OKAN ÜNİVERSİTESİ

İSTANBUL

INTRODUCTION TO KOTLIN

Levent YILDIZ
http://www.leventyildiz.com.tr
https://github.com/lvntyldz

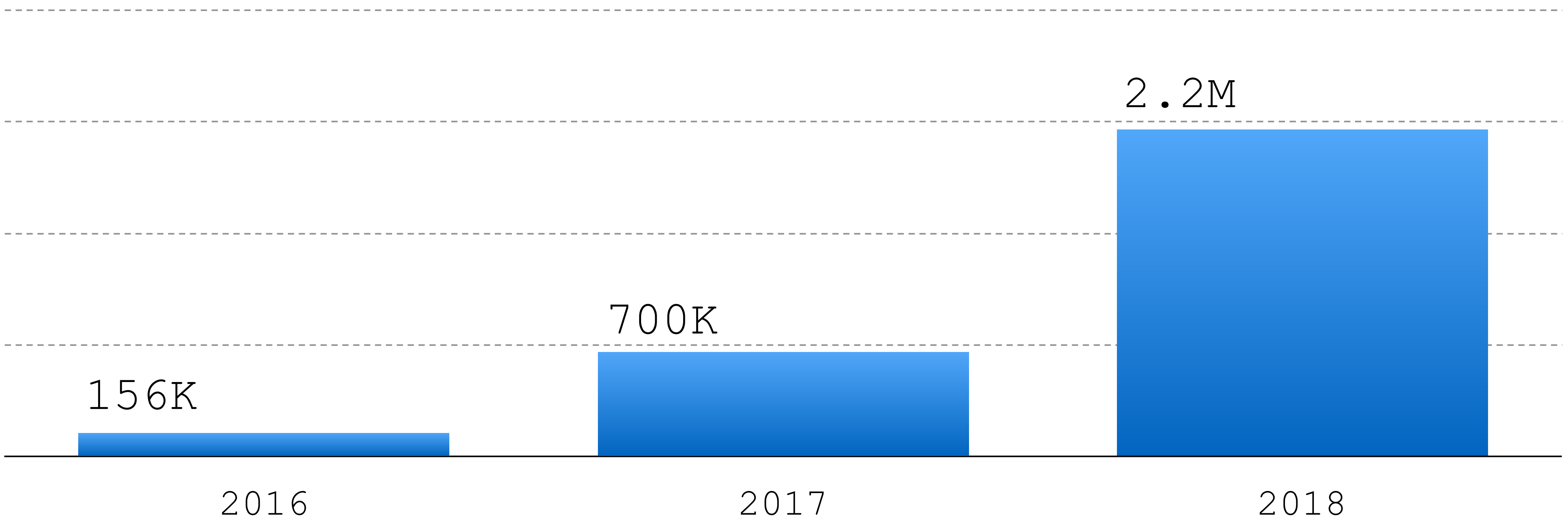
Few words about Kotlin

- Programming language
- Targets JVM, Android and Javascript
- Fully interoperable with Java
- Developed by Jetbrains
- https://kotlinlang.org/

Timeline

- 2010 Project started
- 2016 Kotlin 1.0
- 2017 Official on Android
- 2018 Kotlin 1.3

Kotlin developers



Why do we need Kotlin

- Android is stuck on Java 6
  - No streams
  - No lambdas
  - No try-with-resources

Why do we need Kotlin

- Android is stuck on Java 6

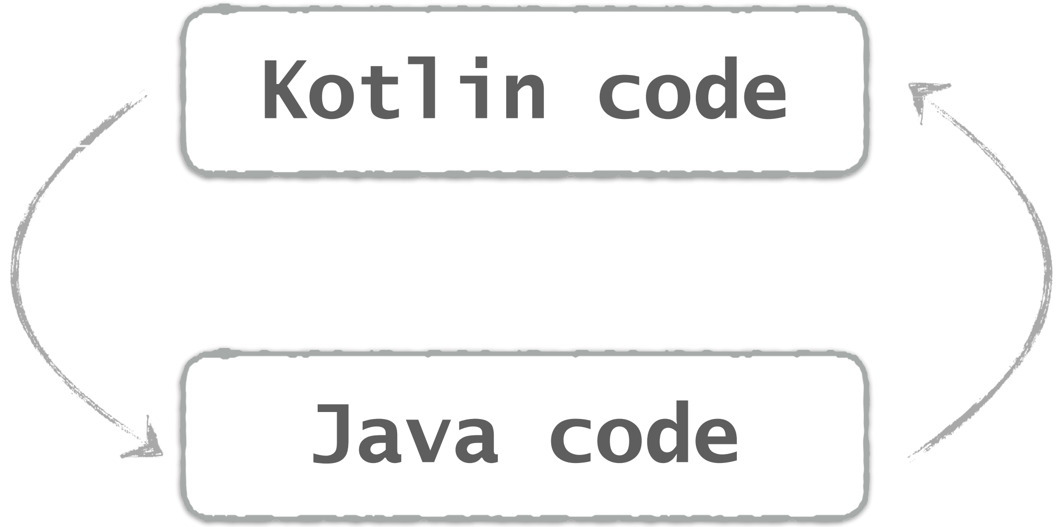
No streams	RxJava
No lambdas	Retrolambda
No try-with-resources	Retrolambda

Why do we need Kotlin

- Android is stuck on Java 6
- Java languagerestrictions
    - Nullability problems
    - Mutability problems
    - No way to add methods to types that we do not control
    - Too muchverbosity

can be easily mixed with Java code

You can have Java & Kotlin code in one project



has good tooling

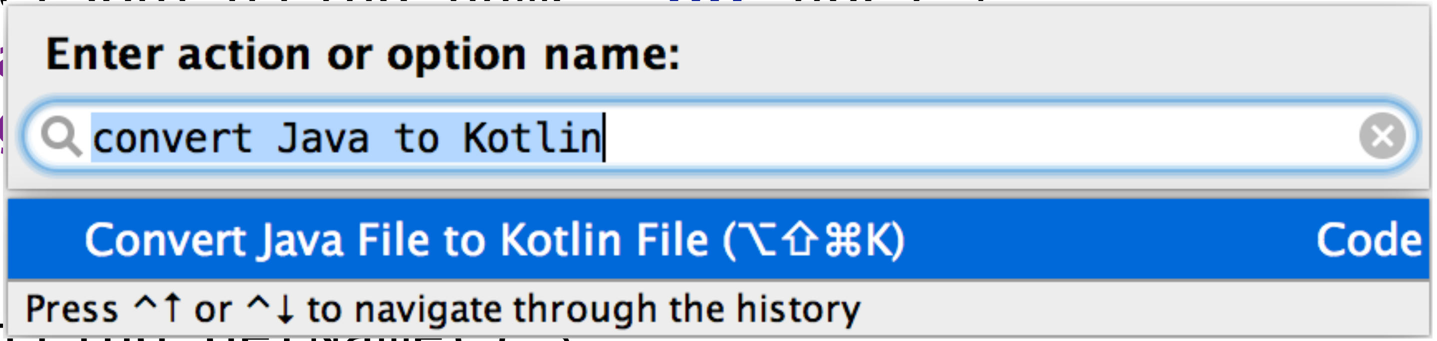
concise & readable

```
public class Person {
  private final String name;
  private final int age;

  public Person(String name, int age) {
    this.na
    this.ag
  }

  public String getName() {
    return name;
  }

  public int getAge() {
    return age;
  }
}
```



Enter action or option name:

convert Java to Kotlin

Convert Java File to Kotlin File (⌥⇧⌘K) Code

Press ^↑ or ^↓ to navigate through the history

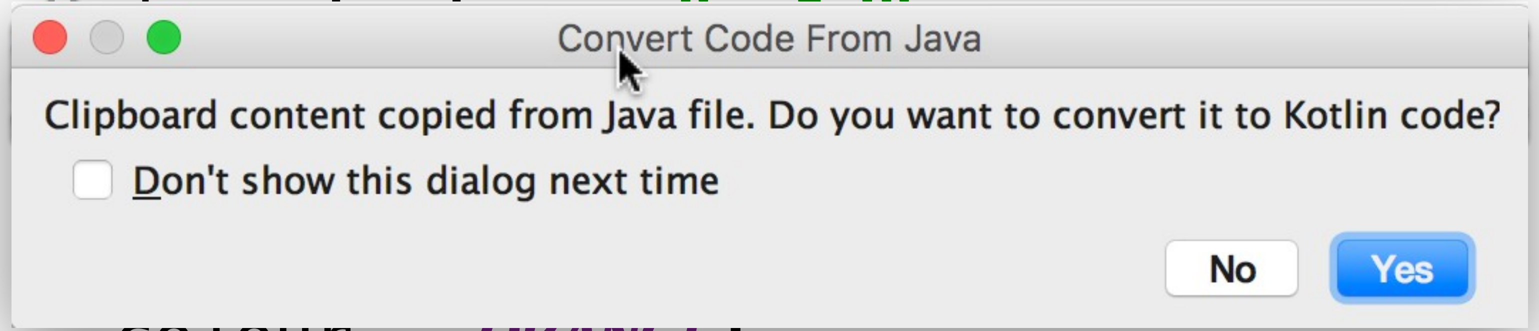
```
data class Person(val name: String, val age: Int)
```

```
- equals
- hashCode
- toString
```

```
public void updateWeather(int degrees) {
    String description;
    Colour colour;

    if (degrees < 5) {

    }
        colour = ORANGE;
    } else {
        description = "hot";
        colour = RED;
    }
    // ...
}
```



Convert Code From Java

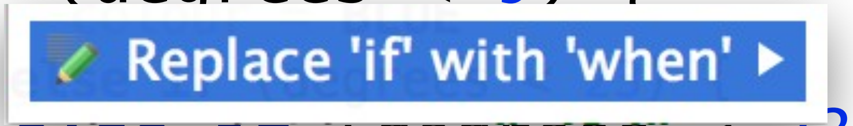
Clipboard content copied from Java file. Do you want to convert it to Kotlin code?

☐ Don't show this dialog next time

```kotlin
fun updateWeather(degrees: Int) {
    val (description: String, colour: Colour) =
        if (degrees < 5) {
            Pair("cold", BLUE)
        } else if (degrees < 23) {
            Pair("mild", ORANGE)
        } else {
            Pair("hot", RED)
        }
    // ...
}
```

```kotlin
fun updateWeather(degrees: Int) {
    val (description, colour) =
        if (degrees < 5) {

        } else if (degrees < 23) {
            Pair("mild", ORANGE)
        } else {
            Pair("hot", RED)
        }
    // ...
}
```



```java
String description;
Colour colour;
if (degrees < 5) {
    description = "cold";
    colour = BLUE;
} else if (degrees < 23) {
    description = "mild";
    colour = ORANGE;
} else {
    description = "hot";
    colour = RED;
}
```

```kotlin
val (description, colour) = when {
    degrees < 5 -> Pair("cold", BLUE)
    degrees < 23 -> Pair("mild", ORANGE)
    else -> Pair("hot" to RED)
}
```

Mutability

An immutable object is an object whose state cannot be changed after instantiation.

```
val name = "Mary" // compile time error if we reassign it
var age  = 20
```

Mutability

An immutable object is an object whose state cannot be changed after instantiation.

```
val name = "Mary" // compile time error if we reassign it
var age  = 20

val numbers: MutableList = mutableListOf(1, 2, 3)
val readOnlyNumbers: List = numbers
```

Dealing with Nullable Types

```
val s: String?

if (s != null) {
   s.
}
```

Replace 'if' expression with safe access expression ▸

Dealing with Nullable Types

```
val s: String?

s?.length
```

Nullability operators

```kotlin
val s: String?
```

```kotlin
val length = if (s != null) s.length else null
```

$$\Big\downarrow$$

```kotlin
val length = s?.length
```

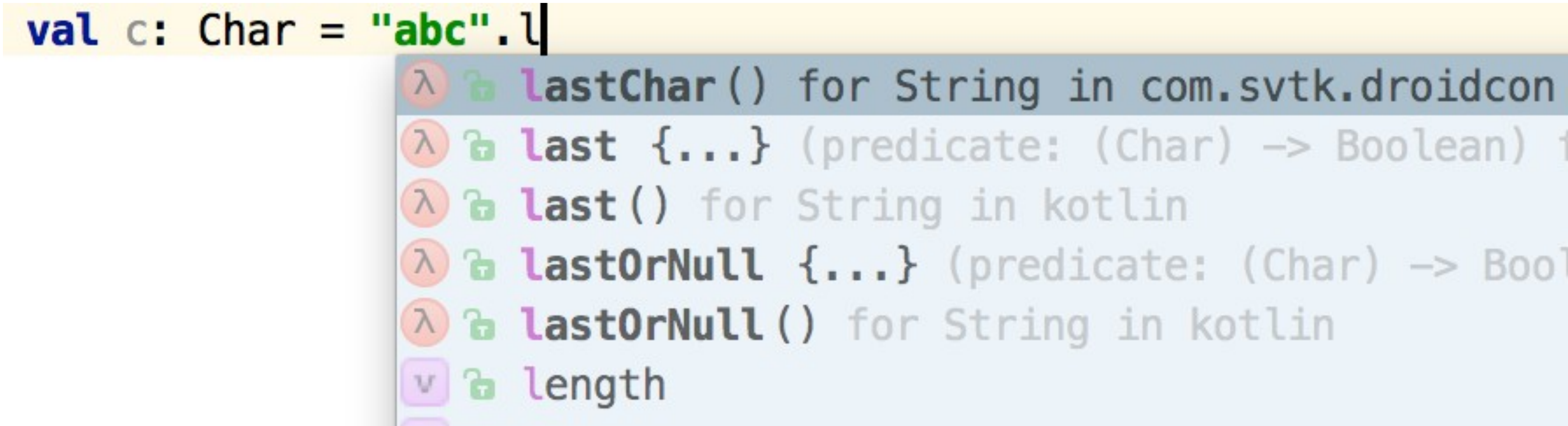
Extension functions

We can extend any class with new features even if we don’t have access to the source code
The extension function acts as part of the class

```
fun Int.isEven(): Boolean { return this%2 == 0 }
println("isEven ${4.isEven()}")
```

Extension Functions

```kotlin
fun String.lastChar() = get(length - 1)
```



```
val c: Char = "abc".l
λ lastChar() for String in com.svtk.droidcon
λ last {...} (predicate: (Char) -> Boolean)
λ last() for String in kotlin
λ lastOrNull {...} (predicate: (Char) -> Bool
λ lastOrNull() for String in kotlin
v length
```

Extension functions

- do not modify the original class
- the function is added as a static import
- can be declared in any file
- common practice: create files which include a set of related functions

Object Oriented & Functional

Uses lambda expressions, to solve some problems in a much easier way.

```
view.setOnClickListener { toast("Hello world!") }
```

```
view.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
           Toast.makeText (this, "Hello world!", Toast.LENGTH_LONG )
           .show();
      }
});
```

enum Orientation { VERTICAL,
HORIZONTAL; }

apply

```
enum class Orientation { VERTICAL, HORIZONTAL }


class LayoutStyle {
   var orientation = HORIZONTAL
}


fun main(args: Array<String>) {
   val layout = LayoutStyle().apply { orientation = VERTICAL }
}
```

```
public class LayoutStyle {
   private Orientation orientation = HORIZONTAL;
   public Orientation getOrientation() {
       return orientation;
   }
   public void setOrientation(Orientation orientation) {
       this.orientation = orientation;
   }
   public static void main(String[] args) {
       LayoutStyle layout = new LayoutStyle();
       layout.setOrientation(VERTICAL);
   }
}
```

TEŞEKKÜRLER..

Levent YILDIZ
http://www.leventyildiz.com.tr
https://github.com/lvntyldz