#7: Java Interop

JUNO fitbit SPACE JET BRAINS

Сергей Крюков

Developer @ Banuba

siarhei.krukau@gmail.com

school.kt

SPACE

juno

fitbit

JET
BRAINS

Менторы

img by http://www.theadhdnerd.com

Kotlin ↔ Java

school.kt

ВСЁ ОЧЕНЬ ХОРОШО

school.kt

Java → Kotlin

Kotlin → Java

Java → Kotlin

Kotlin → Java

school.kt

Getters & setters

```java
public class Person {
   private String name;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }
}
```

```kotlin
person.name = "Jonh Doe"
println(person.name)
```

Getters & setters

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

   public Person(String name) {
       this.name = name;
   }

   public String getName() {
       return name;
   }
}
```

~~person.name = "Jonh Doe"~~

```
println(person.name)
```

Getters & setters

```
public class Person {
    private String name;

    public void setName(String name) {
        this.name = name;
    }
}
```

```
person.setName("John Doe")
```
~~println(person.name)~~

Getters & setters

```java
public class Person {
   private Boolean dead;

   public Boolean getDead() {
       return dead;
   }

   public void setDead(Boolean dead) {
       this.dead = dead;
   }
}
```

```kotlin
person.dead = true
println(person.dead)
```

school.kt

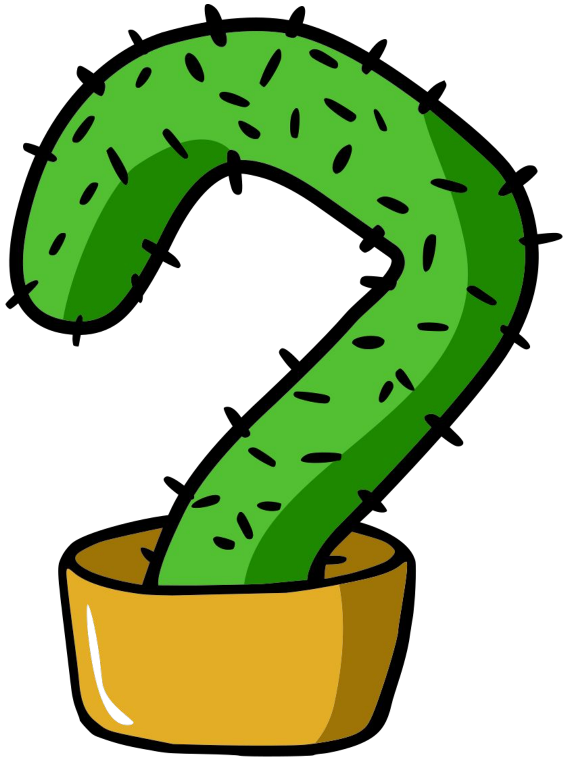
Getters & setters

```
public class Person {
   private boolean dead;

   public boolean isDead() {
       return dead;
   }

   public void setDead(boolean dead) {
       this.dead = dead;
   }
}
```

```
person.isDead = true
println(person.isDead)
```

school.kt

Getters & setters



school.kt

void → Unit

```java
public class Worker {
   public void doWork(){
      …
   }
}
```

```kotlin
val result = worker.doWork()
println(result)  // kotlin.Unit
```

school.kt

Keywords

```java
public class Keywords {
   public Keywords getObject() {
       return new Keywords();
   }

   public boolean when(
         Function<Keywords, Boolean> cb
   ) {
       return cb.apply(this);
   }
}
```

```kotlin
val instance = keywords.`object`
instance.`when` { it == instance }
```

Platform types / Null-safety

```java
public class Citizen {
   private final String citizenship;

   public Citizen(String citizenship) {
       this.citizenship = citizenship;
   }

   public String getCitizenship() {
       return citizenship;
   }
}
```

```kotlin
val belarusian = Citizen("BY")
val c: String = belarusian.citizenship

println(c) // BY
```

school.kt

Platform types / Null-safety

```java
public class Citizen {
   private final String citizenship;

   public Citizen(String citizenship) {
       this.citizenship = citizenship;
   }

   public String getCitizenship() {
       return citizenship;
   }
}
```

```kotlin
val diogenes = Citizen(null)
val c: String? = diogenes.citizenship

println(c) // null
```

school.kt

Platform types / Null-safety

```java
public class Citizen {
   private final String citizenship;

   public Citizen(String citizenship) {
       this.citizenship = citizenship;
   }

   public String getCitizenship() {
       return citizenship;
   }
}
```

```kotlin
val diogenes = Citizen(null)

// IllegalStateException:
// diogenes.citizenship must not be null
val c: String = diogenes.citizenship
```

school.kt

Platform types / Collections

```java
public class SchoolKt {
   public List<String> lections() {
       return new LinkedList() {{
           add("#0: Intro");
           …
           add("#6: Coroutines");
       }};
   }
}
```

```kotlin
val lections: MutableList<String> =
  schoolKt.lections()

lections.add("#7: Kotlin ↔ Java")
println(lections)
```

Platform types / Collections

```
public class SchoolKt {
   public List<String> lections() {
       return List.of(
           "#0: Intro",
           "#6: Coroutines",
           "#7: Kotlin ↔ Java"
       );
   }
}
```

```
val lections: List<String> =
  schoolKt.lections()

~~lections.add("#8: Ecosystem")~~
println(lections)
```

Platform types / Collections

```java
public class SchoolKt {
   public List<String> lections() {
        return List.of(
             "#0: Intro",
             "#6: Coroutines",
             "#7: Kotlin ↔ Java"
        );
   }
}
```

```kotlin
val lections: MutableList<String> =
  schoolKt.lections()

// Exception in thread "main"
// j.l.UnsupportedOperationException
lections.add("#8: Ecosystem")
```

Platform types / Arrays

```java
public class SchoolKt {
   public String[] lections() {
       return new String[]{
           "#0: Intro",
           "#6: Coroutines",
           "#7: Kotlin ↔ Java",
           null
       };
   }
}
```

```kotlin
val lections: Array<out String> =
  schoolKt.lections()

// #0: Intro
// #6: Coroutines
// #7: Kotlin ↔ Java
// null
lections.forEach(::println)
```

Platform types

```
val cz /*: String! */ = citizen.citizenship
val lc /*: (Mutable)List<String!>! */ = schoolKt.lections()
val la /*: Array<out String!>! */ = schoolKt.lections()
```

school.kt

Platform types

```
public class Citizen {
    @NotNull
    private final String citizenship;

    public Citizen(@NotNull String citizenship) {
        this.citizenship = citizenship;
    }

    @NotNull
    public String getCitizenship() {
        return citizenship;
    }
}
```

Platform types

```
public class SchoolKt {
    @NotNull
    public List<@NotNull String> lections() {
        return List.of(
                "#0: Intro",
                "#6: Coroutines",
                "#7: Kotlin ↔ Java"
        );
    }
}
```

Mapped types

`byte`	`kotlin.Byte`
`short`	`kotlin.Short`
`int`	`kotlin.Int`
`long`	`kotlin.Long`
`char`	`kotlin.Char`
`float`	`kotlin.Float`
`double`	`kotlin.Double`
`boolean`	`kotlin.Boolean`

Mapped types

Byte	kotlin.Byte!
Short	kotlin.Short!
Integer	kotlin.Int!
Long	kotlin.Long!
Char	kotlin.Char!
Float	kotlin.Float!
Double	kotlin.Double!
Boolean	kotlin.Boolean!

Mapped types

```
java.lang.Long.toHexString(42)
```

Mapped types

`java.lang.Object`	`kotlin.Any!`
`java.lang.String`	`kotlin.String!`
`java.lang.Throwable`	`kotlin.Throwable!`
`java.lang.Enum`	`kotlin.Enum!`
`java.lang.Deprecated`	`kotlin.Deprecated!`
…	…

Mapped types

`int[]`	`kotlin.IntArray!`
`String[]`	`kotlin.Array<(out) String>!`

Generics

```
class Generics {
    List<? extends Number> outNumber;
    List<? super Integer> inInt;
}
```

```
g.outNumber /*: MutableList<out Number!>! */
g.inInt /*: MutableList<in Int!>! */

g.outNumber = mutableListOf<Number>()
g.outNumber = mutableListOf<Int>()
g.outNumber = mutableListOf<Double>()

g.inInt = mutableListOf<Int>()
g.inInt = mutableListOf<Number>()
g.inInt = mutableListOf<Any>()
```

Generics

```
class Generics {
   List any;
}
```

```
g.any /*: (Mutable)List<*>! */

g.any = listOf<String>()
g.any = listOf<Int>()
```

Arrays

```
class ArraysK {
   fun doStuff(n: Array<Number>) {
      …
   }
}

val arraysJ = ArraysJ()
val arraysK = ArraysK()

val numbers = arrayOf(1, 2, 3);

arraysJ.doStuff(numbers)
```
~~arraysK.doStuff(numbers)~~

```
public class ArraysJ {
   public void doStuff(Number[] n) {
      …
   }
}
```

school.kt

Arrays

```java
class Arrays {
   public void doStuff(int[] ints) {
      …
   }
}
```

```kotlin
val ints: IntArray = intArrayOf(1, 2, 3)

arrays.doStuff(ints)
```

school.kt

Varargs

```
class Varargs {
   public void doStuff(int... ints) {
      …
   }
}
```

```
val ints: IntArray = intArrayOf(1, 2, 3)

varargs.doStuff(*ints)
varargs.doStuff(1, 2, 3)
```

school.kt

Operators

```
class Width {
   public final int value;

   Width(int value) {
       this.value = value;
   }

   public Rectangle times(final Height height) {
       return new Rectangle(this, height);
   }
}
```

Operators

```
class Height {
   public final int value;

   Height(int value) {
       this.value = value;
   }

   public Rectangle times(final Width width) {
       return new Rectangle(width, this);
   }
}
```

Operators

```
class Rectangle {
   public final Width width;
   public final Height height;

   Rectangle(Width width, Height height) {
       this.width = width;
       this.height = height;
   }

   public int getArea() {
       return width.value * height.value;
   }
}
```

Operators

```
val width = Width(3)
val height = Height(2)

println((width * height).area)
```

school.kt

Exceptions

```java
public class Stone extends Exception {}

public class David {
   public void sling() throws Stone {
       // Kill the Goliath
   }
}
```

```kotlin
val david = David()
val goliath = Goliath()


david.sling()
goliath.smile()
```

school.kt

Object methods

```
val bus = "Bus"

(bus as Object).wait()
```

Object methods

```kotlin
val first = 1

first::class.java
```

school.kt

SAM conversions

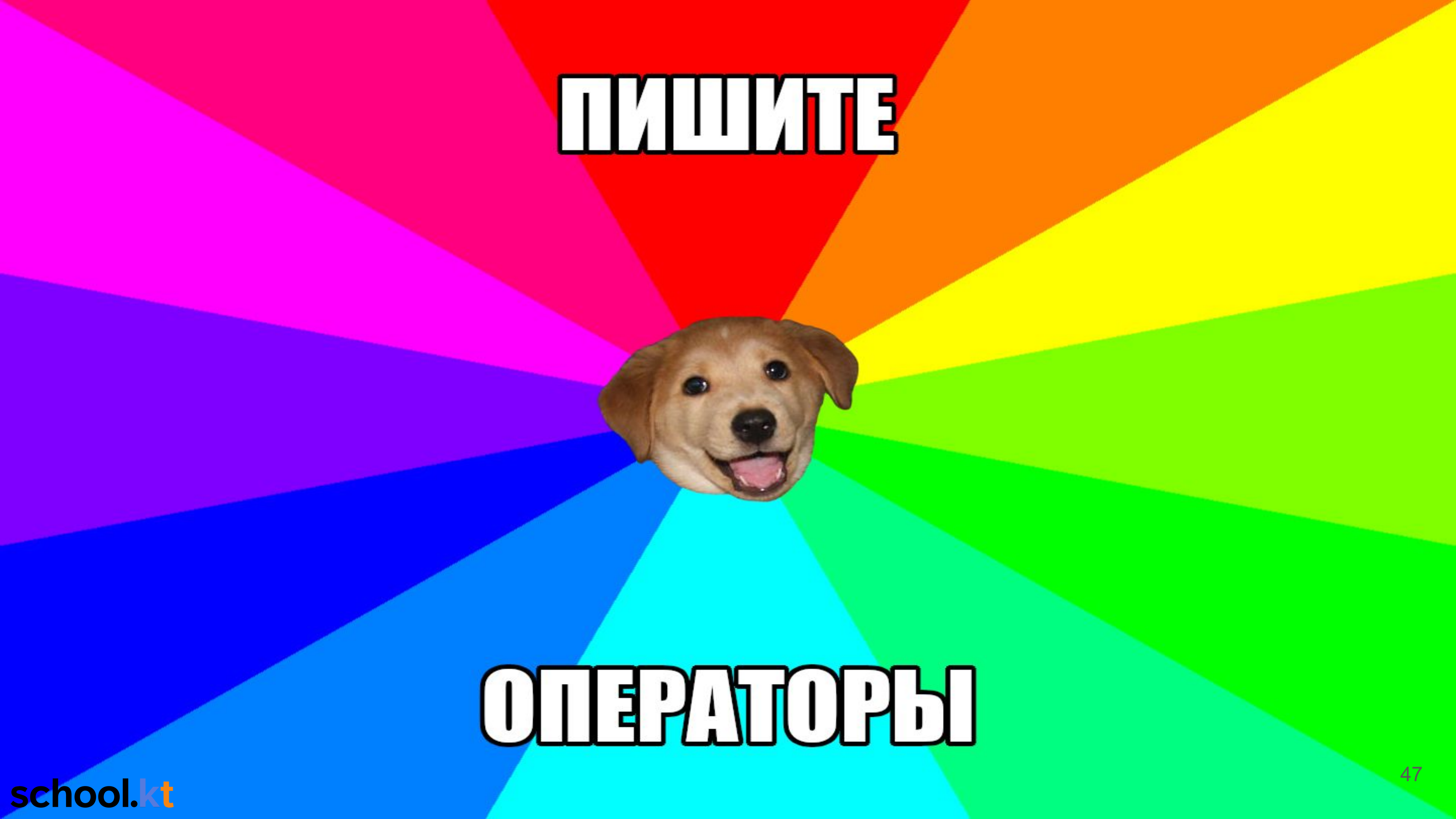
```kotlin
val executor = Executors.newSingleThreadExecutor()

executor.execute {
    println("Running in the pool")
}
```

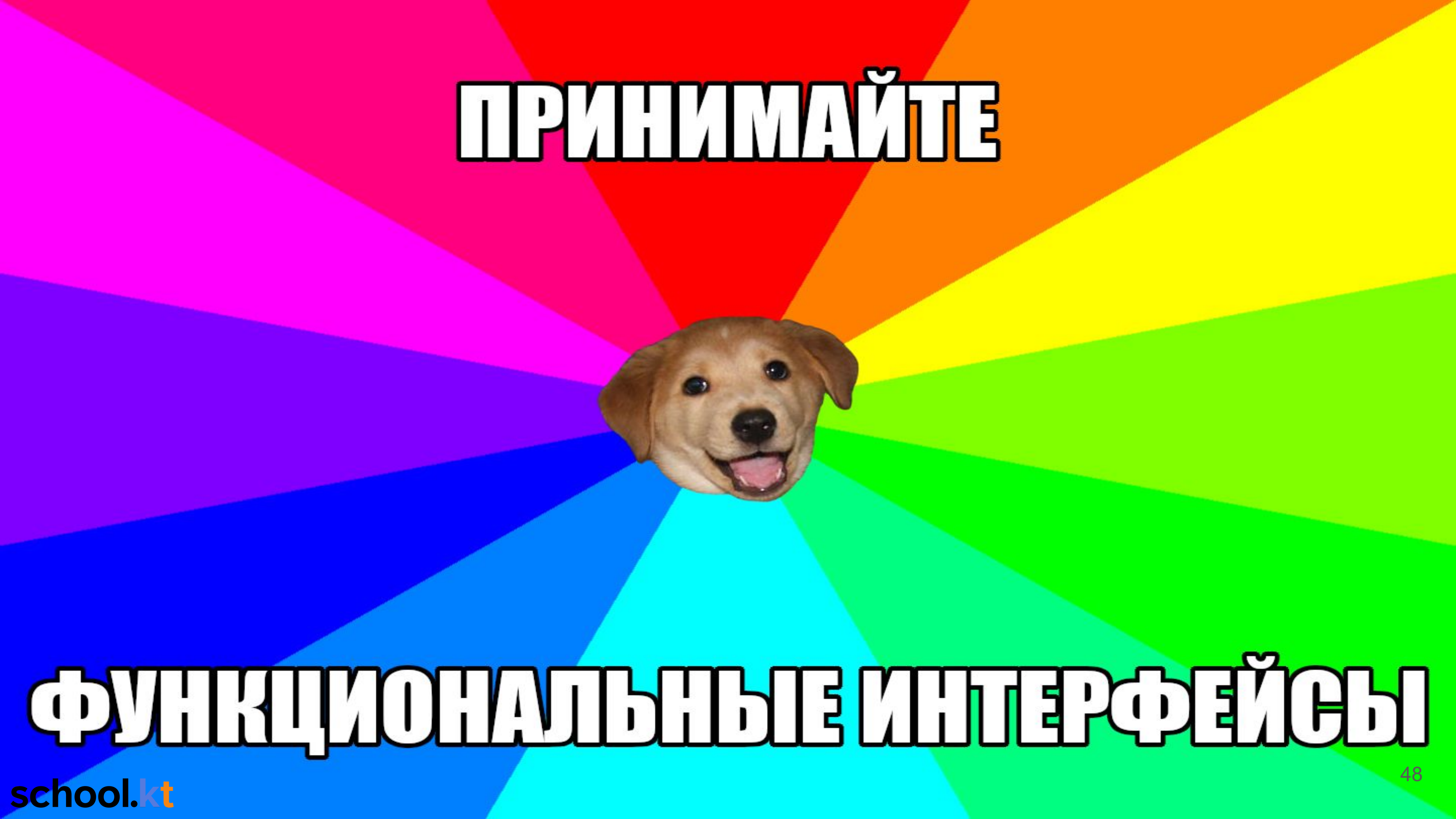
school.kt

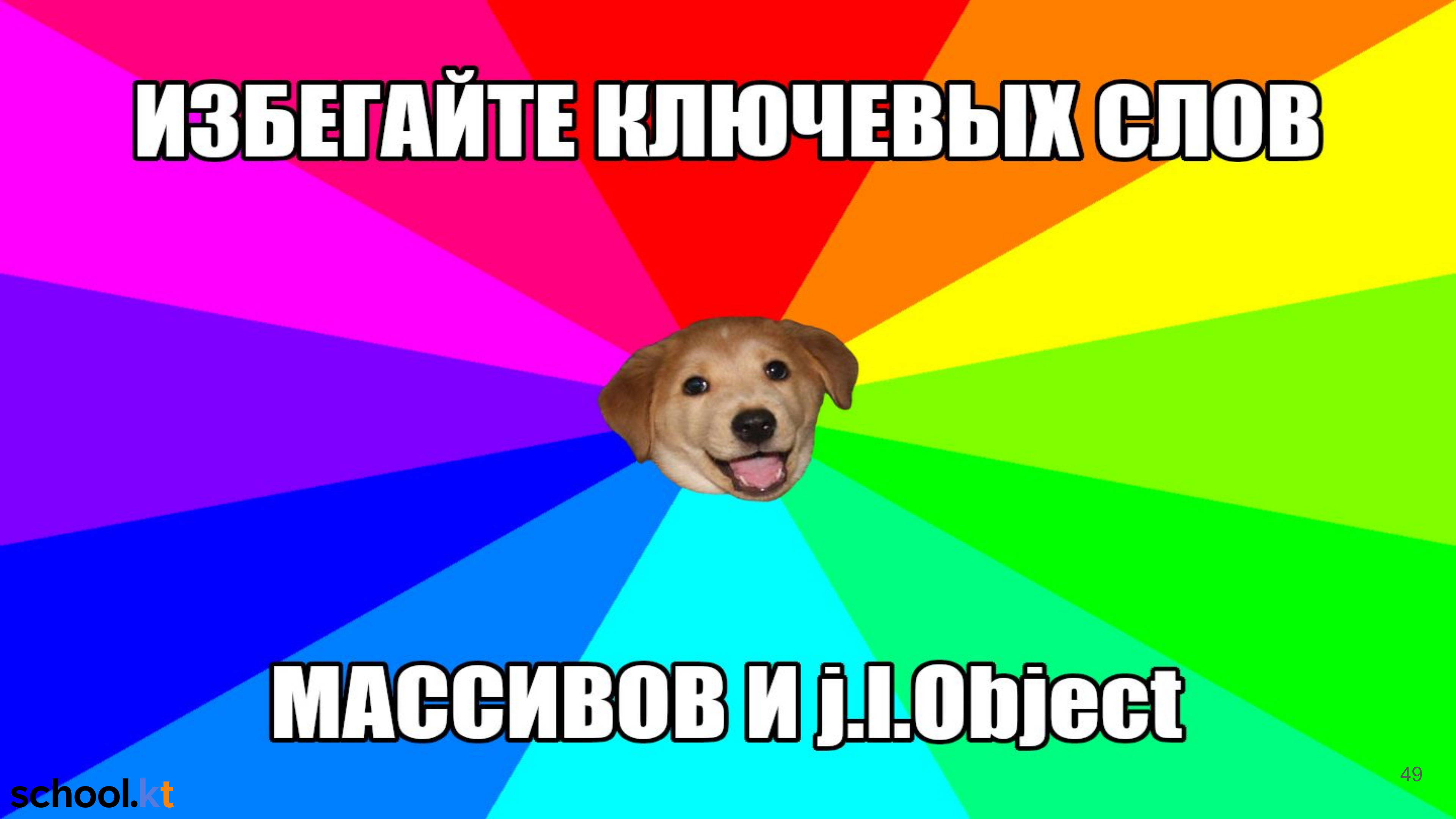
SAM conversions

```kotlin
val executor = Executors.newSingleThreadExecutor()
val runnable = Runnable { println("This runs in a runnable") }

executor.execute(runnable)
runnable.run()
```

ПИШИТЕ АКСЕССОРЫ
СТАВЬТЕ АННОТАЦИИ


ПИШИТЕ
ОПЕРАТОРЫ


ПРИНИМАЙТЕ
ФУНКЦИОНАЛЬНЫЕ ИНТЕРФЕЙСЫ


ИЗБЕГАЙТЕ КЛЮЧЕВЫХ СЛОВ
МАССИВОВ И j.l.Object

Java → Kotlin

Kotlin → Java

school.kt

Properties

```kotlin
data class Person(
   val name: String,
   var age: Int,
   val hasSister: Boolean,
   val isJedi: Boolean
)
```

```java
final Person person = new Person(
   "Luke", 18, true, true
);

System.out.println(person.getName());

person.setAge(19);
System.out.println(person.getAge());

System.out.println(person.getHasSister());

System.out.println(person.isJedi());
```

school.kt

Properties

```kotlin
data class Person(
   val name: String,
   var age: Int,
   val hasSister: Boolean,
   val isJedi: Boolean
)
```

```java
public final class Person {
  @NotNull
  private final String name;
  private int age;
  private final boolean hasSister;
  private final boolean isJedi;

  @NotNull
  public final String getName() { … }
  public final int getAge() { … }
  public final void setAge(int var1) { … }
  public final boolean getHasSister() { … }
  public final boolean isJedi() { … }

  …
}
```

Fields

```kotlin
data class Person(
   @JvmField val name: String,
   @JvmField var age: Int
)
```

```java
System.out.println(person.name);
System.out.println(person.age);
```

school.kt

Fields

```kotlin
data class Person(
   @JvmField val name: String,
   @JvmField var age: Int
)
```

```java
public final class Person {
  @NotNull
  public final String name;
  @JvmField
  public int age;

  ...
}
```

Name overrides

```kotlin
class Person(name: String) {
    var name: String = name
        @JvmName("name")
        get() = field

        @JvmName("name")
        set(value) {
            field = value
        }
}
```

```java
person.name("Yegor '256' Bugaenko");
System.out.println(person.name());
```

school.kt

Name overrides

```kotlin
@JvmName("biggest")
fun List<Int>.greatest() = this.max()

@JvmName("longest")
fun List<String>.greatest() =
this.max().length
```

```java
int biggest(List $receiver)

int longest(List $receiver)
```

school.kt

Package-level declarations

```kotlin
/* school.kt */

package bkug

class KotlinJavaInterop

fun isGreat() = true
```

```java
new bkug.KotlinJavaInterop();

bkug.SchoolKt.isGreat();
```

school.kt

Package-level declarations

```kotlin
/* school.kt */

@file:JvmName("KotlinCourses")

package bkug

class KotlinJavaInterop

fun isGreat() = true
```

```java
new bkug.KotlinJavaInterop();

bkug.KotlinCourses.isGreat();
```

Combining files

```
/* school.kt */
@file:JvmName("BKUG")
@file:JvmMultifileClass

fun knowledge() {}

/* bkug.kt */
@file:JvmName("BKUG")
@file:JvmMultifileClass

fun meetups() {}
```

```
public final class BKUG {
  public static final void knowledge() {  }
  public static final void meetups() {  }
}
```

Type aliases

```kotlin
typealias Width = Int

typealias Height = java.lang.Integer

fun area(width: Width, height: Height)
```

```java
public static final void area(
  int width,
  @NotNull Integer height
) {
  Intrinsics
    .checkParameterIsNotNull(
      height, "height"
    );

  return width * height;
}
```

school.kt

Inline classes

```kotlin
inline class Area(val value: Int)

inline class Width(val value: Int) {
   operator fun times(height: Height)
    = Area(this.value * height.value)
}

inline class Height(val value: Int)

fun area(width: Width, height: Height)
= width * height
```

```java
final Width width = new
Width.constructor-impl(2);

width.times-vX11Ur0(
  Height.constructor-impl(3)
);
```

school.kt

Inline functions

```kotlin
inline fun area(
  width: Int, height: Int
) = width * height
```

```java
System.out.println(area(2, 3));
```

school.kt

Static fields

```kotlin
object Earth {
    val sign = "♁"
}
```

```java
public final class Earth {
  private static final String sign =
"♁";
  public static final Earth INSTANCE;

  public final String getSign() {
     return sign;
  }
}
```

Static fields

```kotlin
object Earth {
    const val sign = "♁"
}
```

```java
public final class Earth {
  public static final String sign = "♁";
  public static final Earth INSTANCE;
}
```

Static fields

```kotlin
object Earth {
    @JvmField
    val sign = "♁"
}
```

```java
public final class Earth {
  public static final String sign = "♁";
  public static final Earth INSTANCE;
}
```

Static fields

```kotlin
object Earth {
   lateinit var exterminated: Date
}
```

```java
public final class Earth {
  public static Date exterminated;
  public static final Earth INSTANCE;

  ...
}
```

Static fields

```kotlin
class SolarSystem {
   companion object {
       const val VENUS = "♀"
       @JvmField
       val EARTH = "♁"
   }
}
```

```java
public final class SolarSystem {
  public static final String VENUS = "♀";
  public static final String EARTH =
"♁";

  public static final
SolarSystem.Companion Companion = …
}
```

school.kt

Static methods

```kotlin
object TV {
   @JvmStatic
   fun turnOn() {}
}
```

```java
public final class TV {
  public static final TV INSTANCE;

  public static final void turnOn() {}
}
```

school.kt

Static methods

```kotlin
class TV {
   companion object {
       @JvmStatic
       fun turnOn() {}
   }
}
```

```java
public final class TV {
  public static final TV.Companion
Companion = ...

  public static final void turnOn() {
     Companion.turnOn();
  }

  public static final class Companion {
     public final void turnOn() {}
  }
}
```

school.kt

Visibilities

```kotlin
class Quote {
    private fun tobacco() {}
    public fun friendship() {}
}
```

```java
public final class Quote {
  private final void tobacco() {}

  public final void friendship() {}
}
```

school.kt

Visibilities

```
/* Top-level class */
private class Tobacco
```

```
final class Tobacco {}
```

school.kt

Visibilities

```kotlin
class Young {
   protected val honor = true
}
```

```java
public final class Young {
  private final boolean honor = true;

  protected final boolean getHonor() {
     return this.honor;
  }
}
```

school.kt

Visibilities

```kotlin
class Belarus {
   internal val tourism = false
}
```

```java
public final class Belarus {
  private final boolean tourism;

  public final boolean
getTourism$Kotlin___Java_main() {
     return this.tourism;
  }
}
```

KClass

```kotlin
fun print(k: KClass<*>) {
   println(k.simpleName)
}
```

```java
print(
  kotlin.jvm.JvmClassMappingKt
    .getKotlinClass(Integer.class)
);
```

school.kt

Overloads Generation

```kotlin
data class MixedNumeral(
   val whole: Int = 1,
   val numerator: Int= 0,
   val denominator: Int= 100
)
```

```java
public MixedNumeral(
  int whole,
  int numerator,
  int denominator
) {
  this.whole = whole;
  this.numerator = numerator;
  this.denominator = denominator;
}

public MixedNumeral() {
  this(1, 0, 1);
}
```

Overloads Generation

```kotlin
data class MixedNumeral(
   val whole: Int,
   val numerator: Int= 0,
   val denominator: Int= 100
)
```

```java
public MixedNumeral(
  int whole,
  int numerator,
  int denominator
) {
  this.whole = whole;
  this.numerator = numerator;
  this.denominator = denominator;
}
```

Overloads Generation

```kotlin
data class MixedNumeral
@JvmOverloads constructor(
   val whole: Int,
   val numerator: Int = 0,
   val denominator: Int = 100
)
```

```java
public MixedNumeral(int w, int n, int d)
public MixedNumeral(int w, int n)
public MixedNumeral(int w)
```

school.kt

Checked Exceptions

```
class David {
   @Throws(Stone::class)
   fun sling() {
       // Kill the Goliath
   }
}
```

```
try {
   david.sling();
} catch (Stone stone) {
   goliath.die();
}
```

school.kt

Null-safety

```kotlin
fun print(arg: Any) {
   println(arg)
}

private fun printPrivate(arg: Any) {
   println(arg)
}
```

```java
public void print(@NotNull Object arg) {
  Intrinsics
    .checkParameterIsNotNull(
      arg, "arg"
    );
  System.out.println(arg);
}

private void printPrivate(Object arg) {
  System.out.println(arg);
}
```

school.kt

Reified generics

```kotlin
inline fun <reified T> foo() {}
```

ПИШИТЕ
НА KOTLIN

ВСЁ ОЧЕНЬ ХОРОШО

Homework

$\frac{1}{2} + \frac{2}{3}$

$\frac{4}{5} \times \frac{7}{8}$

$\frac{1}{2} < \frac{5}{6}$

$\frac{3}{8} / 0$

…

Спасибо!

school.kt

https://kotlinlang.org/docs/reference/java-interop.html

https://kotlinlang.org/docs/reference/java-to-kotlin-interop.html

https://developer.android.com/kotlin/interop