ToiletPaper #2

jambit
WHERE INNOVATION WORKS

Kotlin: Better than Java?

By Michael Hoffmann and Andreas Scharf

✕ Problem

Java is one of the most commonly used programming languages in the world. Unfortunately, it is not a modern programming language, even though Java 8 is a step in the right direction. Java code is susceptible to NullPointerExceptions and demands a lot of typing from the programmer for small tasks (keyword: boilerplate code). In certain Android or Java projects, it may be the case that the latest Java version is not yet used and you have to rely on alternatives.

✓ Solution

The developers of JetBrains (known for the IDE IntelliJ) were bothered by the properties of Java and, without further ado, created their own programming language called “Kotlin”. The name comes from a small island in the Gulf of Finland.

Kotlin was conceived as a general-purpose language and takes on a large portion of the syntax of Java. Consequently, it is object-oriented, statically typed and provides generics. The source code is converted into Java bytecode by the compiler and thus runs on any Java Virtual Machine (JVM). As a result, Kotlin is suitable for use with familiar frameworks such as Spring or Hibernate, as well as for Android development. The language provides 100% interoperability with Java, which means that it is possible to call up Java code from Kotlin code and vice-versa. The language has been licensed under Apache 2.0 since February 2012.

At the same time as the language, JetBrains developed the Kotlin support for IntelliJ. Besides, an Eclipse plug-in is available. Tried-and-tested build tools, such as Maven, Gradle or Ant, can be used for the development.

➔ Example

Here a comparison between the Kotlin and Java syntax:

```kotlin
// Kotlin
fun main(args: Array<String>) {
       println("Hello, World!")
}
```

```java
// Java
public class App {
      public static void main(String[] args) {
      System.out.println("Hello, World!");
      }
}
```

The following code line creates a POJO in Kotlin, inclusive getter, setter, equals(), hashCode(), toString() und copy():

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

Tony Hoare, the inventor of the null reference, now refers to his invention as a billion-dollar mistake because programming is susceptible to errors at null, which has resulted in high costs. Since Java 8, the optional class, which is familiar from other programming languages, has been available as a solution. Kotlin offers a more elegant solution: an explicit distinction is made between optional types which may contain null and the normal types with a mandatory value. The computer complains if you make an untested reference to optional types:

```kotlin
//val o: String = null  // Compilerfehler
val s: String? = null

//s.hashCode()          // Compilerfehler
s?.hashCode()           // Safecall
```

✚ Further aspects

- Official website: https://kotlinlang.org/
- Browser playground: http://try.kotlinlang.org/