Meeting agenda:

1. What is Kotlin?
2. What Java has that Kotlin does not?
3. What Kotlin has that Java does not?
4. What is Kotlin Native?
5. What is Kotlin JS?
6. How to use Kotlin?
7. Playing Fifteen Puzzle Game!

This presentation was prepared by Gennady Agranov (Symmwin Configuration) and would not be possible without help of my teammates who were holding the fort while I was having fun and full support of our manager who allowed it ☺

DELL Technologies

That's what Wikipedia tells us about Kotlin:

- Kotlin (/ˈkɒtlɪn/) is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of its standard library depends on the Java Class Library, but type inference allows its syntax to be more concise. Kotlin mainly targets the JVM, but also compiles to JavaScript or native code (via LLVM). Language development costs are borne by JetBrains, while the Kotlin Foundation protects the Kotlin trademark.[
- JetBrains (formerly IntelliJ Software) is a software development company whose tools are targeted towards software developers and project managers. The company offers an extended family of integrated development environments (IDE) for the programming languages Java, Groovy, Kotlin, Ruby, Python, PHP, C, Objective-C, C++, C#, Go, JavaScript and SQL.
- The company entered a new area in 2011 when it introduced Kotlin, a programming language that runs in a Java virtual machine (JVM).

Red color indicates Kotlin features important for presenter. Presenter does understand why Wikipedia did not mention Scala as it also supported by IntelliJ.

What does it mean “interoperate fully with Java” ?

It means that when you are using IntelliJ you can convert Java file to Kotlin file with one click – IDE would enable Kotlin support and convert Java to Kotlin! Be careful though – you Java file would be removed – as you do not need it anymore ☺

Your application would work as nothing has happened – the methods of your Kotlin class would be called from other classes, and your Kotlin class would call methods of other classes.

But if you convert the whole application to Kotlin you would notice that it may become faster (almost 2x in presenter’s case) and files are now smaller ☺

What Java has that Kotlin does not?

- Checked exceptions – good riddance! – RIP dear checked exceptions, we are not going to miss you ☺
- Primitive types that are not classes - in Kotlin everything is an object in the sense that we can call member functions and properties on any variable. Some of the types (e.g. numbers, characters and booleans) can be represented as primitive values at runtime - but to the user they look like ordinary classes.
- Static members – in Kotlin you do not need to declare a class if you just want to write a function or declare some constant/variable.
- Wildcard-types – in Kotlin Java wildcard generic types were replaced with declaration-site variance and type projections. You can read more here.
- Ternary-operator a ? b : c - In Kotlin, if is an expression, i.e. it returns a value. Therefore there is no ternary operator (condition ? then : else), because ordinary if works fine in this role:

```
val max = if (a > b) a else b

val max = if (a > b) {
    print("Choose a")
    a
} else {
    print("Choose b")
    b
}
```

By the way – did I tell you that you rarely need semicolon (;) in Kotlin?

Kotlin provides null safety!

In Kotlin, the type system distinguishes between references that can hold null (nullable references) and those that can not (non-null references). For example, a regular variable of type String can not hold null:

```
var a: String = "abc" // Regular initialization means non-null by default (type declaration is redundant here)
a = null // compilation error

var b: String? = "abc" // can be set null
b = null // ok
print(b)

val l = a.length

val l = b.length // error: variable 'b' can be null

val l = if (b != null) b.length else -1 // smart casting from String? to String
```

Kotlin has data classes!

Java:

```
class Book {
    private String title;
    private String author;

    public String getTitle() {  return title; }
    public void setTitle(String title) { this.title = title; }
    public String getAuthor() { return author; }
    public void setAuthor(String author) { this.author = author; }
}
```

Kotlin:

```
data class Book(val title: String, val author: String)
// equals, hashCode, toString, getters, setters and clone(copy) methods are generated if not found.
```

Kotlin has smart casts!

```
if (obj is String)  {
    print(obj.length)
}

if (obj !is String)  { // same as !(obj is String)
    print("Not a String")
}  else  {
    print(obj.length)
}

if (x !is String) return
print(x.length) // x is automatically cast to String

if (x !is String || x.length == 0) return
```

```
if (x is String && x.length > 0) {
    print(x.length)
}

when (x)  {
    is Int -> print(x + 1)
    is String -> print(x.length + 1)
    is IntArray -> print(x.sum())
}
```

Note that smart casts do not work when the compiler cannot guarantee that the variable cannot change between the check and the usage – e.g. for var properties.

Quiz – can Java compiler do smart casting for final variables?

What else Kotlin has that Java does not?

- Kotlin functions are *first-class* values - they can be stored in variables and data structures, passed as arguments to and returned from other higher-order functions. You can operate with functions in any way that is possible for other non-function values.
- inline functions
- extension functions – it seems that everybody except Java has them?! (in Java you can use Lombok's annotations though). The following adds a swap function to MutableList<Int>:

```
fun MutableList<Int>.swap(index1: Int, index2: Int) {
val tmp = this[index1] // 'this' corresponds to the list
this[index1] = this[index2]
this[index2] = tmp
}
```

- string templates:

```
val s = "abc"
println("$s.length is ${s.length}") // prints "abc.length is 3"
```

What else Kotlin has that Java does not?

- Properties
- Primary constructors
- First-class delegation
- Type inference for variable and property types
- Singletons
- Declaration-site variance & Type projections
- Range expressions
- Operator overloading
- Companion objects
- Separate interfaces for read-only and mutable collections
- Coroutines

What does it mean "compiles to Javascript"?

When you choose the JavaScript target, any Kotlin code that is part of the project as well as the standard Kotlin library is transpiled to JavaScript.

- The Kotlin compiler tries to comply with the following goals:
  - Provide output that is optimal in size
  - Provide output that is readable JavaScript
- Kotlin compiler generates normal JavaScript classes, functions and properties you can freely use from JavaScript code.
- Creating Kotlin code that targets client-side JavaScript:
  - Kotlin provides a series of typed interfaces to interact with the Document Object Model, allowing creation and update of DOM elements.
  - Kotlin provides JavaScript wrappers for the WebGL API.
- Creating Kotlin code that targets server-side JavaScript:
  - You can use Kotlin to interact with server-side JavaScript such as Node.js
  - Kotlin can be used together with existing third-party libraries and frameworks, such as jQuery or React. To access third-party frameworks with a strongly-typed API, you can convert TypeScript definitions from the Definitely Typed type definitions repository to Kotlin using the Dukat. Alternatively, you can use the dynamic type to access any framework without strong typing.

What does it mean “compiles to native code”?

Kotlin/Native is a technology for compiling Kotlin code to native binaries. It is an LLVM based backend for the Kotlin compiler and native implementation of the Kotlin standard library.

Kotlin/Native is primarily designed to allow compilation for platforms where *virtual machines* are not desirable or possible, for example, embedded devices or iOS. It solves the situations when a developer needs to produce a self-contained program that does not require an additional runtime or virtual machine.

Kotlin/Native supports the following platforms:

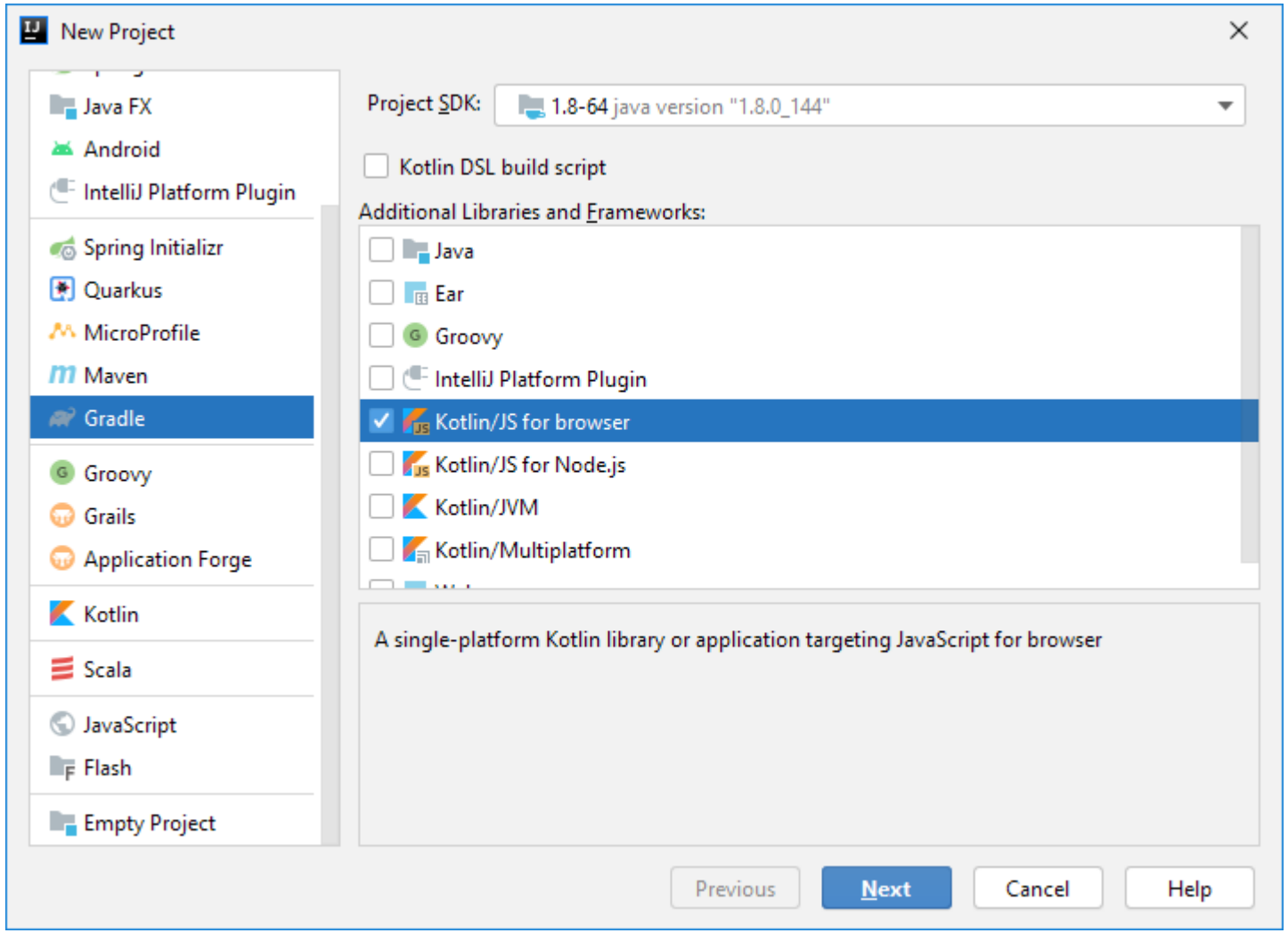
- iOS (arm32, arm64, simulator x86_64)
- macOS (x86_64)
- watchOS (arm32, arm64, x86)
- tvOS (arm64, x86_64)
- Android (arm32, arm64, x86, x86_64)
- Windows (mingw x86_64, x86)
- Linux (x86_64, arm32, arm64, MIPS, MIPS little endian)
- WebAssembly (wasm32)

Kotlin/Native supports two-way interoperability with the Native world:

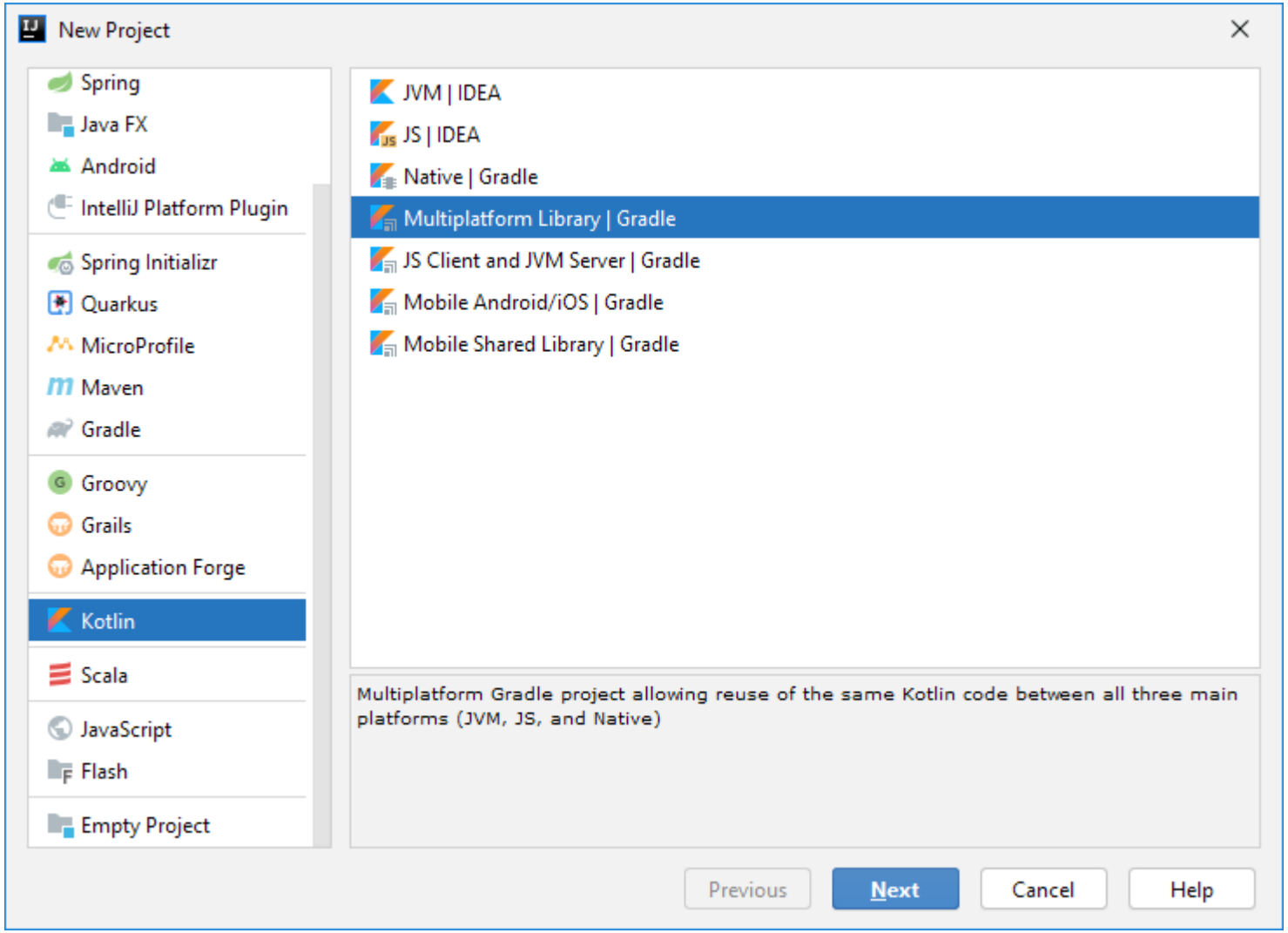
- You can create a standalone executable for many platforms
- You can create static library or dynamic library with C headers for C/C++ projects
- You can create an Apple framework for Swift and Objective-C projects
- You can use existing static or dynamic C Libraries
- You can use existing C, Swift, and Objective-C frameworks

How to use Kotlin?

Install IntelliJ IDEA and create project:



Multiplatform Library Project:



Fifteen Puzzle Game

Some puzzle trivia:

- Only half of all possible starting positions is solvable
- Finding shortest solution is NP-hard problem

Java Puzzle Solver

Java Puzzle Solver converted to Kotlin

Kotlin Puzzle Solver that only uses kotlin.stdlib

Pascal Application calls Kotlin DLL (Lazarus IDE)


DELL Technologies