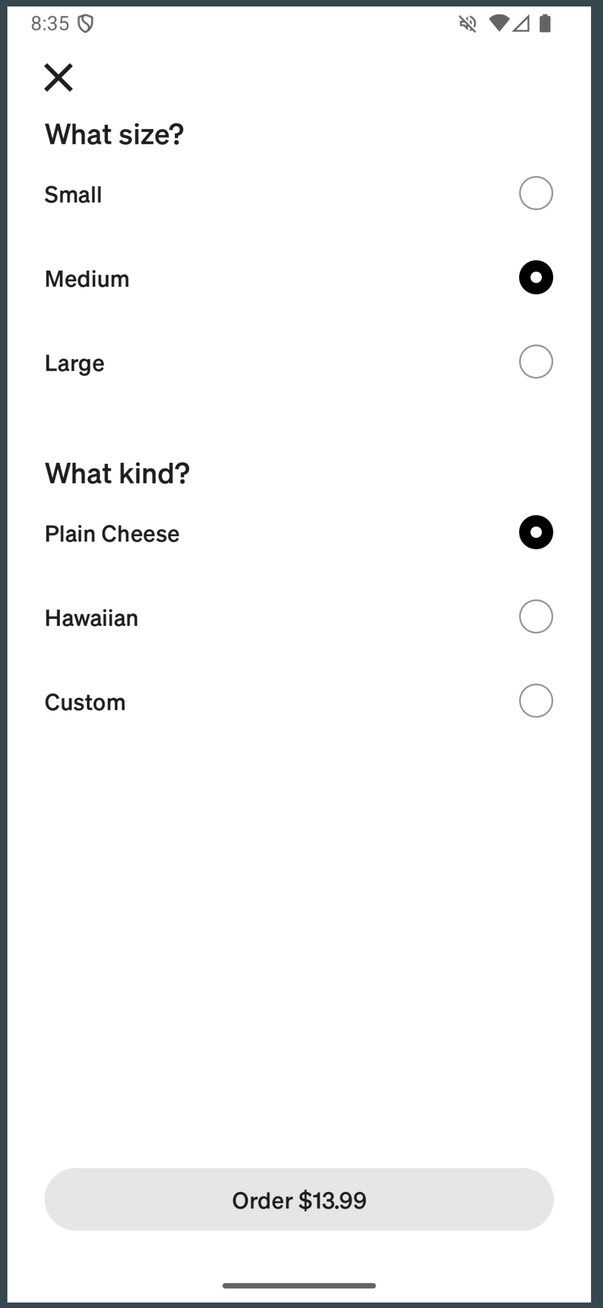
```
class GetPizzaScreen(spec: GetPizzaScreenSpec) {

  @Composable
  override fun Show(onResult: (Pizza) -> Unit) {
    FlowScreen(
      body = {
        SizeSection()
        KindSection()
      },
      footer = {
        button(
          text = "Order ${price().formattedWithCents()}",
          onClick = {
           onResult(Pizza(size.value, kind.value.toppings))
          },
        )
```



Here's a Pizza UI

PARAMETER

```
class GetPizzaScreen(spec: GetPizzaScreenSpec) {

  @Composable
  override fun Show(onResult: (Pizza) -> Unit) {
    FlowScreen(
      body = {
        SizeSection()
        KindSection()
      },
      footer = {
        button(
          text = "Order ${price().formattedWithCents()}",
          onClick = {
           onResult(Pizza(size.value, kind.value.toppings))
          },
        )
```

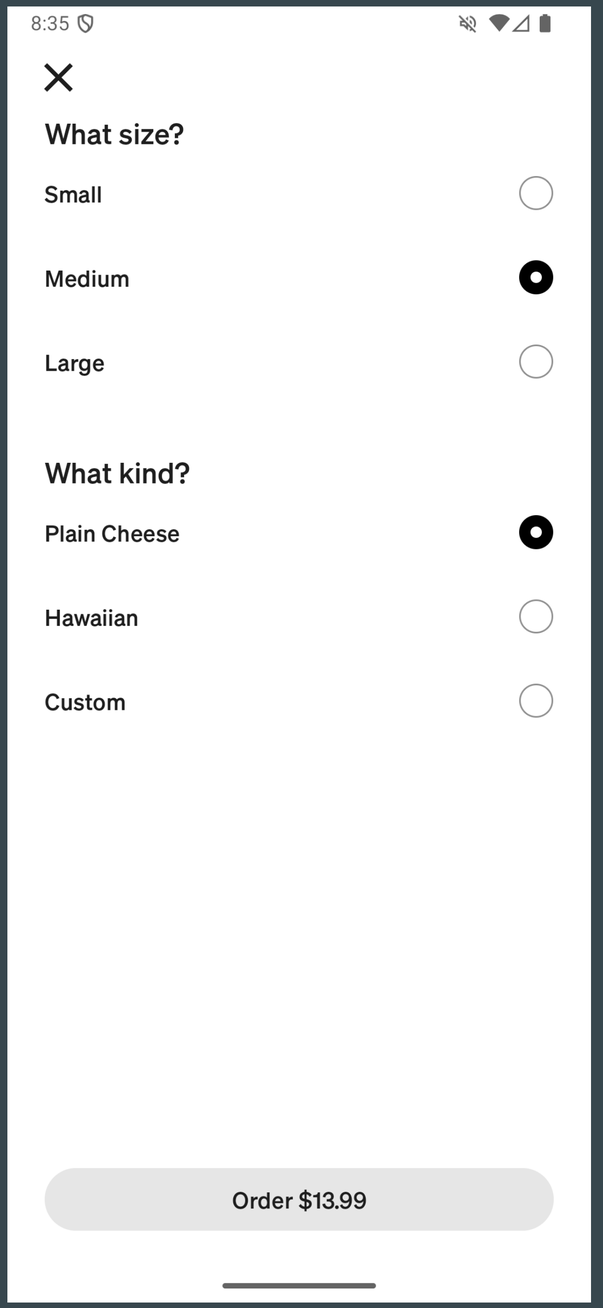


Here's a Pizza UI

PARAMETER

RETURN VALUE!

```
class GetPizzaScreen(spec: GetPizzaScreenSpec) {

  @Composable
  override fun Show(onResult: (Pizza) -> Unit) {
    FlowScreen(
      body = {
        SizeSection()
        KindSection()
      },
      footer = {
        button(
          text = "Order ${price().formattedWithCents()}",
          onClick = {
           onResult(Pizza(size.value, kind.value.toppings))
          },
        )
```



Call the user like a function!

```
val pizza = display.show(GetPizzaScreenSpec())
```

The `display.show()` call suspends while the user does her thing

It returns her decision


```
val pizza = display.show(GetPizzaScreenSpec())
```

Abstracting a server into a function call makes sense, it’s software

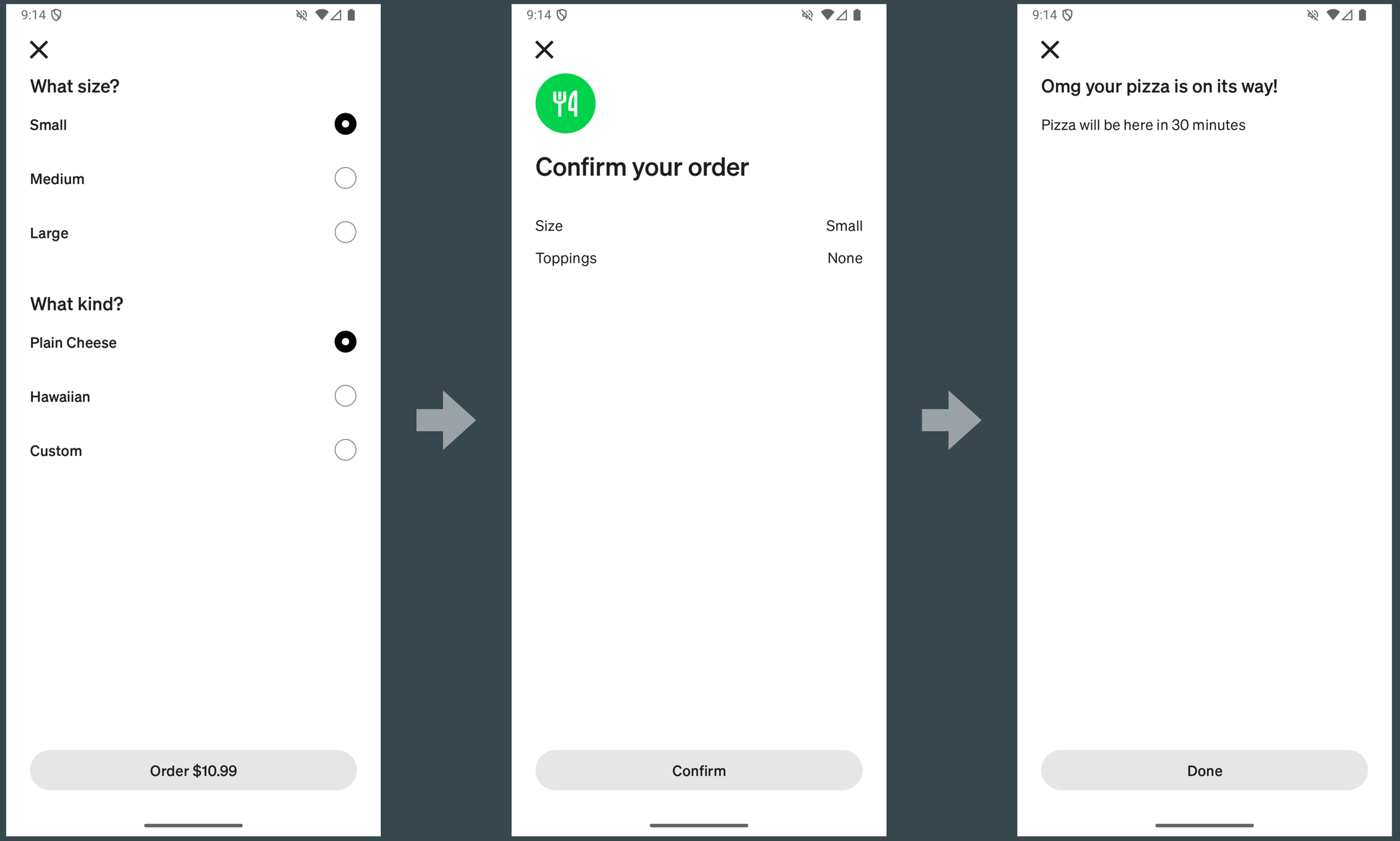
Abstracting a person into a function is a cool party trick, but why?!

It’s not just one screen!

```
override suspend fun execute(display: Display) {
  val pizza = display.show(GetPizzaScreenSpec())

  display.show(ConfirmScreenSpec(pizza))

  val orderResponse = pizzaApi.order(
    OrderRequest(pizza)
  )

  display.show(
    ResultScreenSpec(
     eta = orderResponse.eta,
    )
  )
}
```

It’s not just one screen!

```
override suspend fun execute(display: Display) {
  val pizza = display.show(GetPizzaScreenSpec())

  display.show(ConfirmScreenSpec(pizza))

  val orderResponse = pizzaApi.order(
    OrderRequest(pizza)
  )

  display.show(
    ResultScreenSpec(
     eta = orderResponse.eta,
    )
  )
}
```


9:14
What size?
Small
Medium
Large
What kind?
Plain Cheese
Hawaiian
Custom
Order $10.99
9:14
Confirm your order
Size
Small
Toppings
None
Confirm
9:14
Omg your pizza is on its way!
Pizza will be here in 30 minutes
Done

Let’s Change it Up

Let’s upsell:

- The confirm screen used to return `Unit`
- Now it’ll return either or `AddAnother` or `Proceed`

And our navigation function needs a loop

```
override suspend fun execute(display: Display) {
  var pizzas = listOf<Pizza>()

  while (true) {
    pizzas += display.show(GetPizzaScreenSpec())
    when (display.show(ConfirmScreenSpec(pizzas))) {
      AddAnother -> continue
      Proceed -> break
    }
  }

  val orderResponse = pizzaApi.order(
    OrderRequest(pizzas)
  )

  display.show(
    ResultScreenSpec(
      eta = orderResponse.eta
    )
  )
}
```

```
override suspend fun execute(display: Display) {
  var pizzas = listOf<Pizza>()

  while (true) {
    pizzas += display.show(GetPizzaScreenSpec())
    when (display.show(ConfirmScreenSpec(pizzas))) {
      AddAnother -> continue
      Proceed -> break
    }
  }

  val orderResponse = pizzaApi.order(
    OrderRequest(pizzas)
  )

  display.show(
    ResultScreenSpec(
      eta = orderResponse.eta
    )
  )
}
```

NavController

Navigation is configured as a set of edges in a graph

Navigation is separate from application data

Extremely flexible for deep links & synthetic back stacks

Awkward to test!

Functions

Navigation is what happens when you call a function to show a screen

Navigation is integrated with application data

Strictly structured - No GOTO

Easy to test

NavController

Navigation is configured as a set of edges in a graph

Navigation is separate from application data

Extremely flexible for deep links & synthetic back stacks

Awkward to test!

Functions

Navigation is what happens when you call a function to show a screen

Navigation is integrated with application data

Strictly structured - No GOTO

Easy to test

NavController

Navigation is configured as a set of edges in a graph

Navigation is separate from application data

Extremely flexible for deep links & synthetic back stacks

Awkward to test!

Functions

Navigation is what happens when you call a function to show a screen

Navigation is integrated with application data

Strictly structured - No GOTO

Easy to test

NavController

Navigation is configured as a set of edges in a graph

Navigation is separate from application data

Extremely flexible for deep links & synthetic back stacks

Awkward to test!

Functions

Navigation is what happens when you call a function to show a screen

Navigation is integrated with application data

Strictly structured - No GOTO

Easy to test

NavController

Navigation is configured as a set of edges in a graph

Navigation is separate from application data

Extremely flexible for deep links & synthetic back stacks

Awkward to test!

Functions

Navigation is what happens when you call a function to show a screen

Navigation is integrated with application data

Strictly structured - No GOTO

Easy to test

A Structured Navigation API

```
interface ScreenSpec<T>

interface Screen<T> {
  @Composable
  fun Show(onResult: (T) -> Unit)

  interface Factory {
    fun <T> create(
      spec: ScreenSpec<T>,
    ): Screen<T>?
  }
}

interface App {
  suspend fun execute(display: Display)
}
```

```
interface Display {
  suspend fun <T> show(
    spec: ScreenSpec<T>
  ): T
}

@Composable
fun Show(
  app: App,
  screenFactory: Screen.Factory,
) {
  ...
}

object LoadingScreenSpec
  : ScreenSpec<Nothing>
```

Advice

Coroutines can model business processes directly

Using coroutines for navigation is pretty fun

TRICK #9

Back Navigation

Let’s Change it Up, Again

Let’s support the back button

Using Coroutines

Back Button

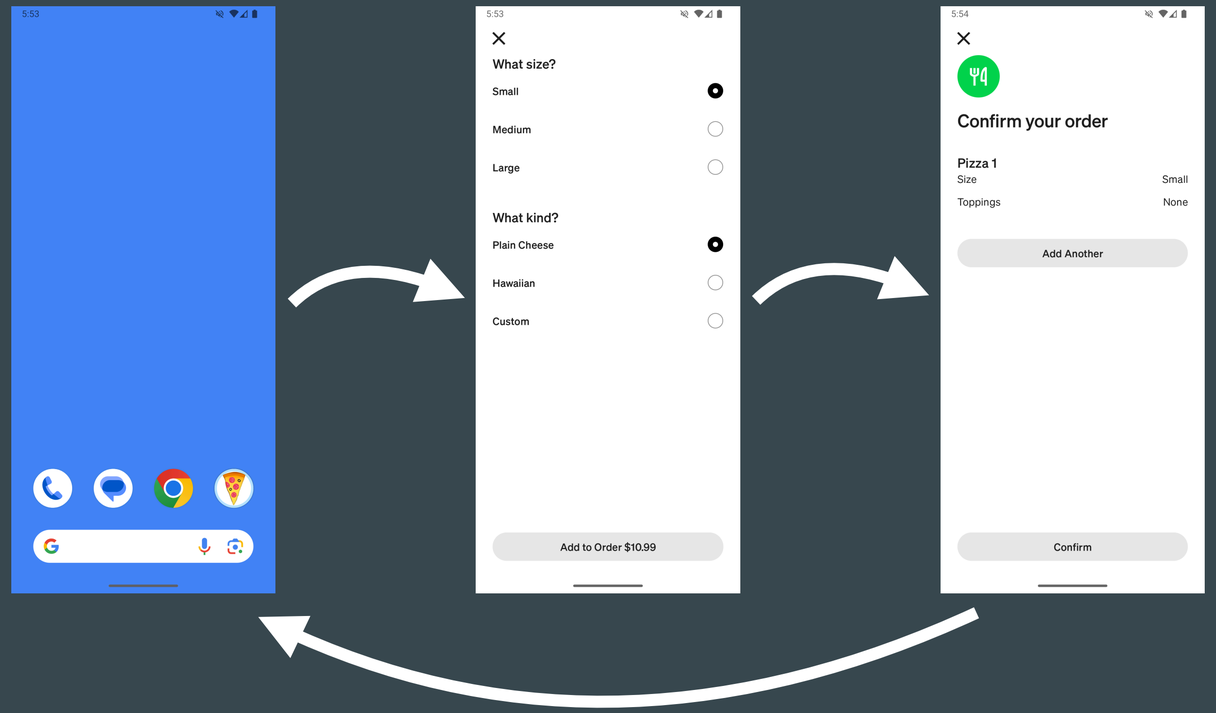
Design idea: the call stack is the back stack

Add a new `show()` overload that accepts a lambda

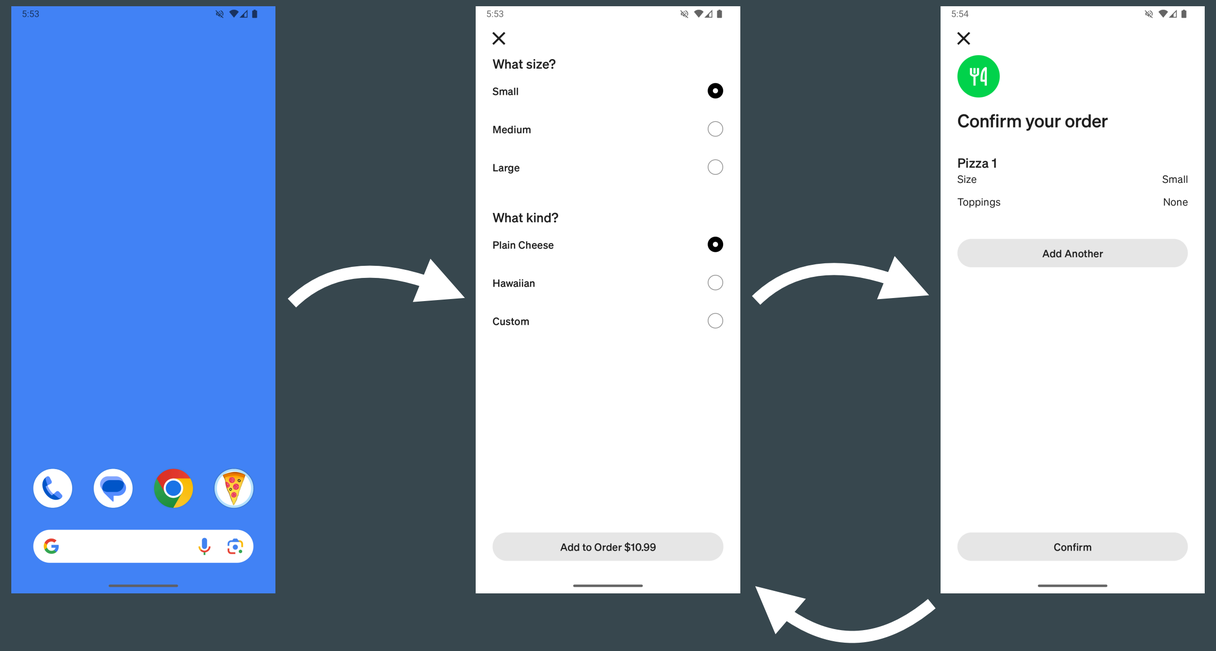
```
interface Display {
  suspend fun <T> show(spec: ScreenSpec<T>): T
  suspend fun <T, R> show(spec: ScreenSpec<T>, block: suspend (T) -> R): R
}
```

Pressing back while executing that lambda will return to the screen

```
override suspend fun execute(display: Display) {
  val pizza = display.show(GetPizzaScreenSpec())
  display.show(ConfirmScreenSpec(pizza))

  ...
}
```



```
override suspend fun execute(display: Display) {
  val pizza = display.show(GetPizzaScreenSpec()) { pizza ->
    display.show(ConfirmScreenSpec(pizza))
    return@show pizza
  }

  ...
}
```



Demo

How it works

1. Run the lambda as a job using `async()`
2. If the user presses back, cancel that job
3. Recover from canceled jobs by showing the previous screen

Advice

Business processes sometimes cancel stuff

Coroutines can do that too

TRICK #10

Detect Idling

Goal

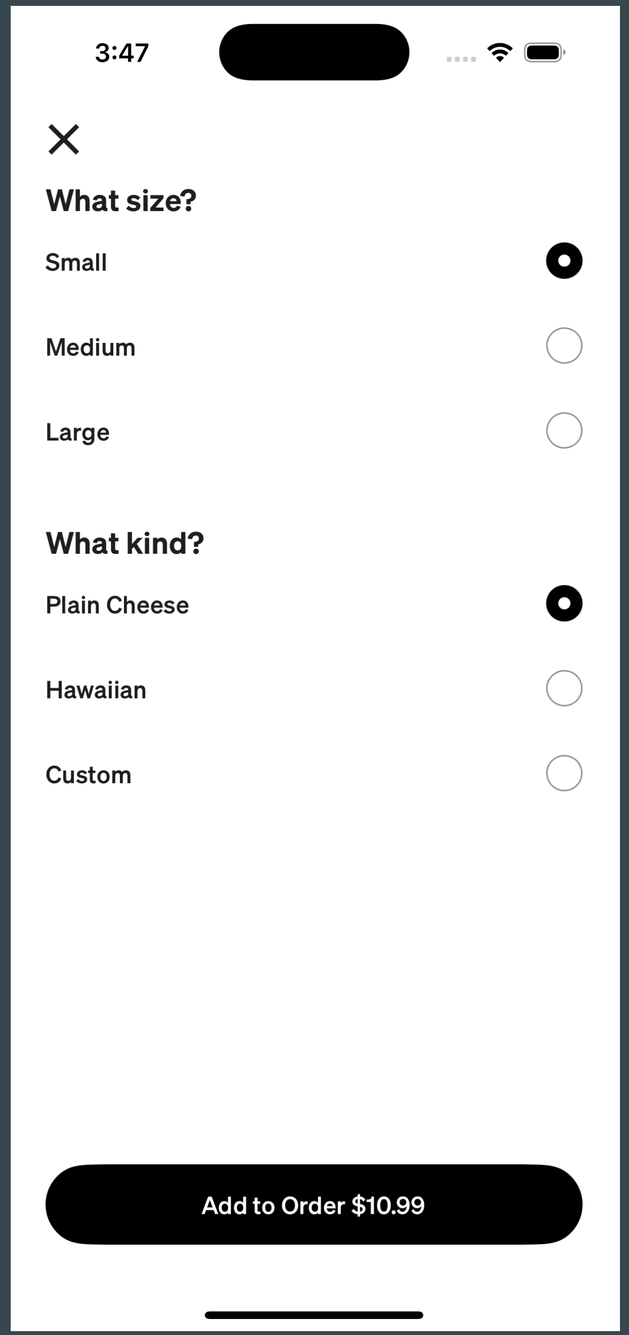
We want a push transition between the current and next screen

Unless there’s a loading delay, in which case we’ll fade to a spinner

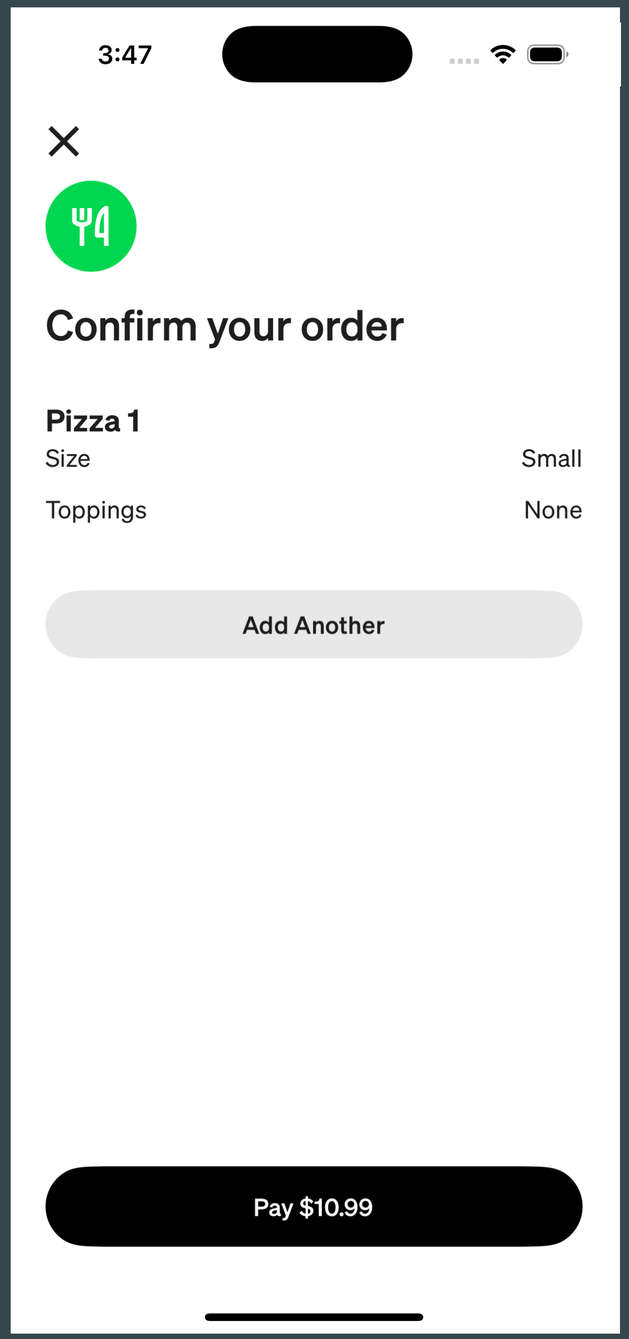
We don’t know ahead of time if there will be a loading delay

```
override suspend fun execute(display: Display) {
  display.show(GetPizzaScreenSpec())
  display.show(ConfirmScreenSpec())
  pizzaApi.order()
  display.show(ResultScreenSpec())
}
```

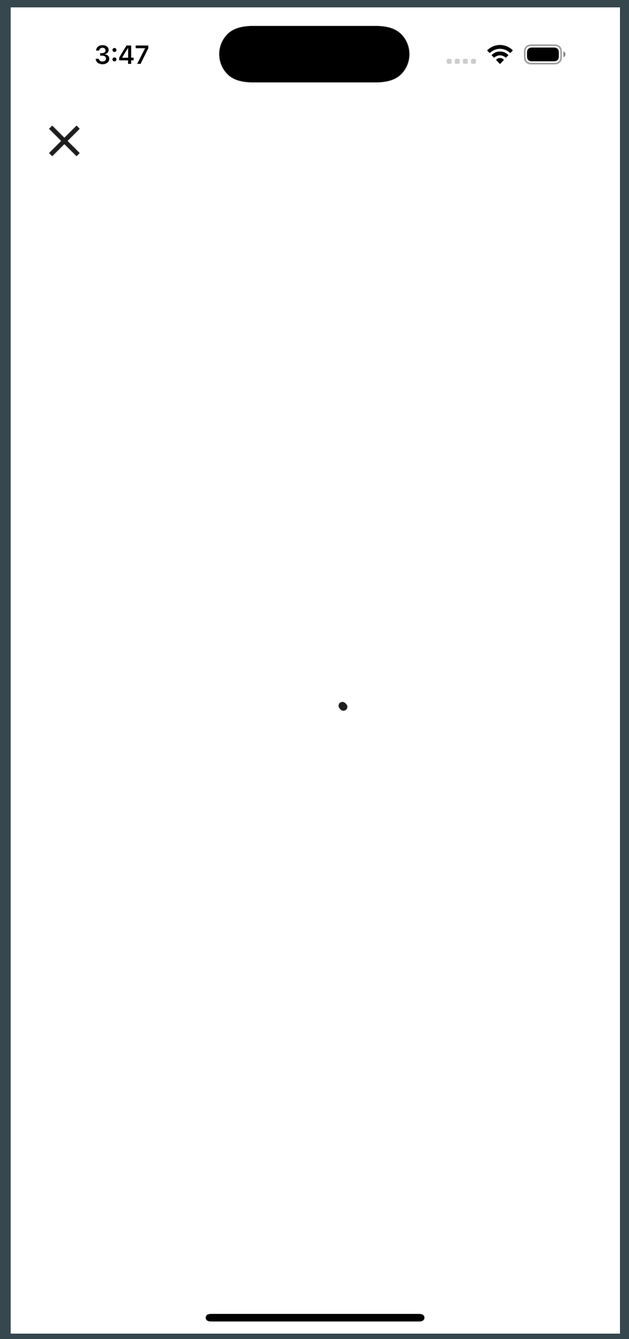
```
override suspend fun execute(display: Display) {
  display.show(GetPizzaScreenSpec())
  display.show(ConfirmScreenSpec())
  pizzaApi.order()
  display.show(ResultScreenSpec())
}
```



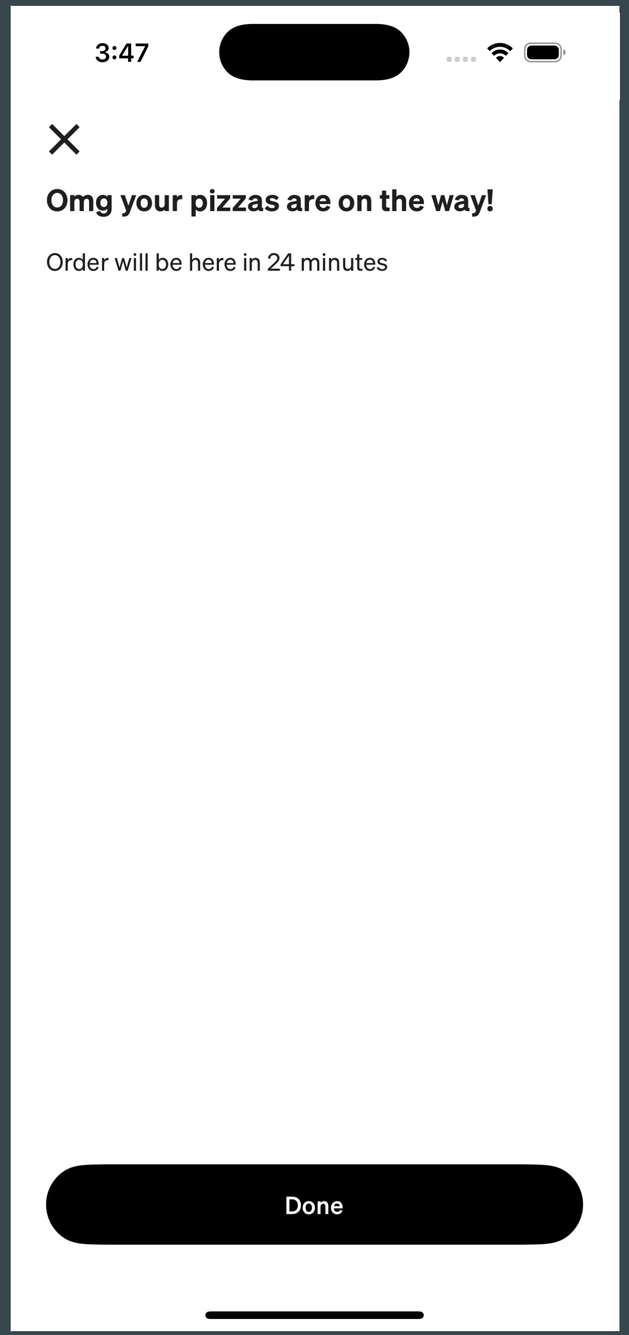
```
override suspend fun execute(display: Display) {
  display.show(GetPizzaScreenSpec())
  display.show(ConfirmScreenSpec())
  pizzaApi.order()
  display.show(ResultScreenSpec())
}
```



```
override suspend fun execute(display: Display) {
  display.show(GetPizzaScreenSpec())
  display.show(ConfirmScreenSpec())
  pizzaApi.order()
  display.show(ResultScreenSpec())
}
```



```
override suspend fun execute(display: Display) {
  display.show(GetPizzaScreenSpec())
  display.show(ConfirmScreenSpec())
  pizzaApi.order()
  display.show(ResultScreenSpec())
}
```



Coroutine Dispatchers

When a coroutine suspends, it builds a `Runnable` to run when it's resumed later

Upon resume, it needs *something* to execute that `Runnable`

That something is a `CoroutineDispatcher`

Coroutine Dispatcher API

```
public abstract class CoroutineDispatcher
  : AbstractCoroutineContextElement(ContinuationInterceptor),
    ContinuationInterceptor {

  abstract fun dispatch(context: CoroutineContext, block: Runnable)

  ...
}
```

Strategy

Decorate `CoroutineDispatcher`

Every time a block is enqueued, increment `enqueueCount`

Every time a block completes, increment `completeCount`

Any time `enqueueCount` equals `completeCount`, we're idle!

```
class IdleCallbackCoroutineDispatcher(
  private val delegate: CoroutineDispatcher,
  private val onIdle: () -> Unit,
) : CoroutineDispatcher() {
  private var enqueueCount = 0
  private var completeCount = 0

  override fun dispatch(context: CoroutineContext, block: Runnable) {
    enqueueCount++
    delegate.dispatch(context) {
      try {
        block.run()
      } finally {
        completeCount++
        if (enqueueCount == completeCount) {
          onIdle()
        }
      }
    }
  }
}
```

```
class IdleCallbackCoroutineDispatcher(
  private val delegate: CoroutineDispatcher,
  private val onIdle: () -> Unit,
) : CoroutineDispatcher() {
  private var enqueueCount = 0
  private var completeCount = 0

  override fun dispatch(context: CoroutineContext, block: Runnable) {
    enqueueCount++
    delegate.dispatch(context) {
      try {
        block.run()
      } finally {
        completeCount++
        if (enqueueCount == completeCount) {
          onIdle()
        }
      }
    }
  }
}
```

```
class IdleCallbackCoroutineDispatcher(
  private val delegate: CoroutineDispatcher,
  private val onIdle: () -> Unit,
) : CoroutineDispatcher() {
  private var enqueueCount = 0
  private var completeCount = 0

  override fun dispatch(context: CoroutineContext, block: Runnable) {
    enqueueCount++
    delegate.dispatch(context) {
      try {
        block.run()
      } finally {
        completeCount++
        if (enqueueCount == completeCount) {
          onIdle()
        }
      }
    }
  }
}
```

```
class IdleCallbackCoroutineDispatcher(
  private val delegate: CoroutineDispatcher,
  private val onIdle: () -> Unit,
) : CoroutineDispatcher() {
  private var enqueueCount = 0
  private var completeCount = 0

  override fun dispatch(context: CoroutineContext, block: Runnable) {
    enqueueCount++
    delegate.dispatch(context) {
      try {
        block.run()
      } finally {
        completeCount++
        if (enqueueCount == completeCount) {
          onIdle()
        }
      }
    }
  }
}
```

```
class IdleCallbackCoroutineDispatcher(
  private val delegate: CoroutineDispatcher,
  private val onIdle: () -> Unit,
) : CoroutineDispatcher() {
  private var enqueueCount = 0
  private var completeCount = 0

  override fun dispatch(context: CoroutineContext, block: Runnable) {
    enqueueCount++
    delegate.dispatch(context) {
      try {
        block.run()
      } finally {
        completeCount++
        if (enqueueCount == completeCount) {
          onIdle()
        }
      }
    }
  }
}
```

```
class IdleCallbackCoroutineDispatcher(
  private val delegate: CoroutineDispatcher,
  private val onIdle: () -> Unit,
) : CoroutineDispatcher() {
  private var enqueueCount = 0
  private var completeCount = 0

  override fun dispatch(context: CoroutineContext, block: Runnable) {
    enqueueCount++
    delegate.dispatch(context) {
      try {
        block.run()
      } finally {
        completeCount++
        if (enqueueCount == completeCount) {
          onIdle()
        }
      }
    }
  }
}
```

```
class IdleCallbackCoroutineDispatcher(
  private val delegate: CoroutineDispatcher,
  private val onIdle: () -> Unit,
) : CoroutineDispatcher() {
  private var enqueueCount = 0
  private var completeCount = 0

  override fun dispatch(context: CoroutineContext, block: Runnable) {
    enqueueCount++
    delegate.dispatch(context) {
      try {
        block.run()
      } finally {
        completeCount++
        if (enqueueCount == completeCount) {
          onIdle()
        }
      }
    }
  }
}
```

```
class IdleCallbackCoroutineDispatcher(
  private val delegate: CoroutineDispatcher,
  private val onIdle: () -> Unit,
) : CoroutineDispatcher() {
  private var enqueueCount = 0
  private var completeCount = 0

  override fun dispatch(context: CoroutineContext, block: Runnable) {
    enqueueCount++
    delegate.dispatch(context) {
      try {
        block.run()
      } finally {
        completeCount++
        if (enqueueCount == completeCount) {
          onIdle()
        }
      }
    }
  }
}
```

Using It

```
val originalDispatcher = scope.coroutineContext[CoroutineDispatcher]
  ?: error("no coroutine dispatcher?")

val onIdleDispatcher = originalDispatcher.withOnIdleCallback {
  maybeShowLoading()
}

scope.launch(onIdleDispatcher) {
  app.execute(display)
}
```

Gotcha: Infinite Idle

The idle callback might dispatch a task

When *that* task completes, we're idle again

Oh no! Infinite loop

Gotcha: Delay

The built-in implementations of `CoroutineDispatcher` implement an internal API, `kotlin.coroutines.Delay`

When we decorate, we lose the interface

This breaks tests!

Advice

The decorator pattern is powerful

Decorating core framework types can be fragile

Watch out for internal-facing APIs like `Delay`

FIN

Takeaways

Let’s Party

Coroutines do a lot for you

They’re fast but not free

Don’t go to programmer jail

github.com/swankjesse/coroutines-party-tricks

Thanks