#2: stdlib

Сергей Крюков

Developer @ Banuba

siarhei.krukau@gmail.com

school.kt

SPACE

JUNO

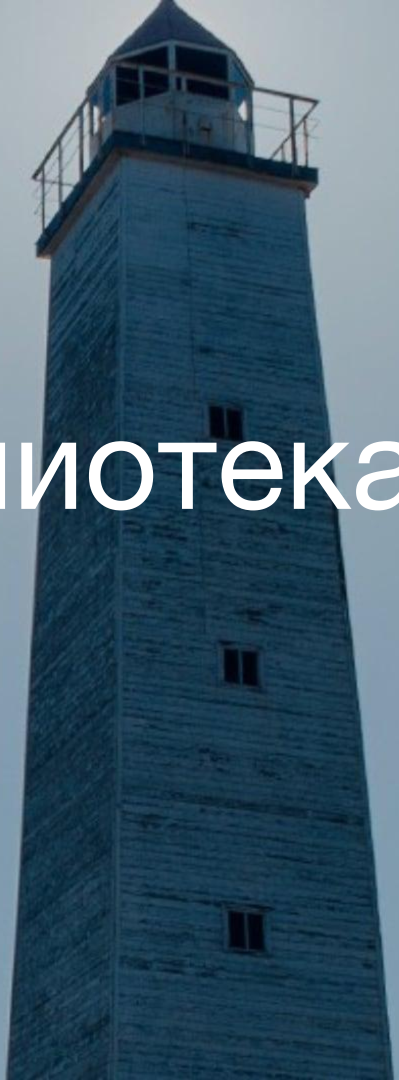
fitbit

JET BRAINS

Менторы

img by http://www.theadhdnerd.com

Стандартная библиотека Kotlin



school.kt

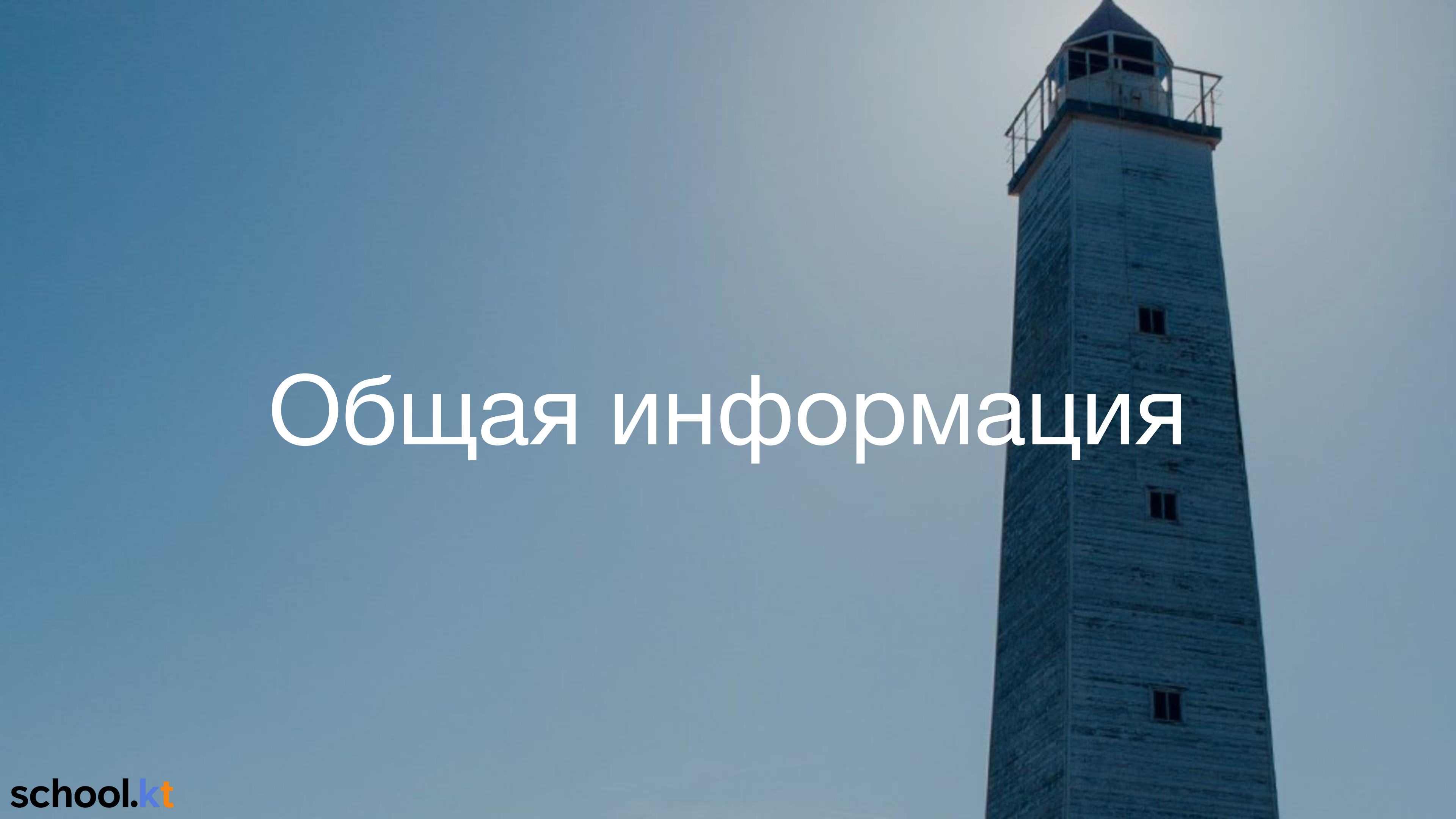
kotlin.jvm
kotlin.coroutines.experimental.intrinsics
kotlin.native.concurrent
kotlin.comparisons
kotlin.reflect
kotlin.annotation
org.w3c.dom.parsing
org.w3c.dom.url
kotlin.dom
org.w3c.dom.css
org.w3c.performance
kotlin.streams
org.w3c.workers
org.khronos.webgl
kotlinx.cinterop
kotlinx.wasm.jsinterop
kotlin.reflect.full
kotlin.reflect.jvm
kotlin.sequences
kotlin.ioorg.w3c.dom.events
org.w3c.notifications
kotlin.properties
kotlin.mathkotlin.collections

kotlin.randomkotlin.native
kotlin.experimentalkotlin.js
kotlin.coroutines.intrinsics
kotlin.concurrent
kotlin.coroutines
org.w3c.fetch
org.w3c.files
org.w3c.dom
kotlin.text
kotlin.system
kotlin
org.w3c.xhr
org.w3c.dom.svg
kotlin.ranges
kotlin.contracts
kotlin.browser
kotlin.coroutines.experimental
kotlin.native.ref

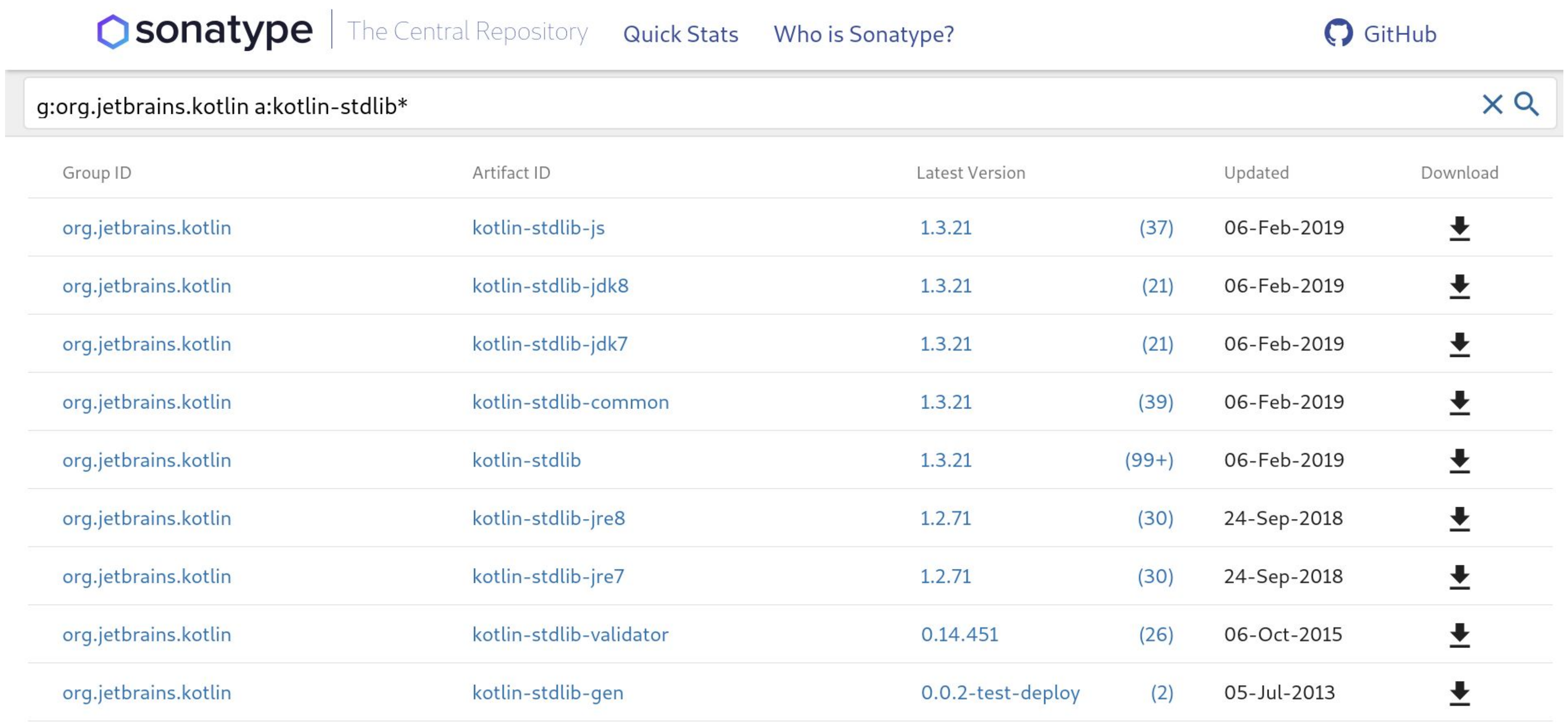
kotlin.coroutines.experimental.intrinsics
kotlin.jvm
kotlin.native.concurrent
kotlin.comparisons
kotlin.reflect
kotlin.annotation
org.w3c.dom.parsing
org.w3c.dom.url
kotlin.dom
org.w3c.dom.css
org.w3c.performance
kotlin.streams
org.w3c.workers
org.khronos.webgl
kotlinx.cinterop
kotlinx.wasm.jsinterop
kotlin.reflect.full
kotlin.reflect.jvm
kotlin.sequences
kotlin.io
org.w3c.dom.events
org.w3c.notifications
kotlin.properties
kotlin.math
kotlin.collections
kotlin.random
kotlin.native
kotlin.experimental
kotlin.js
kotlin.coroutines.intrinsics
kotlin.concurrent
kotlin.coroutines
org.w3c.fetch
org.w3c.files
org.w3c.dom
kotlin.text
kotlin.system
kotlin
org.w3c.xhr
org.w3c.dom.svg
kotlin.ranges
kotlin.contracts
kotlin.browser
kotlin.coroutines.experimental
kotlin.native.ref

Стандартная библиотека Kotlin

- Общая информация
- Основные типы
- Основные функции
- Числа, логика и математика
- Контейнеры
- Строки и регулярные выражения
- Ввод-вывод
- Работа с многопоточностью
- Свойства и делегаты, рефлексия



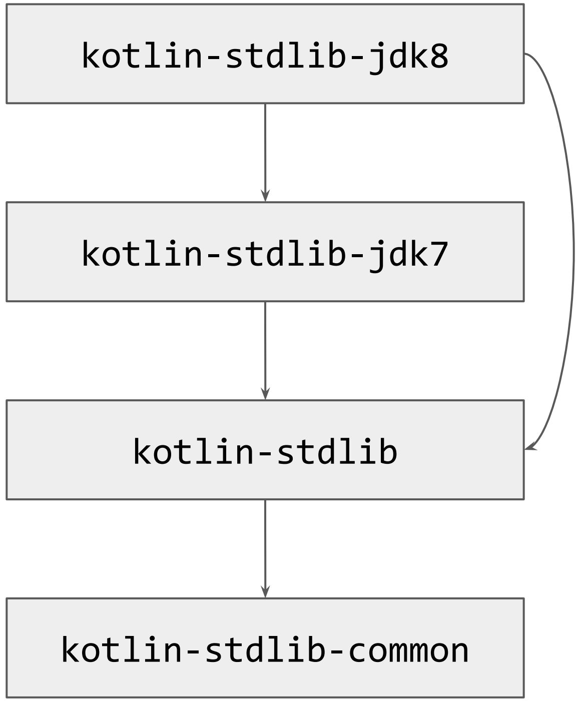
Общая информация


sonatype | The Central Repository   Quick Stats   Who is Sonatype?   GitHub

g:org.jetbrains.kotlin a:kotlin-stdlib*

Group ID	Artifact ID	Latest Version		Updated	Download
org.jetbrains.kotlin	kotlin-stdlib-js	1.3.21	(37)	06-Feb-2019	⬇
org.jetbrains.kotlin	kotlin-stdlib-jdk8	1.3.21	(21)	06-Feb-2019	⬇
org.jetbrains.kotlin	kotlin-stdlib-jdk7	1.3.21	(21)	06-Feb-2019	⬇
org.jetbrains.kotlin	kotlin-stdlib-common	1.3.21	(39)	06-Feb-2019	⬇
org.jetbrains.kotlin	kotlin-stdlib	1.3.21	(99+)	06-Feb-2019	⬇
org.jetbrains.kotlin	kotlin-stdlib-jre8	1.2.71	(30)	24-Sep-2018	⬇
org.jetbrains.kotlin	kotlin-stdlib-jre7	1.2.71	(30)	24-Sep-2018	⬇
org.jetbrains.kotlin	kotlin-stdlib-validator	0.14.451	(26)	06-Oct-2015	⬇
org.jetbrains.kotlin	kotlin-stdlib-gen	0.0.2-test-deploy	(2)	05-Jul-2013	⬇

Что выбрать?


kotlin-stdlib-jdk8
kotlin-stdlib-jdk7
kotlin-stdlib
kotlin-stdlib-common

Maven

```xml
<dependencies>
   <dependency>
       <groupId>org.jetbrains.kotlin</groupId>
       <artifactId>kotlin-stdlib-jdk8</artifactId>
       <version>${kotlin.version}</version>
   </dependency>
</dependencies>
```

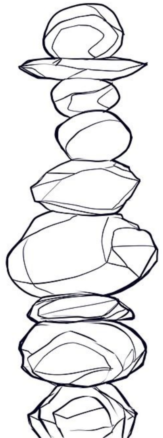
Gradle

```
dependencies {
   implementation("org.jetbrains.kotlin:kotlin-stdlib-jdk8")
}
```

kotlin-stdlib (1.3.21)

1.2 M

7955 методов



school.kt

Версия stdlib

```
System.out.println(
    System.getProperty("java.version")
);

>1.8.0_202
```

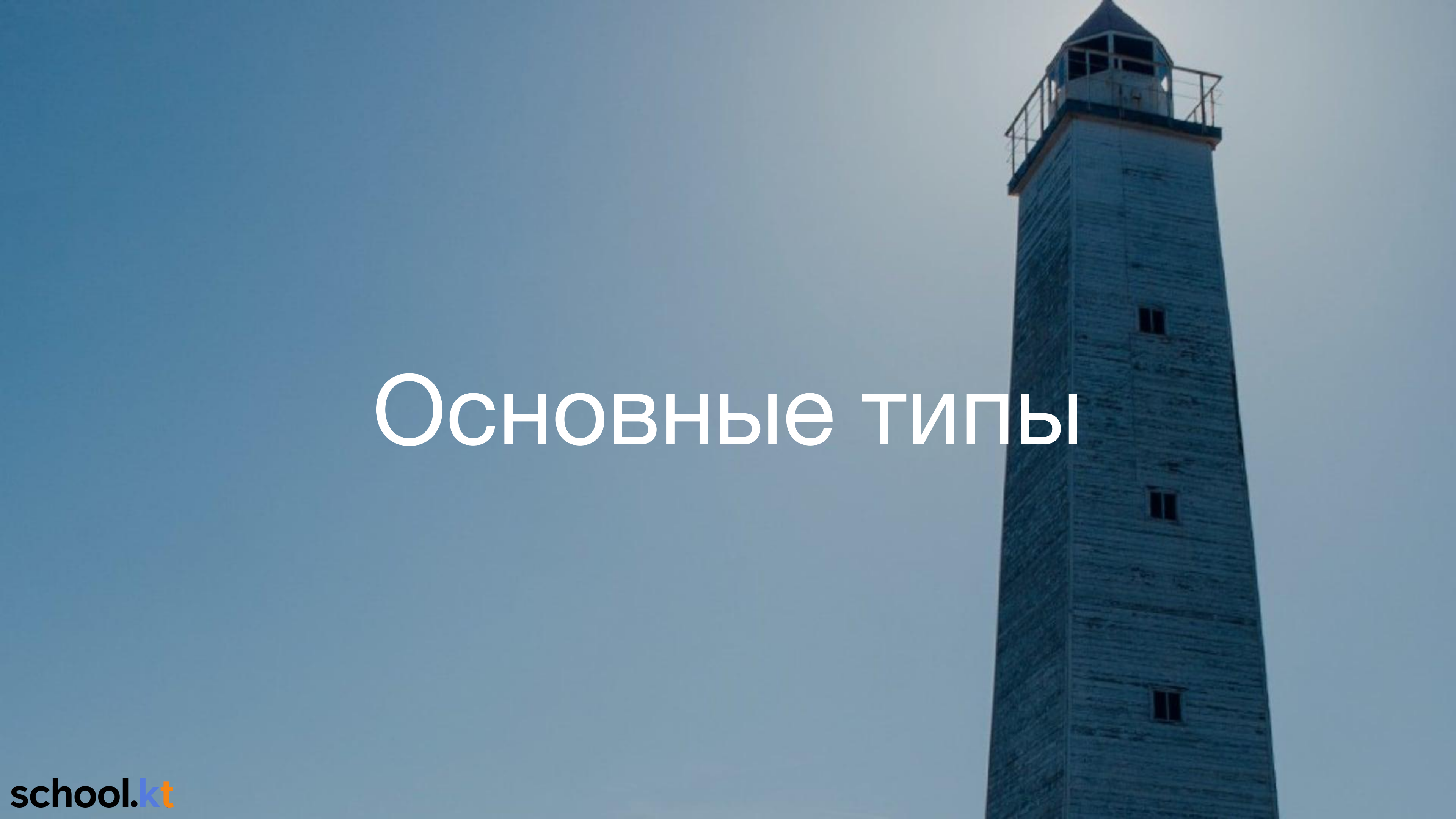
```
println(KotlinVersion.CURRENT)

>1.3.21
```

Версия stdlib

```
KotlinVersion.CURRENT.major          // 1
KotlinVersion.CURRENT.minor          // 3
KotlinVersion.CURRENT.patch          // 21
KotlinVersion.CURRENT.isAtLeast(1, 2) // true
```

school.kt

school.kt

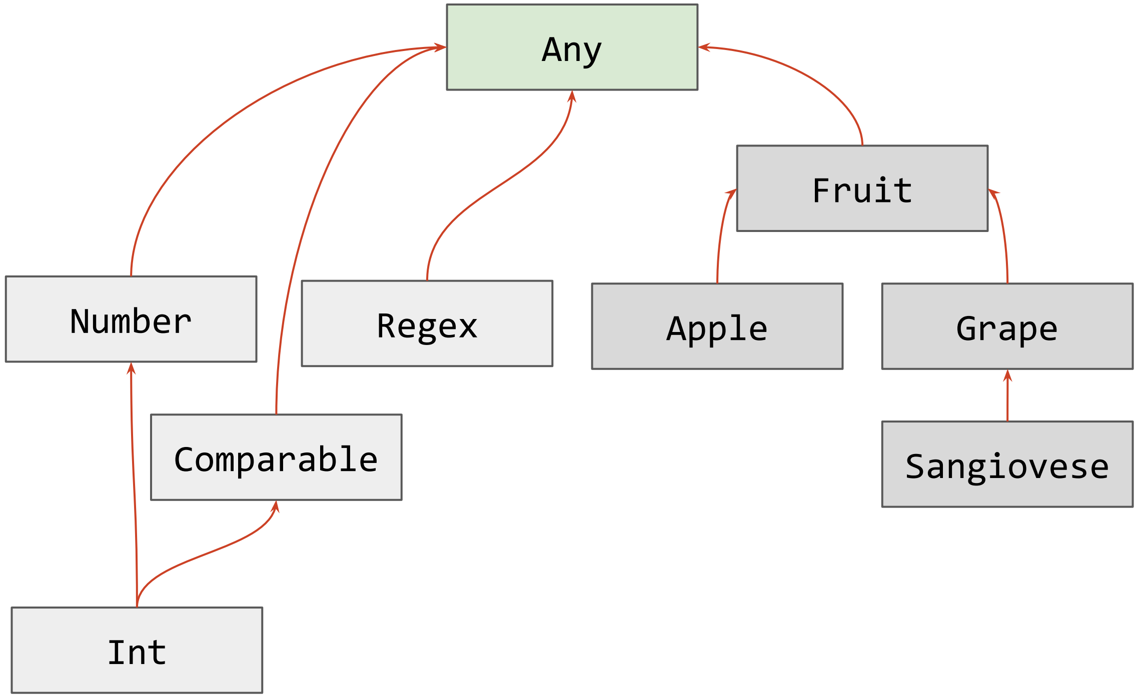


Основные типы

school.kt

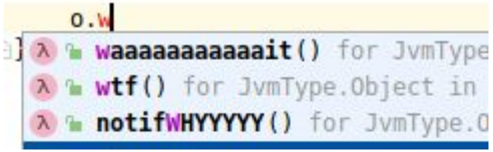
Any

```kotlin
interface Fruit

class Apple : Fruit

open class Grape : Fruit

class Sangiovese : Grape()
```



Object vs Any

- Object.equals()
- Object.hashCode()
- Object.toString()
- Object.getClass()
- Object.clone()
- Object.notify()
- Object.notifyAll()
- Object.wait()
- Object.wait(long)
- Object.wait(long, int)
- Object.finalize()



```
o.w
waaaaaaaaaaait() for JvmType
wtf() for JvmType.Object in
notifWHYYYYY() for JvmType.O
```

- `Any.equals()`
- `Any.hashCode()`
- `Any.toString()`

- `Any.javaClass()`
- `Any.let()`
- `Any....()`

school.kt

Object vs Any

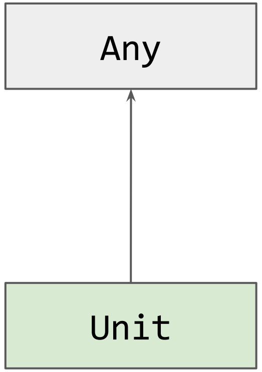
- Примитивы, не наследующие Object
- Неявное nullability
- 8 “лишних” методов!

- Нет примитивов
- Явное nullability
- Нет “лишних” методов

Unit

```
fun unit() {}

fun main() {
    println(unit())
}
```

```
>kotlin.Unit
```



Unit vs void

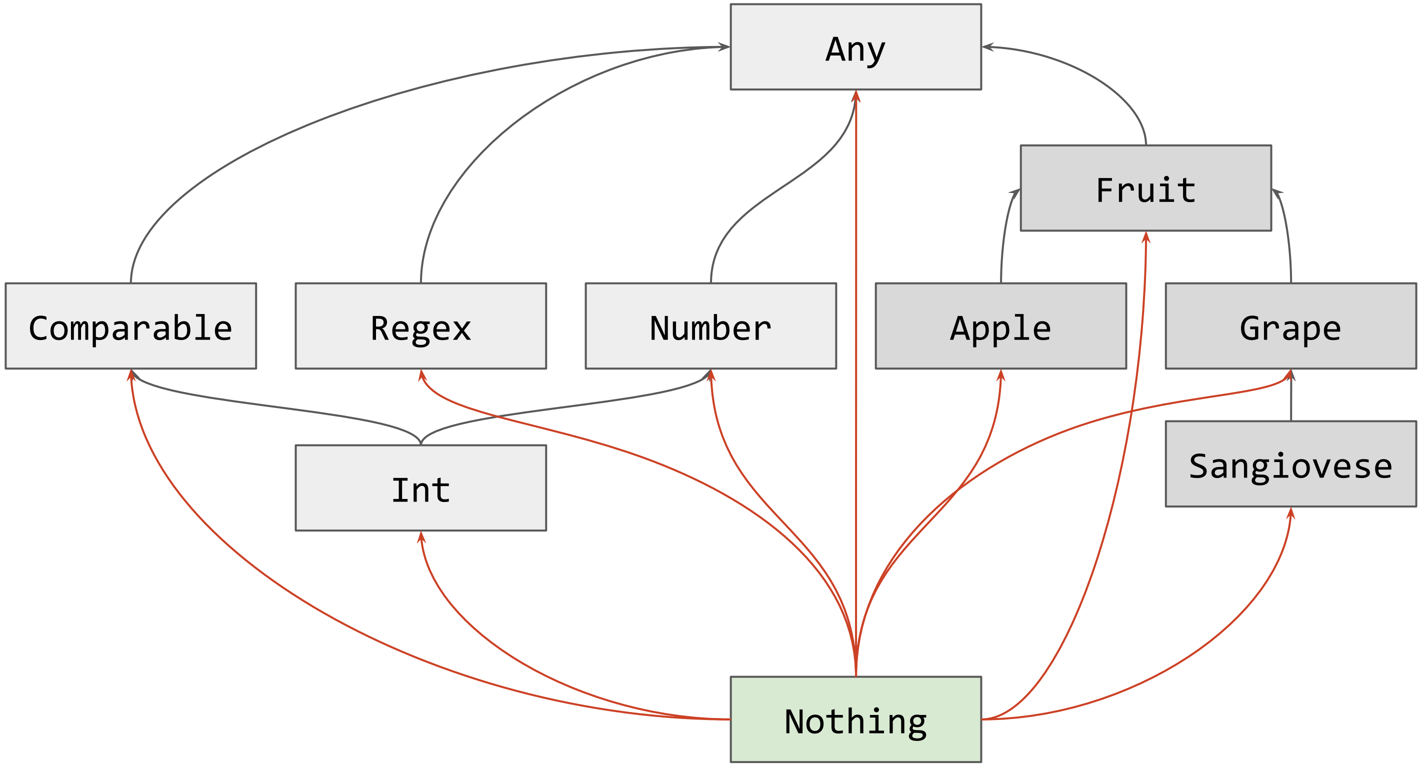
```kotlin
fun unit() {}

fun main() {
   println(unit())
}

>kotlin.Unit
```

```java
public class Demo {
   private static void unit() {}

   public static void main(String[] args) {
       System.out.println(unit());
   }
}
```

```
>Error:(6, 32) java: 'void' type not allowed here
```

school.kt

Nothing



school.kt

Nothing vs Unit

```kotlin
fun unit() {}
fun nothing(): Nothing = throw NotImplementedError()

fun main() {
   val u = unit()
   println(u)

   val n = nothing()
   println(n)
}
```

```
Error:(10, 5) Kotlin: Overload resolution ambiguity:
@InlineOnly public inline fun println(message: Any?): Unit defined in kotlin.io
@InlineOnly public inline fun println(message: Boolean): Unit defined in kotlin.io
```

Nothing vs Unit

```kotlin
fun unit() {}
fun nothing(): Nothing = throw NotImplementedError()

fun main() {
   unit()
   println("u")

   nothing()
   println("n")
}
```

```
>u
>Exception in thread "main" kotlin.NotImplementedError: An operation is not implemented.
    at StdlibKt.nothing(stdlib.kt:13)
```

school.kt

Nothing: недостижимый код

```kotlin
fun main() {
    return

    println("Hello? Anybody?") // Unreachable code
}
```

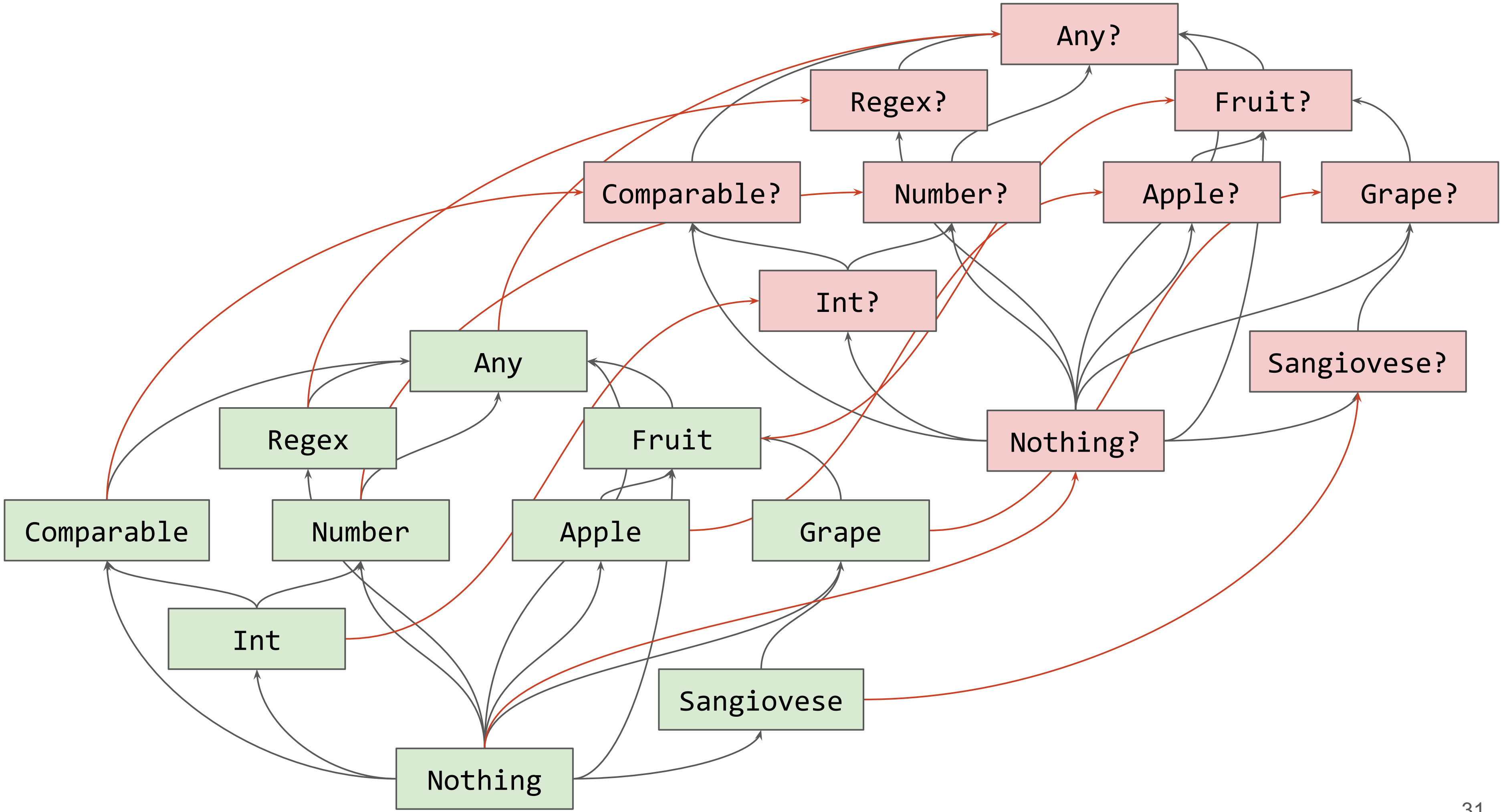
school.kt

Nothing: “чистые” типы

```kotlin
fun s(): String? = null

fun fail(): Nothing = throw IllegalArgumentException()

fun pass(): Unit {}

fun main() {
   val s1 /*String*/ = s() ?: return
   val s2 /*String*/ = s() ?: throw IllegalArgumentException()
   val s3 /*String*/ = s() ?: fail()
   val s4 /* Any */  = s() ?: pass()
}
```

А КАК ЖЕ
NULL?
risovach.ru

$$\begin{tikzcd}
\texttt{Any} \arrow[r] & \texttt{Any?} \\
\texttt{Fruit} \arrow[u] \arrow[r] & \texttt{Fruit?} \arrow[u]
\end{tikzcd}$$


Any?
Regex?
Fruit?
Comparable?
Number?
Apple?
Grape?
Int?
Any
Sangiovese?
Regex
Fruit
Nothing?
Comparable
Number
Apple
Grape
Int
Sangiovese
Nothing

А как в Java?

```
fun main() {
   println(Any::class.java)
   println(Unit::class.java)
   println(Nothing::class.java)
}
```

```
>class java.lang.Object
>class kotlin.Unit
>class java.lang.Void
```

school.kt

Какой тип у null?

```
Nothing?
```

Алиасы типов

```kotlin
typealias Width = Int
typealias Height = Int
typealias Size = Pair<Width, Height>

infix fun Width.x(h: Height) = Size(this, h)

fun window(size: Size) {}

fun main() {
    window(5 x 10)
}
```

Алиасы типов

```kotlin
@SinceKotlin("1.1") public actual typealias Appendable = java.lang.Appendable
```

school.kt

Алиасы в stdlib

```
typealias Comparator<T> = java.util.Comparator<T>
typealias Exception = java.lang.Exception
typealias Error = java.lang.Error
typealias RandomAccess = java.util.RandomAccess
typealias ArrayList<E> = java.util.ArrayList<E>
typealias LinkedHashMap<K, V> = java.util.LinkedHashMap<K, V>
typealias HashMap<K, V> = java.util.HashMap<K, V>
typealias LinkedHashSet<E> = java.util.LinkedHashSet<E>
typealias HashSet<E> = java.util.HashSet<E>
typealias Appendable = java.lang.Appendable
typealias StringBuilder = java.lang.StringBuilder
```

Pair и Triple

```
val p = Pair("stdlib", 3)
val t = Triple("stdlib", 3, LocalDate.now())
```

Result

```
val r = Result.success("Kotlin")

r.isFailure                                // false
r.isSuccess                                // true
r.exceptionOrNull()                        // null
r.getOrNull()                              // Kotlin
r.getOrDefault("Java")                     // Kotlin
r.fold(
      onSuccess = ::println,               // Kotlin
      onFailure = { exitProcess(1) }
)
r.map { it.reversed() }                    // niltoK
```

@Deprecated

```kotlin
@Deprecated(
        message = "Use error from stdlib",
        replaceWith = ReplaceWith(
                expression = "error",
                imports = ["kotlin.errror"]
        ),
        level = DeprecationLevel.WARNING
)
fun fail(): Nothing = throw IllegalArgumentException()
```

@Supress

```kotlin
@file:Suppress("UNREACHABLE_CODE")

fun main(args: Array<String>) {
   error("I'm done!")
   println("Who's left?")
}
```

school.kt

@JvmName

```kotlin
@file:JvmName("CallMeMaybe")

fun main(args: Array<String>) {
}
```

@Experimental

```
@Experimental(Experimental.Level.ERROR)
annotation class Experiment

@Experiment
fun watchOut() = ((0..1).random()).takeIf { it == 0 } ?: throw NotImplementedError()

fun main(args: Array<String>) {
   watchOut()
}
```

```
>Error:(10, 5) Kotlin: This declaration is experimental and its usage must be marked
with '@kt.school.Experiment' or '@UseExperimental(kt.school.Experiment::class)'
```

school.kt

@UseExperimental

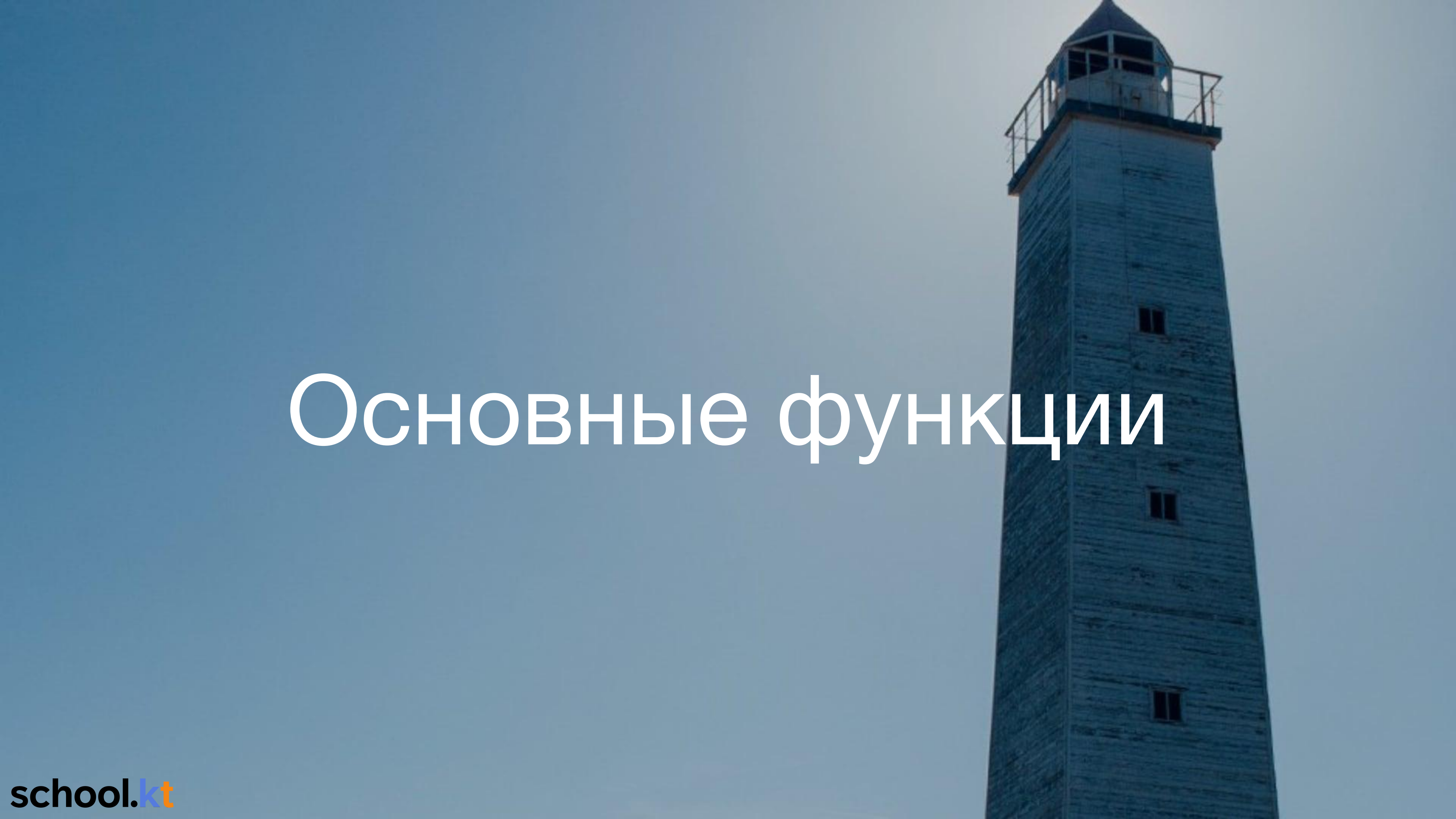
```
@Experimental(Experimental.Level.ERROR)
annotation class Experiment

@Experiment
fun watchOut() = ((0..1).random()).takeIf { it == 0 } ?: throw NotImplementedError()

@UseExperimental(Experiment::class)
fun main(args: Array<String>) {
    watchOut()
}
```

school.kt

Основные типы

- 🍹 Kotlin basics: types. Any, Unit and Nothing
- 🔥 A Whirlwind Tour of the Kotlin Type Hierarchy
- 🔥 The Kotlin Type Hierarchy From Top to Bottom
- 👷 The Nature of Nothing in Kotlin
- 👷 Nothing (else) matters in Kotlin

school.kt



Основные функции

school.kt

TODO

```
fun main() {
    TODO()
}
```

```
>Exception in thread "main" kotlin.NotImplementedError: An operation is not
implemented.
```

Бенчмарки для бедных

```
val b1 = measureNanoTime {
    repeat(100_000) {
        (1..10).toList().map { it * 2 }.filter { it >= 10 }.first { it % 2 == 0 }
    }
} // 339534319

val b2 = measureTimeMillis {
    repeat(100_000) {
        (1..10).asSequence().map { it * 2 }.filter { it >= 10 }.first { it % 2 == 0 }
    }
} // 156
```

System.exit()

```
fun main() {
    exitProcess(0)

    println("Hello, world!") // Unreachable code
}
```

let

```
fun <T, R> T.let(block: (T) -> R): R {
    return block(this)
}
```

school.kt

let

```
val result = listOf(1, 2, 3).let {
    it.sum()
}

println(result)   // 6
```

school.kt

let

```
System.getProperty("java.version")?.let {
    println("Known Java version: $it")
}
```

school.kt

apply

```kotlin
fun <T> T.apply(block: T.() -> Unit): T {
    block()
    return this
}
```

apply

```
val preparedMap = java.util.HashMap<String, String>().apply {
    this["❤"] = "Kotlin"
    this["†"] = "Java"
}

println(preparedMap)   // {†=Java, ❤=Kotlin}
```

with

```
fun <T, R> with(receiver: T, block: T.() -> R): R {
   return receiver.block()
}
```

school.kt

with

```
val map = java.util.HashMap<String, String>()
val empty = with(map) {
    this["❤"] = "Kotlin"
    this["†"] = "Java"

    isEmpty()
}

if (!empty) {
    println(map)   // {†=Java, ❤=Kotlin}
}
```

run

```
fun <R> run(block: () -> R): R {
    return block()
}
```

school.kt

run

```kotlin
val a = "Hello"
val answer = run {
   val a = 21 * 2

   a
}

if (answer > 0) {
   println(a)   // Hello
}
```

run

```kotlin
fun <T, R> T.run(block: T.() -> R): R {
    return block()
}
```

run

```kotlin
val question = "Life?"
val answer = question.run {
   "$this: 42"
}

println(answer)   // Life? 42
```

school.kt

also

```kotlin
fun <T> T.also(block: (T) -> Unit): T {
   block(this)
   return this
}
```

school.kt

also

```kotlin
val map = java.util.HashMap<String, String>()
    .also {
        it["❤"] = "Groovy"
    }
    .also {
        it.clear()
    }
    .also {
        it["❤"] = "Kotlin"
    }

println(map)   // {❤=Kotlin}
```

use

```kotlin
DriverManager.getConnection("...").use { connection ->
    connection.prepareStatement("...").use { statement ->
        statement.executeQuery().use { resultSet ->
            TODO()
        }
    }
}
```

require

```
require(false)
require(false) { "Failed requirement." }
requireNotNull(null)
requireNotNull(null) { "Failed requirement." }
```

```
>Exception in thread "main" java.lang.IllegalArgumentException: Failed requirement.
```

check / error

```
check(false)
check(false) { "Check failed." }
checkNotNull(null)
checkNotNull(null) { "Check failed." }

error("Check failed.")

>Exception in thread "main" java.lang.IllegalStateException: Check failed.
```

school.kt

assert

```
assert(false)
assert(false) { "Assertion failed" }

>Exception in thread "main" java.lang.AssertionError: Assertion failed
```

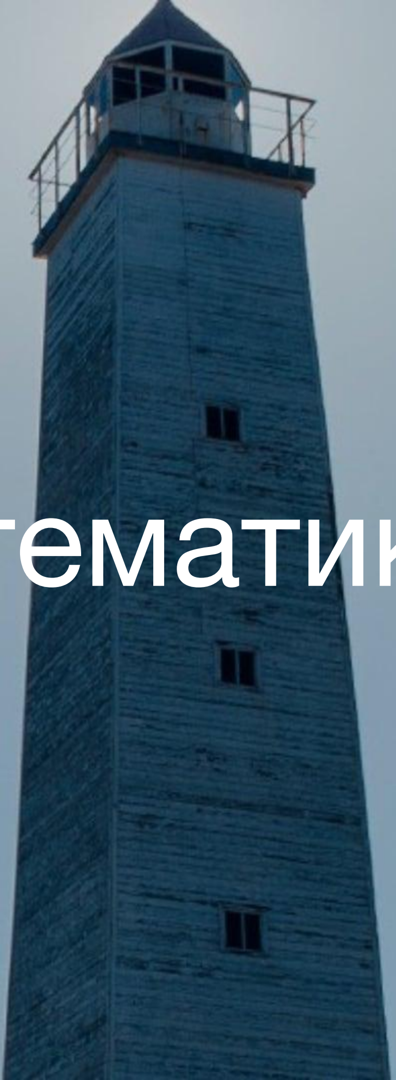
takeIf / takeUnless

```
"".takeIf { it.isNotBlank() }      // null
"".takeUnless { !it.isNotBlank() } // null
```

school.kt

Основные функции

- 👷 Exploring the Kotlin standard library
- 👷 The difference between Kotlin’s functions: ‘let’, ‘apply’, ‘with’, ‘run’ and ‘also’



Числа, логика и математика

school.kt

Математика

```
kotlin.math ≈ java.lang.Math
```

Числа

```
"127".toByte()                // 127
32767.toShort()               // 32767
2.5.toInt()                   // 2
2.5.roundToInt()              // 3
"NaN".toFloat()               // NaN
BigDecimal(2.5).toDouble()    // 2.5
```

Числа в Kotlin: операторы

```
val a: BigInteger = 100.toBigInteger()
var b: BigDecimal = 100.toBigDecimal()
val c: BigInteger = "10".toBigInteger()
val d: BigDecimal = "2.5".toBigDecimal()

println(-a);      println(b++)
println(a + c);   println(a - c)
println(b * d);   println(b / d)
println(a % c);   println(a and c)
println(a shl 2); println(a xor c)
```

Числа: расширения

```
println(42.0.isFinite())                         // true
println(Double.POSITIVE_INFINITY.isInfinite()) // true
println(Double.NaN.isNaN())                      // true
```

inline-классы

```kotlin
inline class Width(val value: Int)
inline class Height(val value: Int)
typealias Size = Pair<Width, Height>

infix fun Width.x(h: Height) = Size(this, h)

fun window(size: Size) {}

fun main() {
   window(Width(5) x Height(10))
}
```

school.kt

Беззнаковые типы

```
public inline class UInt internal constructor(
    internal val data: Int
) : Comparable<UInt>
```

school.kt

Беззнаковые типы

```
typealias u_int = Int

fun main() {
    val a = 0xFF.toUByte();          val b = 0xFFFFFFFFu
    val c = "FFFFFF".toUInt(16);  val d: u_int = 5

    println(a);          println(b);
    println(c);          println(d);
    println(a::class);  println(d::class);
}

>255                  4294967295
>16777215             5
>class kotlin.UByte   class kotlin.Int
```

Random

```
kotlin.Random ≈ java.util.Random
```

school.kt

Random

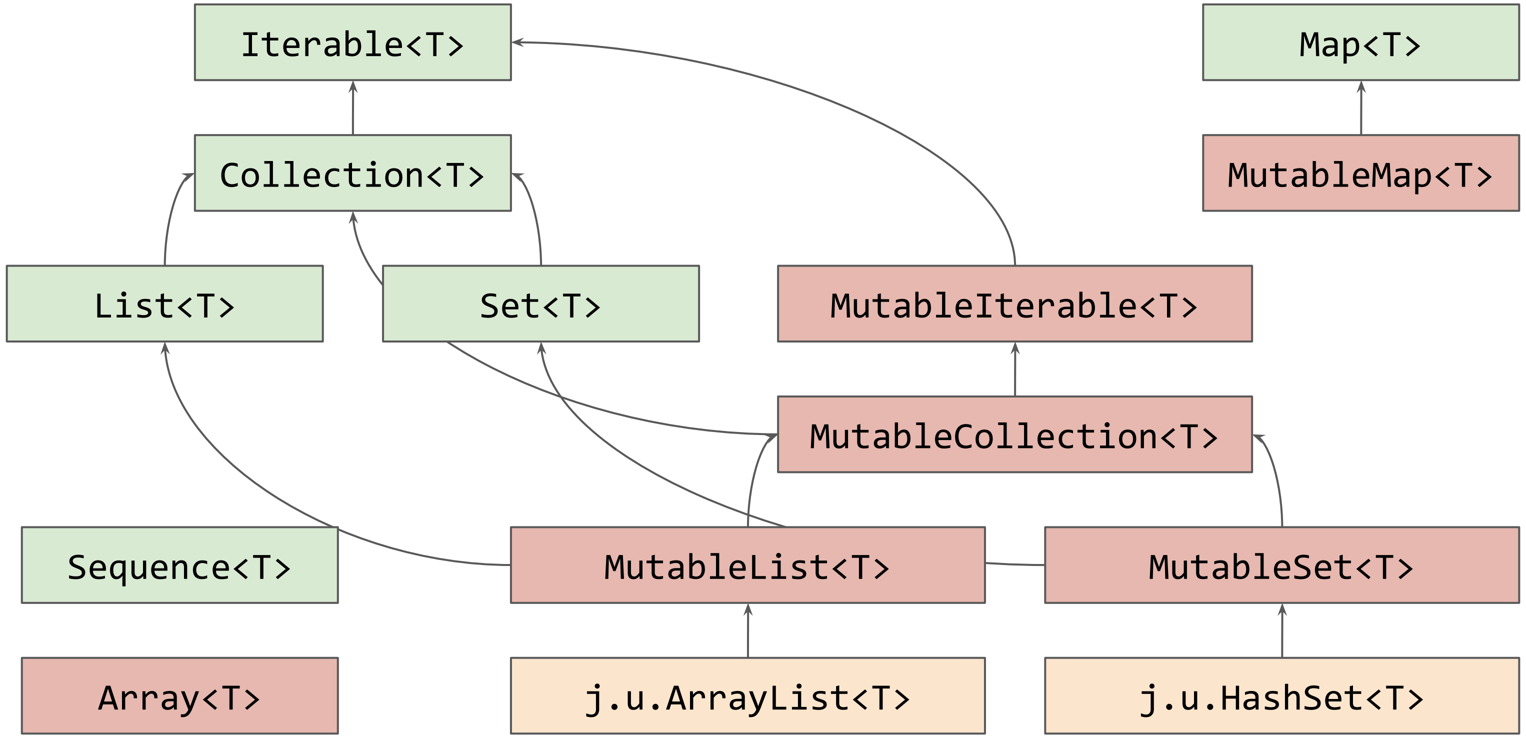
```
@SinceKotlin("1.3")
public fun Random.asJavaRandom(): java.util.Random = …

@SinceKotlin("1.3")
public fun java.util.Random.asKotlinRandom(): Random = …
```

Boolean

```kotlin
val a = true
val b = false

println(!a)
println(a and b)
println(a or b)
println(a xor b)
```

```
withTimeout(5 * 60 * 1000) {
    val relaxation = rest()
}
```

Контейнеры


Iterable<T>
Map<T>
Collection<T>
MutableMap<T>
List<T>
Set<T>
MutableIterable<T>
MutableCollection<T>
Sequence<T>
MutableList<T>
MutableSet<T>
Array<T>
j.u.ArrayList<T>
j.u.HashSet<T>

school.kt

Массивы: Java vs Kotlin

```java
Fruit a[] = {};
Apple b[] = {};
Grape c[] = {};

a = b;    // OK
// b = c; // Incompatible types.
```

```kotlin
var a = arrayOf<Fruit>()
var b: Array<Apple?> = arrayOfNulls(3)
var c = emptyArray<Grape>()

// a = b // Type mismatch.
// b = c // Type mismatch.
```

school.kt

Массивы примитивных типов

```kotlin
val a: BooleanArray = booleanArrayOf()
val b: ByteArray = byteArrayOf()
val c: CharArray = charArrayOf()
val d: DoubleArray = doubleArrayOf()
val e: FloatArray = floatArrayOf()
val f: IntArray = intArrayOf()
val g: LongArray = longArrayOf()
val h: ShortArray = shortArrayOf()
val i: UByteArray = ubyteArrayOf()
val j: UIntArray = uintArrayOf()
val k: ULongArray = ulongArrayOf()
val l: UShortArray = ushortArrayOf()
```

```java
boolean[] a
byte[] b
char[] c
double[] d
float[] e
int[] f
long[] g
short[] h
byte[] i
int[] j
long[] k
short[] l
```

Массивы примитивных типов

```kotlin
val a: BooleanArray = emptyArray<Boolean>().toBooleanArray()
val b: ByteArray = emptyArray<Byte>().toByteArray()
val c: CharArray = emptyArray<Char>().toCharArray()
val d: DoubleArray = emptyArray<Double>().toDoubleArray()
val e: FloatArray = emptyArray<Float>().toFloatArray()
val f: IntArray = emptyArray<Int>().toIntArray()
val g: LongArray = emptyArray<Long>().toLongArray()
val h: ShortArray = emptyArray<Short>().toShortArray()
val i: UByteArray = byteArrayOf().toUByteArray()
val j: UIntArray = intArrayOf().toUIntArray()
val k: ULongArray = longArrayOf().toULongArray()
val l: UShortArray = shortArrayOf().toUShortArray()
```

Массивы примитивных типов

```
val a: ByteArray = ubyteArrayOf().asByteArray()
val b: IntArray = uintArrayOf().asIntArray()
val c: LongArray = ulongArrayOf().asLongArray()
val d: ShortArray = ushortArrayOf().asShortArray()
```

Списки / Lists

```
val a = emptyList<String>()                   // kotlin.collections.EmptyList
val b = listOf<String>()                      // kotlin.collections.EmptyList
val c = listOf("Kotlin")                      // java.util.Collections$SingletonList
val d = listOf("I", "❤", "Kotlin")            // java.util.Arrays$ArrayList
val e = mutableListOf<String>()               // java.util.ArrayList
val f = listOfNotNull("I", null, "Java")      // java.util.ArrayList
val g = arrayListOf<String>()                 // java.util.ArrayList
```

Множества / Sets

```
val a = emptySet<String>()           // kotlin.collections.EmptySet
val b = setOf<String>()              // kotlin.collections.EmptySet
val c = setOf("Kotlin")              // java.util.Collections$SingletonSet
val d = setOf("I", "❤", "Kotlin")  // java.util.LinkedHashSet
val e = mutableSetOf<String>()       // java.util.LinkedHashSet
val f = hashSetOf<String>()          // java.util.HashSet
val j = linkedSetOf<String>()        // java.util.LinkedHashSet
val h = sortedSetOf<String>()        // java.util.TreeSet
```

school.kt

Словари / Maps

```
val a = emptyMap<String, String>()                       // kotlin.collections.EmptyMap
val b = mapOf("❤" to "Kotlin")                           // java.util.Collections$SingletonMap
val c = mapOf("❤" to "Kotlin", "†" to "Java")            // java.util.LinkedHashMap
val d = mutableMapOf<String, String>()                   // java.util.LinkedHashMap
val e = hashMapOf<String, String>()                      // java.util.HashMap
val f = linkedMapOf<String, String>()                    // java.util.LinkedHashMap
val g = sortedMapOf<String, String>()                    // java.util.TreeMap
```

school.kt

JDK 8+ потоки / Streams

```
val strings = Stream.of("I", "❤", "Kotlin")

strings.asSequence()
strings.toList()
```

school.kt

Последовательности / Sequences

```
val a = emptySequence<String>()
val b = sequenceOf("I", "❤", "Kotlin")
val c = sequence {
   yield("I")
   yield("❤")
   yield("Kotlin")
}
val d = Collections.enumeration(listOf(1, 2, 3)).asSequence()
val e = listOf(1, 2, 3).asSequence()
```

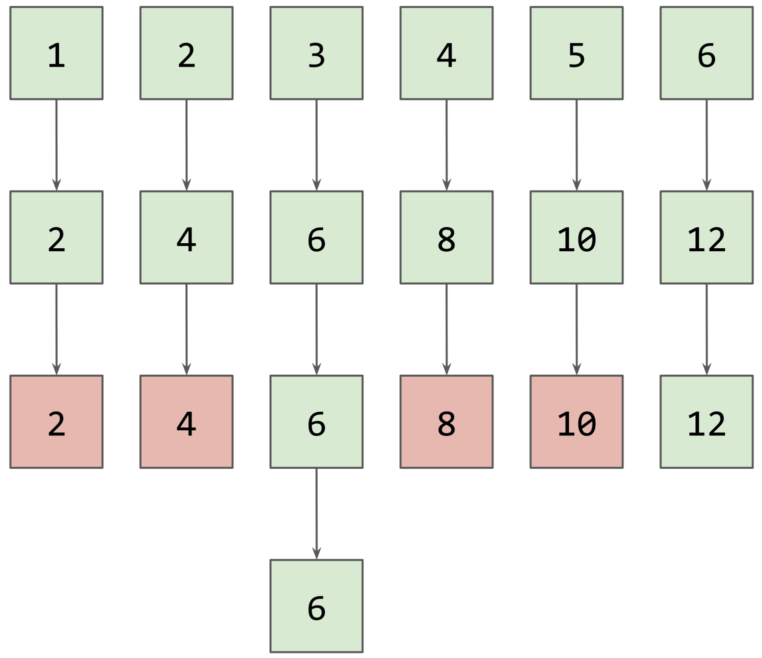
Списки vs последовательности

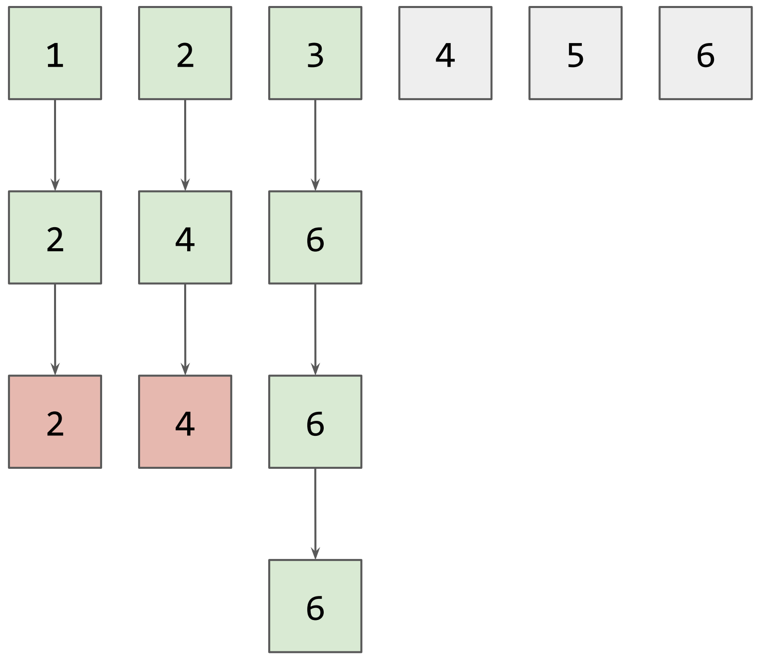
```
var ops = 0
val list = listOf(1, 2, 3, 4, 5, 6)
val result = list
    .map { ops++; it * 2 }
    .filter { ops++; it % 3 == 0 }
    .first { ops++; it > 3 }

println("$result in $ops ops")

>6 in 13 ops
```

```
var ops = 0
val list = listOf(1, 2, 3, 4, 5, 6)
val result = list.asSequence()
    .map { ops++; it * 2 }
    .filter { ops++; it % 3 == 0 }
    .first { ops++; it > 3 }

println("$result in $ops ops")

>6 in 7 ops
```

school.kt

Списки vs последовательности





school.kt

Последовательности vs JDK 8+ потоки

```
var ops = 0
val list = listOf(1, 2, 3, 4, 5, 6)
val result = list.asSequence()
    .map { ops++; it * 2 }
    .filter { ops++; it % 3 == 0 }
    .first { ops++; it > 3 }

println("$result in $ops ops")

>6 in 7 ops
```

```
var ops = 0
val list = listOf(1, 2, 3, 4, 5, 6)
val result = list.stream()
    .map { ops++; it * 2 }
    .filter { ops++; it % 3 == 0 }
    .first { ops++; it > 3 }

println("$result in $ops ops")


>Unresolved reference.
```

school.kt

Последовательности vs JDK 8+ потоки

- Больше операций
- Последовательные
- Можно использовать на JDK 8-, Kotlin/JS и Kotlin/Native

- Меньше операций
- Последовательные и параллельные
- Только JDK 8+

school.kt

Интервалы / Ranges

```
val a = 1..10                    // 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
val b = 1 until 10 step 2        // 1, 3, 5, 7, 9
val c = 10 downTo 1              // 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
val d = (10 downTo 1).reversed() // 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

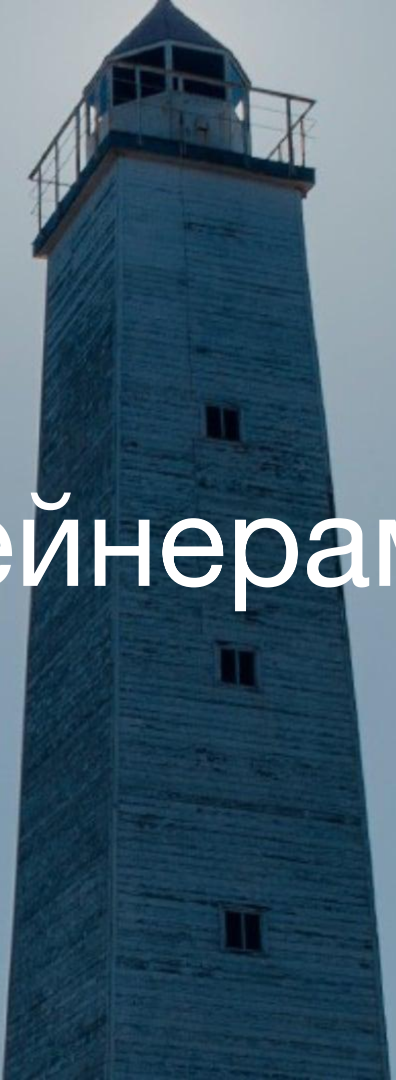
Интервалы / Ranges

```kotlin
val a = 0.coerceIn(1..10)  // 1
val b = 5.coerceIn(1..10)  // 5
val c = 11.coerceIn(1..10) // 10
```

school.kt

Прогрессии / Progressions

```
(1..5).joinToString()                            // 1, 2, 3, 4, 5
("Ada".."Visual Basic").contains("Kotlin")  // true
```



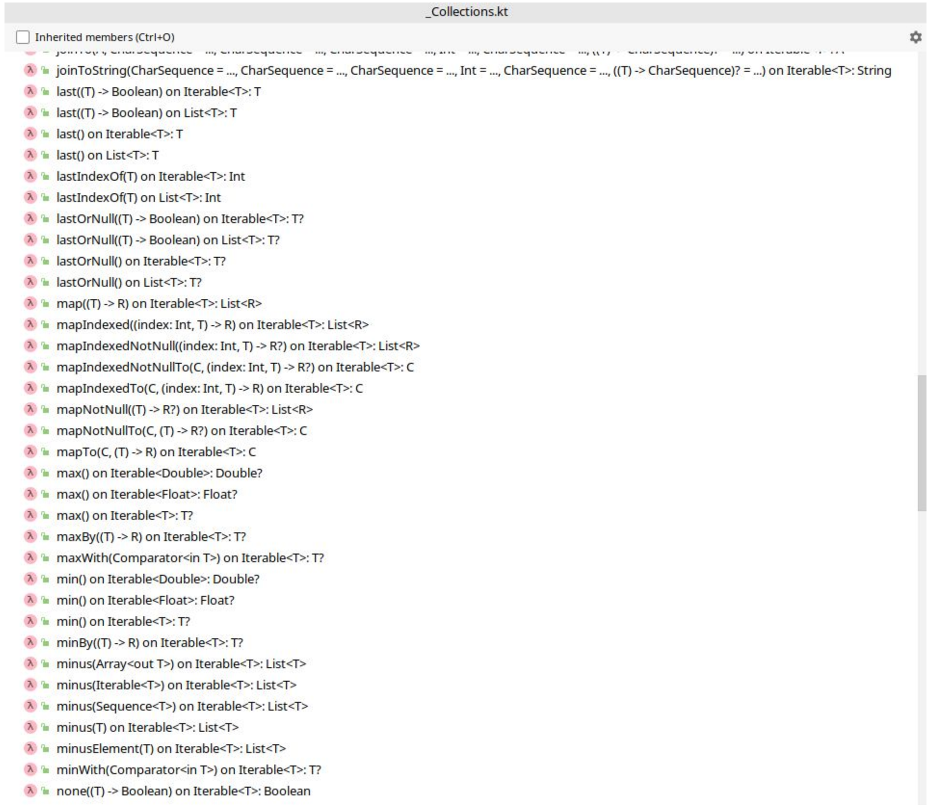
Операции над контейнерами

school.kt

Где все методы?

```kotlin
public interface Iterable<out T> {
   public operator fun iterator(): Iterator<T>
}

public interface Collection<out E> : Iterable<E> {
   public val size: Int
   public fun isEmpty(): Boolean
   public operator fun contains(element: @UnsafeVariance E): Boolean
   override fun iterator(): Iterator<E>
   public fun containsAll(elements: Collection<@UnsafeVariance E>): Boolean
}
```

_Collections.kt



_Collections.kt

Inherited members (Ctrl+O)

```
joinToString(CharSequence = ..., CharSequence = ..., CharSequence = ..., Int = ..., CharSequence = ..., ((T) -> CharSequence)? = ...) on Iterable<T>: String
last((T) -> Boolean) on Iterable<T>: T
last((T) -> Boolean) on List<T>: T
last() on Iterable<T>: T
last() on List<T>: T
lastIndexOf(T) on Iterable<T>: Int
lastIndexOf(T) on List<T>: Int
lastOrNull((T) -> Boolean) on Iterable<T>: T?
lastOrNull((T) -> Boolean) on List<T>: T?
lastOrNull() on Iterable<T>: T?
lastOrNull() on List<T>: T?
map((T) -> R) on Iterable<T>: List<R>
mapIndexed((index: Int, T) -> R) on Iterable<T>: List<R>
mapIndexedNotNull((index: Int, T) -> R?) on Iterable<T>: List<R>
mapIndexedNotNullTo(C, (index: Int, T) -> R?) on Iterable<T>: C
mapIndexedTo(C, (index: Int, T) -> R) on Iterable<T>: C
mapNotNull((T) -> R?) on Iterable<T>: List<R>
mapNotNullTo(C, (T) -> R?) on Iterable<T>: C
mapTo(C, (T) -> R) on Iterable<T>: C
max() on Iterable<Double>: Double?
max() on Iterable<Float>: Float?
max() on Iterable<T>: T?
maxBy((T) -> R) on Iterable<T>: T?
maxWith(Comparator<in T>) on Iterable<T>: T?
min() on Iterable<Double>: Double?
min() on Iterable<Float>: Float?
min() on Iterable<T>: T?
minBy((T) -> R) on Iterable<T>: T?
minus(Array<out T>) on Iterable<T>: List<T>
minus(Iterable<T>) on Iterable<T>: List<T>
minus(Sequence<T>) on Iterable<T>: List<T>
minus(T) on Iterable<T>: List<T>
minusElement(T) on Iterable<T>: List<T>
minWith(Comparator<in T>) on Iterable<T>: T?
none((T) -> Boolean) on Iterable<T>: Boolean
```

school.kt

Поиск

`allMatch`	→	`all`
`anyMatch`	→	`any`
`noneMatch`	→	`none`
`max`	→	`max`
		`maxBy`
		`maxWith`
`min`	→	`min`
		`minBy`
		`minWith`

Фильтрация

```
filter      →    filter
                 filterIndexed
                 filterIsInstance
                 filterNotNull
                 filterNot
```

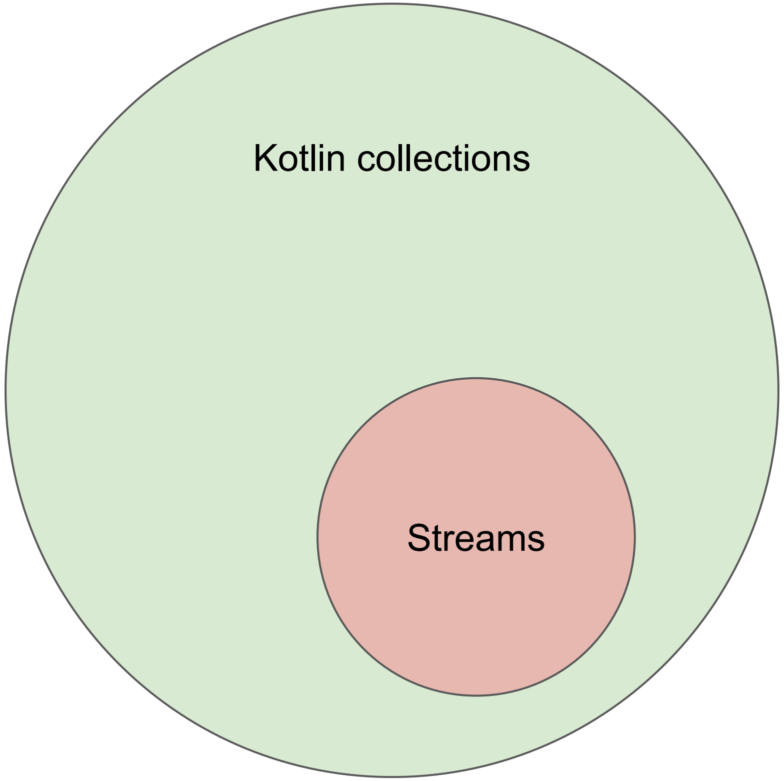
Сортировка

```
sorted      →     sorted
                  sortedBy
                  sortedByDescending
                  sortedDescending
                  sortedWith
```

skip / limit

`skip`	→	`drop`
		`dropLast`
		`dropLastWhile`
`limit`	→	`take`
		`takeLast`
		`takeLastWhile`

Kotlin collections vs Streams



school.kt

Операторы

```
setOf(1, 2, 3, 4) + setOf(2, 4, 6, 8, 10)  // [1, 2, 3, 4, 6, 8, 10]
listOf(1, 2, 3, 4) - setOf(2, 4, 6, 8, 10) // [1, 3]
mapOf(1 to "1", 2 to "2") + mapOf(2 to "II", 3 to "III") // {1=1, 2=II, 3=III}
```

Деструктуризация

```
val (a, _, c) = (1..10).toList()  // 1, 3
```

school.kt

zip / zipWithNext

```kotlin
val a = listOf(1, 2, 3, 4)
val b = listOf("A", "B", "C", "D", "E")
val c = listOf(1, "A", 2, "B", 3)

a zip b          // [(1, A), (2, B), (3, C), (4, D)]
c.zipWithNext() // [(1, A), (A, 2), (2, B), (B, 3)]
```

windowed

```
val a = listOf(1, 2, 3, 4, 5, 6, 7)
a.windowed(size = 3, step = 2) // [[1, 2, 3], [3, 4, 5], [5, 6, 7]]
```

school.kt

groupBy

```kotlin
val langs = listOf("Java", "Kotlin", "Groovy", "Scala").groupBy { it.length }

langs   // {4=[Java], 6=[Kotlin, Groovy], 5=[Scala]}
```

school.kt

fold / reduce

```
val balance = listOf('(', '(', '(', ')', ')', ')', '(', ')', ')', ')')
    .fold(0) { i, c ->
        when (c) {
            '(' -> i + 1
            ')' -> i - 1
            else -> throw IllegalStateException()
        }
    }

println(balance)  // -2
```

sorted

```
data class Record(val id: Long, val favorite: Boolean, val created: Long)

fun main(args: Array<String>) {
   val list = listOf(Record(1, true, 10), Record(2, false, 20), Record(3, true, 30),
Record(4, true, 30))

   list.sortedWith(
      compareBy({ !it.favorite }, Record::created).thenByDescending { it.id }
   )
}

// [Record(id=1), Record(id=4), Record(id=3), Record(id=2)]
```

school.kt

school.kt

Контейнеры

👷 Kotlin : Slow List and Lazy Sequence

🔥 Java 8 Stream API Analogies in Kotlin

school.kt

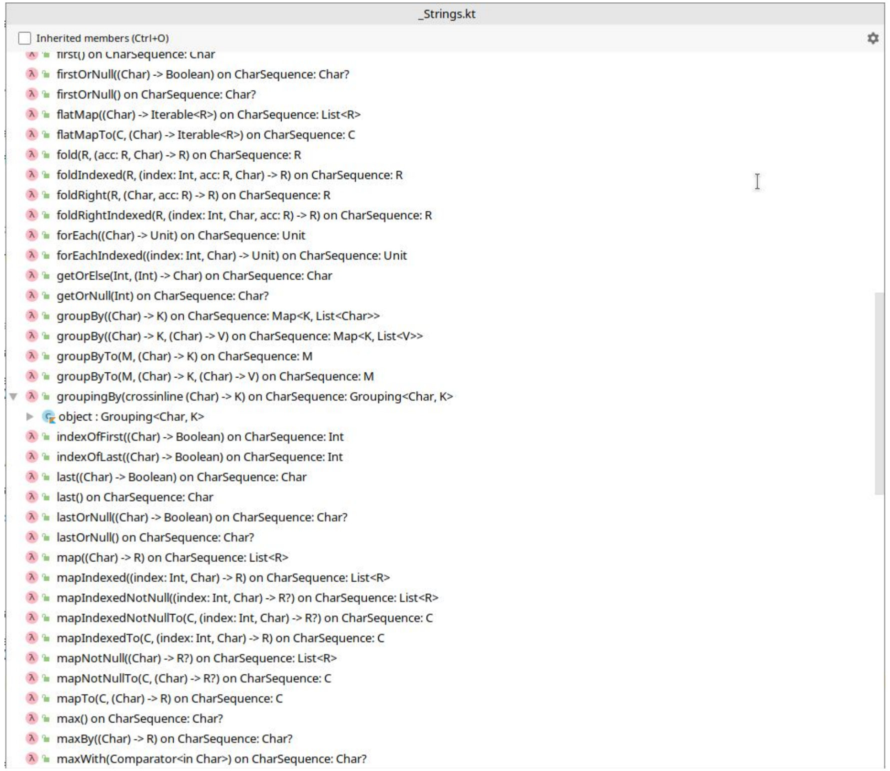
Строки и регулярные выражения

school.kt

Где все методы?

```
public interface CharSequence {
   public val length: Int
   public operator fun get(index: Int): Char
   public fun subSequence(startIndex: Int, endIndex: Int): CharSequence
}

public class String : Comparable<String>, CharSequence {
   public operator fun plus(other: Any?): String
   public override val length: Int
   public override fun get(index: Int): Char
   public override fun subSequence(startIndex: Int, endIndex: Int): CharSequence
   public override fun compareTo(other: String): Int
}
```

_String.kt


_Strings.kt

Inherited members (Ctrl+O)

```
first() on CharSequence: Char
firstOrNull((Char) -> Boolean) on CharSequence: Char?
firstOrNull() on CharSequence: Char?
flatMap((Char) -> Iterable<R>) on CharSequence: List<R>
flatMapTo(C, (Char) -> Iterable<R>) on CharSequence: C
fold(R, (acc: R, Char) -> R) on CharSequence: R
foldIndexed(R, (index: Int, acc: R, Char) -> R) on CharSequence: R
foldRight(R, (Char, acc: R) -> R) on CharSequence: R
foldRightIndexed(R, (index: Int, Char, acc: R) -> R) on CharSequence: R
forEach((Char) -> Unit) on CharSequence: Unit
forEachIndexed((index: Int, Char) -> Unit) on CharSequence: Unit
getOrElse(Int, (Int) -> Char) on CharSequence: Char
getOrNull(Int) on CharSequence: Char?
groupBy((Char) -> K) on CharSequence: Map<K, List<Char>>
groupBy((Char) -> K, (Char) -> V) on CharSequence: Map<K, List<V>>
groupByTo(M, (Char) -> K) on CharSequence: M
groupByTo(M, (Char) -> K, (Char) -> V) on CharSequence: M
groupingBy(crossinline (Char) -> K) on CharSequence: Grouping<Char, K>
    object : Grouping<Char, K>
indexOfFirst((Char) -> Boolean) on CharSequence: Int
indexOfLast((Char) -> Boolean) on CharSequence: Int
last((Char) -> Boolean) on CharSequence: Char
last() on CharSequence: Char
lastOrNull((Char) -> Boolean) on CharSequence: Char?
lastOrNull() on CharSequence: Char?
map((Char) -> R) on CharSequence: List<R>
mapIndexed((index: Int, Char) -> R) on CharSequence: List<R>
mapIndexedNotNull((index: Int, Char) -> R?) on CharSequence: List<R>
mapIndexedNotNullTo(C, (index: Int, Char) -> R?) on CharSequence: C
mapIndexedTo(C, (index: Int, Char) -> R) on CharSequence: C
mapNotNull((Char) -> R?) on CharSequence: List<R>
mapNotNullTo(C, (Char) -> R?) on CharSequence: C
mapTo(C, (Char) -> R) on CharSequence: C
max() on CharSequence: Char?
maxBy((Char) -> R) on CharSequence: Char?
maxWith(Comparator<in Char>) on CharSequence: Char?
```

school.kt

Строки

```
String ≈ Collection / Iterable / Sequence
```

repeat

```
"*".repeat(10) // **********
```

capitalize / upperCase

```
"i ❤ kotlin".toUpperCase()  // I ❤ KOTLIN
"i ❤ kotlin".capitalize()   // I ❤ kotlin
"I ❤ Java".toLowerCase()    // i ❤ java
"I ❤ Java".decapitalize()   // i ❤ Java
```

school.kt

commonPrefixWith / commonSuffixWith

```
"I ❤ Kotlin".commonPrefixWith("I ❤ Java")           // I ❤
"I ❤ Kotlin".commonSuffixWith("Everybody ❤ Kotlin")  //  ❤ Kotlin
```

school.kt

StringUt…

```
"I ❤ Kotlin".isBlank()              // false
"I ❤ Kotlin".isEmpty()              // false
"I ❤ Kotlin".isNotBlank()           // true
"".isNotEmpty()                     // false
"   ".isNullOrBlank()               // true
(null as String?).isNullOrEmpty()   // true
"   ".ifBlank { "N/A" }             // N/A
"".ifEmpty { "N/A" }                // N/A
```

school.kt

lines

```
"Too\nmany\nlines".lines()             // [Too, many, lines]
"Too\nmany\nlines".lineSequence()      // [Too, many, lines]
```

school.kt

pad

```
"Kotlin".padEnd(10, '_')   // Kotlin____
"Kotlin".padStart(10, '_') // ____Kotlin
```

slice / remove

```
"Kotlin".slice(0..2)                                // Kot
"___Kotlin".removePrefix("___")                     // Kotlin
"Kotlin___".removeSuffix("___")                     // Kotlin
"Kotlin".removeRange(1 until 3)                     // Klin
"===Kotlin!!!".removeSurrounding("===", "!!!") // Kotlin
```

trim

```
"""I
   |❤
   |Kotlin""".trimMargin(marginPrefix = "|").replace("\n", " ")  // I ❤ Kotlin
" Kotlin ".trim()                                                // Kotlin
" Kotlin ".trim()                                                // Kotlin
" Kotlin".trimStart()                                            // Kotlin
"Kotlin ".trimEnd()                                              // Kotlin
```

school.kt

replace

```
// Java
"I.❤.Kotlin".replaceAll(".", " "); //

// Kotlin
"I.❤.Kotlin".replace(".", " ")               // I ❤ Kotlin
"I.❤.Kotlin".replace("\\.".toRegex(), " ") // I ❤ Kotlin
"I ❤ Java".replaceAfter("I ❤ ", "Kotlin") // I ❤ Kotlin
"I ❤ Java".replaceBefore("❤", "Nobody ")  // Nobody ❤ Java
```

school.kt

matches

```
"I ❤ Kotlin".matches(Regex(".*❤.*")) // true
```

school.kt

Деструктуризация Regex

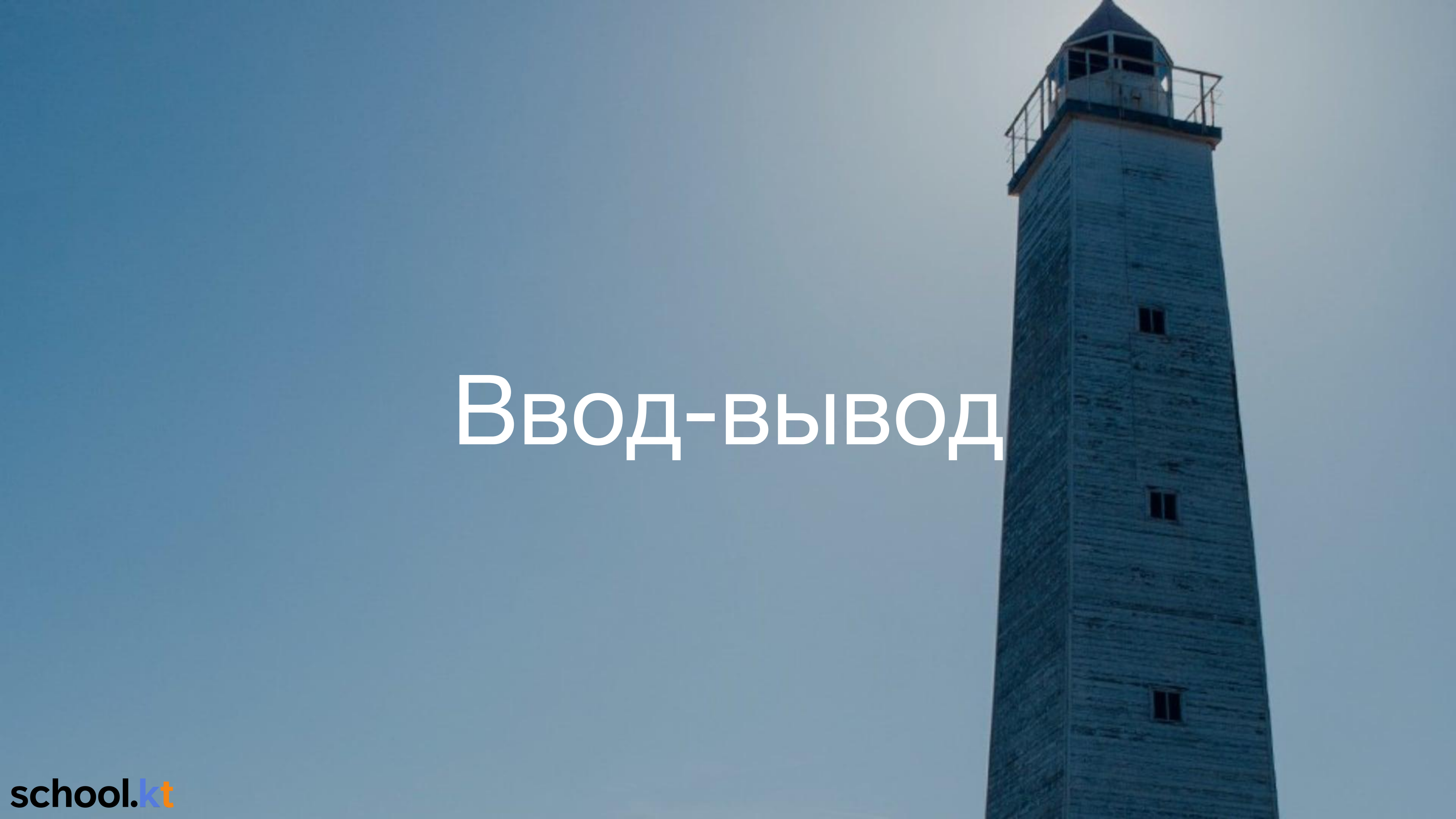
```
Regex("""([\w\s]+) ❤ ([\w]+)""")
    .find("I ❤ Kotlin")
    ?.destructured
    ?.let { (who, language) ->
        println(who)          // I
        println(language)     // Kotlin
    }
```

school.kt

Типографика

```
"JVM ${Typography.bullet} Native ${Typography.bullet} JS"  // JVM • Native • JS
"2 ${Typography.times} 2 ${Typography.almostEqual} 5"      // 2 × 2 ≈ 5
```

school.kt



Ввод-вывод

readLine / println

```
val s = readLine()

println(s)          // ???
```

Потоки из контейнеров

```
"I ❤ Kotlin".byteInputStream(Charsets.UTF_8) // [73, 32, ...]
byteArrayOf(1, 2, 3).inputStream()          // [1, 2, 3]
```

IOUt…

```
val i = File("/input").inputStream()
val o = File("/output").outputStream()

i.buffered()       // BufferedInputStream
o.buffered()       // BufferedOutputStream
i.reader()         // InputStreamReader
o.writer()         // OutputStreamWriter
i.bufferedReader() // BufferedReader
o.bufferedWriter() // BufferedWriter
i.copyTo(o)        // Yay!
i.readBytes()      // Yay!
```

StringReader

```kotlin
Scanner("1 2.5 3".reader())
    .also { println(it.nextInt()) }        // 1
    .also { println(it.nextDouble()) }     // 2.5
    .also { println(it.nextBigInteger()) } // 3
```

FileUt…

```kotlin
val root = File("/school.kt")
val presentation = File("/school.kt/stdlib/lection.pptx")

root.extension                    // kt
root.isRooted                     // true
presentation.relativeTo(root) // stdlib/lection.pptx
```

FileUt…

```
val i = File("/input")

i.readLines() // List<String>
i.useLines { lines ->
   // Sequence<String>
}
i.forEachLine { line ->
   // String
}
i.readText()  // String
i.readBytes() // ByteArray
```

FileUt…

```
val o = File("/output")

o.writeBytes(byteArrayOf(1, 2, 3))
o.writeText("Overwrite with this!")
o.appendBytes(byteArrayOf(1,2,3))
o.appendText("Append this!")
```

Временные файлы

```
val file = createTempFile() // /tmp/tmp11594971236686759544.tmp
val dir = createTempDir()   // /tmp/tmp6294446080465211194.tmp
```

school.kt

FileTreeWalk

```kotlin
val a = File("/path/to/some/dir")
    .walk()
    .onEnter { println("Entering $it"); true }
    .onLeave { println("Exiting $it") }   // FileTreeWalk : Sequence<File>

a.forEach(::println)
```

HTTP для бедных

```
URL("https://httpbin.org/ip").readText()   // {"origin": "1.1.1.1"}
URL("https://httpbin.org/ip").readBytes()  // [123, 10, 32, ...]
```

school.kt

Работа с многопоточностью

school.kt

Потоки

```
thread {
    Thread.sleep(1000)
    println("World!")
}

print("Hello, ")


>Hello, World!
```

Локи

```kotlin
val lock = ReentrantReadWriteLock()

fun main(args: Array<String>) {
    lock.readLock().withLock { /* Access shared resource */ }

    lock.read { /* Access shared resource */ }

    lock.write { /* Access shared resource */ }
}
```

school.kt

ThreadLocal

```kotlin
val mdc = ThreadLocal<Int>()

fun main(args: Array<String>) {
    mdc.get()              // null
    mdc.getOrSet { 42 } // 42
    mdc.get()              // 42
}
```

school.kt

Таймеры

```
fixedRateTimer(period = 1_000) {
    print("Heartbeat")
}

timer(period = 1_000) {
    print("Heartbeat")
}
```

JVM-интероп

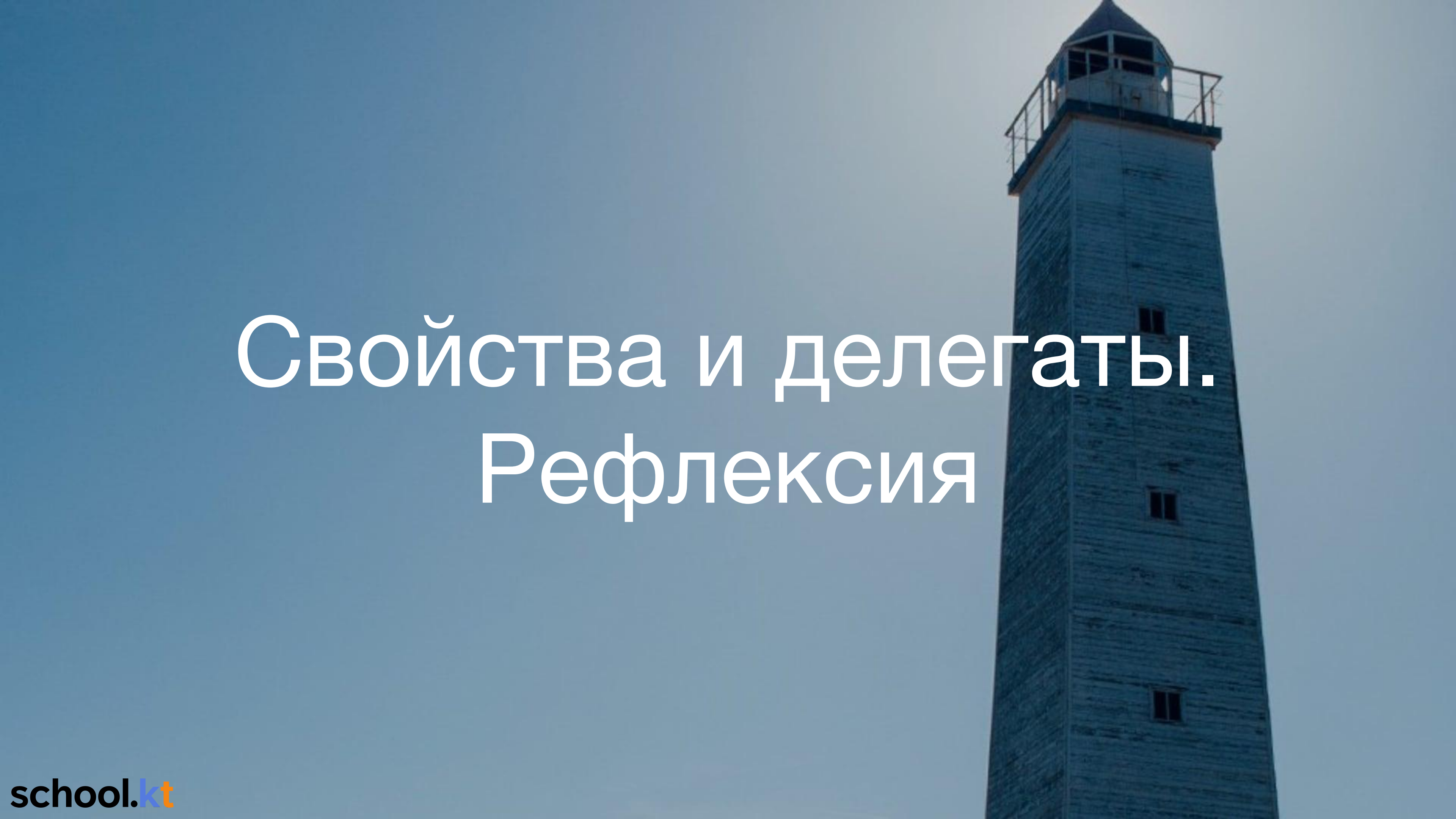
```
class Concurrent {
   @Volatile
   var e: Long = 57005

   @Synchronized
   fun synchronizedMethod() {
       println("Synchronized method")
   }

   fun syncronizedBlock() {
       synchronized(e) {
           println("Synchronized block")
       }
   }
}
```

school.kt

Основные функции

👷 Concurrency Primitives in Kotlin

school.kt



Свойства и делегаты. Рефлексия

Делегирование свойств

```
fun randomString() = ('A'..'z').map { it }.shuffled().subList(0, 6).joinToString("")

object SP∆CE {
   operator fun getValue(thisRef: Any?, property: KProperty<*>): String {
       return randomString()
   }
   operator fun setValue(thisRef: Any?, property: KProperty<*>, value: String) {
       println("$thisRef.${property.name} ← $value")
   }
}
```

Делегирование свойств

```
var SPΔCE by SPΔCE

println(SPΔCE)       // YwuUIM
println(SPΔCE)       // YcCpIG
SPΔCE = "istheplace" //null.SPΔCE ← istheplace
```

school.kt

Свойства и делегаты

```
public interface ReadOnlyProperty<in R, out T> {
   public operator fun getValue(thisRef: R, property: KProperty<*>): T
}

public interface ReadWriteProperty<in R, T> {
   public operator fun getValue(thisRef: R, property: KProperty<*>): T
   public operator fun setValue(thisRef: R, property: KProperty<*>, value: T)
}
```

Свойства и делегаты

```
public abstract class ObservableProperty<T>(initialValue: T)
    : ReadWriteProperty<Any?, T>
{
   protected open fun beforeChange(property: KProperty<*>, oldValue: T, newValue: T)

   protected open fun afterChange(property: KProperty<*>, oldValue: T, newValue: T)

   public override fun getValue(thisRef: Any?, property: KProperty<*>): T

   public override fun setValue(thisRef: Any?, property: KProperty<*>, value: T)
}
```

Delegates.observable

```kotlin
var observable: String by Delegates.observable("I ❤ Java") {
  property, old, new ->
    println("That's better!")
}

observable = "I ❤ Kotlin"  // That's better!
println(observable)         // I ❤ Kotlin
```

Delegates.vetoable

```kotlin
var vetoable: String by Delegates.vetoable("Kotlin one love!") {
  property, old, new ->
    false
}

vetoable = "I ❤ Java"
println(vetoable)       // Kotlin one love!
```

school.kt

Delegates.notNull

```kotlin
var notNull: String by Delegates.notNull()

// println(notNull)
notNull = "I ❤ Kotlin"
println(notNull)  // I ❤ Kotlin
```

school.kt

Свойства и делегаты

```
public interface Lazy<out T> {
   public val value: T
   public fun isInitialized(): Boolean
}

operator fun <T> Lazy<T>.getValue(thisRef: Any?, property: KProperty<*>): T
= value
```

school.kt

lazy

```kotlin
val lazy: String by lazy { "I do slides" }
val eager: String by lazyOf("Others do slides")

println(lazy)   // I do slides
println(eager)  // Others do slides
```

school.kt

kotlin-reflect

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

fun main(args: Array<String>) {
   val luke = Jedi("Luke Skywalker", 19)

   println(luke::class.allSuperclasses)
}
```

```
>Error:(8, 25) Kotlin: Unresolved reference: allSuperclasses
```

school.kt

kotlin-reflect

```
val luke = Jedi("Luke Skywalker", 19)

println(luke::class)  // class Jedi (Kotlin reflection is not available)
```

kotlin-reflect

```
dependencies {
  implementation(
    "org.jetbrains.kotlin:kotlin-reflect"
  )
}
```

```
<dependency>
  <groupId>org.jetbrains.kotlin</groupId>
  <artifactId>kotlin-reflect</artifactId>
  <version>${kotlin.version}</version>
</dependency>
```

KClass

```
val k = luke::class

k.simpleName       // Jedi
k.isAbstract       // false
k.isCompanion      // false
k.isData           // true
k.isFinal          // true
k.isInner          // false
k.isOpen           // false
k.isSealed         // false
k.memberProperties // [val Jedi.age: kotlin.Int, val Jedi.name: 
kotlin.String]
```

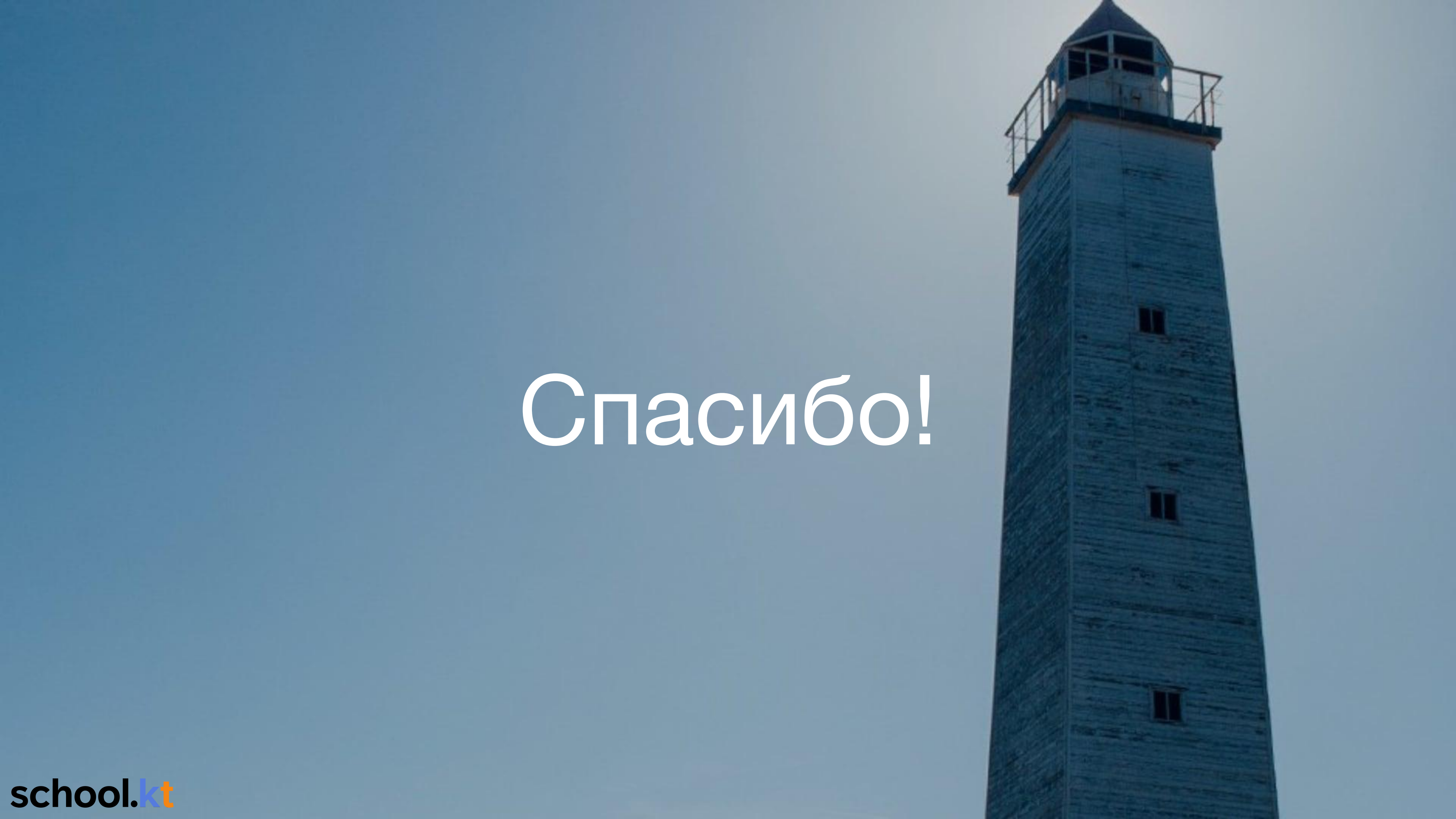
KProperty

```kotlin
val p = luke::age

p.get()       // 19
p.isLateinit  // false
p.isOpen      // false
p.isFinal     // true
```

school.kt

Стандартная библиотека Kotlin

- Не только JVM
- Активно разрабатывается
- Extensions, extensions, extensions
- Крайне обширна :)



Спасибо!

school.kt