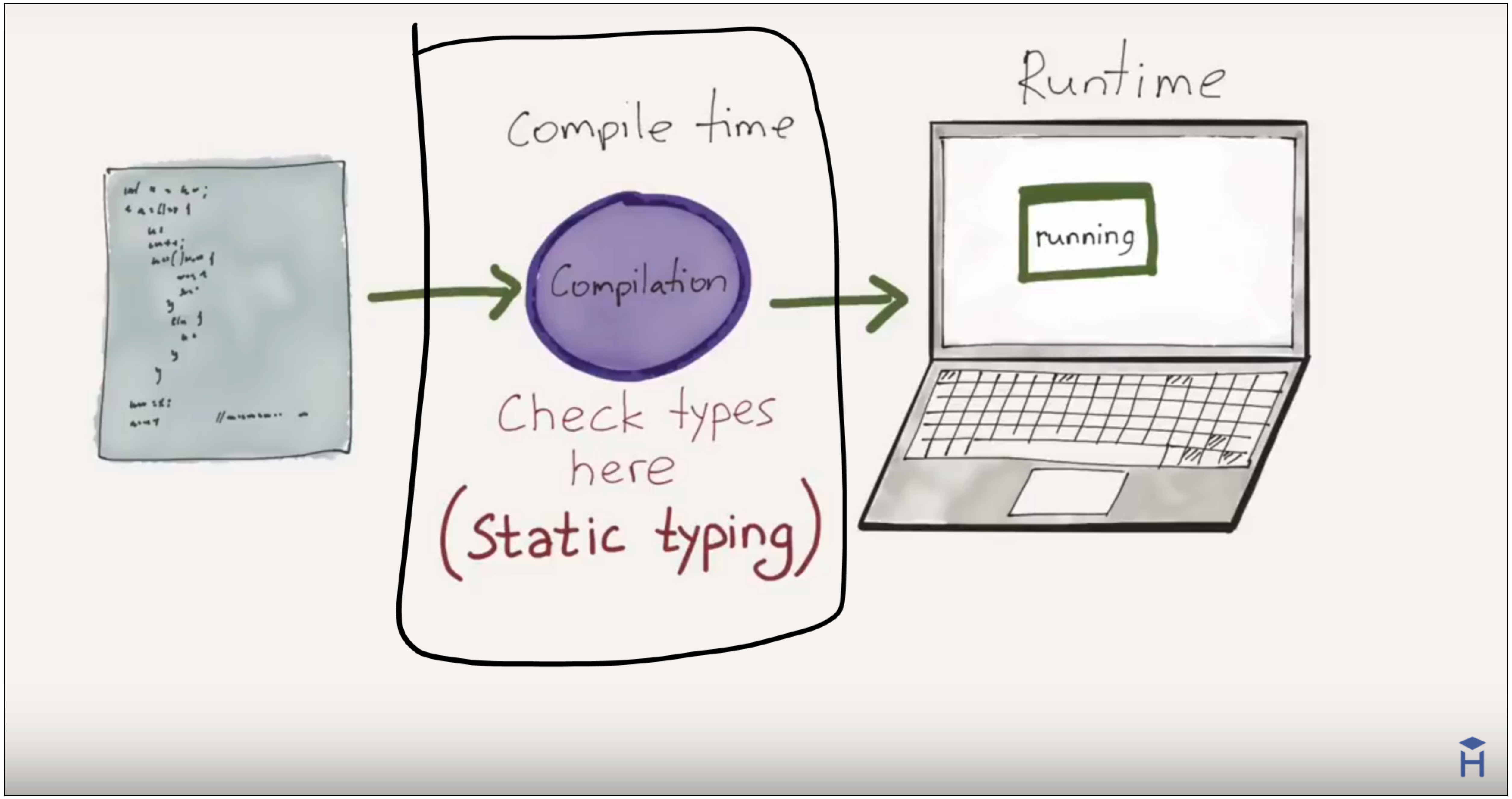
Kotlin: Types & Null



STATIC TYPING

“Variable declarations are mandatory before usage, else results in a compile-time error"

Static Typing – Example

```
String greeting = "Hello!";
int someRandomInteger = 100;
double aDoubleVariable = 2.2;
```

A type is assigned to each variable.

In Java, if we don't assign a type, we get a compiler error
→ Java is statically typed.

Types determine the operations we can perform on the variables.

https://howtoprogramwithjava.com/dynamic-typing-vs-static-typing/

Static Typing – Example

In Kotlin, you don't have to specify the type of each variable explicitly, even though Kotlin ***is*** *statically-typed.*

Here, Kotlin determines the type from the initialisation.

```
fun main(args : Array<String>)
{
    var someRandomInteger = 100
    var aDoubleVariable = 2.2
    println (someRandomInteger)
    println (aDoubleVariable)
}
```

http://petersommerhoff.com/dev/kotlin/kotlin-for-java-devs/#type-inference

Static Typing – Example

However, you can choose to explicitly define a data type.

```
fun main(args : Array<String>)
{
    var someRandomInteger : Int = 100
    var aDoubleVariable : Double = 2.2
    println (someRandomInteger)
    println (aDoubleVariable)
}
```

Static Typing – Example

Kotlin

With Kotlin, you have to either define a type or initialise the variable (kotlin then determines the type!).

```
fun main(args : Array<String>)
{
    var someRandomInteger //compile error
    var aDoubleVariable : Double = 2.2
    println (someRandomInteger)
    println (aDoubleVariable)
}
```

http://petersommerhoff.com/dev/kotlin/kotlin-for-java-devs/#type-inference

Static Typing – Example

```
fun main(args : Array<String>)
{
   var someRandomInteger : Int = 100
   var aDoubleVariable : Double = 2.2

   someRandomInteger = 2.65     //compile error
   aDoubleVariable = 233        //compile error

   println (someRandomInteger)
   println (aDoubleVariable)
}
```

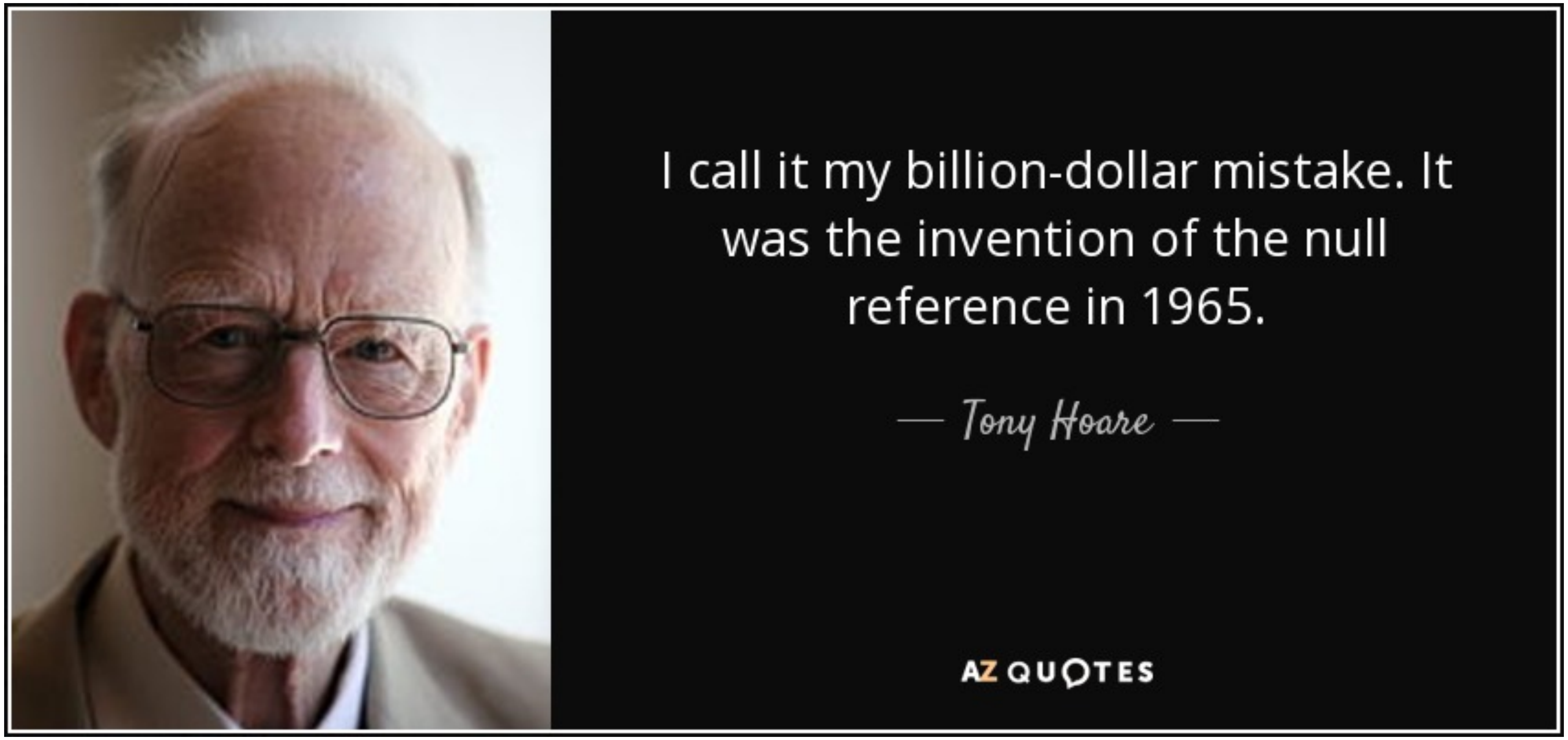
http://petersommerhoff.com/dev/kotlin/kotlin-for-java-devs/#type-inference

Kotlin

- runs on Java Virtual Machine.
- is an evolution of the Java syntax but is more concise and has cleaner syntax.
- is not syntax compatible with Java; but is interoperable with Java.
- relies on some Java Class Libraries e.g. Collections framework.
- is a statically-typed programming language.
- offers null safety.

Based on : https://en.wikipedia.org/wiki/Kotlin_(programming_language)

Null – Billion Dollar Mistake



I call it my billion-dollar mistake. It was the invention of the null reference in 1965.

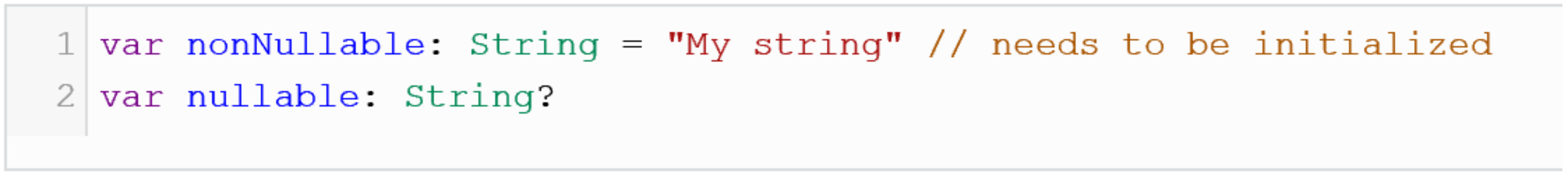
— Tony Hoare —

AZ QUOTES

Kotlin and Null Safety

- Kotlin eliminates most sources of null references by making all types non-nullable by default — meaning that the compiler won't let you use a non-initialized, non-nullable variable.

- If you need a variable to hold a null value, you have to declare the type as nullable, adding a question mark after the type (more on this in later lectures).

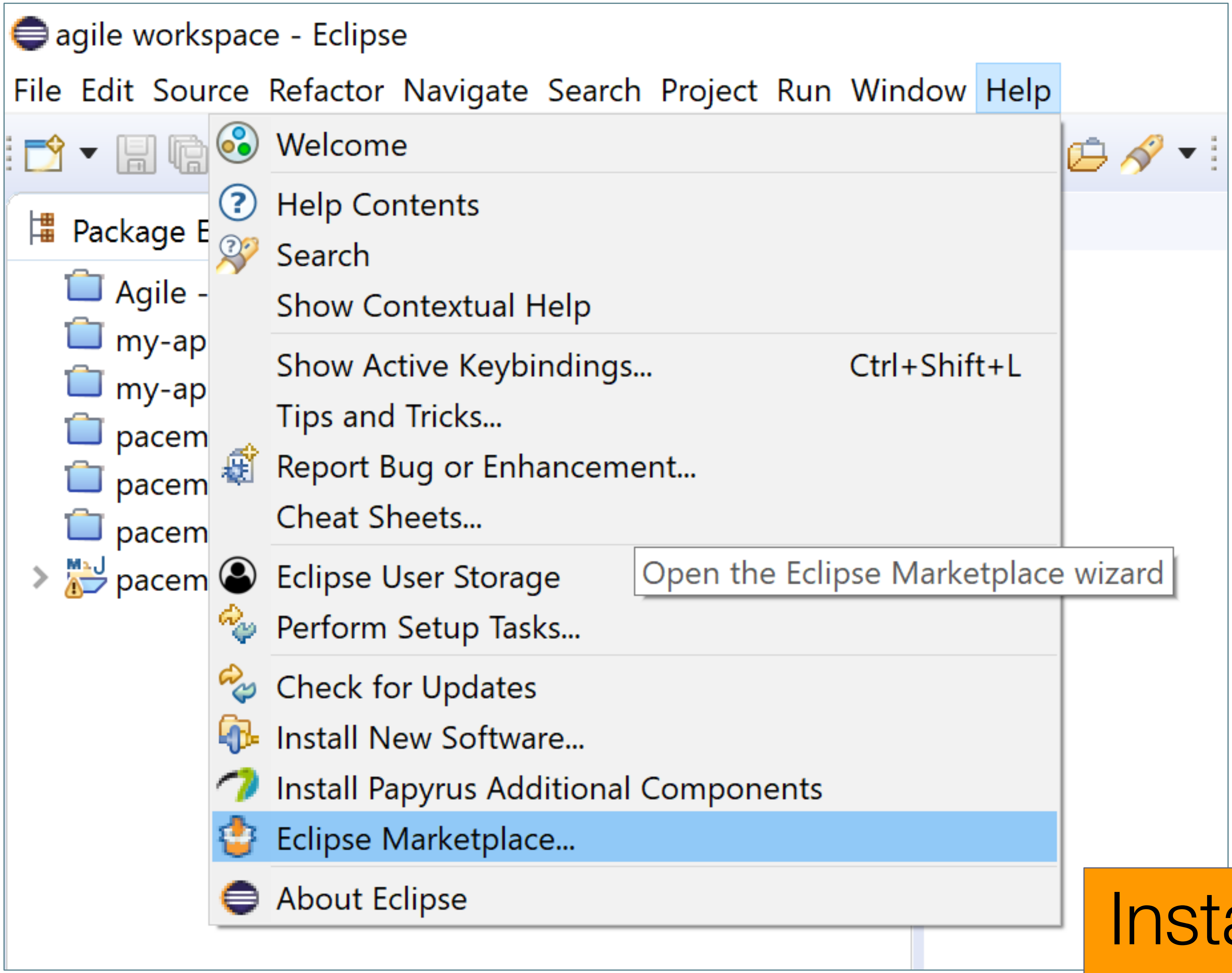


```kotlin
var nonNullable: String = "My string" // needs to be initialized
var nullable: String?
```

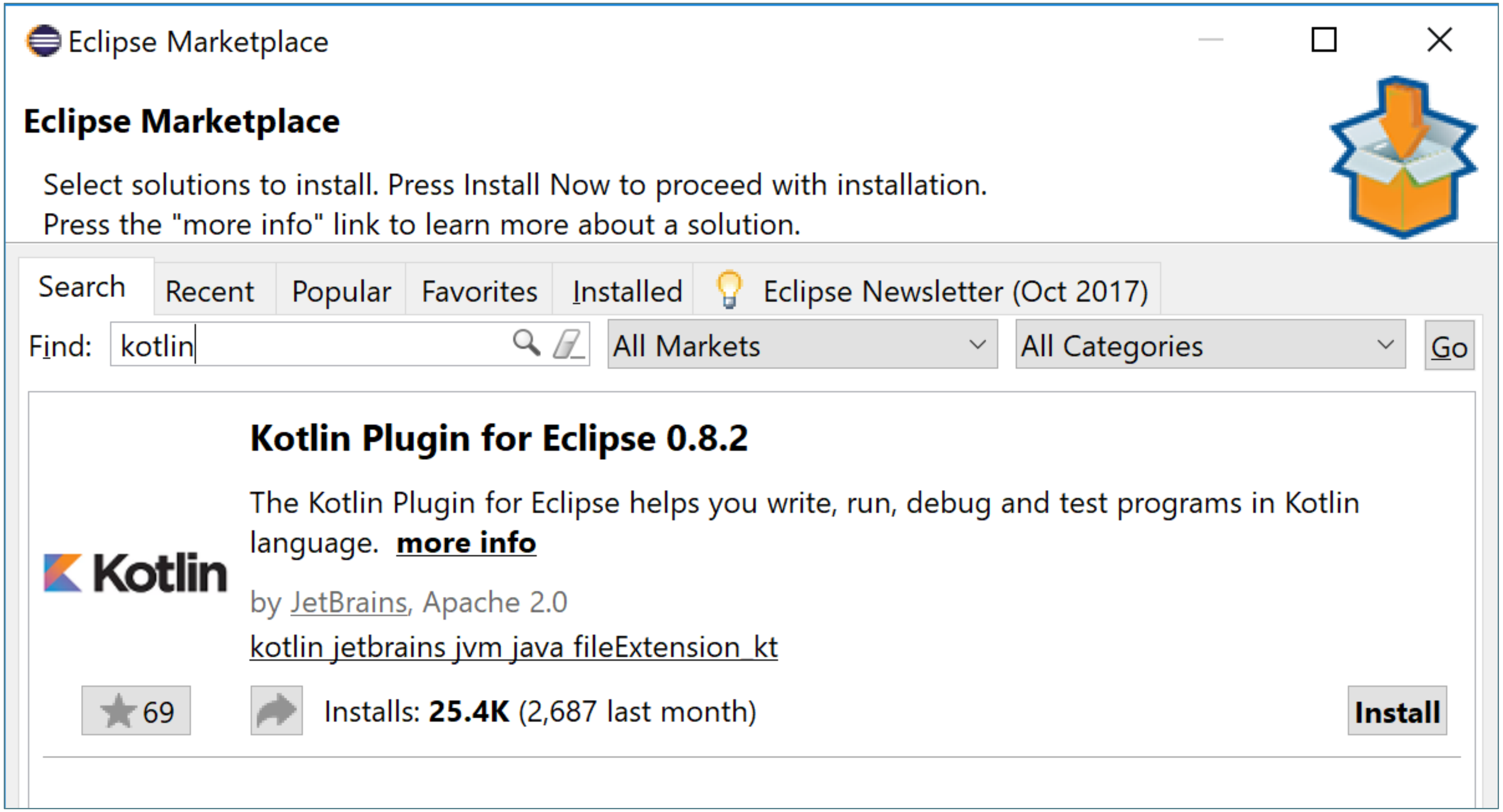
[Source: https://dzone.com/articles/kotlin-vs-java-first-impressions-using-kotlin-for](https://dzone.com/articles/kotlin-vs-java-first-impressions-using-kotlin-for)

Kotlin

eclipse



Install Kotlin plugin into Eclipse



Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

Kotlin Plugin for Eclipse 0.8.2

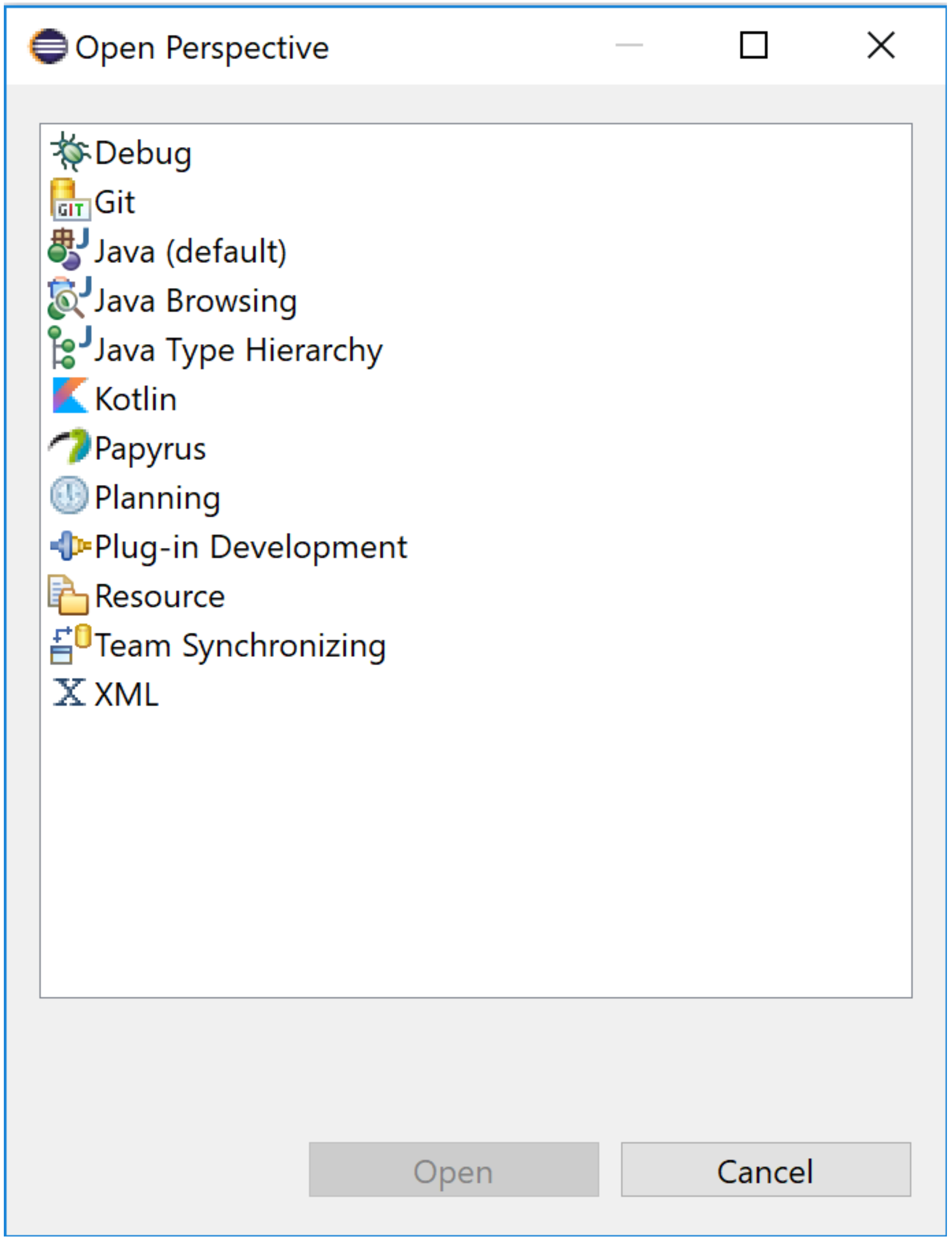
The Kotlin Plugin for Eclipse helps you write, run, debug and test programs in Kotlin language. **more info**

by JetBrains, Apache 2.0

kotlin jetbrains jvm java fileExtension_kt

Installs: **25.4K** (2,687 last month)

Install Kotlin plugin into Eclipse

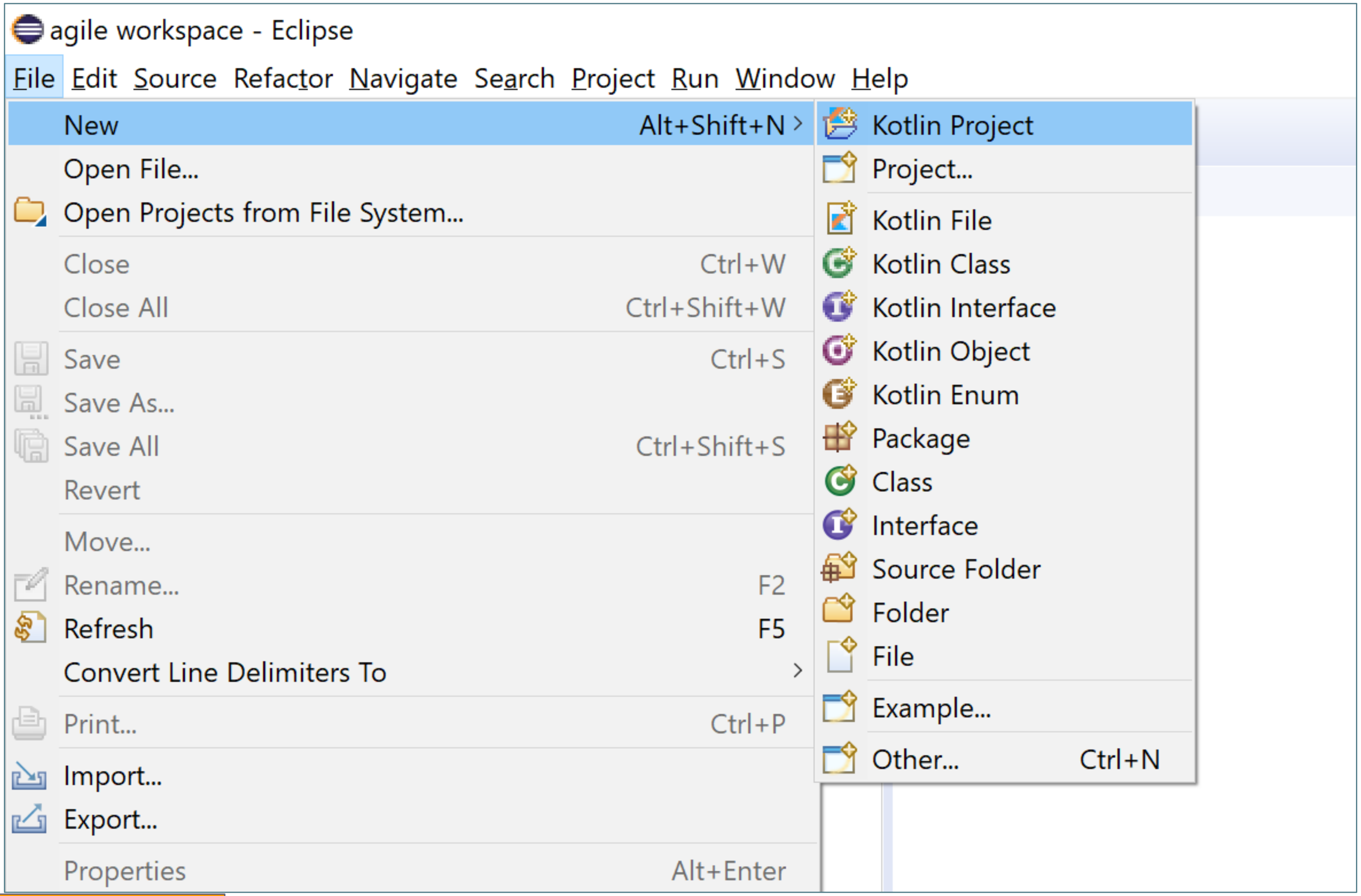


Open Perspective

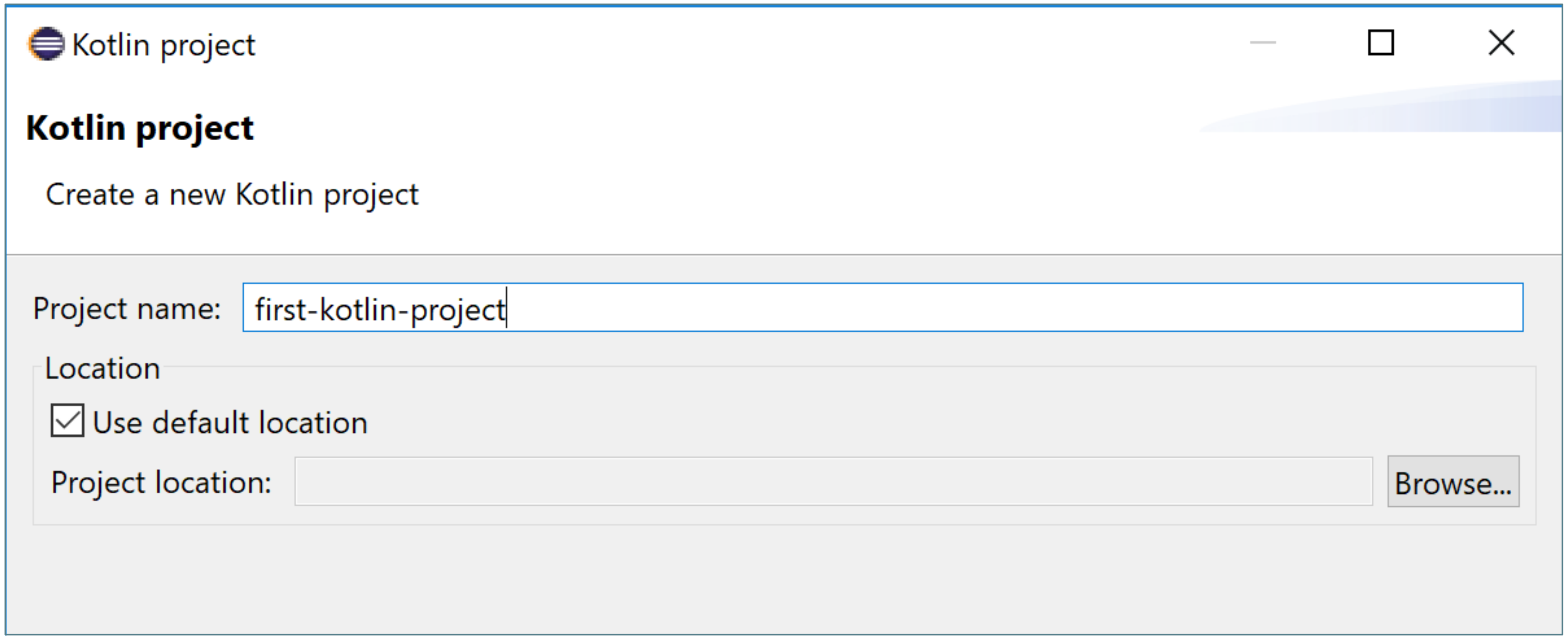
- Debug
- Git
- Java (default)
- Java Browsing
- Java Type Hierarchy
- Kotlin
- Papyrus
- Planning
- Plug-in Development
- Resource
- Team Synchronizing
- XML

Change perspective to Kotlin.

Install Kotlin plugin into Eclipse



Create a new Kotlin project



Kotlin project

Kotlin project

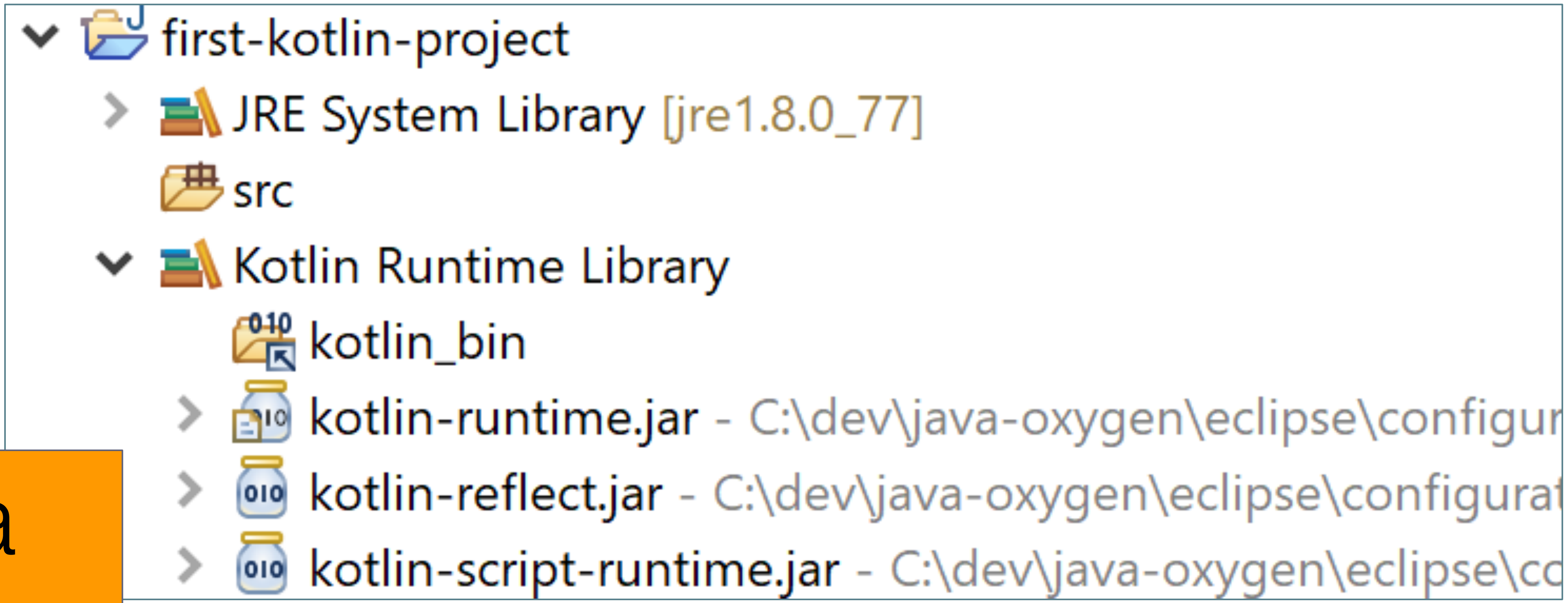
Create a new Kotlin project

Project name: first-kotlin-project

Location

☑ Use default location

Project location:



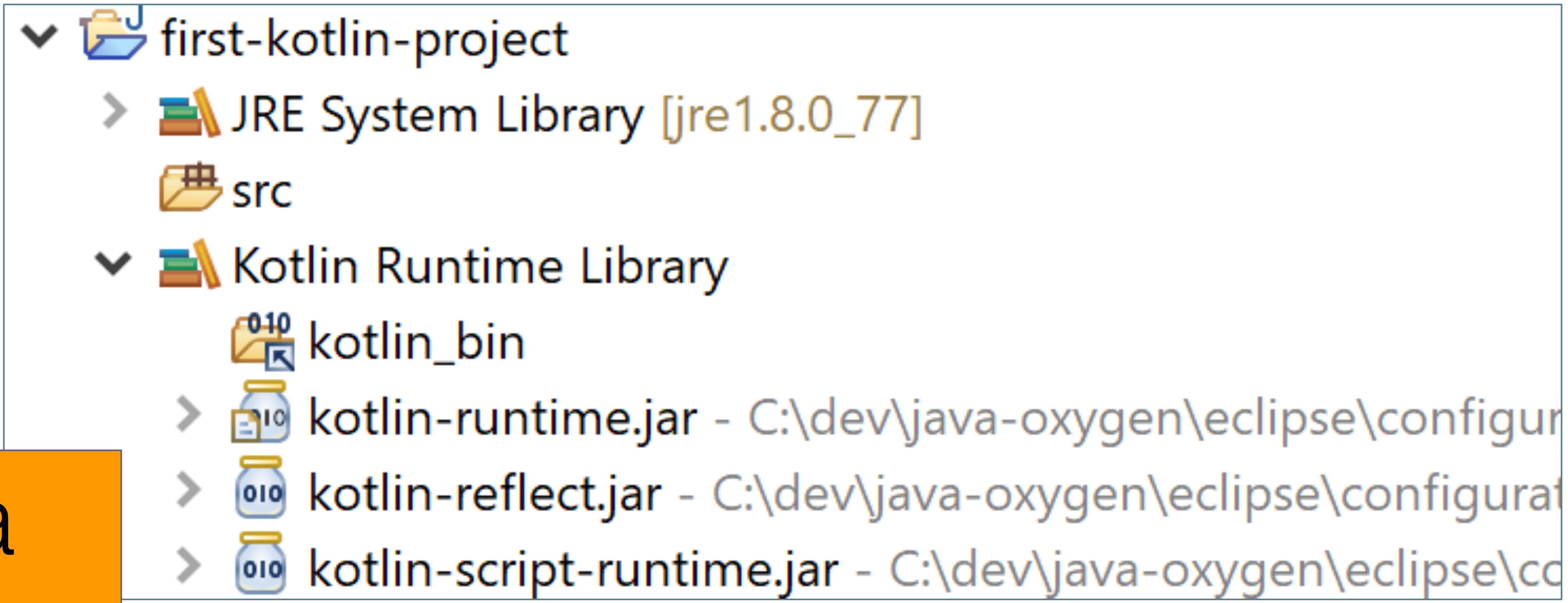
first-kotlin-project
- JRE System Library [jre1.8.0_77]
- src
- Kotlin Runtime Library
  - kotlin_bin
  - kotlin-runtime.jar - C:\dev\java-oxygen\eclipse\configur
  - kotlin-reflect.jar - C:\dev\java-oxygen\eclipse\configurat
  - kotlin-script-runtime.jar - C:\dev\java-oxygen\eclipse\cc

Create a new Kotlin project

Eclipse project is also a Java project with a:

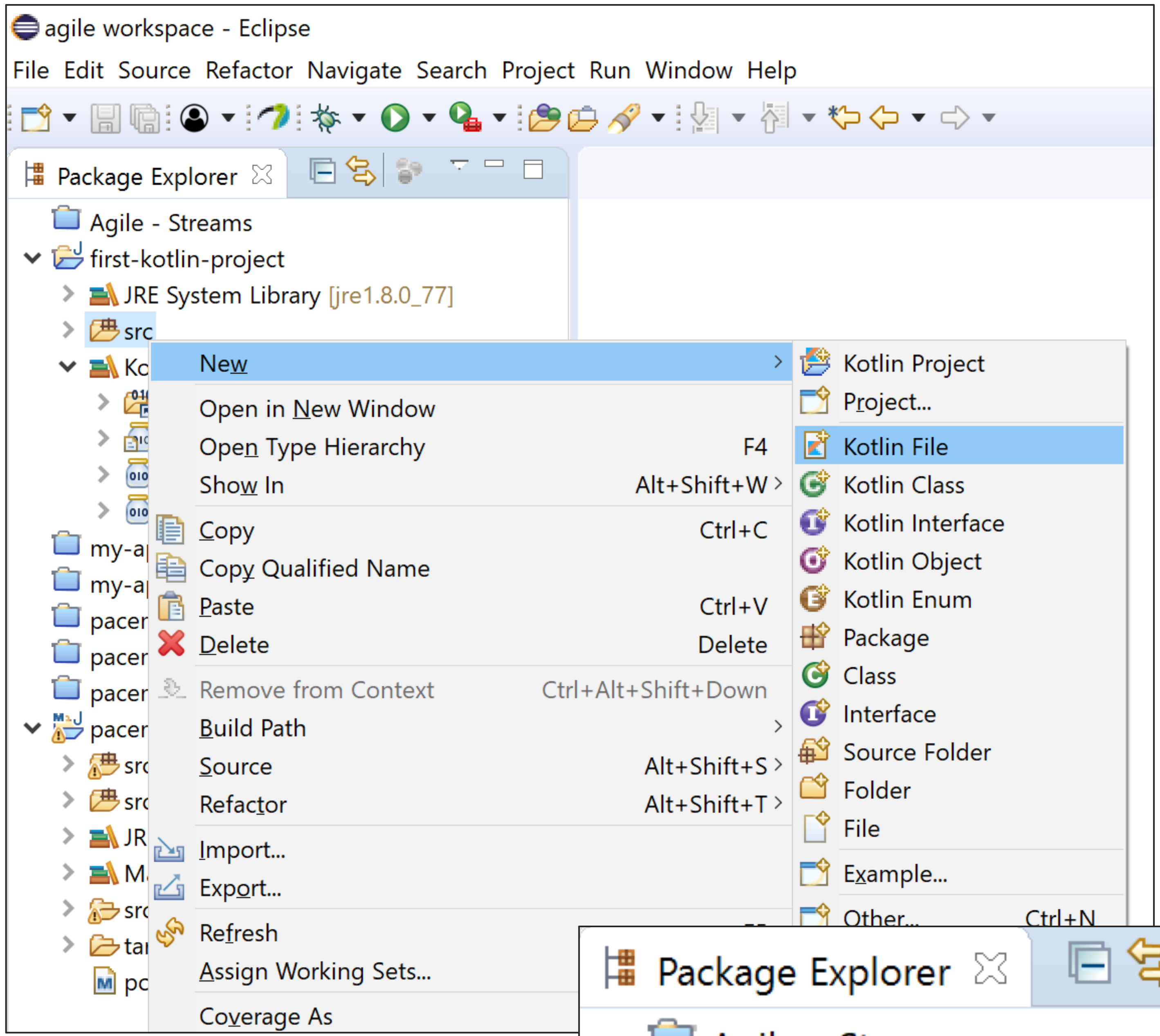
- Kotlin Builder and
- Kotlin Runtime Library.

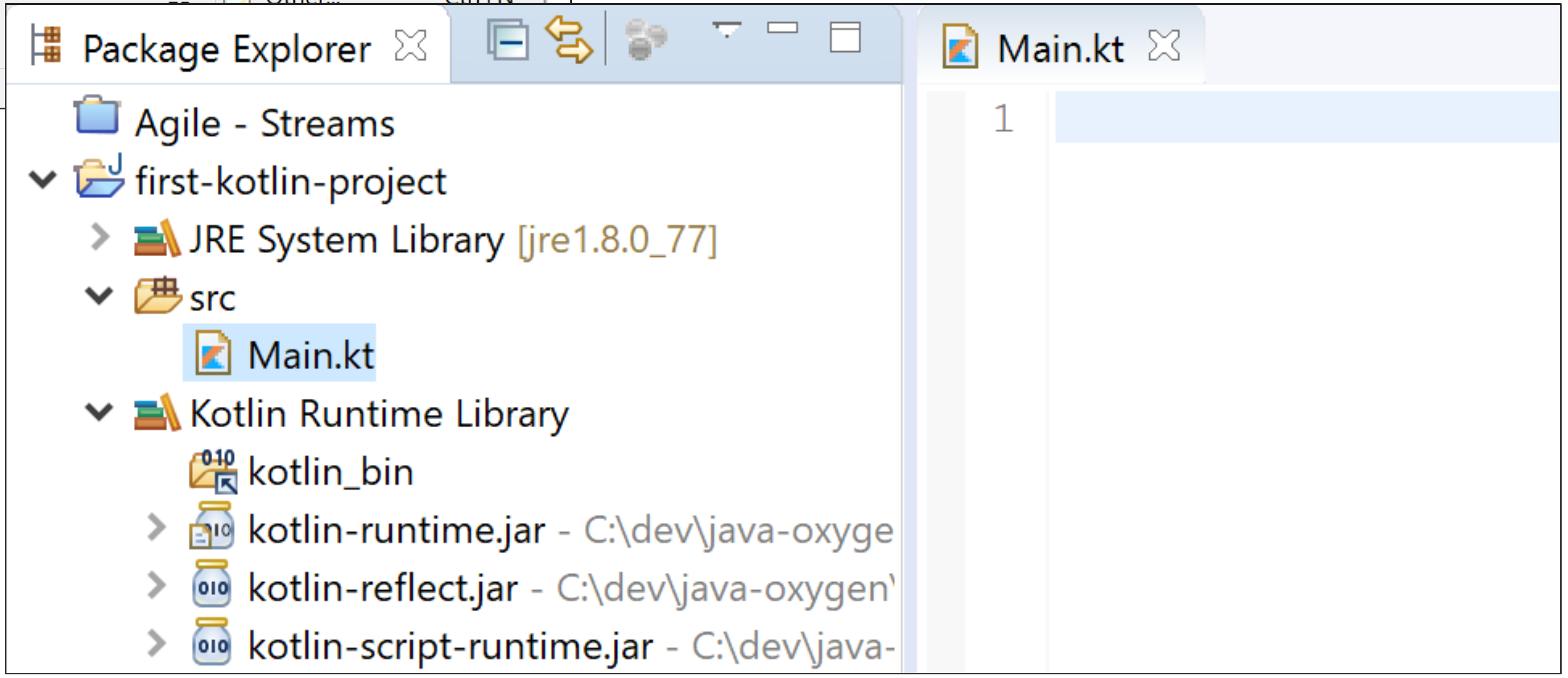
→ Can add Java classes to the project, mixing and matching Kotlin and Java code where required.

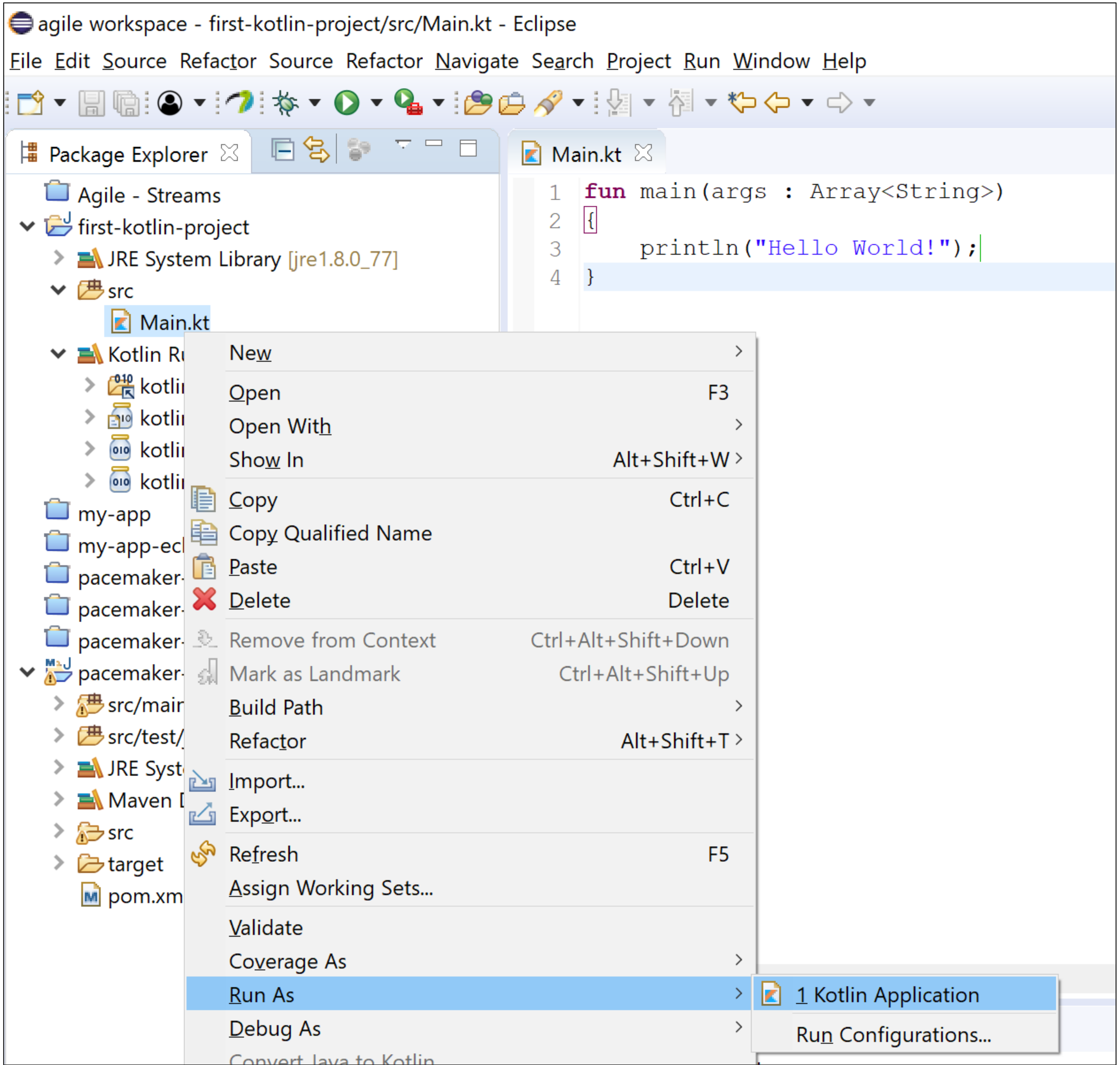

```
first-kotlin-project
  JRE System Library [jre1.8.0_77]
  src
  Kotlin Runtime Library
    kotlin_bin
    kotlin-runtime.jar - C:\dev\java-oxygen\eclipse\configur
    kotlin-reflect.jar - C:\dev\java-oxygen\eclipse\configurat
    kotlin-script-runtime.jar - C:\dev\java-oxygen\eclipse\cc
```

Create a new Kotlin project

Create a new Kotlin file



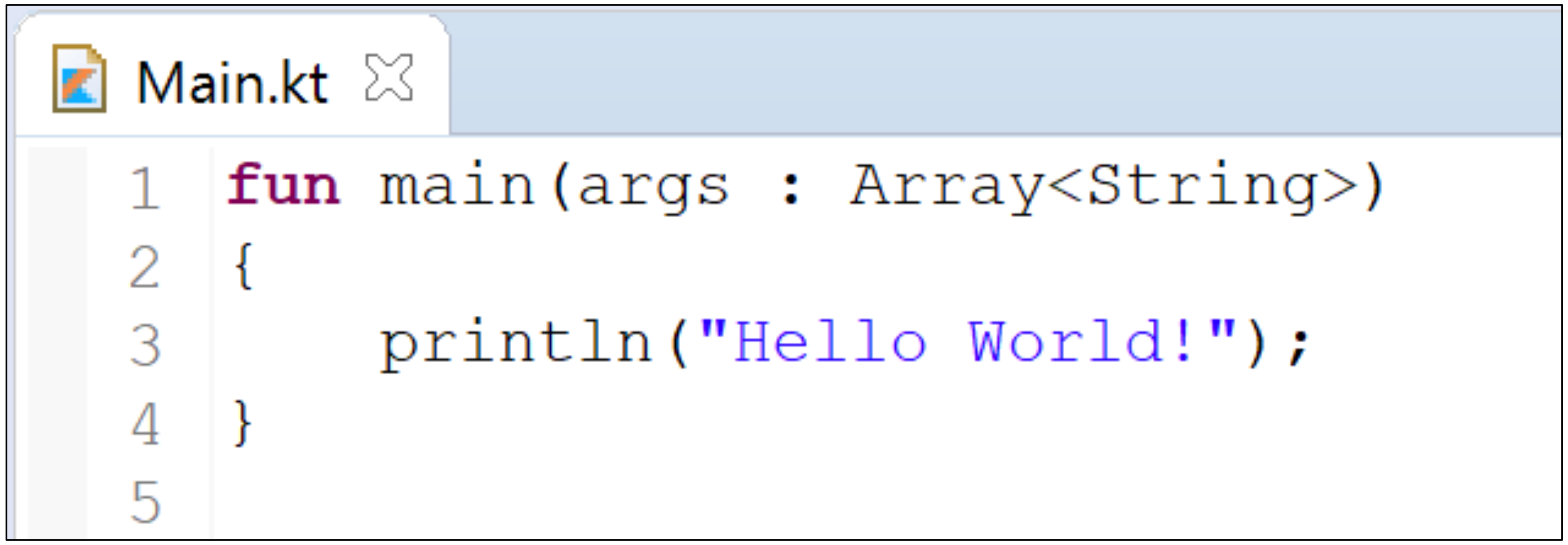

Package Explorer
Main.kt
Agile - Streams
first-kotlin-project
JRE System Library [jre1.8.0_77]
src
Main.kt
Kotlin Runtime Library
kotlin_bin
kotlin-runtime.jar - C:\dev\java-oxyge
kotlin-reflect.jar - C:\dev\java-oxygen
kotlin-script-runtime.jar - C:\dev\java-



agile workspace - first-kotlin-project/src/Main.kt - Eclipse

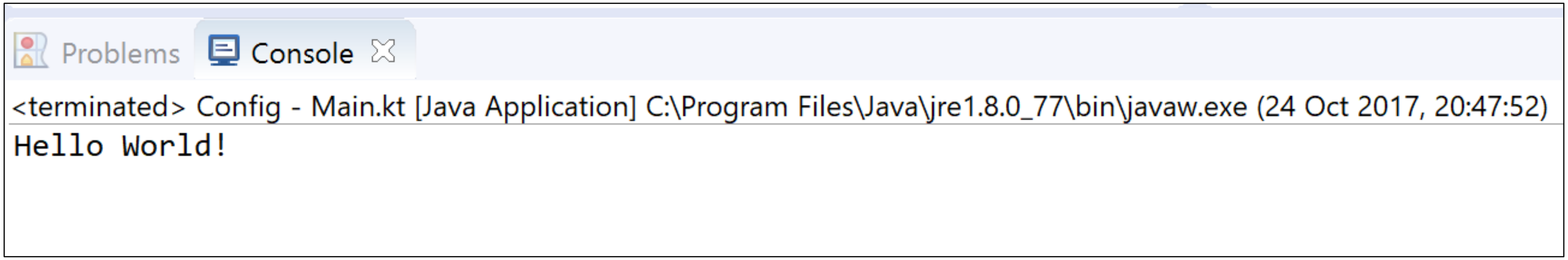
```
fun main(args : Array<String>)
{
    println("Hello World!");
}
```

Hello World!



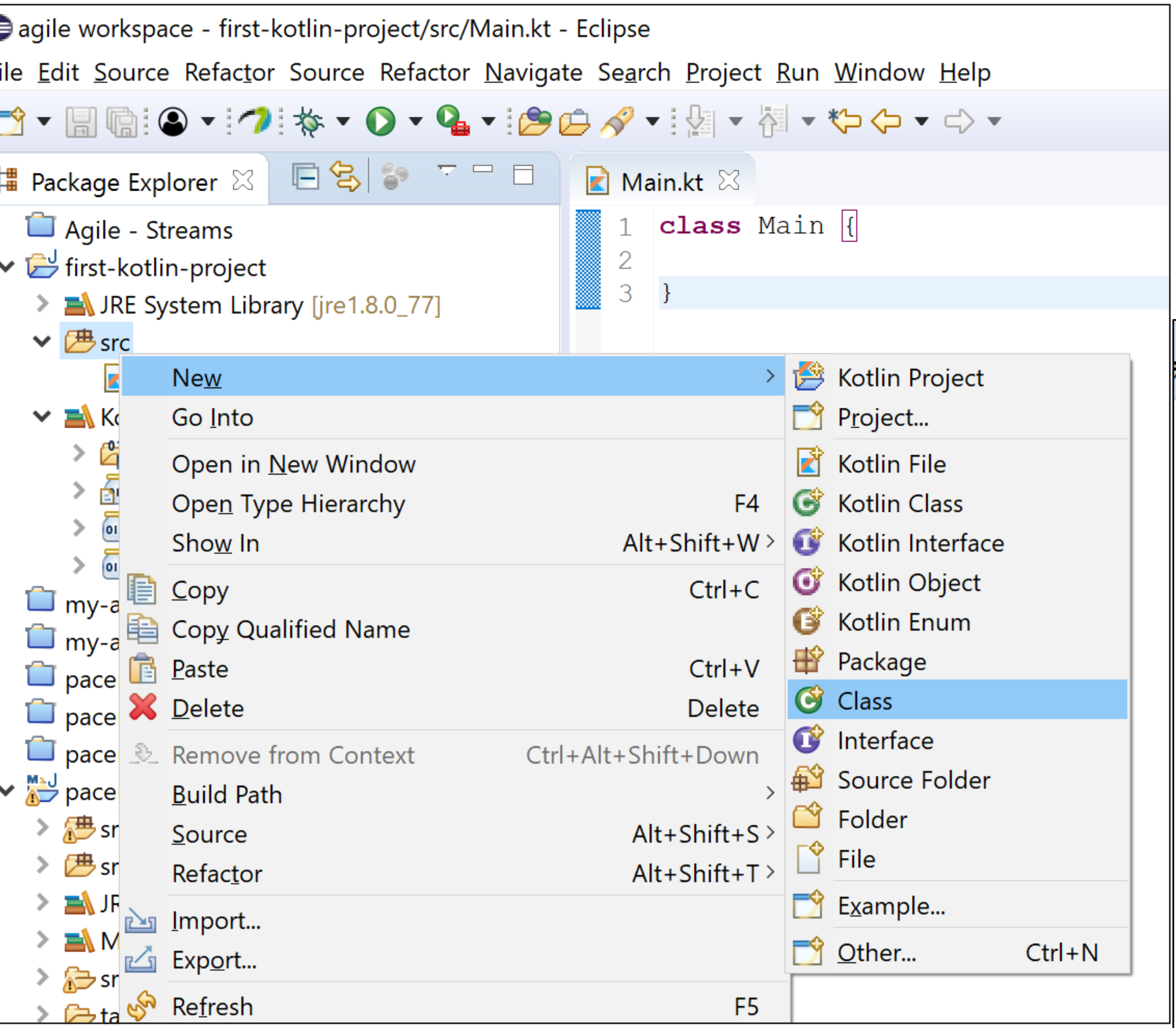
```
fun main(args : Array<String>)
{
    println("Hello World!");
}

```

Hello World!

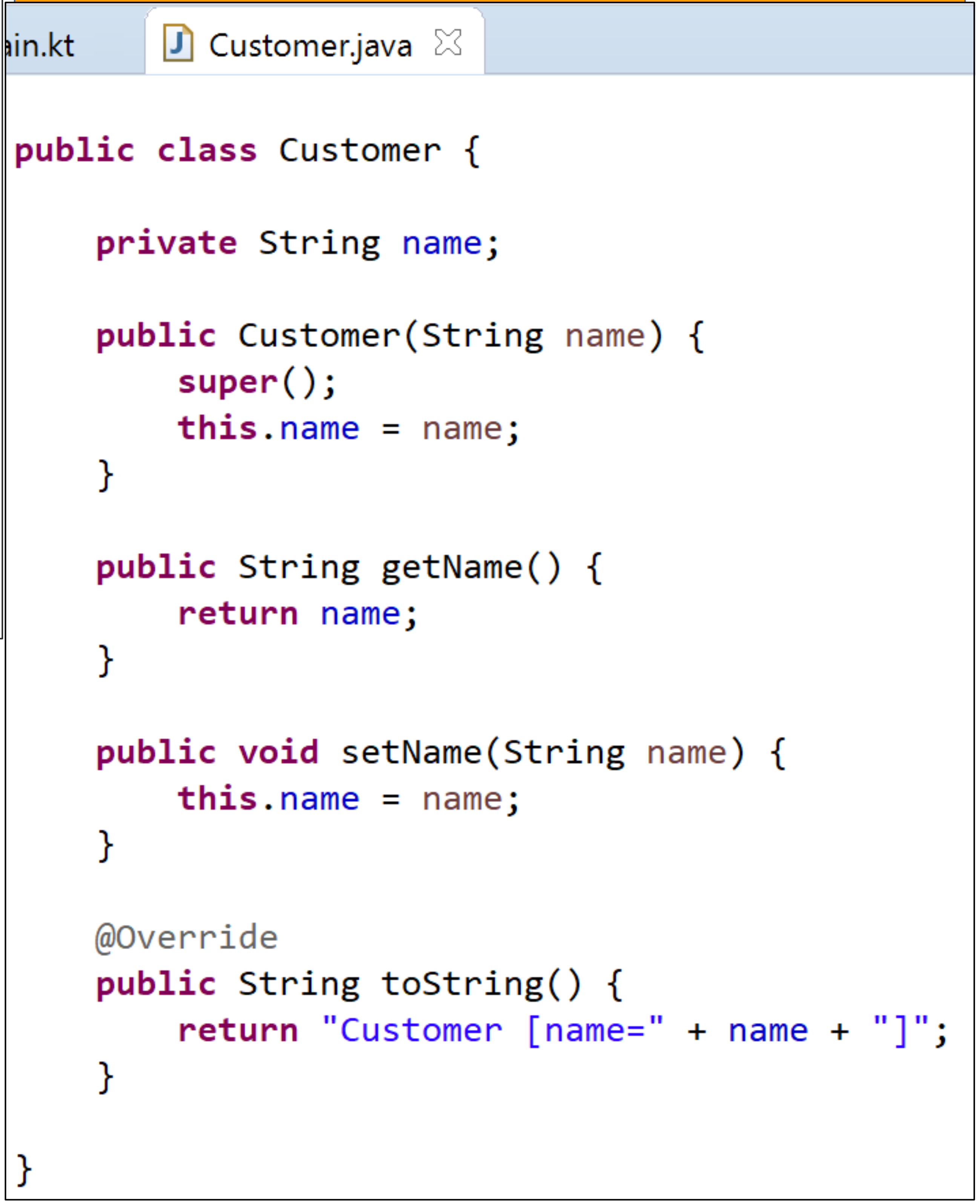


```
<terminated> Config - Main.kt [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (24 Oct 2017, 20:47:52)
Hello World!
```

Interoperability: Create a new Java class

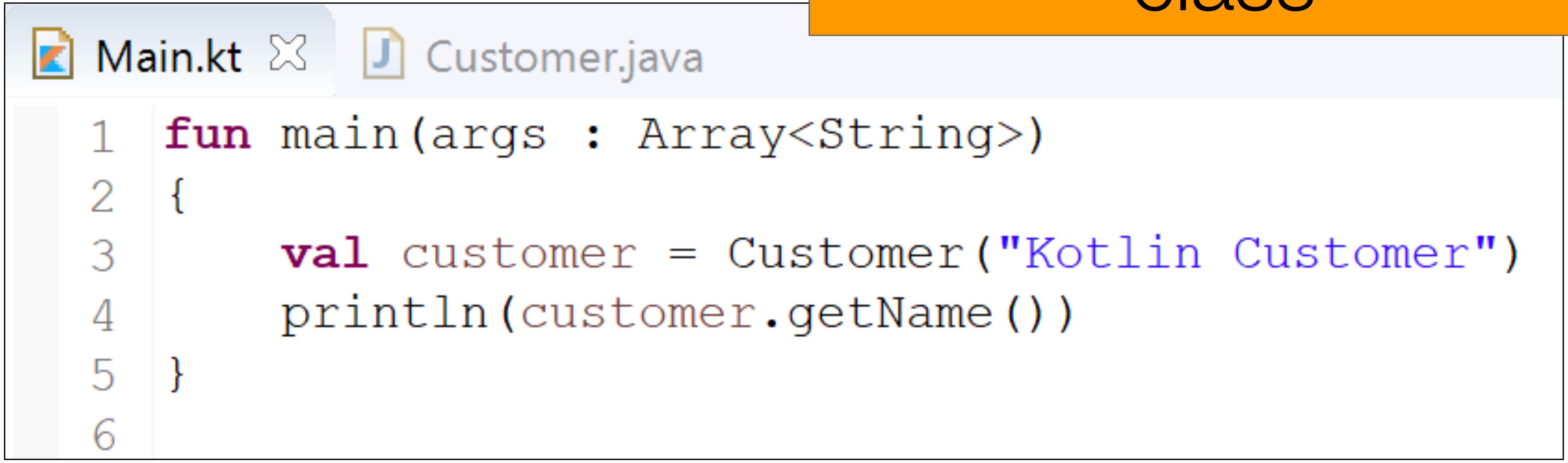


```
class Main {

}
```

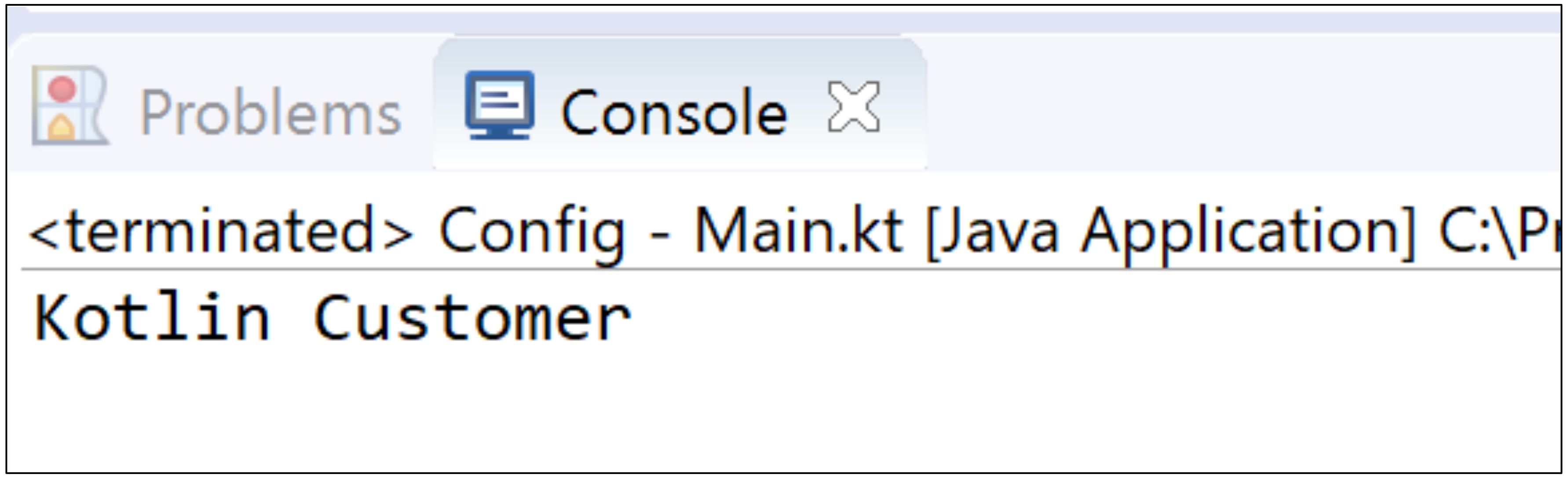


```
public class Customer {

    private String name;

    public Customer(String name) {
        super();
        this.name = name;
    }

    public String getName() {
        return name;
    }

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

    @Override
    public String toString() {
        return "Customer [name=" + name + "]";
    }

}
```

Interoperability: Create a new Java class

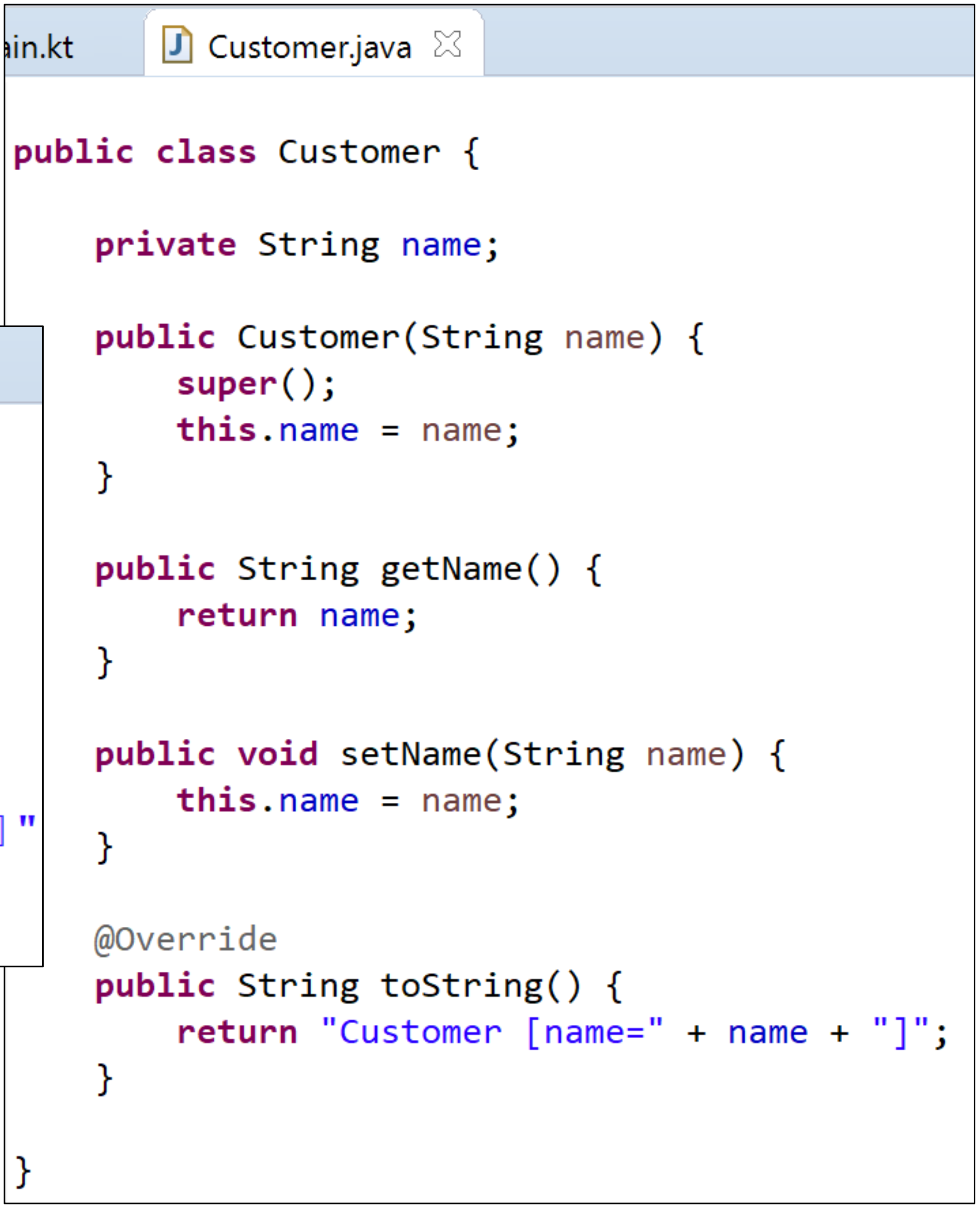


```
fun main(args : Array<String>)
{
     val customer = Customer("Kotlin Customer")
     println(customer.getName())
}
```



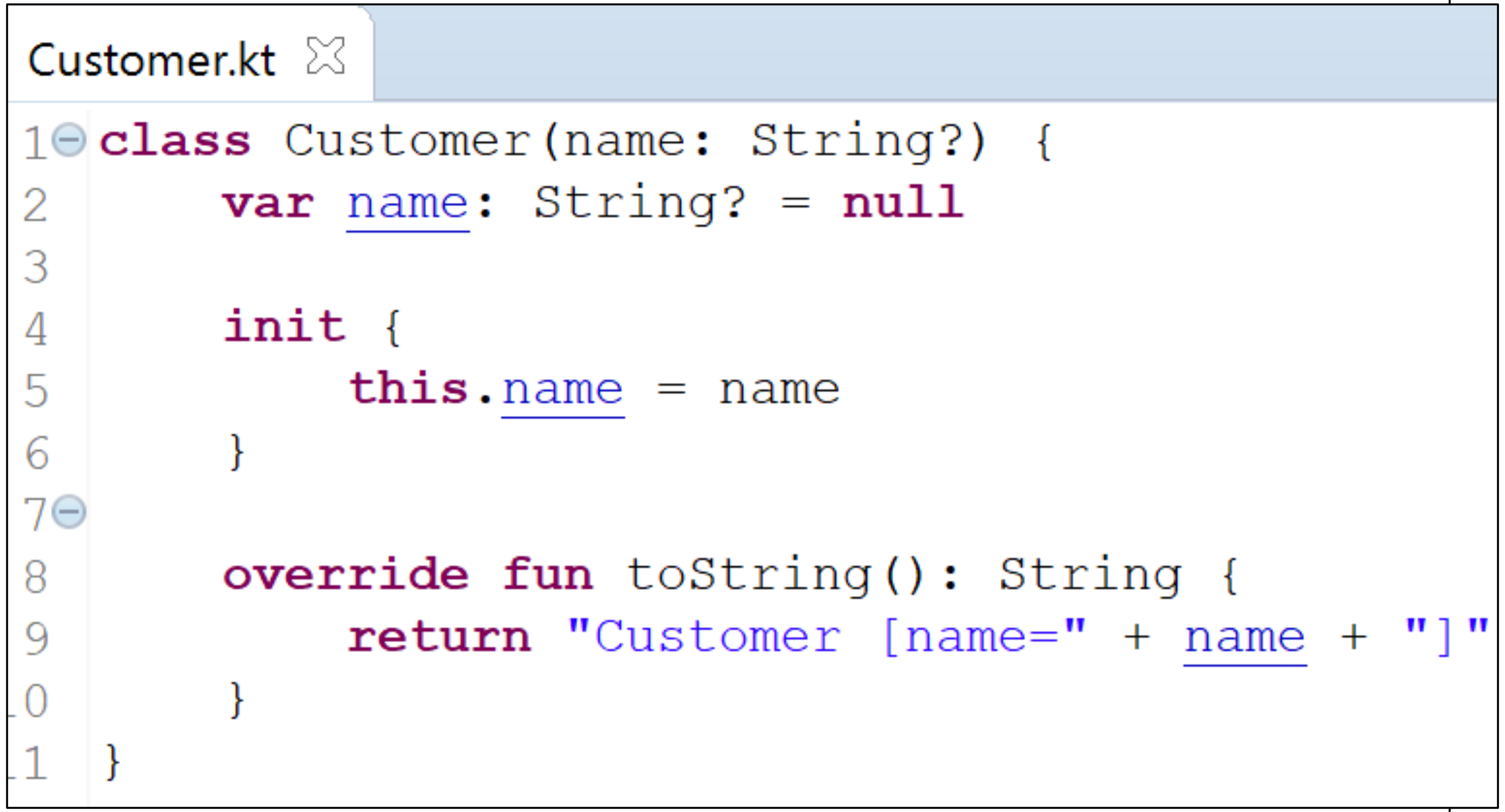
```
<terminated> Config - Main.kt [Java Application] C:\Pr
Kotlin Customer
```

Kotlin Vs Java



ain.kt | Customer.java

```java
public class Customer {

    private String name;

    public Customer(String name) {
        super();
        this.name = name;
    }

    public String getName() {
        return name;
    }

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

    @Override
    public String toString() {
        return "Customer [name=" + name + "]";
    }

}
```



Customer.kt

```kotlin
class Customer(name: String?) {
    var name: String? = null

    init {
        this.name = name
    }

    override fun toString(): String {
        return "Customer [name=" + name + "]"
    }
}
```