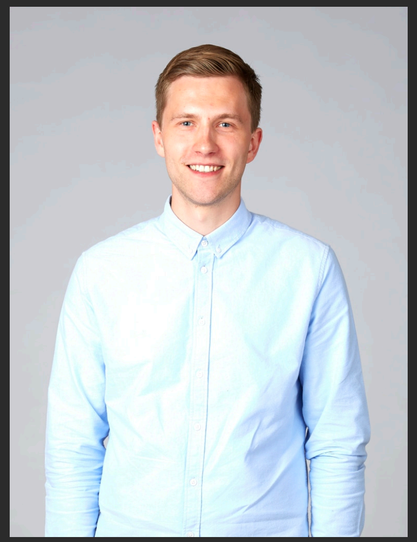
Kotlin Workshop

Kotlin Workshop

Vegard



Sondre

Tia

Nicklas

Thomas

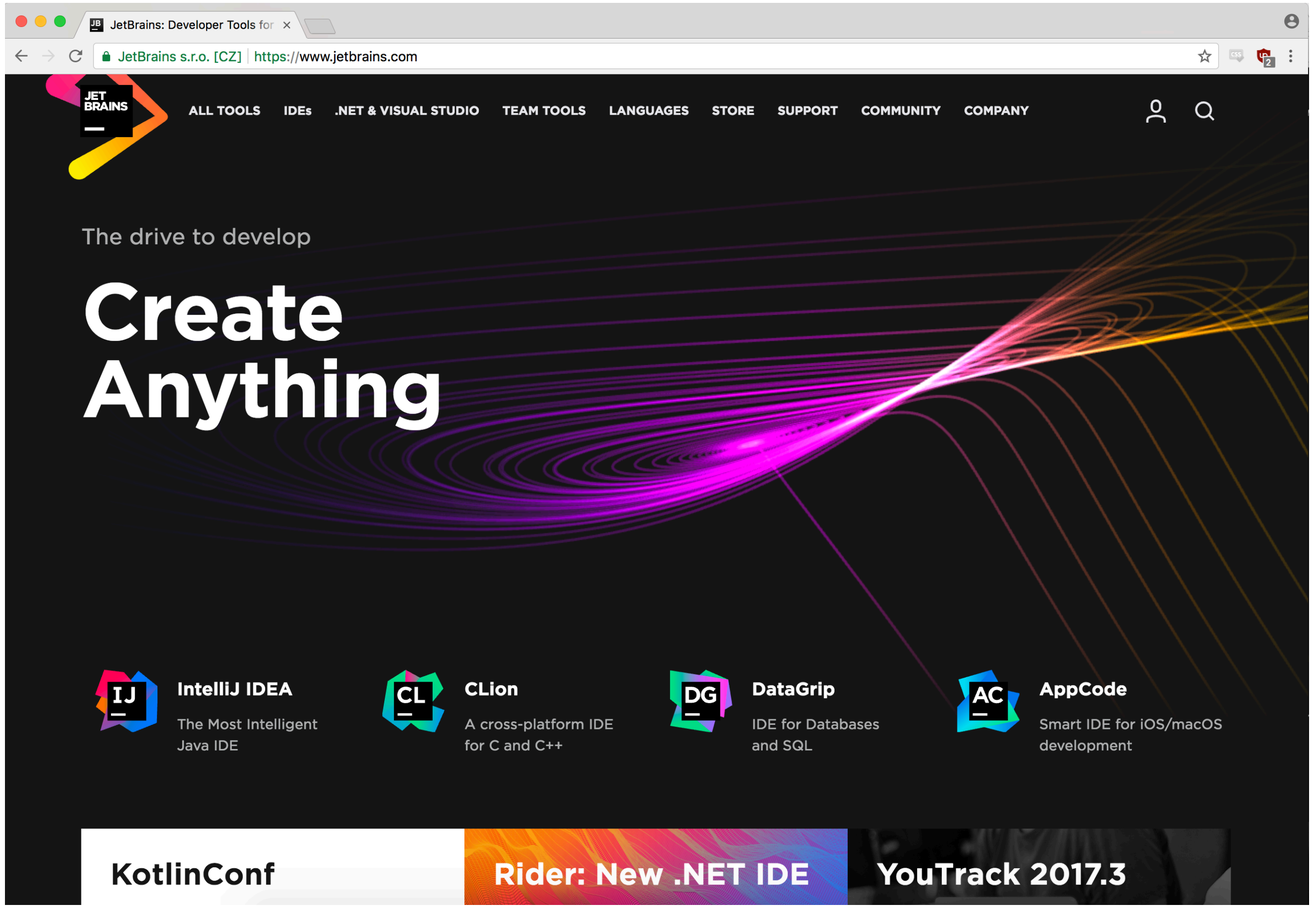
Vetle

Yrjan

Herman


Kotlin

Utviklet av JetBrains



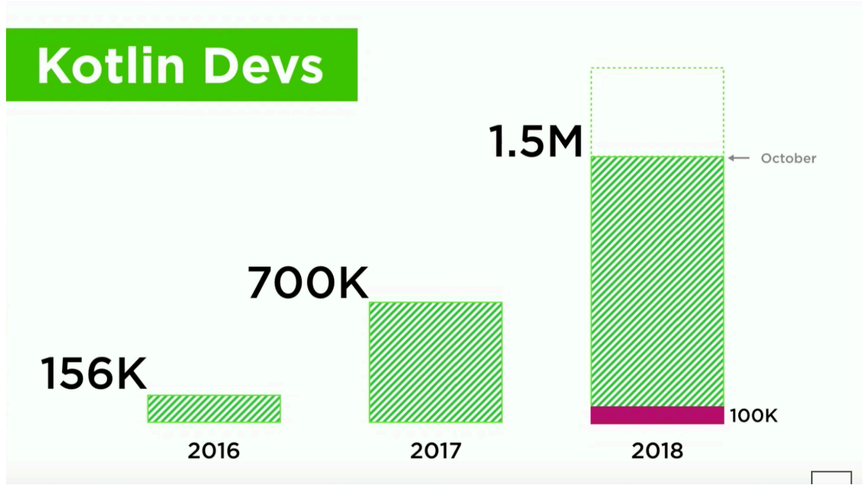
JVM

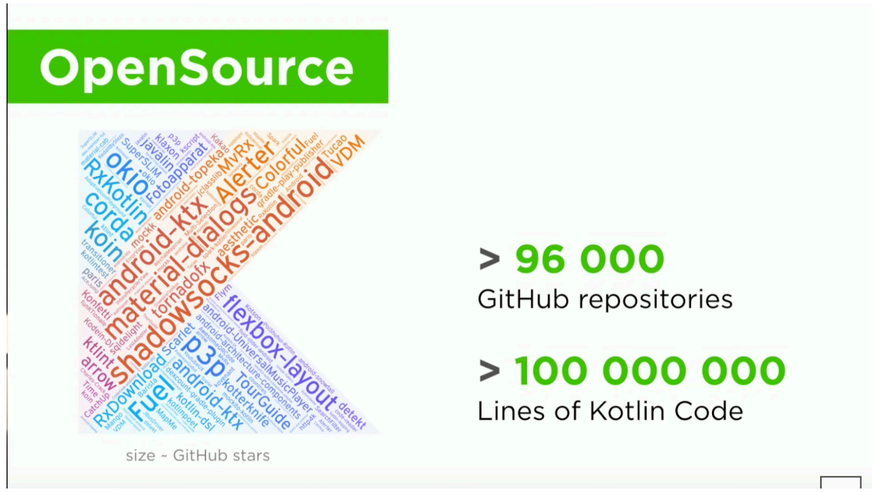

Scala


Groovy


Java

KotlinConf 2018 - Opening Keynote





https://www.youtube.com/watch?v=PsaFVLr8t4E

100% Java interop

Kode

Java

```java
package com.example.demo;

public class UnicornJava {


    private int age;
    private String name;
    private String color;

    public UnicornJava(int age, String name, String color) {
        this.age = age;
        this.name = name;
        this.color = color;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        UnicornJava that = (UnicornJava) o;

        if (age != that.age) return false;
        if (name != null ? !name.equals(that.name) : that.name != null) return false;
        return color != null ? color.equals(that.color) : that.color == null;
    }

    @Override
    public int hashCode() {
        int result = age;
        result = 31 * result + (name != null ? name.hashCode() : 0);
        result = 31 * result + (color != null ? color.hashCode() : 0);
        return result;
    }

    @Override
    public String toString() {
        return "UnicornJava{" +
                "age=" + age +
                ", name='" + name + '\'' +
                ", color='" + color + '\'' +
                '}';
    }
}
```

```
private int age;
private String name;
private String color;

public UnicornJava(int age, String name, String color
    this.age = age;
    this.name = name;
    this.color = color;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public String getName() {
    return name;
```

Java

```java
package com.example.demo;

public class UnicornJava {


    private int age;
    private String name;
    private String color;

    public UnicornJava(int age, String name, String color) {
        this.age = age;
        this.name = name;
        this.color = color;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        UnicornJava that = (UnicornJava) o;

        if (age != that.age) return false;
        if (name != null ? !name.equals(that.name) : that.name != null) return false;
        return color != null ? color.equals(that.color) : that.color == null;
    }

    @Override
    public int hashCode() {
        int result = age;
        result = 31 * result + (name != null ? name.hashCode() : 0);
        result = 31 * result + (color != null ? color.hashCode() : 0);
        return result;
    }

    @Override
    public String toString() {
        return "UnicornJava{" +
                "age=" + age +
                ", name='" + name + '\'' +
                ", color='" + color + '\'' +
                '}';
    }
}
```

Java

```java
package com.example.demo;

public class UnicornJava {


    private int age;
    private String name;
    private String color;

    public UnicornJava(int age, String name, String color) {
        this.age = age;
        this.name = name;
        this.color = color;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        UnicornJava that = (UnicornJava) o;

        if (age != that.age) return false;
        if (name != null ? !name.equals(that.name) : that.name != null) return false;
        return color != null ? color.equals(that.color) : that.color == null;
    }

    @Override
    public int hashCode() {
        int result = age;
        result = 31 * result + (name != null ? name.hashCode() : 0);
        result = 31 * result + (color != null ? color.hashCode() : 0);
        return result;
    }

    @Override
    public String toString() {
        return "UnicornJava{" +
                "age=" + age +
                ", name='" + name + '\'' +
                ", color='" + color + '\'' +
                '}';
    }
}
```

Kotlin

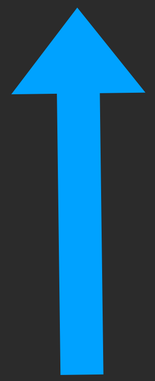
```kotlin
package com.example.demo

data class Unicorn(var age: Int, val name: String, var color: String)
```

Kotlin

```kotlin
package com.example.demo

data class Unicorn(var age: Int, val name: String, var color: String)
```

Kotlin

```
package com.example.demo

data class Unicorn(var age: Int, val name: String, var color: String)
```

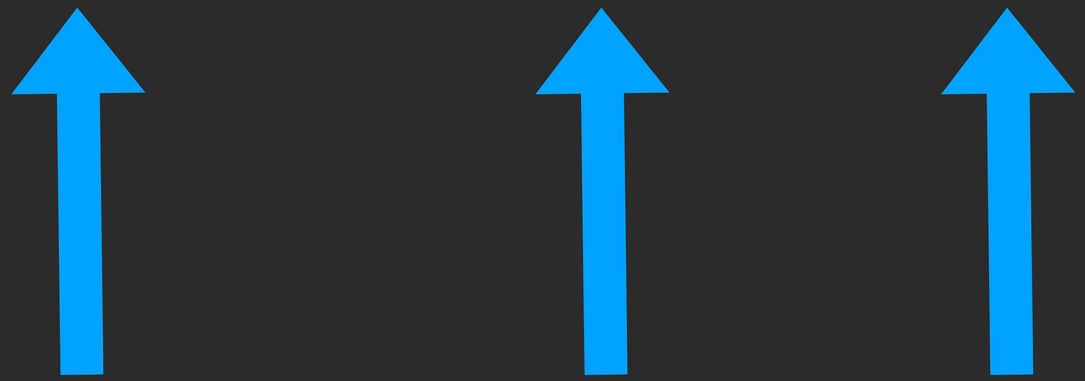


Kotlin

```
package com.example.demo

data class Unicorn(var age: Int, val name: String, var color: String)
```

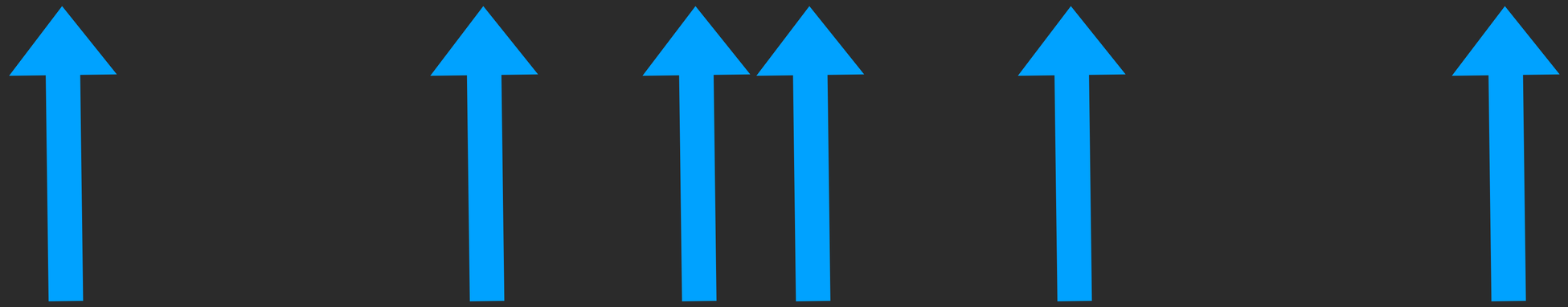
Kotlin

```
package com.example.demo

data class Unicorn(var age: Int, val name: String, var color: String)
```

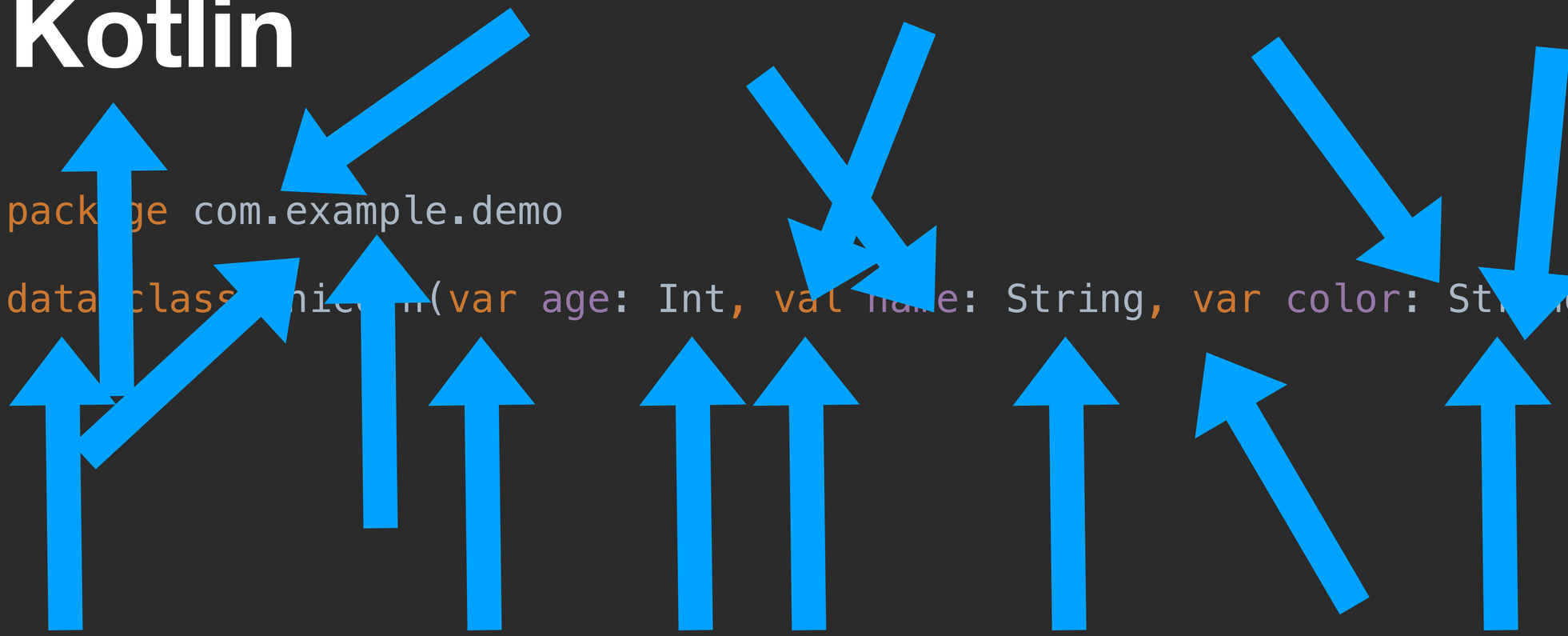


Kotlin

```
package com.example.demo

data class Unicorn(var age: Int, val name: String, var color: String)
```



Kotlin

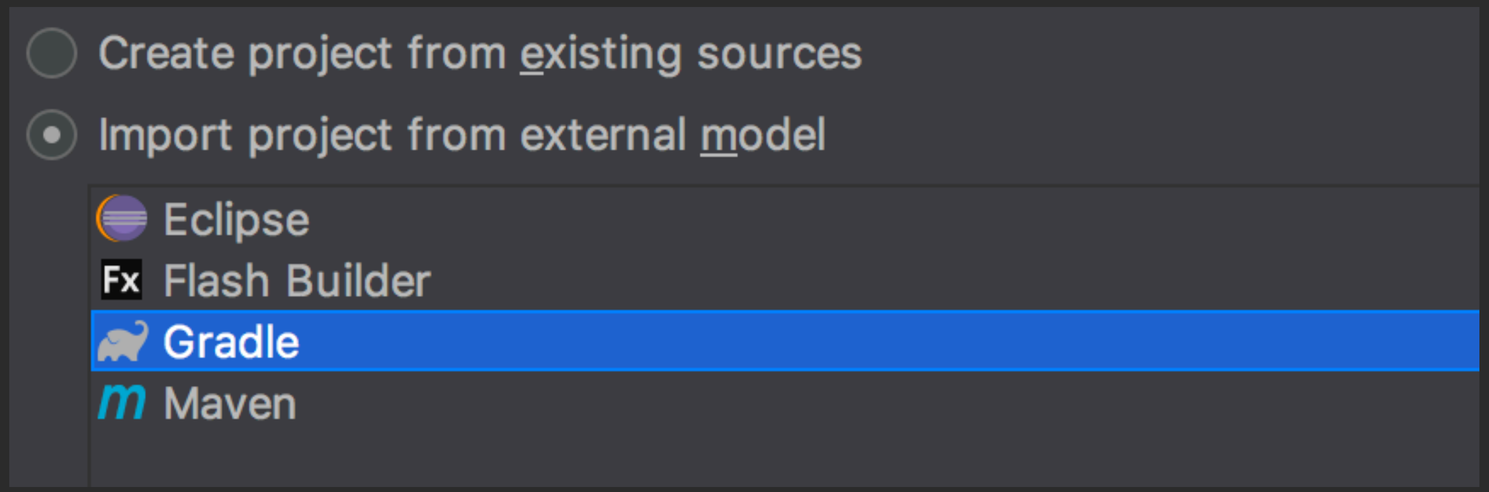


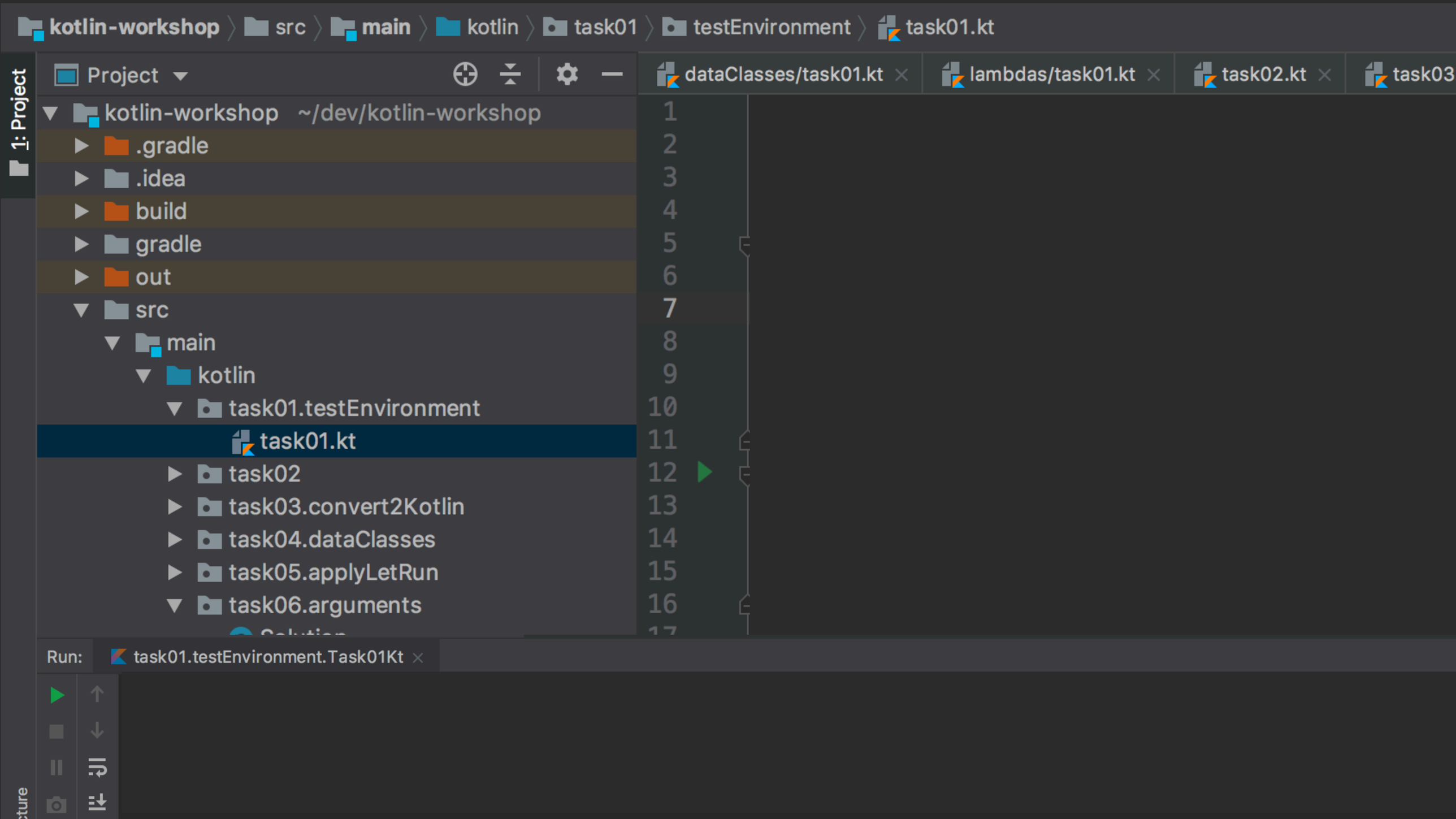
```
package com.example.demo

data class Animal(var age: Int, val name: String, var color: String)
```

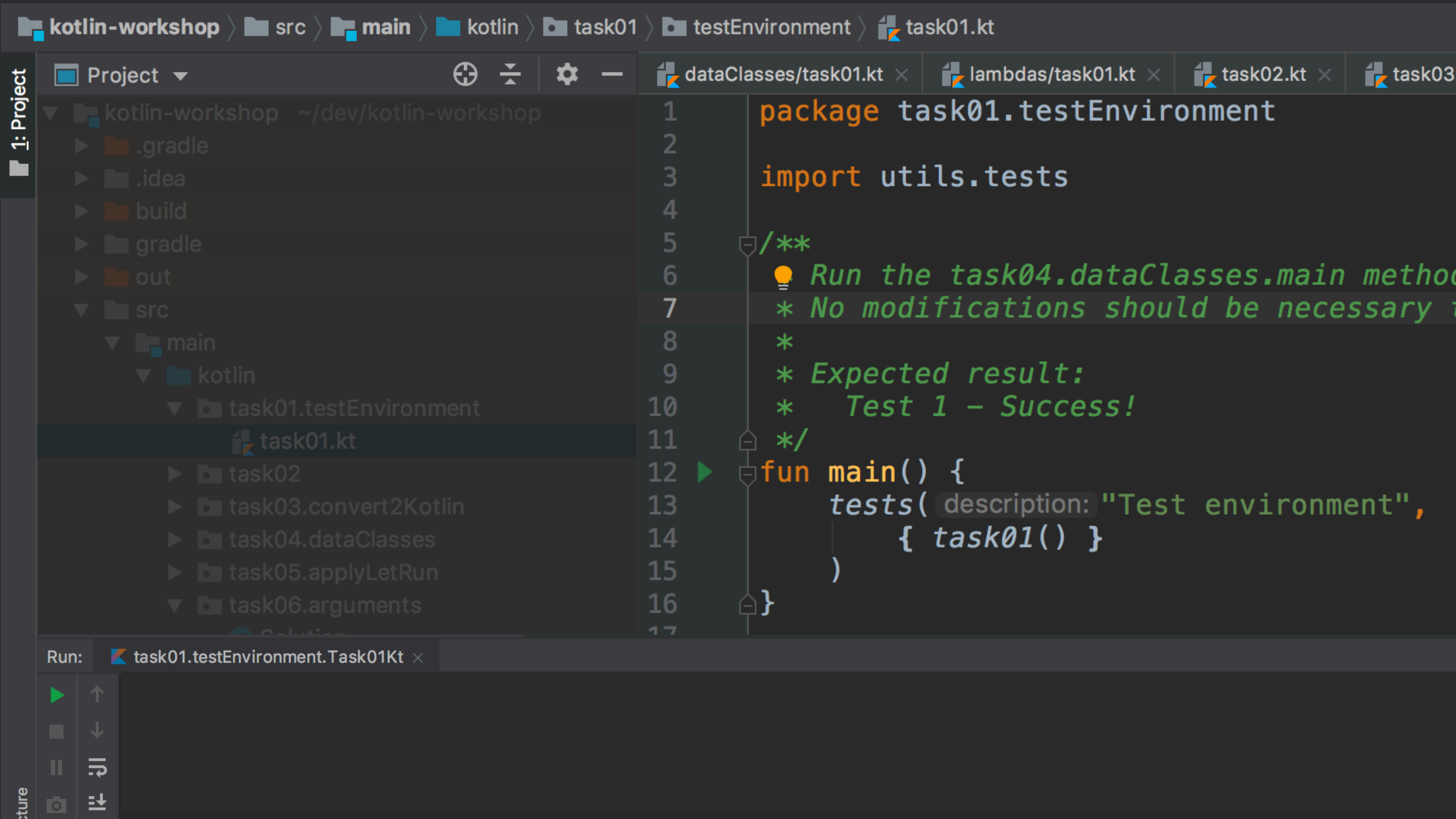
git clone https://github.com/bekk/kotlin-workshop.git

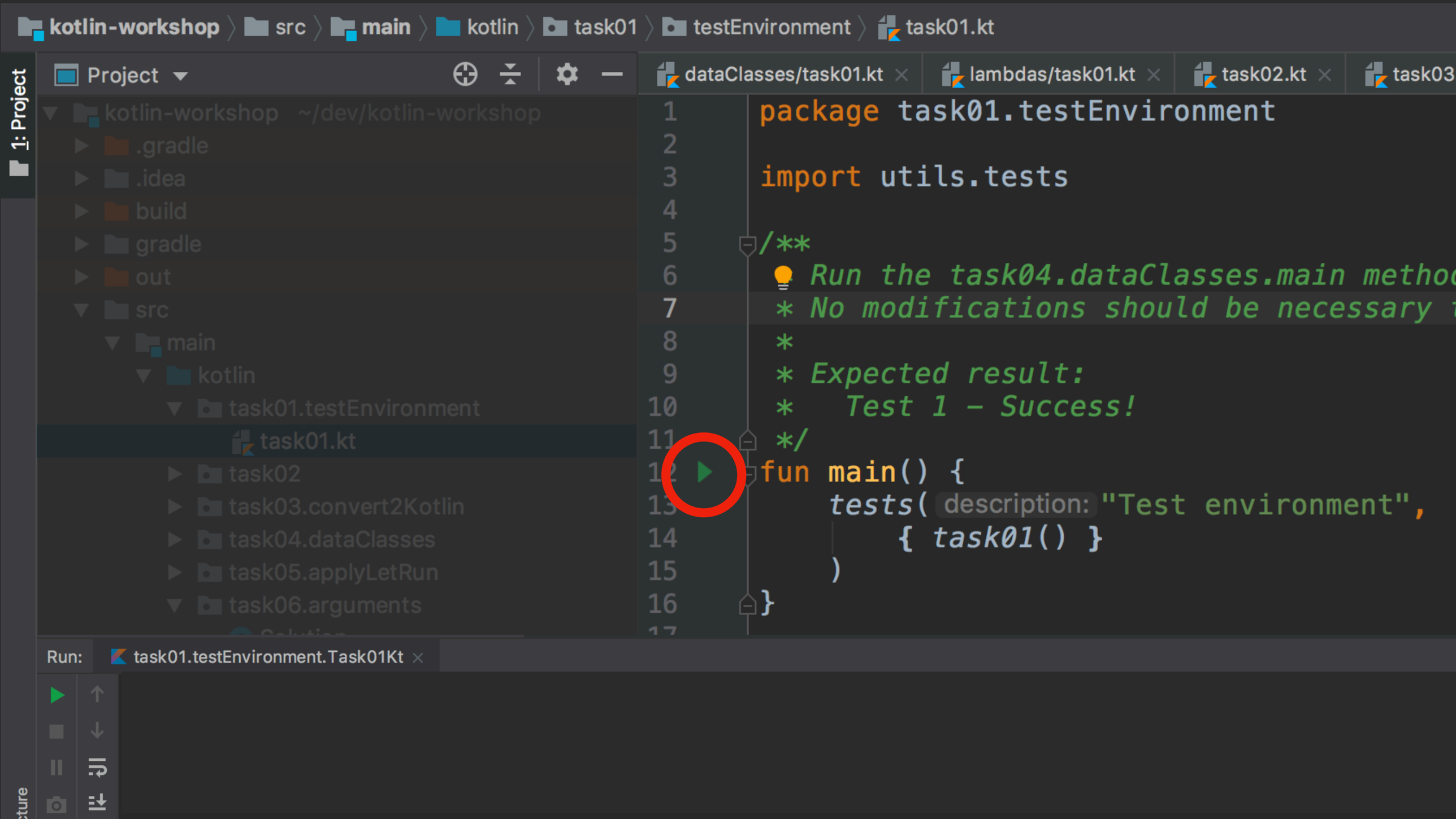
git clone https://github.com/bekk/kotlin-workshop.git

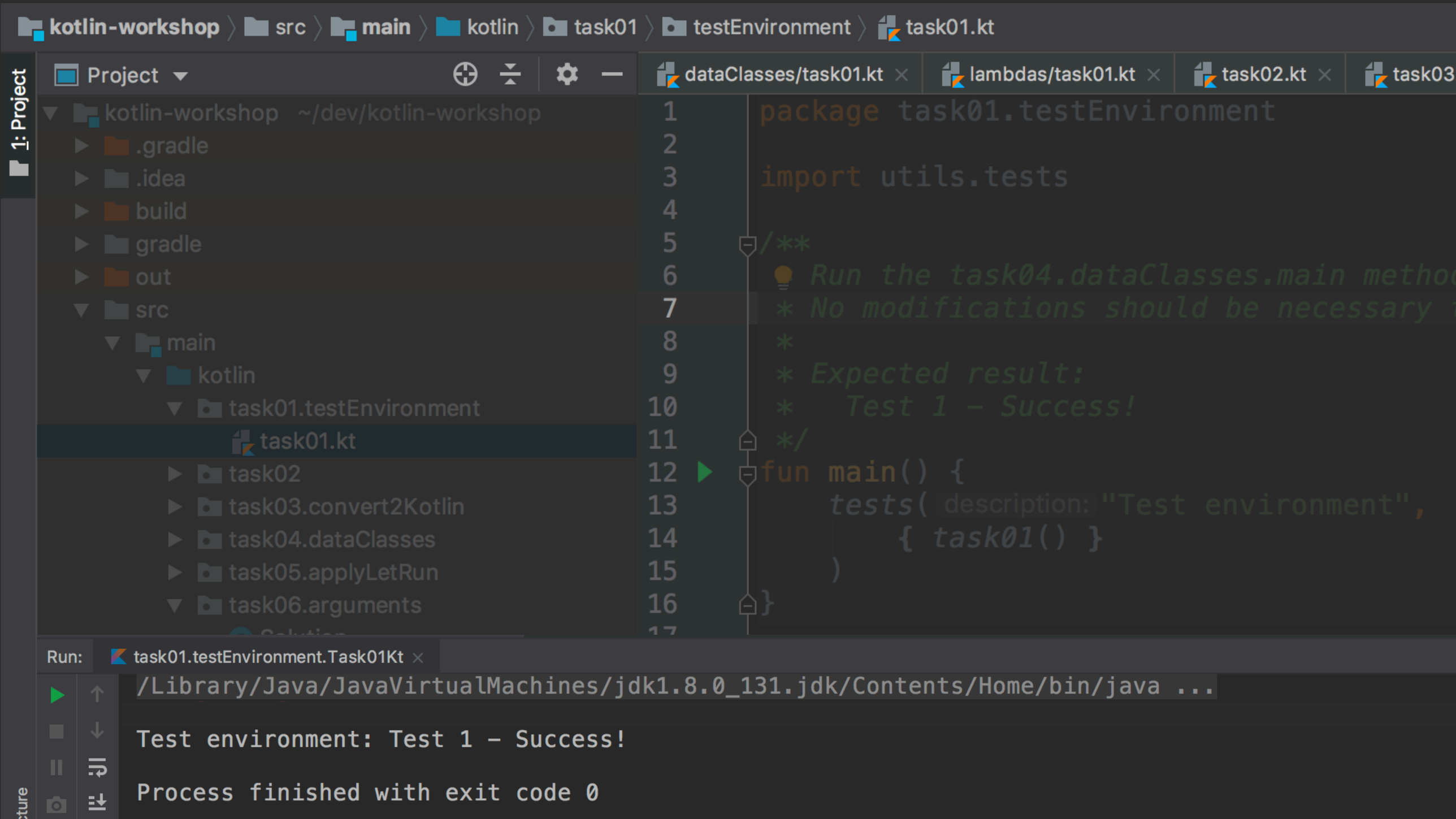
File > New > Project from Existing Sources




kotlin-workshop › src › main › kotlin › task01 › testEnvironment › task01.kt
1: Project
Project
kotlin-workshop ~/dev/kotlin-workshop
.gradle
.idea
build
gradle
out
src
main
kotlin
task01.testEnvironment
task01.kt
task02
task03.convert2Kotlin
task04.dataClasses
task05.applyLetRun
task06.arguments
dataClasses/task01.kt
lambdas/task01.kt
task02.kt
task03
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Run:
task01.testEnvironment.Task01Kt


kotlin-workshop › src › main › kotlin › task01 › testEnvironment › task01.kt
Project
1: Project
kotlin-workshop ~/dev/kotlin-workshop
.gradle
.idea
build
gradle
out
src
main
kotlin
task01.testEnvironment
task01.kt
task02
task03.convert2Kotlin
task04.dataClasses
task05.applyLetRun
task06.arguments
dataClasses/task01.kt
lambdas/task01.kt
task02.kt
task03
1 package task01.testEnvironment
2
3 import utils.tests
4
5 /**
6 * Run the task04.dataClasses.main metho
7 * No modifications should be necessary
8 *
9 * Expected result:
10 * Test 1 - Success!
11 */
12 fun main() {
13 tests(description: "Test environment",
14 { task01() }
15)
16 }
Run:
task01.testEnvironment.Task01Kt


kotlin-workshop › src › main › kotlin › task01 › testEnvironment › task01.kt
Project
1: Project
kotlin-workshop ~/dev/kotlin-workshop
.gradle
.idea
build
gradle
out
src
main
kotlin
task01.testEnvironment
task01.kt
task02
task03.convert2Kotlin
task04.dataClasses
task05.applyLetRun
task06.arguments
dataClasses/task01.kt
lambdas/task01.kt
task02.kt
task03
1 package task01.testEnvironment
2
3 import utils.tests
4
5 /**
6 * Run the task04.dataClasses.main metho
7 * No modifications should be necessary
8 *
9 * Expected result:
10 * Test 1 - Success!
11 */
12 fun main() {
13 tests(description: "Test environment",
14 { task01() }
15)
16 }
Run:
task01.testEnvironment.Task01Kt


kotlin-workshop › src › main › kotlin › task01 › testEnvironment › task01.kt
Project
kotlin-workshop ~/dev/kotlin-workshop
.gradle
.idea
build
gradle
out
src
main
kotlin
task01.testEnvironment
task01.kt
task02
task03.convert2Kotlin
task04.dataClasses
task05.applyLetRun
task06.arguments
dataClasses/task01.kt
lambdas/task01.kt
task02.kt
task03
1 package task01.testEnvironment
2
3 import utils.tests
4
5 /**
6 * Run the task04.dataClasses.main metho
7 * No modifications should be necessary
8 *
9 * Expected result:
10 * Test 1 - Success!
11 */
12 fun main() {
13 tests(description: "Test environment",
14 { task01() }
15)
16 }
Run:
task01.testEnvironment.Task01Kt
/Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java ...
Test environment: Test 1 - Success!
Process finished with exit code 0

```kotlin
package task01.testEnvironment

import utils.tests

/**
 * Run the main method to check that everything is working correctly.
 * No modifications should be necessary to get this running.
 *
 * Expected result:
 *   Test 1 - Success!
 */
fun main() {
    tests("Test environment",
        { task01() }
    )
}

/**
 * This is where we typically will solve the tasks at hand.
 */
fun task01() = true
```

Oppgave 1

Test at oppsettet fungerer

Basic syntax

```
fun main() {

    val immutable: String = "hello"
    var mutable: String = "hello"

    println("$immutable $mutable")

}
```

```
fun main() {

    val immutable = "hello"
    var mutable = "hello"

    println("$immutable $mutable")

}
```

```
val users = listOf("Vegard", "Thomas", "Vetle", "Yrjan", "Herman")

for (user in users) {
    println(user)
}
```

```
class User(val name: String)

val users2 = listOf(User("Ola"), User("Kari"))

for (user in users2) {
    println(user.name)
}
```

```
fun printUser(user: User) {
```

```
    class User(val name: String)

    val users2 = listOf(User("Ola"), User("Kari"))

    for (user in users2) {
        println(user.name)
    }




    fun printUser(user: User) {
        println(user)
    }

    fun shoutUsername(user: User): String = user.name + "!"

}
```

```kotlin
fun printUser(user: User) {
    println(user)
}

fun shoutUsername(user: User): String = user.name + "!"
```

```kotlin
fun printUser(user: User) {
    println(user)
}

fun shoutUsername(user: User) = user.name + "!"
```

```
data class User(val age: Int)
val old = User(100).age * 2
```

```
data class User(val age: Int)
val old = User(null).age * 2
```

```
data class User(val age: Int?)
val old = User(null).age * 2
```

```
data class User(val age: Int?)
val old: Int? = User(null).age?.times(2)
```

```
data class User(val age: Int?)
val old: Int = User(null).age?.times(2) ?: 0
```

```
data class User(val age: Int)
val old = User(100).age * 2
```

```
data class User(val age: Int)
val old = User(null).age * 2
```

```
data class User(val age: Int?)
val old = User(null).age * 2
```

```
data class User(val age: Int?)
val old: Int? = User(null).age?.times(2)
```

```
data class User(val age: Int?)
val old: Int = User(null).age?.times(2) ?: 0
```

```
data class User(val age: Int)
val old = User(100).age * 2
```

```
data class User(val age: Int)
val old = User(null).age * 2
```

```
data class User(val age: Int?)
val old = User(null).age * 2
```

```
data class User(val age: Int?)
val old: Int? = User(null).age?.times(2)
```

```
data class User(val age: Int?)
val old: Int = User(null).age?.times(2) ?: 0
```

```
data class User(val age: Int)
val old = User(100).age * 2
```

```
data class User(val age: Int)
val old = User(null).age * 2
```

```
data class User(val age: Int?)
val old = User(null).age * 2
```

```
data class User(val age: Int?)
val old: Int? = User(null).age?.times(2)
```

```
data class User(val age: Int?)
val old: Int = User(null).age?.times(2) ?: 0
```

```
data class User(val age: Int)
val old = User(100).age * 2
```

```
data class User(val age: Int)
val old = User(null).age * 2
```

```
data class User(val age: Int?)
val old = User(null).age * 2
```

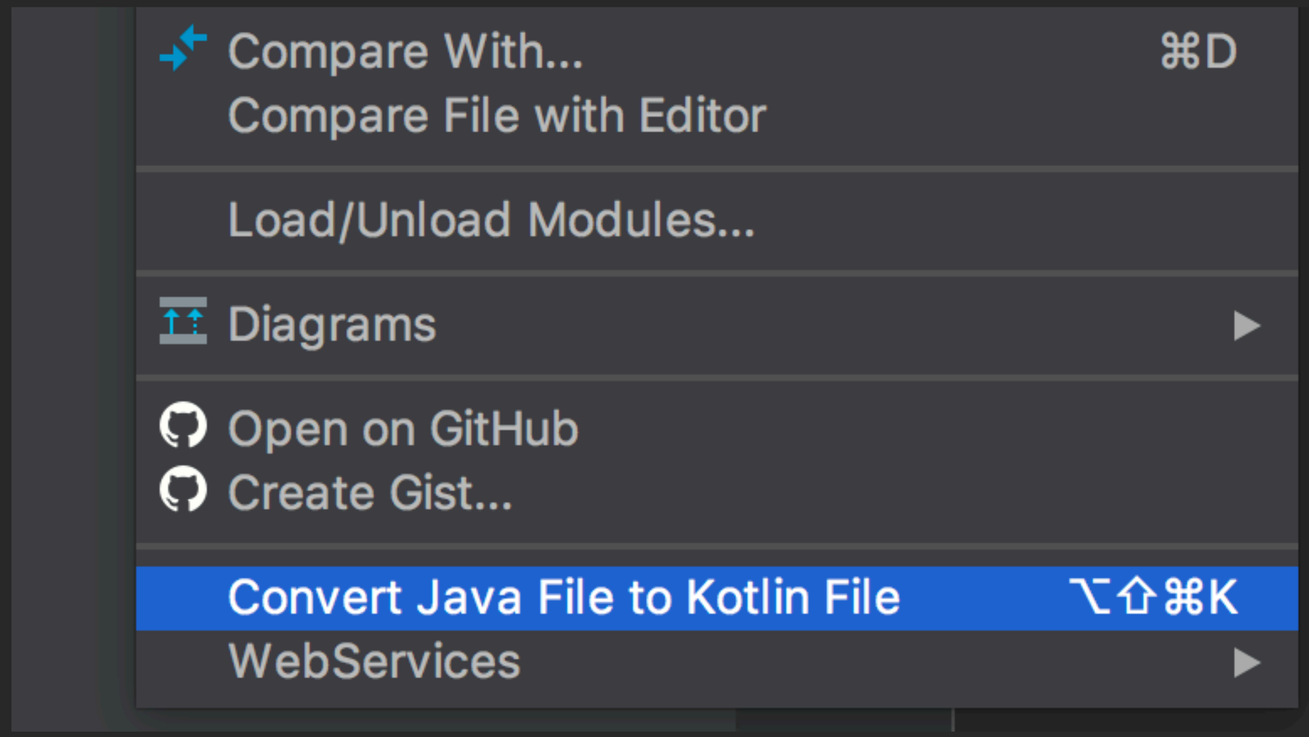
```
data class User(val age: Int?)
val old: Int? = User(null).age?.times(2)
```

```
data class User(val age: Int?)
val old: Int = User(null).age?.times(2) ?: 0
```

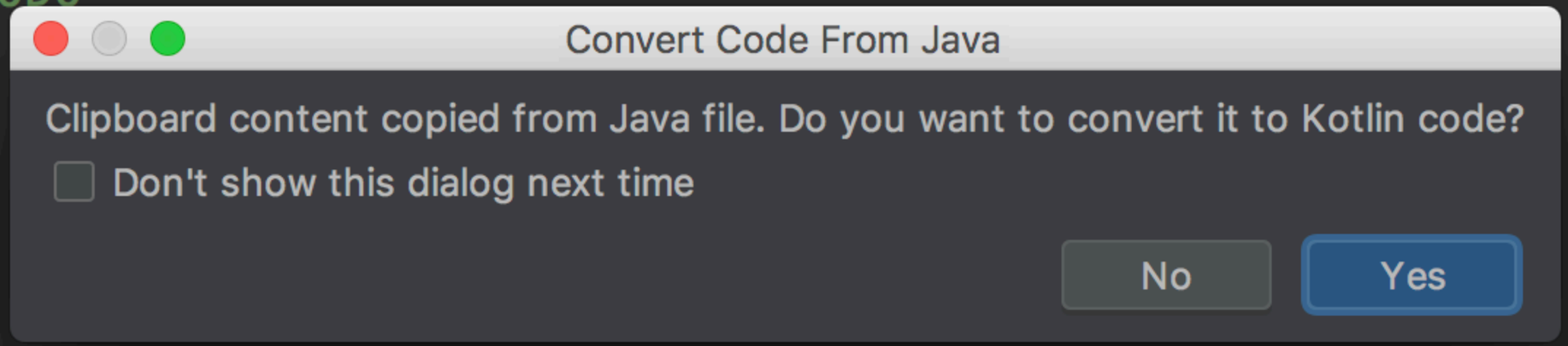
Oppgave 2

Basic syntax

Convert Java 2 Kotlin


Compare With... ⌘D
Compare File with Editor
Load/Unload Modules...
Diagrams
Open on GitHub
Create Gist...
Convert Java File to Kotlin File ⌥⇧⌘K
WebServices

```
case 6: return "sunday";
default: return "not a valid day number";

'TODO"
```



Oppgave 3

Convert Java 2 Kotlin

Kotlin Data Classes

```
class User(
    val name: String,
    val age: Int
)
```

```
data class User(
    val name: String,
    val age: Int
)
```

```
data class User(
    val name: String,
    val age: Int
)
```

```
equals()
hashCode()
toString()
componentN()
copy()
```

Oppgave 4

Kotlin Data Classes

apply, let, run

```kotlin
val user = User("Vegard", 30)

user.apply { age = 50 } // Samme objekt, nå med age = 50

user.let { it.name + it.age } // "Vegard30"

user.run { name + age } // "Vegard30"

user.also { println(it.name) } // user

with(user) { name + age } // "Vegard30"
```

Oppgave 5

apply, let, run

Arguments

```kotlin
data class User(
    val name: String = "Default name",
    val age: Int = 40
)

User()

User("Vegard", 30)

User(name = "Vegard", age = 30)

User(name = "Vegard")

User(age = 30)

User("Vegard", 30).copy(age = 31)
```

Oppgave 6

Arguments

Smart casting

```
fun someFunction(): Any = "hello"

val result = someFunction()

if (result is String) {
    result.capitalize()
}
```

```
fun someFunction(): Any = "hello"

val result = someFunction()

val str = when (result) {
    is String -> result
    is Int -> (result * 2).toString()
    else -> "not a string or a number"
}
```

Oppgave 7

Smart casting

Lambdas

```kotlin
fun something(lambda: () -> String) {
    lambda()
}

something({ "hello" })
something { "hello" }


fun withArgs(lambda: (String) -> String) = lambda("hello")

withArgs { arg -> "$arg!" }

withArgs { "$it!" }


fun prefix(arg: String): (String) -> String = { it + arg }

val exclaim = prefix("!")
exclaim("hello")
```

```
fun something(lambda: () -> String) {
    lambda()
}

something({ "hello" })
something { "hello" }


fun withArgs(lambda: (String) -> String) = lambda("hello")

withArgs { arg -> "$arg!" }

withArgs { "$it!" }


fun prefix(arg: String): (String) -> String = { it + arg }

val exclaim = prefix("!")
exclaim("hello")
```

```
fun something(lambda: () -> String) {
    lambda()
}

something({ "hello" })
something { "hello" }


fun withArgs(lambda: (String) -> String) = lambda("hello")

withArgs { arg -> "$arg!" }

withArgs { "$it!" }


fun prefix(arg: String): (String) -> String = { it + arg }

val exclaim = prefix("!")
exclaim("hello")
```

Oppgave 8

Lambdas

Map, Filter, Reduce, etc…

```kotlin
data class User(val name: String, val age: Int)

val users = listOf(User("a", 20), User("b", 25), User("c",30))

users.map(User::toString)

users.map { it.toString() }
users.mapIndexed { index, user -> user.age + index }
users.sortedBy { it.age }.first()

val averageAge: Double = users
    .map { it.age }
    .average()

val wiseAndOld: List<String> = users
    .filter { it.age > 23 }
    .map { it.name }

users
    .map { it.age }
    .fold(0) { acc, it -> acc + it }
```

```kotlin
data class User(val name: String, val age: Int)

val users = listOf(User("a", 20), User("b", 25), User("c",30))

users.map(User::toString)

users.map { it.toString() }
users.mapIndexed { index, user -> user.age + index }
users.sortedBy { it.age }.first()

val averageAge: Double = users
    .map { it.age }
    .average()

val wiseAndOld: List<String> = users
    .filter { it.age > 23 }
    .map { it.name }

users
    .map { it.age }
    .fold(0) { acc, it -> acc + it }
```

```kotlin
data class User(val name: String, val age: Int)

val users = listOf(User("a", 20), User("b", 25), User("c",30))

users.map(User::toString)

users.map { it.toString() }
users.mapIndexed { index, user -> user.age + index }
users.sortedBy { it.age }.first()

val averageAge: Double = users
    .map { it.age }
    .average()

val wiseAndOld: List<String> = users
    .filter { it.age > 23 }
    .map { it.name }

users
    .map { it.age }
    .fold(0) { acc, it -> acc + it }
```

```kotlin
data class User(val name: String, val age: Int)

val users = listOf(User("a", 20), User("b", 25), User("c",30))

users.map(User::toString)

users.map { it.toString() }
users.mapIndexed { index, user -> user.age + index }
users.sortedBy { it.age }.first()

val averageAge: Double = users
    .map { it.age }
    .average()

val wiseAndOld: List<String> = users
    .filter { it.age > 23 }
    .map { it.name }

users
    .map { it.age }
    .fold(0) { acc, it -> acc + it }
```

Oppgave 9

Map, Filter, Reduce, etc…

Extension functions

```kotlin
data class User(val name: String, val age: Int)

fun User.funkyName() = name.reversed().capitalize() + age

println(User("Vegard", 30).funkyName()) // DrageV30
```

```kotlin
fun String.funkyName() = this.reversed().capitalize() + this.first()

println("Vegard".funkyName()) // DrageVV
```

```kotlin
data class User(val name: String, val age: Int)

fun User.funkyName() = name.reversed().capitalize() + age

println(User("Vegard", 30).funkyName()) // DrageV30
```

```kotlin
fun String.funkyName() = this.reversed().capitalize() + this.first()

println("Vegard".funkyName()) // DrageVV
```

Oppgave 10

Extension functions

Ktor

```
fun main() {
    embeddedServer(Netty, 8089) {}.start(wait = true)
}
```

```
fun main() {
    @Location("/test/get")
    class Test
    embeddedServer(Netty, 8089) {
        install(Locations)
        routing {
            get<Test> {
                call.respond("Imma here")
            }
        }
    }.start(wait = true)
}
```

```
fun main() {
    @Location("/test/get")
    data class Test(val name: String)
    embeddedServer(Netty, 8089) {
        install(Locations)
        routing {
            get<Test> { test ->
                call.respond("Imma here, ${test.name}")
            }
        }
    }.start(wait = true)
}
```

```kotlin
fun main() {
    @Location("/test/get")
    data class Test(val name: String)
    embeddedServer(Netty, 8089) {
        install(Locations)
        install(ContentNegotiation) {
            jackson {
                registerKotlinModule()
            }
        }
        routing {
            get<Test> { test ->
                call.respond(TestResponse("Imma here, ${test.name}"))
            }
        }
    }.start(wait = true)
}
```

```
class CocktailClient {
    private val config: HttpClientConfig<ApacheEngineConfig>.() -> Unit = {
        install(JsonFeature) {
            serializer = JacksonSerializer {
                registerKotlinModule()
                configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false)
            }
        }
    }
    private val httpClient = HttpClient(Apache, config)
    suspend fun getCocktail(cocktailName: String) {
        httpClient.get<HttpResponse>(
                "https://www.thecocktaildb.com/api/json/v1/1/search.php") {
            accept(ContentType.Application.Json)
            parameter("s", cocktailName)
        }.call.response.receive<Drinks>()
    }
}
```

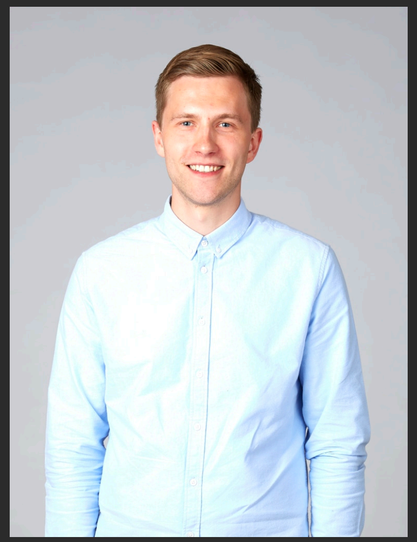
ktor.io

Oppgave 11

Ktor

Takk for oss!

Vegard



Sondre

Tia

Nicklas

Inline classes

```
inline class SuperInt(val value: Int) {
    fun double(): Int = value * 2
}

SuperInt(50).value
SuperInt(50).double()
```

Oppgave 12

Inline classes

Coroutines

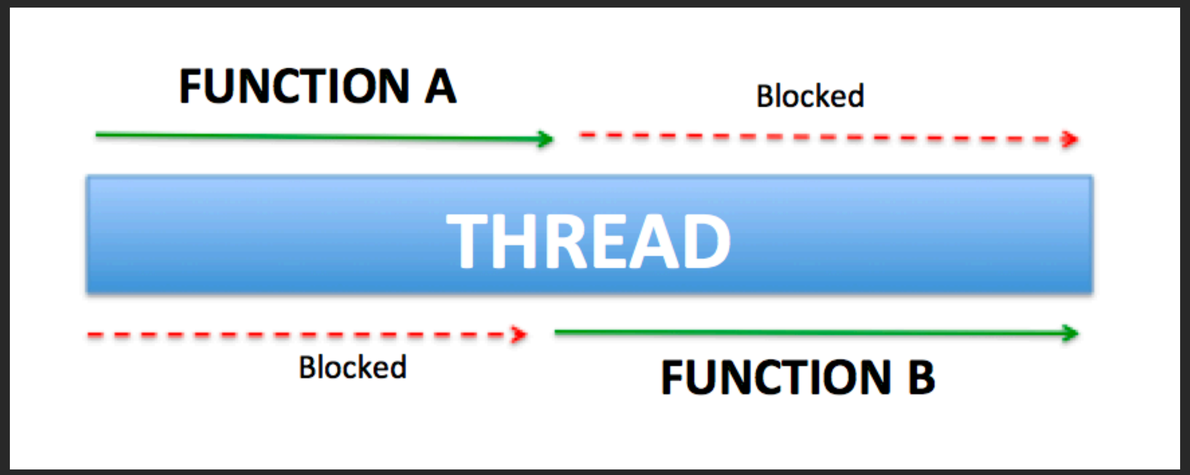
```
var text = "hei"
GlobalScope.launch {
    text = "hei2"
    delay(1000)
}
println(text)
Thread.sleep(2000)
println(text)
```

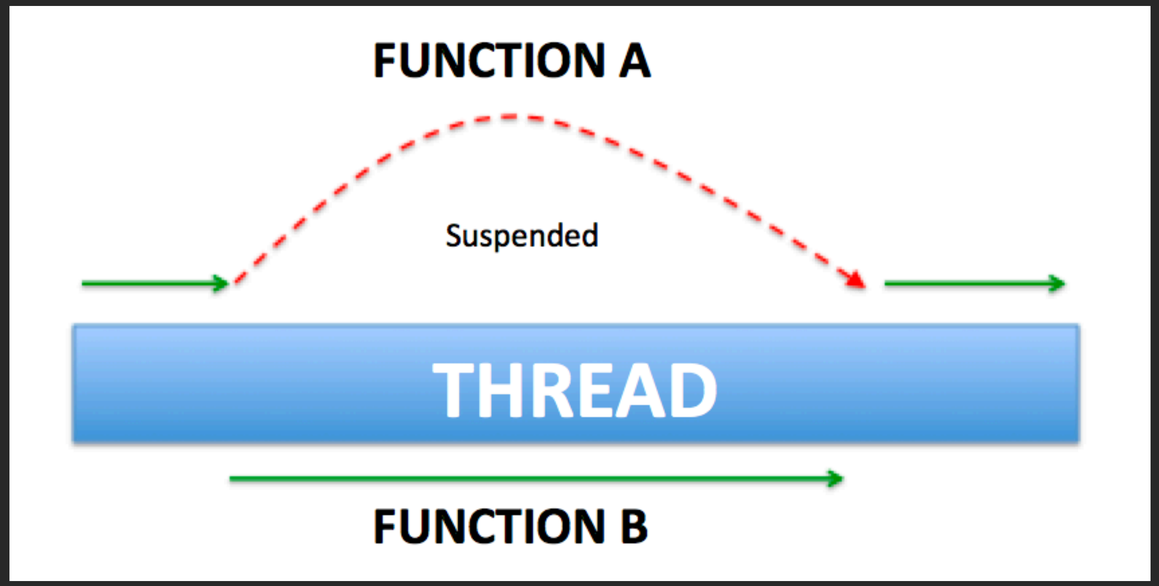
```
runBlocking {
    text = "hei2"
    delay(1000)
}
```

```
var text = "hei"
GlobalScope.launch {
    text = "hei2"
    delay(1000)
}
println(text)
Thread.sleep(2000)
println(text)
```

```
runBlocking {
    text = "hei2"
    delay(1000)
}
```

```
suspend fun api(): String = withContext(Dispatchers.IO) {
    "hallo"
}
```





https://medium.com/@elye.project/understanding-suspend-function-of-coroutines-de26b070c5ed

Oppgave 13

Coroutines