



Introduction to Kotlin

About me



Trayan Iliev

- CEO of IPT – Intellectual Products & Technologies
<http://www.iproduct.org>
- Oracle® certified programmer 15+ Y
- end-to-end reactive fullstack apps with [Java](#), [ES6+](#), [TypeScript](#), [Angular](#), [React](#) and [Vue.js](#)
- 12+ years IT trainer: [Spring](#), [Java EE](#), [Node.js](#), [Express](#), [GraphQL](#), [SOA](#), [REST](#), [DDD](#) & [Reactive Microservices](#)
- Voxxed Days, jPrime, Java2Days, jProfessionals, BGOUG, BGJUG, DEV.BG speaker
- Organizer RoboLearn hackathons and IoT enthusiast

Packages and Imports

Package declarations, imports and default imports, static imports, classes and interfaces, visibility modifiers, modules



Packages

```
package course.kotlin
```

```
fun printMessage() { /*...*/}  
class Message { /*...*/}  
// ...
```

- All the contents, such as classes and functions, of the source file are included in this package. So, in the example above, the full name of `printMessage()` is `org.example.printMessage`, and the full name of `Message` is `org.example.Message`
- If the package is not specified, the contents of such a file belong to the `default package` with no name.

Default imports

Kotlin packages imported by default:

`kotlin.*`

`kotlin.annotation.*`

`kotlin.collections.*`

`kotlin.comparisons.*`

`kotlin.io.*`

`kotlin.ranges.*`

`kotlin.sequences.*`

`kotlin.text.*`

JVM packages imported by default:

`java.lang.*`

`kotlin.jvm.*`

JS packages imported by default:

`kotlin.js.*`

Imports

- The import keyword is not restricted to importing classes – you can also use it to import other declarations:
 - top-level **functions** and **properties**
 - **functions** and **properties** declared in **object declarations**
 - **enum** constants
- If a **top-level declaration** is marked **private**, it is private to the **file it's declared in**

```
package course.kotlin.other
import course.kotlin.Message // Message is now accessible without qualification
import course.kotlin.* // everything in 'course.kotlin' becomes accessible
import course.kotlin.Message2 // Message is accessible
import course.kotlin.Message as MyMessage // MyMessage for 'course.kotlin.Message'
```

Visibility modifiers

- **public** – used by default, which means that your declarations will be visible everywhere;
- **private** – only visible inside the file that contains the declaration.
- **internal** – visible everywhere in the same module
- protected modifier is **not available for top-level declarations**.

```
private fun foo() { } // visible inside packages.kt
```

```
public var bar: Int = 5 // property is visible everywhere  
    private set         // setter is visible only in packages.kt
```

```
internal val baz = 6 // visible inside the same module
```


Modules

- The **internal** visibility modifier means that the member is visible within the same **module**. More specifically, a **module is a set of Kotlin files compiled together**, for example:
 - An IntelliJ IDEA module.
 - A Maven project.
 - A Gradle source set (with the exception that the test source set can access the internal declarations of main).
 - A set of files compiled with one invocation of the `<kotlinc>` Ant task.

Class members visibility modifiers

- **private** – the member is visible inside this class only (including all its members);
- **protected** – the member has the same visibility as one marked as private, but that it is also visible in subclasses;
- **internal** – any client inside this module who sees the declaring class sees its internal members;
- **public** – any client who sees the declaring class sees its public members.

Class members visibility modifiers

```
open class Outer {  
    private val a = 1  
    protected open val b = 2  
    internal open val c = 3  
    val d = 4 // public by default  
  
    protected class Nested {  
        public val e: Int = 5  
    }  
}
```

```
class Unrelated(o: Outer) {  
    // o.a, o.b are not visible  
    // o.c and o.d are visible (same module)  
    // Outer.Nested is not visible, and Nested::e is not visible either  
}
```

```
class Subclass : Outer() {  
    // a is not visible  
    // b, c and d are visible  
    // Nested and e are visible  
  
    override val b = 5 // 'b' is protected  
    override val c = 7 // 'c' is internal  
}
```

And solve programming problem:

<https://codeforces.com/contest/1157/problem/B>

Learn Kotlin by Example & Kotlin idioms

<https://play.kotlinlang.org/byExample/>

<https://kotlinlang.org/docs/idioms.html>

Thank's for Your Attention!



Trayan Iliev

IPT – Intellectual Products & Technologies

<http://iproduct.org/>

<https://github.com/iproduct>

<https://twitter.com/trayaniliev>

<https://www.facebook.com/IPT.EACAD>