#0: Introduction

school.kt

Kirill Rozov

Lead Software Engineer@EPAM

krlrozov

krlrozov@gmail.com

fitbit

SPACE

JUNO

JET BRAINS

School.kt Program

0. Intro
1. Object-oriented programming
2. Standard library
3. Functional programming
4. Generics
5. Kotlin DSL & Multiplatform projects
6. Coroutines
7. Interoperability with Java
8. Kotlin ecosystem

Mentors

Certificate

school.kt

bkug.by/course/certificate

LET'S GO!!!

school.kt

0. Introduction

- Kotlin history
- Variables
- Type system
- Null safety
- Conditional operators
- Cycles
- Functions
- Exceptions
- First home task

school.kt

Kotlin main facts

- Was Developed in JetBrains
- The name comes from Kotlin Island, near St. Petersburg
- 1.0 was released on February 15, 2016
- Free to use
- Open source under the Apache 2 license
- Officially supported by Google for mobile development on Android

Companies with Kotlin in production apps

Square

UBER

coursera

JUNO

Expedia®

corda

Google

Gradle

JET BRAINS

Atlassian

NETFLIX

EVERNOTE®

Pivotal®

Kotlin

- Cross-platform
- Statically-typed
- General-purpose
- Multi paradigms
- Backward compatibility
- Two way interoperability with Java/JS/Native

In Kotlin, **everything is an object** in the sense that we can call member functions and properties on any variable

Any - the root of the Kotlin class hierarchy. Every Kotlin class has *Any* as a superclass.

school.kt

Any

```
open operator fun equals(other: Any?): Boolean
open fun hashCode(): Int
open fun toString(): String

inline val <T : Any> T.javaClass: Class<T>
```

Variables

```kotlin
// Can't be modified after initialization
val readOnly: String = "immutable"
readOnly = "newValue"


// Can be modified
var mutable: String = "mutable"
mutable = "newValue"
```

Variables

```
// Can't be modified after initialization
val readOnly: String = "immutable"
readOnly = "newValue"


// Can be modified
var mutable: String = "mutable"
mutable = "newValue"
```

Variables

```
// Can't be modified after initialization
val readOnly: String = "immutable"
readOnly = "newValue"  //Error


// Can be modified
var mutable: String = "mutable"
mutable = "newValue"
```

Numbers

```
val b: Byte = 12;
val i: Int = 12;
val s: Short = 12;
val l: Long = 12;
val f: Float = 12.0F;
val d: Double = 12.0;
```

Numbers

```
val b: Byte = 12;
val i: Int = 12;
val s: Short = 12;
val l: Long = 12;
val f: Float = 12.0F;
val d: Double = 12.0;
```

Numbers

```kotlin
val b: Byte = 12
val i: Int = 12
val s: Short = 12
val l: Long = 12
val f: Float = 12.0F
val d: Double = 12.0
```

Numbers

```
val b: Byte = 12
val i: Int = 12
val s: Short = 12
val l: Long = 12
val f: Float = 12.0F
val d: Double = 12.0
```

Numbers

```kotlin
val b = 12
val i = 12
val s = 12
val l = 12
val f = 12.0F
val d = 12.0
```

Numbers

```kotlin
val b = 12 // Int
val i = 12 // Int
val s = 12 // Int
val l = 12 // Long
val f = 12.0F // Float
val d = 12.0 // Double
```

Numbers

```kotlin
val b = 12.toByte()
val i = 12
val s = 12.toShort()
val l = 12L
val f = 12.0F
val d = 12.0
```

Numbers

```
val binaries = 0b00001011
val hexI = 0x0F
val expD = 123.5e10

val i = 123_456_789
```

Numbers

```kotlin
// Kotlin 1.3 Experimental
val ub: UByte = 1u
val us: UShort = 2u
val ui: UInt = 3u
val ul: ULong = 4u
```

Numbers

```
val sum = i1 + i2
val diff = i1 - i2
val divide = i1 / i2
val multiple = i1 * i2
```

Numbers

```
// Structural equality
i1 == i2          i1 != i2

// Referential equality
i1 === i2         i1 !== i2

val great = i1 > i2
val greatOrEquals = i1 >= i2
val lessOrEquals = i1 <= i2
val less = i1 < i2
```

Ranges

```
val range: IntRange = 1..10

15 in  1..10 -> false
15 !in 1..10 -> true
10 in  1 until 10 -> false
```

Characters

```
val c = 'c'
c in 'a'..'z'
```

Booleans

```
val b = true

val and = b && true
val and = b and true

val or = b || true
val or = b or true
```

Strings

```kotlin
val string = "Hello, Kotlin"
val concat = string + " Wow"
val template = "$string Wow"
val format = "%s Wow".format(string)
```

Raw Strings

```
val rawString = """
    Multiline string
          that saves all spaces and tabulations!
"""
```

school.kt

Raw Strings

```
val rawString = """
    Multiline string
          that saves all spaces and tabulations!
"""


print(rawString)
Multiline string
      that saves all spaces and tabulations!
```

Raw Strings

```
val rawString = """
    Multiline string
          that saves all spaces and tabulations!
""".trimIndent()
```

Raw Strings

```
val rawString = """
    Multiline string
          that saves all spaces and tabulations!
""".trimIndent()


print(rawString)
Multiline string
that saves all spaces and tabulations!
```

Create object

```
StringBuilder builder = new StringBuilder();
```

Create object

```
val builder = new StringBuilder();
```

Create object

```
val builder = new StringBuilder()
```

school.kt

Create object

```
val builder = StringBuilder()
```

Create object

```
val builder = StringBuilder()


// Java way
StringBuilder builder = new StringBuilder();
```

school.kt

Null safety

```kotlin
var s: String = "value"
s = null
```

school.kt

Null safety

```
var s: String = "value"
s = null //Error
```

school.kt

Null safety

```
var s: String = "value"
s = null //Error
```

school.kt

Null safety

```kotlin
var s: String? = "value"
s = null
```

Null safety

```
public static String format(String s) {
    // ...
}
```

Null safety

```kotlin
var s: String? = "value"
val result = format(s)
```

Null safety

```
var s: String? = "value"
val result = format(s)
```

school.kt

Null safety

```
var s: String? = "value"
val result = format(s)
```

school.kt

Null safety

```
var s: String? = "value"
val result = format(s) // String!
```

school.kt

Null safety

```java
public static String format(String s) {
    // ...
}
```

Null safety

```
@NonNull
public static String format(@Nullable String s) {
    // ...
}
```

Types

- Nullable (String?)
- Non-nullable (String)
- Platform (String!)

Null safety

```
var s: String? = "value"
s.trim()
```

Null safety

```kotlin
var s: String? = "value"
s.trim() //Error
```

Null safety

```kotlin
var s: String? = "value"
if (s != null) {
    s.trim()
}
```

Null safety

```
var s: String? = "value"
s?.trim()
```

Null safety

```
var s: String? = "value"
s?.trim()
```

Null safety

```
var s: String? = "value"
val s1 = s?.trim() ?: ""
```

Null safety

```kotlin
var s: String? = "value"
val s1 = s?.trim() ?: throw Exception("s is null")
```

Type checks

```kotlin
// Type checks
obj is String     // is `obj` of String
obj is String?    // is `obj` of String or null
obj !is String    // Negation
```

Casts

```
// Unsafe cast. Throw exception
obj as String    // Success if `obj` is String
obj as String?   // Success if `obj` is String or null

// Safe cast: return null if `obj` is not String
obj as? String
```

Casts

	Any? = null	Any? = ""	Any? = 1	String? = ""
as String	kotlin.TypeCastException	""	ClassCastException	kotlin.TypeCastException
as String?	null	""	ClassCastException	null
as? String	null	""	null	null
as? String?	null	""	null	null

Type checks

```
val obj: Any? = ""
if (obj is String) {
    // `obj` is automatically cast to `String`
    obj.length
}

// `obj` is still of type `Any?` outside
```

Type checks

```kotlin
val obj: Any? = ""
if (obj !is String) return null

// `obj` is automatically cast to `String`
obj.length
```

Type checks

```kotlin
val obj: Any? = ""
// `obj` is automatically cast to `String`
//  on the right-hand side of `&&`
if (obj is String && obj.length > 0) {
    obj.length
}
```

school.kt

Arrays

```kotlin
val array = Array<Int>(10) { 0 }
val array: Array<Int> = arrayOf(1, 2, 3, 4, 5)
val array: Array<Int?> = arrayOfNulls(size = 10)
```

Special Array Types

- ByteArray
- ShortArray
- IntArray
- LongArray
- FloatArray
- DoubleArray
- BooleanArray
- CharArray

Arrays

Kotlin	Java
Array<Int>	Integer[]
Array<Int?>	Integer[]
IntArray	int[]

school.kt

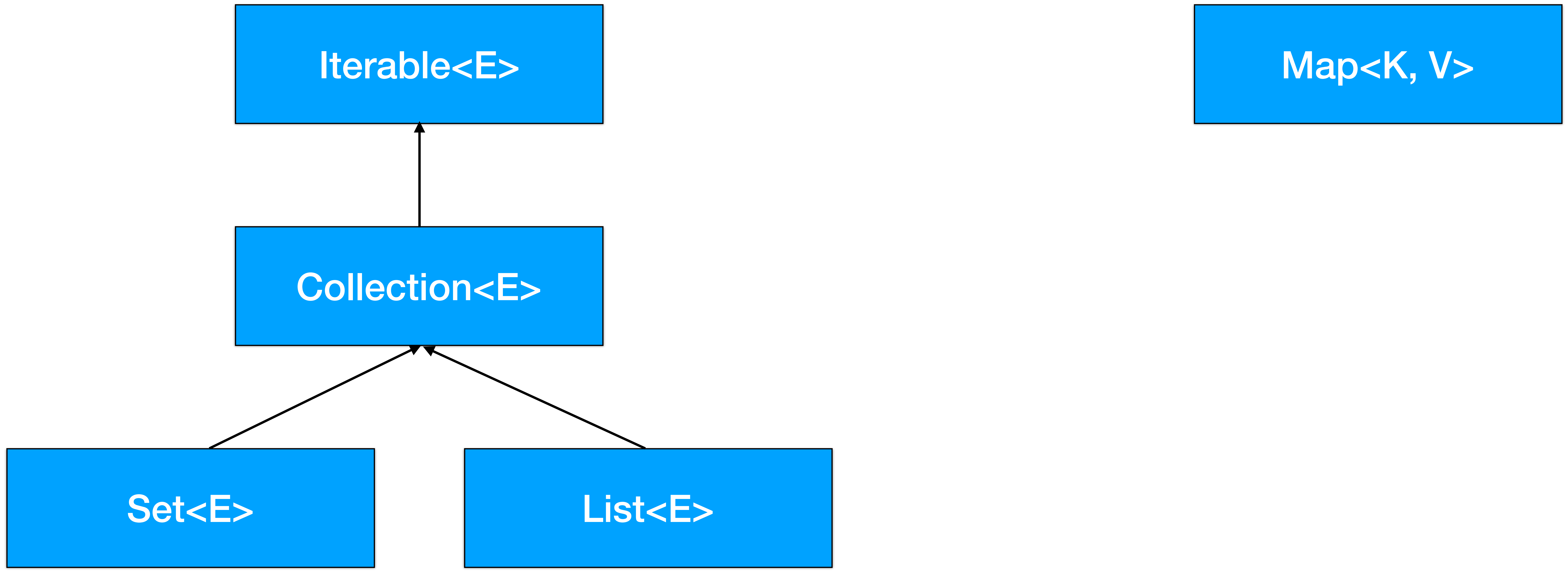
Arrays

```kotlin
array.size // Number of items
array.isEmpty() // Is the array empty


array[1] // Get item in specified position
array.get(1) // The same
array[2] = 9 // Set item in specified position
```

Arrays

```kotlin
// Creates an array with values [0, 1, 4, 9, 16]
val array = Array(5) { i -> i * i }

// Print all values of the array
array.forEach { println(it) }
```

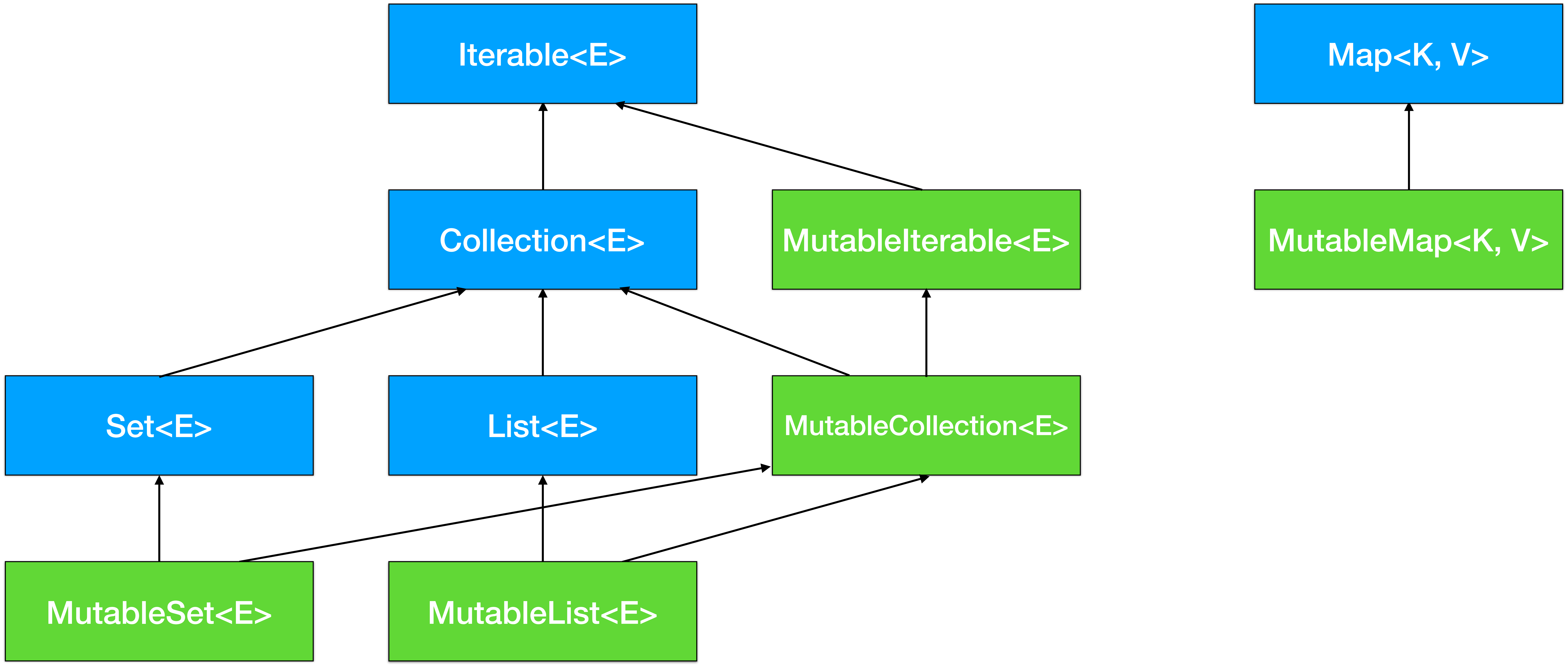
Java Collections



Immutable collections are good because

- Limitation of access
- Stricter APIs
- Thread safe

Kotlin Collections



school.kt

Create List

```
listOf<Value>()
listOf(1, 2, 3)
emptyList<Value>()
List(size = 10) { it }


mutableListOf<Value>()
mutableListOf(1, 2, 3)
MutableList(size = 10) { it }
```

List

```kotlin
// Creates an list with values [0, 1, 4, 9, 16]
val list: List<Int> = List(5) { i -> i * i }

// Print all values of the list
list.forEach { print(it) }
```

List

```kotlin
val list: List<Int> = List(5) { it }
list.map { it * it }
    .filter { it % 3 != 0 }
    .onEach { print(it) }
    .fold(0) { sum, item -> sum + item }
```

List

```
val list: List<Int> = List(5) { it }
list.map { it * it }                // New list object
    .filter { it % 3 != 0 }     // New list object
    .onEach { print(it) }
    .fold(0) { sum, item -> sum + item }
```

Create map

```
val map = emptyMap<KeyType, ValueType>()
val map = mapOf("key" to "value")

val mutableMap = mutableMapOf("key" to "value")
```

Map

```
map["key"] // Get value
map.get("key")

map["key"] = "value" // Set value
map.set("key", "value")
```

Map

```
map.forEach { key, value -> … }

for(entry in map) {
    …
}

for((key, value) in map) {
    …
}
```

school.kt

if…else

```
if (obj == 1) {
    "One"
} else if (obj == "Hello") {
    "Greeting"
} else if (obj is Long) {
    "Long"
} else if (obj !is String) {
    "Not a string"
} else {
    "Unknown"
}
```

if…else

```
val msg: String
if (count == 0) {
    msg = "zero"
} else if (count == 1) {
    msg = "one"
} else {
    msg = "many"
}
```

if expression

```kotlin
val msg = if (count == 0) {
    "zero"
} else if (count == 1) {
    "one"
} else {
    "many"
}
```

school.kt

if expression

```
// Analogue of the Java ternary operator
// String msg = count == 1 ? "one" : "many";
val msg = if (count == 1) "one" else "many"
```

when

```
when (obj) {
    1            -> "One"
    "Hello"      -> "Greeting"
    is Long      -> "Long"
    !is String -> "Not a string"
    else         -> "Unknown"
}
```

when

```
when {
    obj == 1         -> "One"
    obj == "Hello" -> "Greeting"
    obj is Long      -> "Long"
    obj !is String -> "Not a string"
    else             -> "Unknown"
}
```

when expression

```kotlin
val msg = when {
    obj == 1          -> "One"
    obj == "Hello" -> "Greeting"
    obj is Long       -> "Long"
    obj !is String -> "Not a string"
    else              -> "Unknown"
}
```

for cycle

```
val list = listOf(...)
for (item in list) {
    print(item)
}
```

for cycle

```
val list = listOf(...)
for (item in list) {
    print(item)
}
```

```
list.forEach { item -> print(item) }
```

for cycle with indexes

```
val list = listOf(...)
for(i in 0 until list.size) {
    list[i]
}
```

while

```
while (x > 0) {
    x--
}
```

do...while

```
do {
    x--
} while (x > 0)
```

Functions

```
fun sample(arg1: String, arg2: Int): String? {
    // Function body
}
```

Functions

```kotlin
fun sample(arg1: String, arg2: Int): String? {
    // Function body
}
```

school.kt

Functions

```kotlin
fun sample(arg1: String, arg2: Int): String? {
    // Function body
}
```

Functions

```
fun sample(arg1: String, arg2: Int): String? {
    // Function body
}
```

Functions

```kotlin
fun sample(arg1: String, arg2: Int): String? {
    // Function body
}
```

school.kt

Functions

```kotlin
fun sample(arg1: String, arg2: Int): String? {
    // Function body
}
```

school.kt

Functions

```kotlin
fun sample(arg1: String, arg2: Int): Unit {
    // Function body
}
```

school.kt

Functions

```kotlin
fun sample(arg1: String, arg2: Int) {
    // Function body
}
```

school.kt

Functions

```
fun sample(arg1: String, arg2: Int) {
    // Function body
}
```

Functions

```
fun join(values: Iterable<String>): String {
    …
}
```

Functions

```kotlin
val list = listOf("one", "two", "three")
join(list)
```

Functions

```
val list = listOf("one", "two", "three")
list.join()
```

Extension Functions

```
fun join(values: Iterable<String>): String {
    …
}
```

Extension Functions

```kotlin
fun Iterable<String>.join(): String {
    …
}
```

Functions

```
// Join items of a list with separator = " | ", prefix =
"(" and postfix ")"
val list = listOf("a", "b", "c")
println(list.joinToString("(", " | ", “)"))
```

school.kt

Functions

```
// Join items of a list with separator = " | ", prefix =
"(" and postfix ")"
val list = listOf("a", "b", "c")
println(list.joinToString("(", " | ", “)"))


// | a(b(c)
```

Functions

```
fun <T> Iterable<T>.joinToString(
    separator: String,
    prefix: String,
    postfix: String
): String
```

Functions

```
fun <T> Iterable<T>.joinToString(
    separator: String,                     // "("
    prefix: String,                        // " | "
    postfix: String                        // ")"
): String
```

school.kt

Functions

```kotlin
val list = listOf("a", "b", "c")
println(list.joinToString(" | ", "(", ")"))
```

school.kt

Functions

```kotlin
val list = listOf("a", "b", "c")
println(list.joinToString(" | ", "(", ")"))

// (a | b | c)
```

school.kt

Functions

```kotlin
list.joinToString(" | ", "(", ")")
```

Functions Named Arguments

```
list.joinToString(
    separator = " | ",
    prefix = "(",
    postfix = ")"
)
```

Default arguments

```
fun sample(arg1: String = "empty", arg2: Int = 0) {
    // Function body
}
```

Default arguments

```
fun sample(arg1: String = "empty", arg2: Int = 0) {
    // Function body
}


sample()
sample("value")
sample("value", 2)
```

school.kt

Default arguments

```kotlin
fun sample(arg1: String = "empty", arg2: Int = 0) {
    // Function body
}
```

school.kt

Default arguments

```kotlin
fun sample(arg1: String = "empty", arg2: Int) {
    // Function body
}
```

Default arguments

```
fun sample(arg1: String = "empty", arg2: Int) {
    // Function body
}

sample(2)
sample("value", 2)
```

Default arguments

```
fun sample(arg1: String = "empty", arg2: Int) {
    // Function body
}

sample(2) //Error
sample("value", 2)
```

Default arguments

```
fun sample(arg1: String = "empty", arg2: Int) {
    // Function body
}

sample(arg2 = 2)
sample("value", 2)
```

Default arguments

```kotlin
fun sample(arg1: String = "empty", arg2: Int) {
    // Function body
}
```

school.kt

Default arguments

```kotlin
fun sample(arg2: Int, arg1: String = "empty") {
    // Function body
}
```

Default arguments

```kotlin
fun sample(arg2: Int, arg1: String = "empty") {
    // Function body
}
```

Rule: All function parameters with default values must be at the end of parameters list

Functions

```kotlin
fun join(vararg values: String): String {
    // Function body
}
```

Functions

```
join("one", "two", "three")
```

Functions

```
val array = arrayOf("one", "two", "three")
join(array)
```

Functions

```
val array = arrayOf("one", "two", "three")
join(array) //Error
```

Functions

```kotlin
val array = arrayOf("one", "two", "three")
join(*array) // The spread operator
```

school.kt

Functions

```
val array = arrayOf("one", "two", "three")
join(*array)
```

Functions

```kotlin
val array = arrayOf("one", "two", "three")
join("zero", *array, "four", "five")
```

school.kt

Kotlin doesn’t divide exceptions on checked and unchecked. **All exceptions are unchecked**.

Exceptions

```kotlin
throw Exception("Hi There!")
```

Exceptions

```
try {
    // some code
} catch (e: SomeException) {
    // handler
} finally {
    // optional finally block
}
```

Exceptions

```
val a: Int? = try {
    parseInt(input)
} catch (e: NumberFormatException) {
    null
}
```

Try-with-resource.java

```java
try (InputStream inputStream = openFile()) {
    // Read data
} catch (IOException e) {
    // Handle exception
}
```

Try-with-resource.kt

```kotlin
openFile().use { inputStream ->
    // Read data
}
```

school.kt

THE END

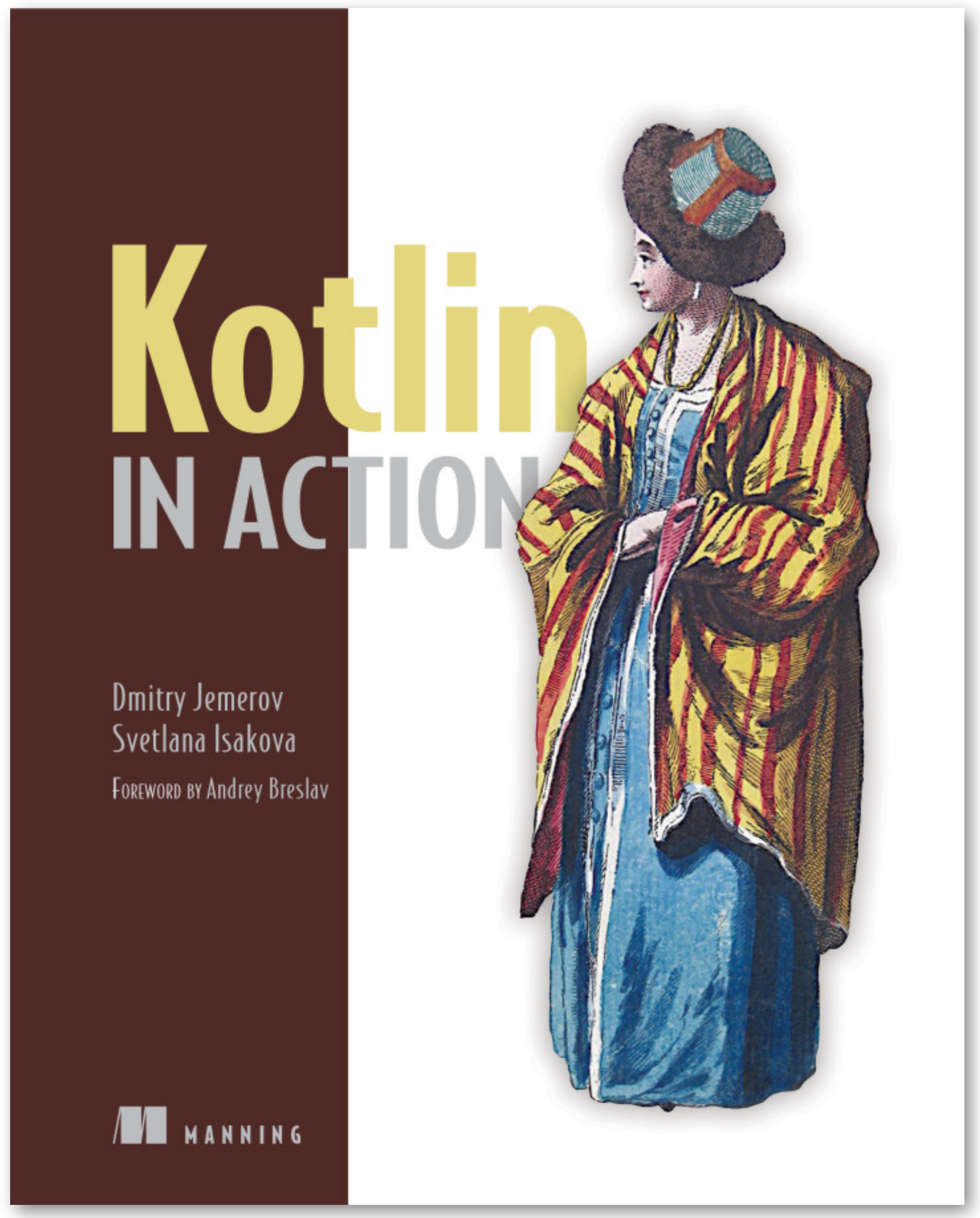
school.kt

What you need to start

- IntelliJ IDEA or Android Studio
- The latest Kotlin Plugin (1.3.21)

Materials

- Official Kotlin Site
  kotlinlang.org
- Kotlin Coding Convention
  kotlinlang.org/docs/reference/coding-conventions.html
- Kotlin in Action, Dmitry Jemerov and Svetlana Isakova (eng & rus)
  manning.com/books/kotlin-in-action
- Coursera: Kotlin For Java Developer (eng & rus)
  coursera.org/learn/kotlin-for-java-developers
- Kotlin Koans: online or in IntelliJ IDEA
  kotlinlang.org/docs/tutorials/koans.html



First homework

- Setup environment
- Kotlin Koans: Introduction

Thanks!!!

bit.ly/SchoolKt_Intro

school.kt