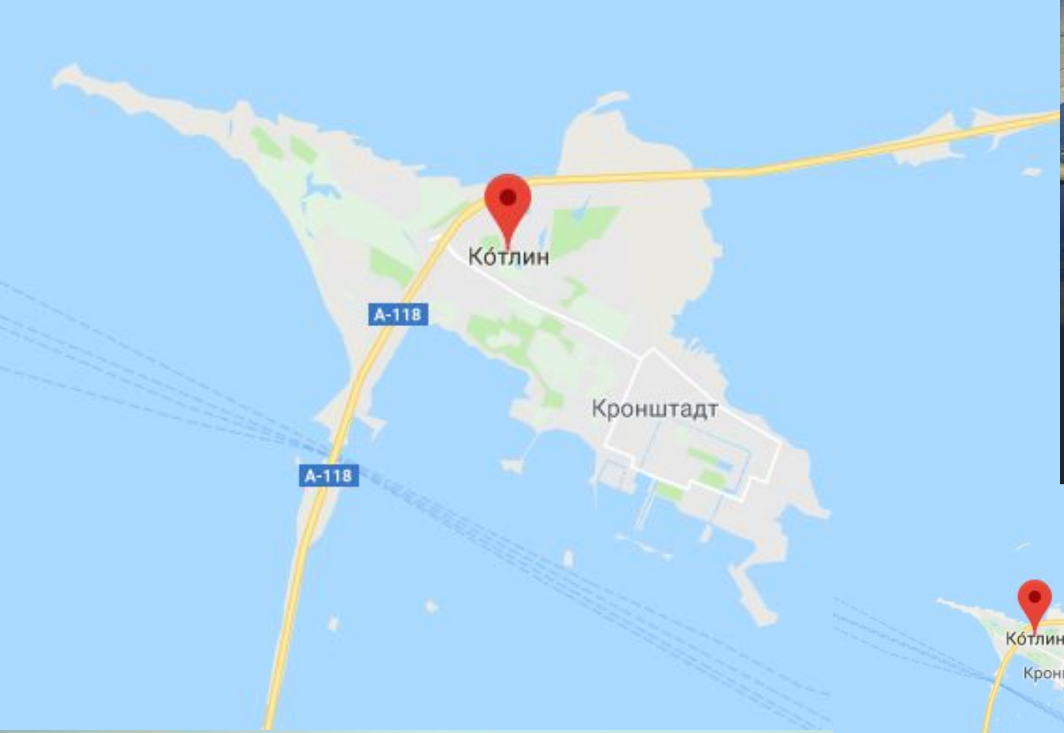
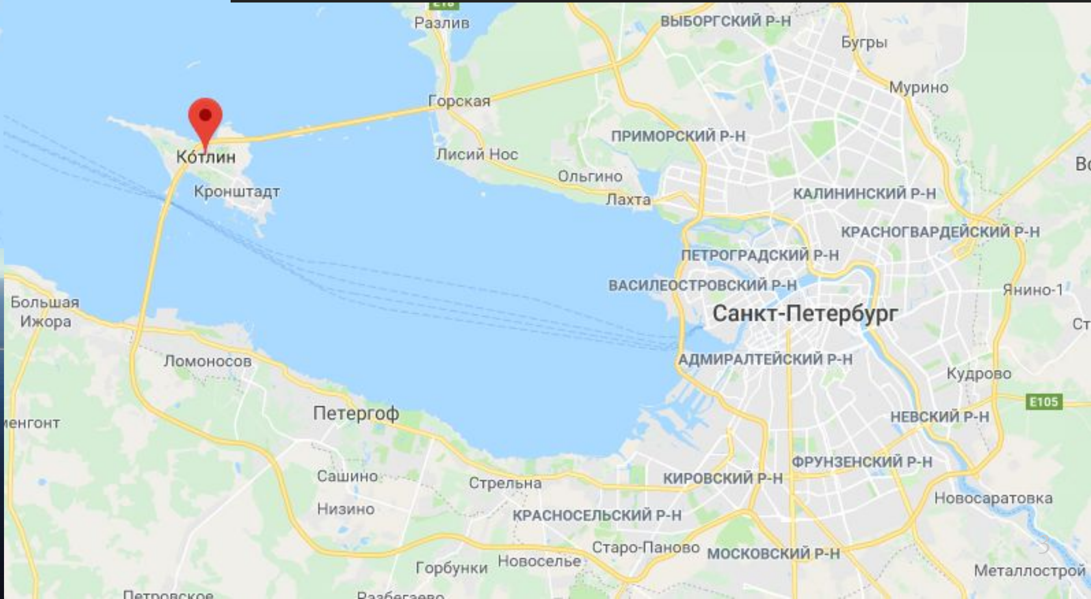
Kotlin

История

- JetBrains
- Язык разрабатывается с 2010 года
- 15 февраля 2016 года - релиз
- Май 2017 года - Kotlin официальный инструмент разработки для ОС Android
- Ноябрь 2017 года - выход Android Studio 3.0 с доступным по умолчанию Kotlin-ом
- Текущая версия 1.2.31
- Май 2018 года - основной язык разработки для ОС Android???


Ко́тлин
А-118
Кронштадт
А-118


Разлив
ВЫБОРГСКИЙ Р-Н
Бугры
Мурино
Горская
Ко́тлин
ПРИМОРСКИЙ Р-Н
Лисий Нос
Ольгино
Кронштадт
Лахта
КАЛИНИНСКИЙ Р-Н
КРАСНОГВАРДЕЙСКИЙ Р-Н
ПЕТРОГРАДСКИЙ Р-Н
ВАСИЛЕОСТРОВСКИЙ Р-Н
Янино-1
Большая Ижора
Санкт-Петербург
АДМИРАЛТЕЙСКИЙ Р-Н
Ломоносов
Кудрово
E105
Петергоф
НЕВСКИЙ Р-Н
енгонт
ФРУНЗЕНСКИЙ Р-Н
Сашино
КИРОВСКИЙ Р-Н
Стрельна
Новосаратовка
Низино
КРАСНОСЕЛЬСКИЙ Р-Н
Старо-Паново
МОСКОВСКИЙ Р-Н
Горбунки
Новоселье
Металлострой
Петровское
Разбегаево

Компилируется в

- JVM
- JavaScript
- Machine code
    - Windows (x86_64 only at the moment)
    - Linux (x86_64, arm32, MIPS, MIPS little endian)
    - MacOS (x86_64)
    - iOS (arm64 only)
    - Android (arm32 and arm64)
    - WebAssembly (wasm32 only)

Why Kotlin?

Concise

Drastically reduce the amount of boilerplate code.

See example

Safe

Avoid entire classes of errors such as null pointer exceptions.

See example

Interoperable

Leverage existing libraries for the JVM, Android, and the browser.

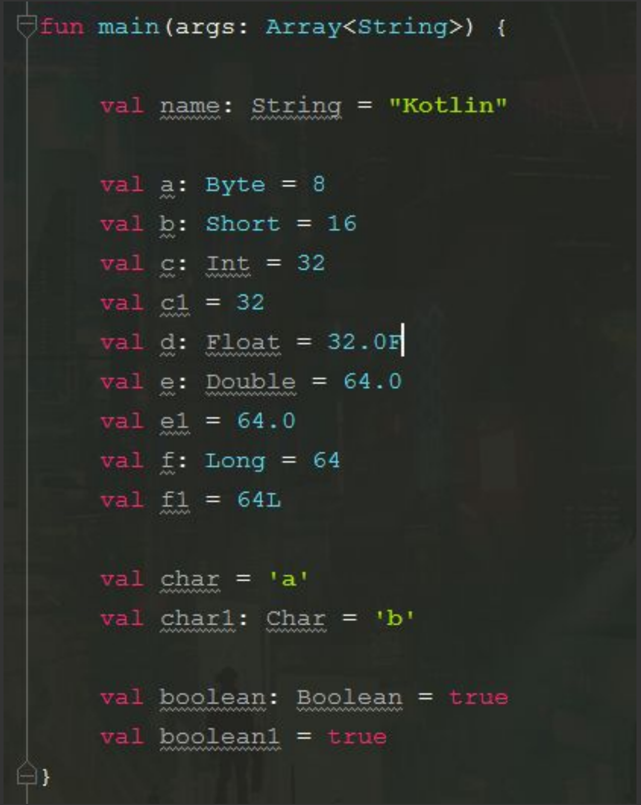
See example

Tool-friendly

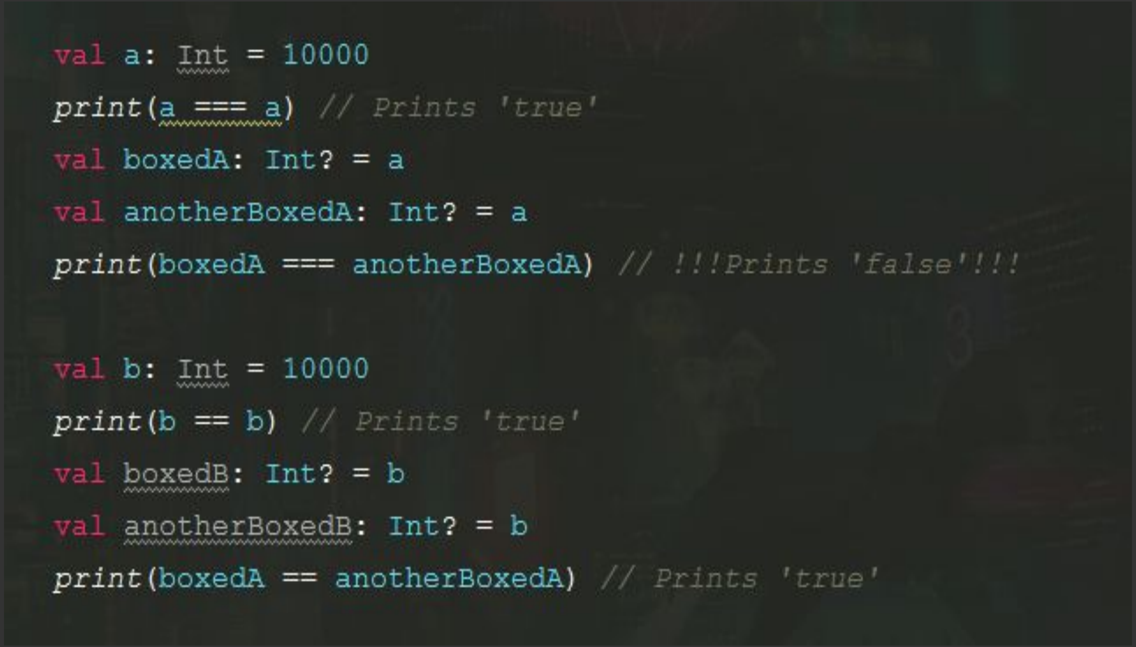
Choose any Java IDE or build from the command line.

See example

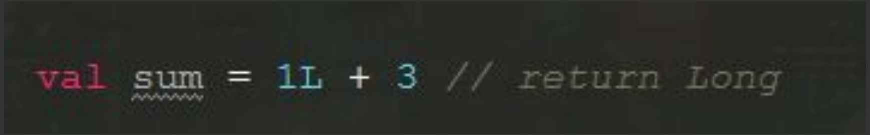
Базовые типы



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

    val name: String = "Kotlin"

    val a: Byte = 8
    val b: Short = 16
    val c: Int = 32
    val c1 = 32
    val d: Float = 32.0F
    val e: Double = 64.0
    val e1 = 64.0
    val f: Long = 64
    val f1 = 64L

    val char = 'a'
    val char1: Char = 'b'

    val boolean: Boolean = true
    val boolean1 = true
}
```

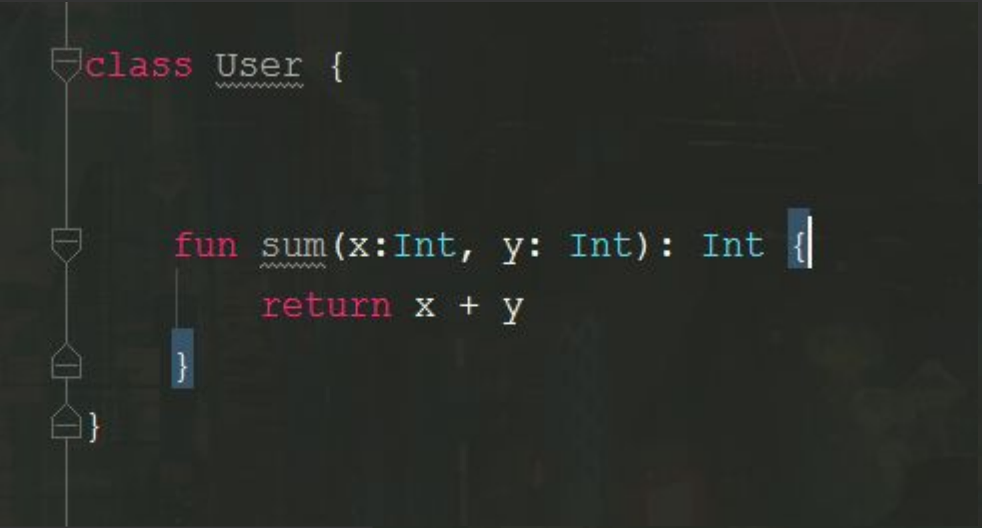


```
val a: Int = 10000
print(a === a) // Prints 'true'
val boxedA: Int? = a
val anotherBoxedA: Int? = a
print(boxedA === anotherBoxedA) // !!!Prints 'false'!!!

val b: Int = 10000
print(b == b) // Prints 'true'
val boxedB: Int? = b
val anotherBoxedB: Int? = b
print(boxedA == anotherBoxedA) // Prints 'true'
```

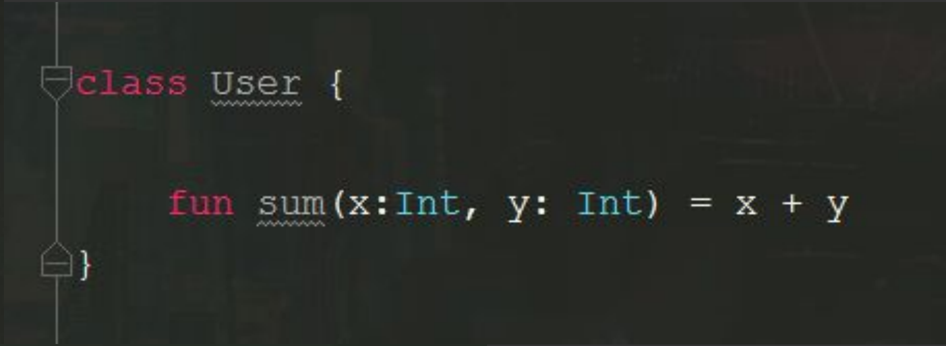


```
val sum = 1L + 3 // return Long
```

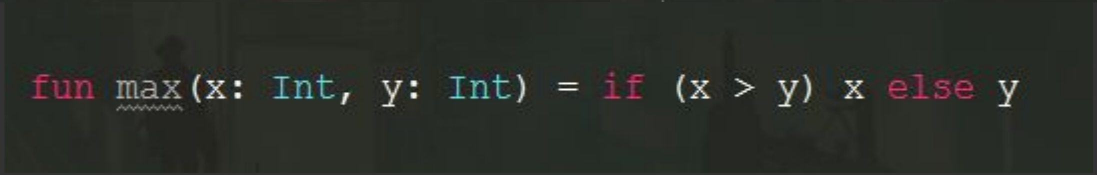
Функция



```
class User {


    fun sum(x:Int, y: Int): Int {
        return x + y
    }
}
```

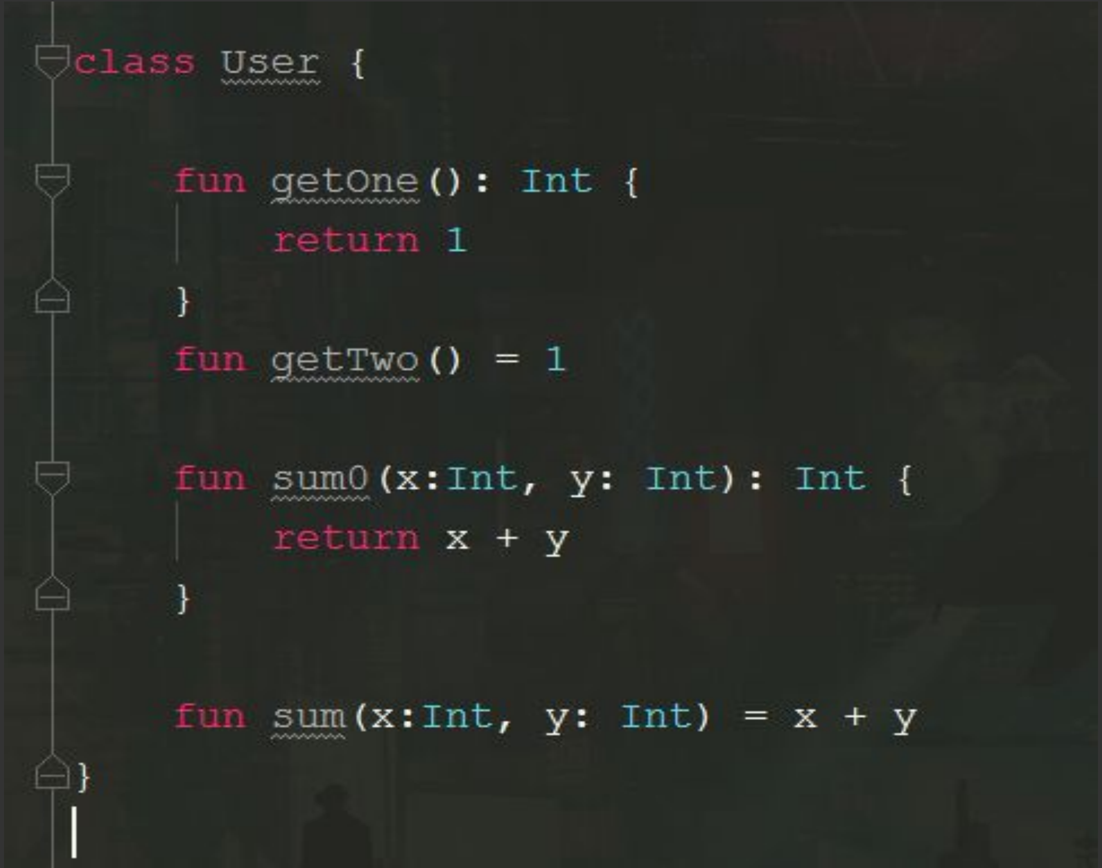


```
class User {

    fun sum(x:Int, y: Int) = x + y

}
```



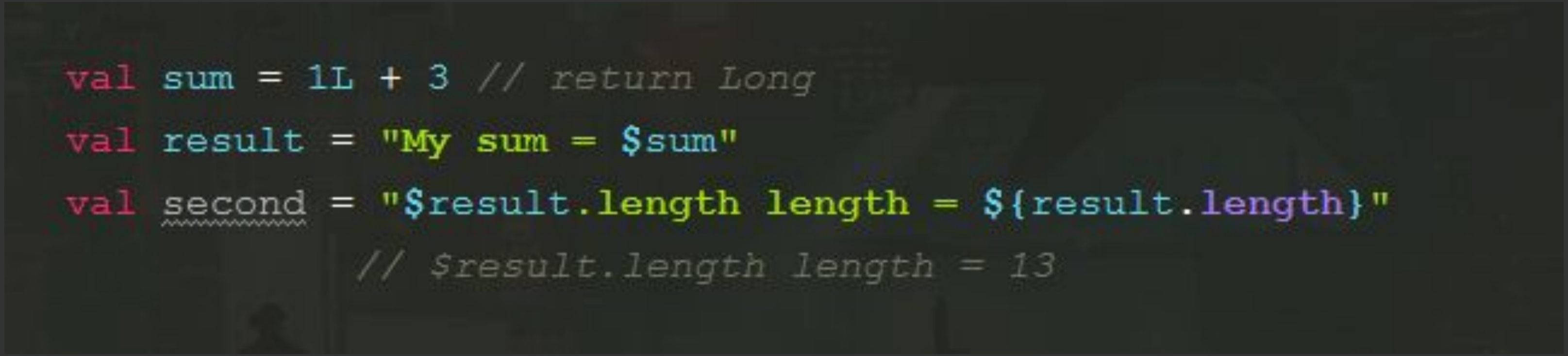
```
fun max(x: Int, y: Int) = if (x > y) x else y
```

Функция



```
class User {

    fun getOne(): Int {
        return 1
    }
    fun getTwo() = 1

    fun sum0(x:Int, y: Int): Int {
        return x + y
    }


    fun sum(x:Int, y: Int) = x + y
}
```

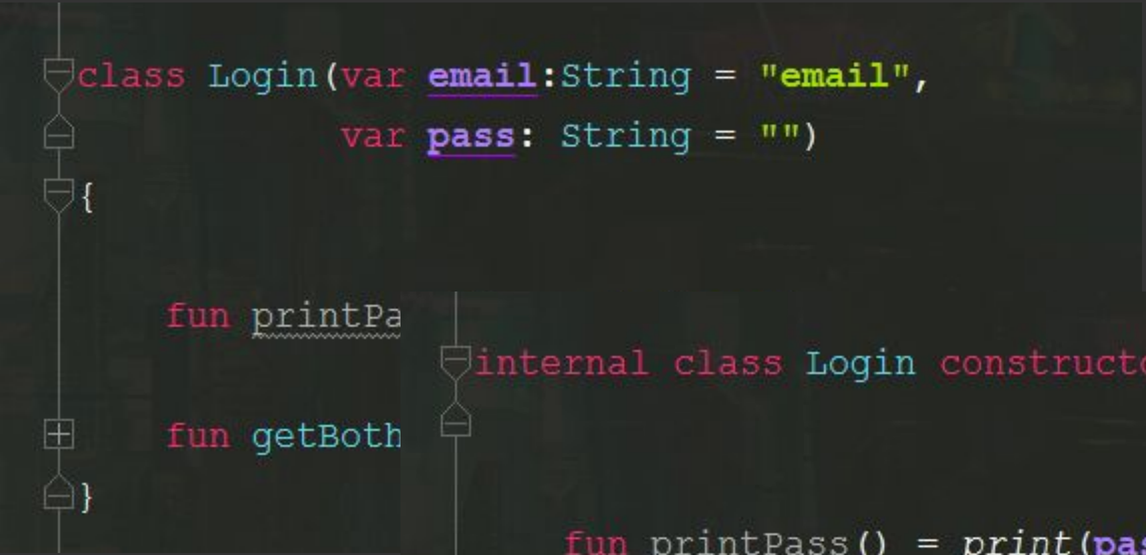
Строковые шаблоны


```
val sum = 1L + 3 // return Long
val result = "My sum = $sum"
val second = "$result.length length = ${result.length}"
            // $result.length length = 13
```

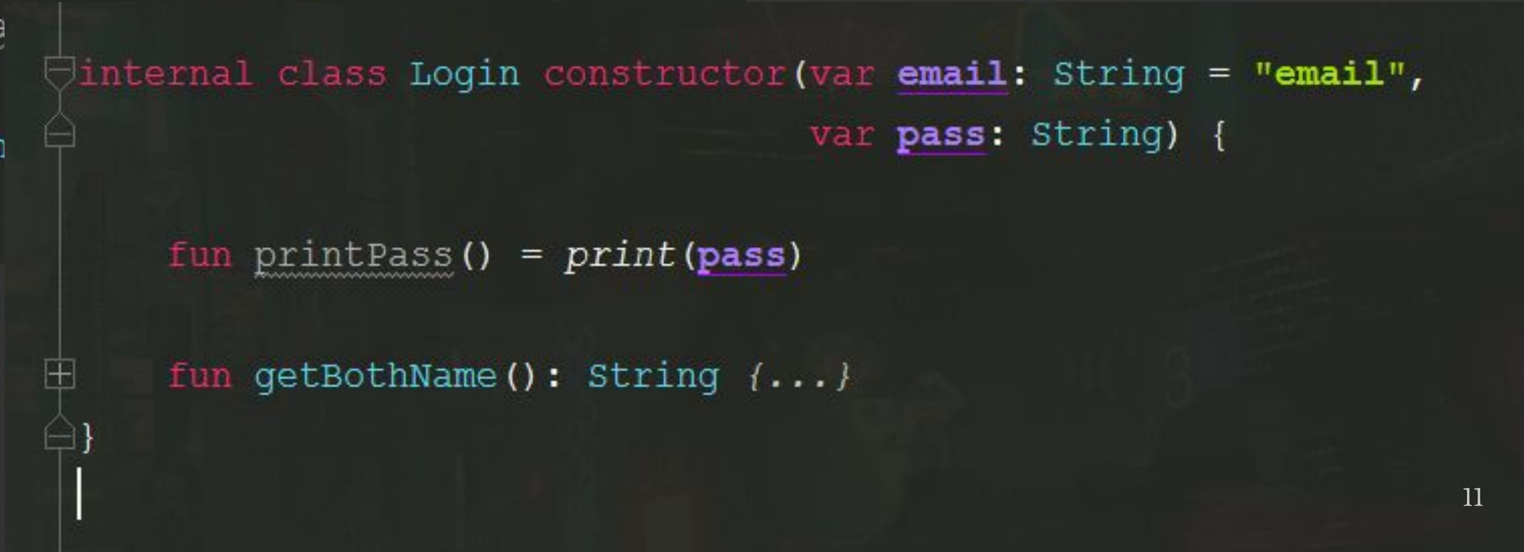
Модификаторы доступа

1. **public** - по умолчанию. Не пишется в явном виде
2. **private** - видимость внутри данного класса
   в Kotlin внешний класс не видит **private** члены своих вложенных классов.
1. **protected** - видимость для наследников
   Если вы переопределите protected член и явно не укажете его видимость, переопределённый элемент также будет иметь модификатор доступа **protected**.
1. **internal** - видимость в области модуля

Класс. Constructor

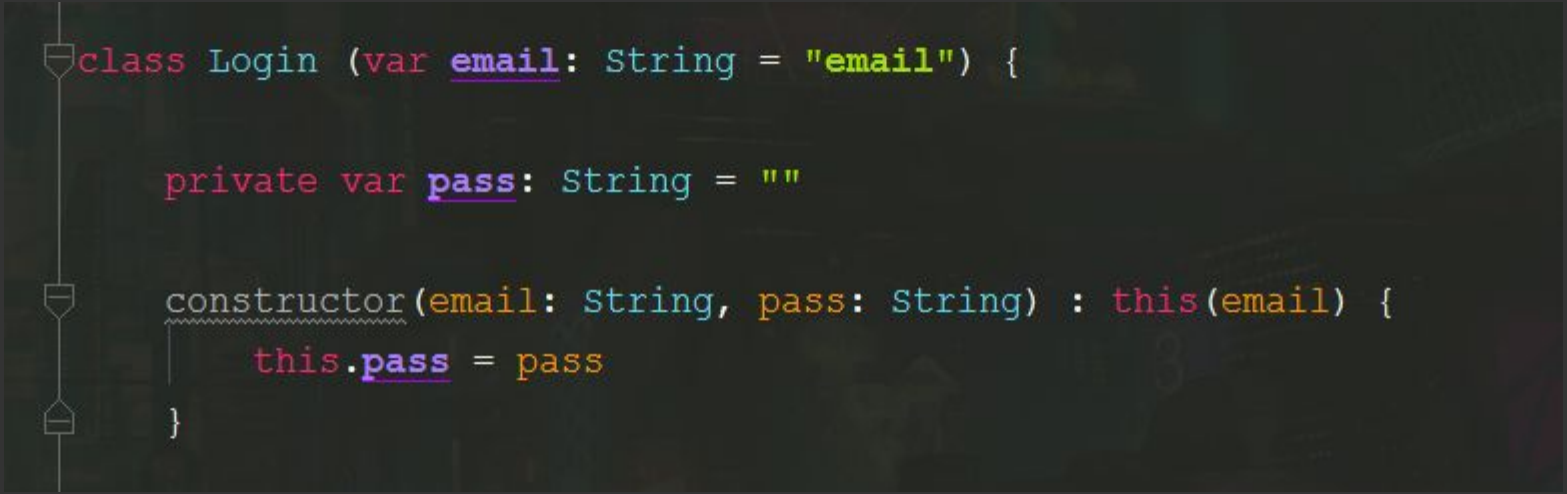


```
class Login(var email:String = "email",
            var pass: String = "")
{

    fun printPa

    fun getBoth
}
```



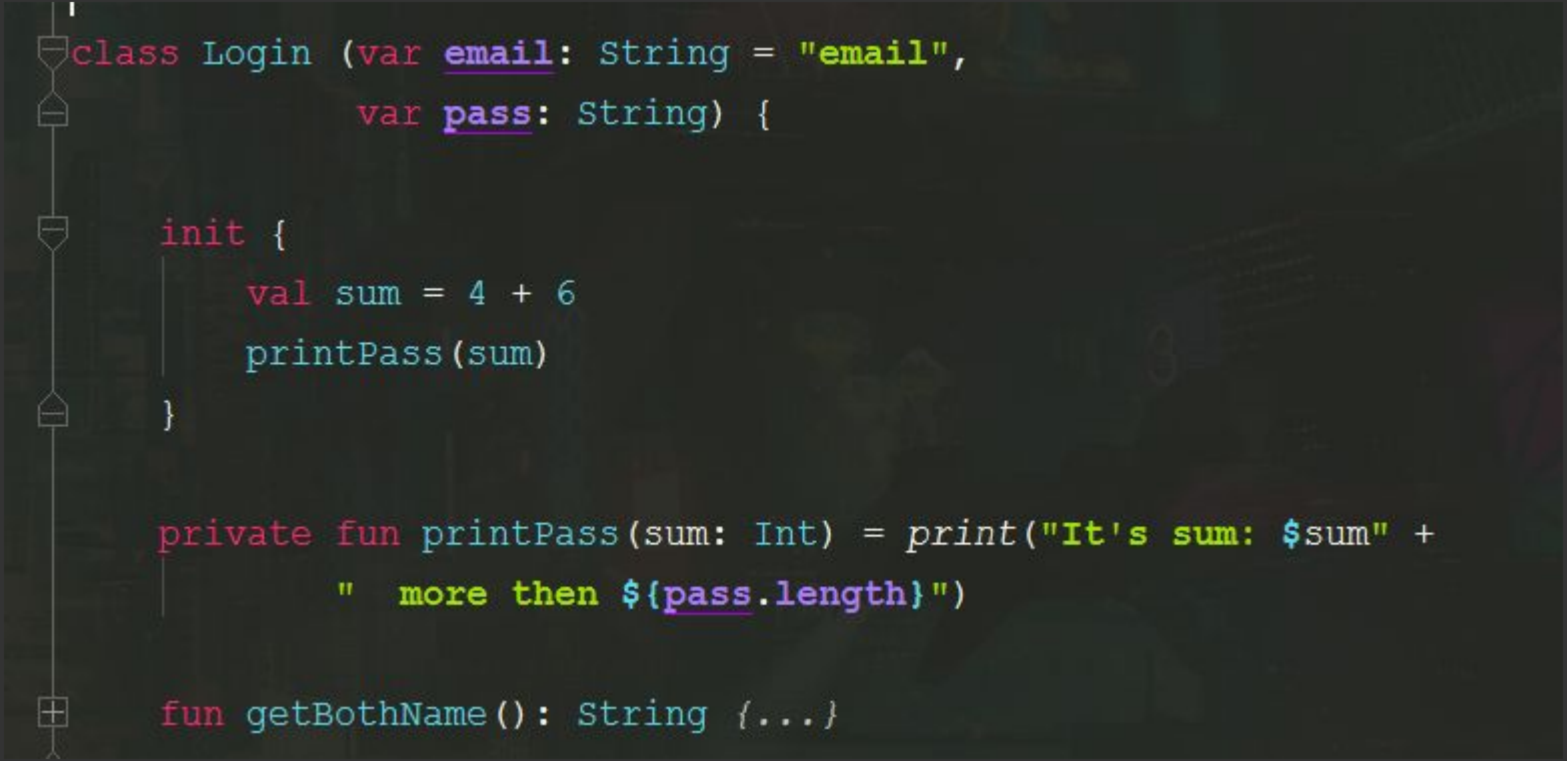
```
internal class Login constructor(var email: String = "email",
                                 var pass: String) {

    fun printPass() = print(pass)

    fun getBothName(): String {...}
}
```

Класс. Constructor. Вторичный



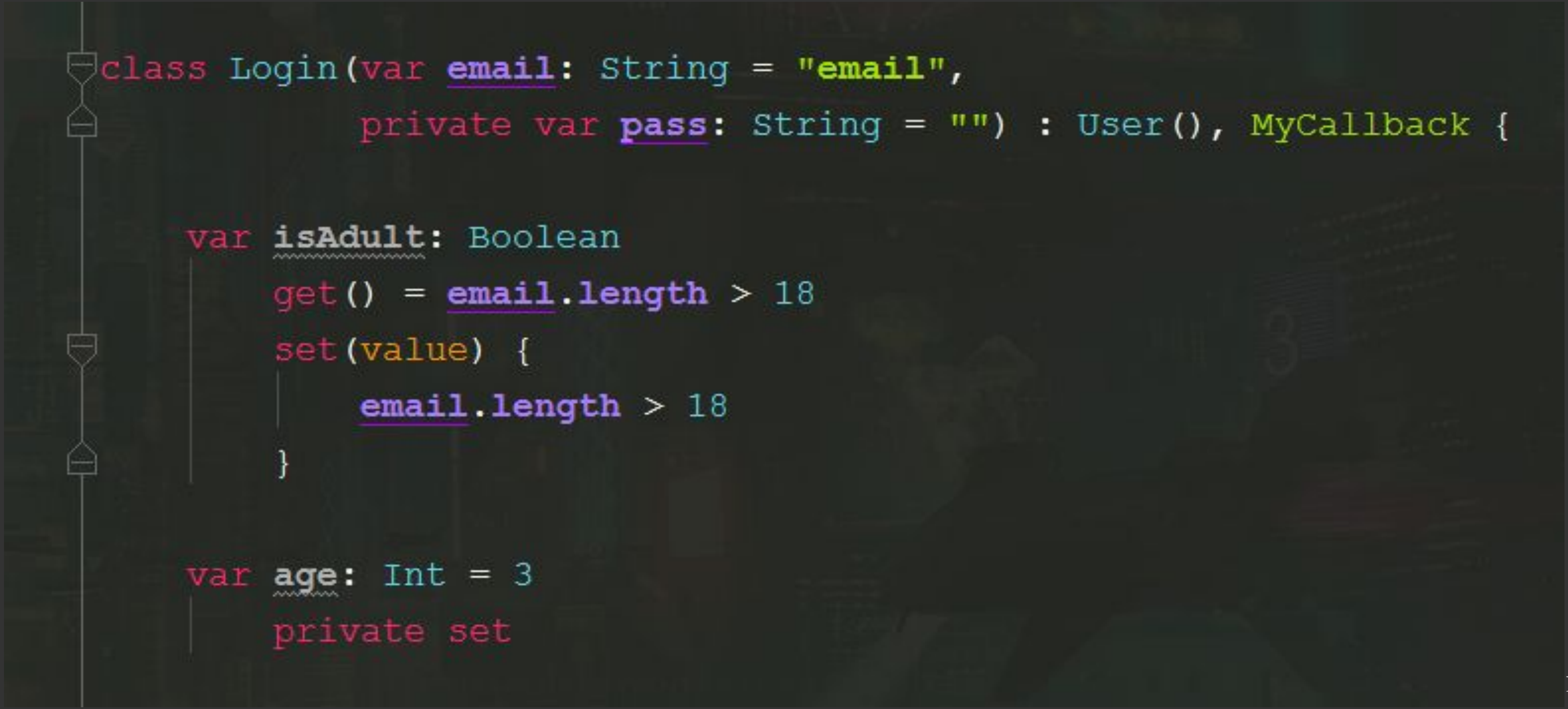
```
class Login (var email: String = "email") {

    private var pass: String = ""

    constructor(email: String, pass: String) : this(email) {
        this.pass = pass
    }
```

Класс. init



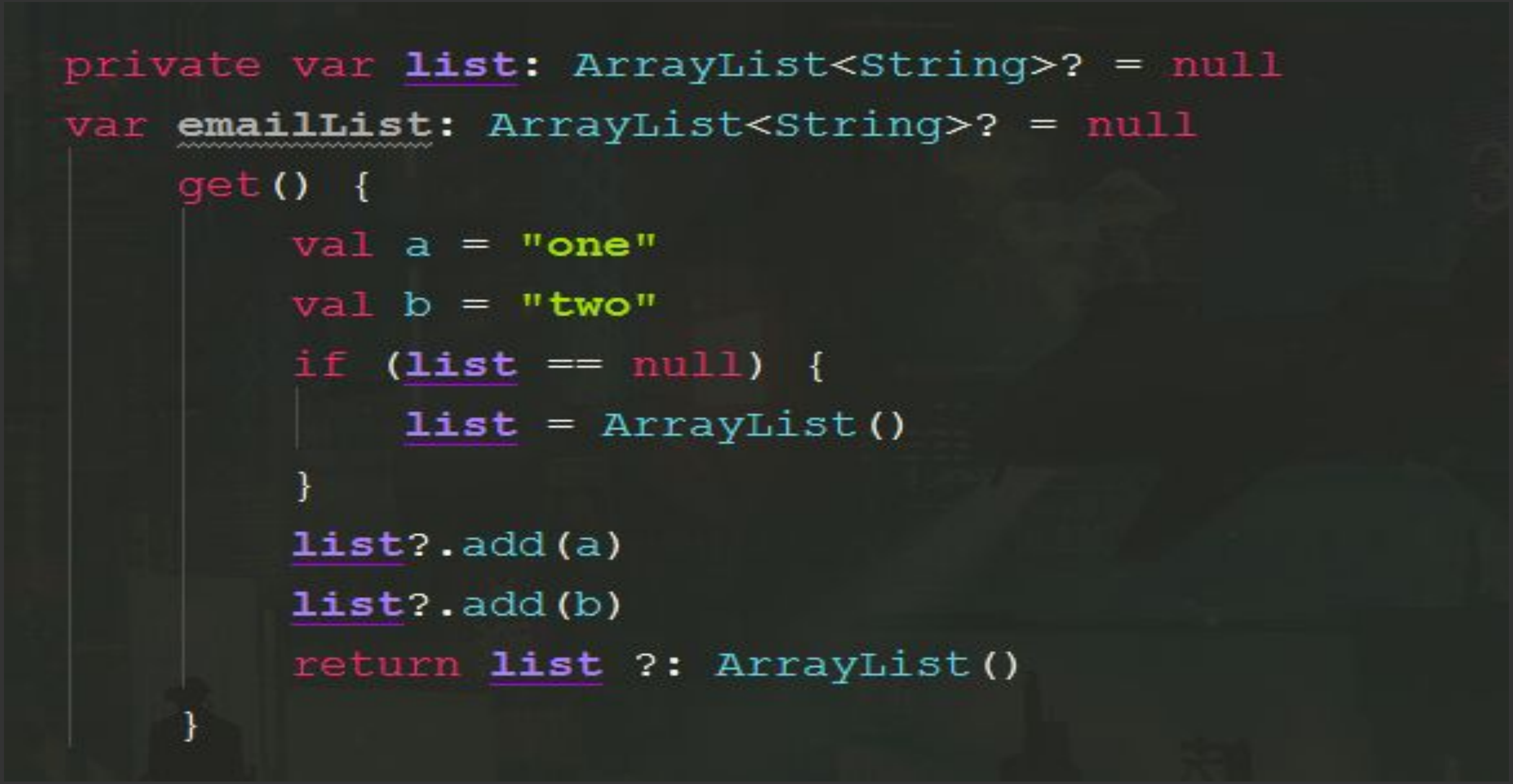
```
class Login (var email: String = "email",
             var pass: String) {

    init {
        val sum = 4 + 6
        printPass(sum)
    }

    private fun printPass(sum: Int) = print("It's sum: $sum" +
            "  more then ${pass.length}")

    fun getBothName(): String {...}
```

Класс. Getter. Setter.



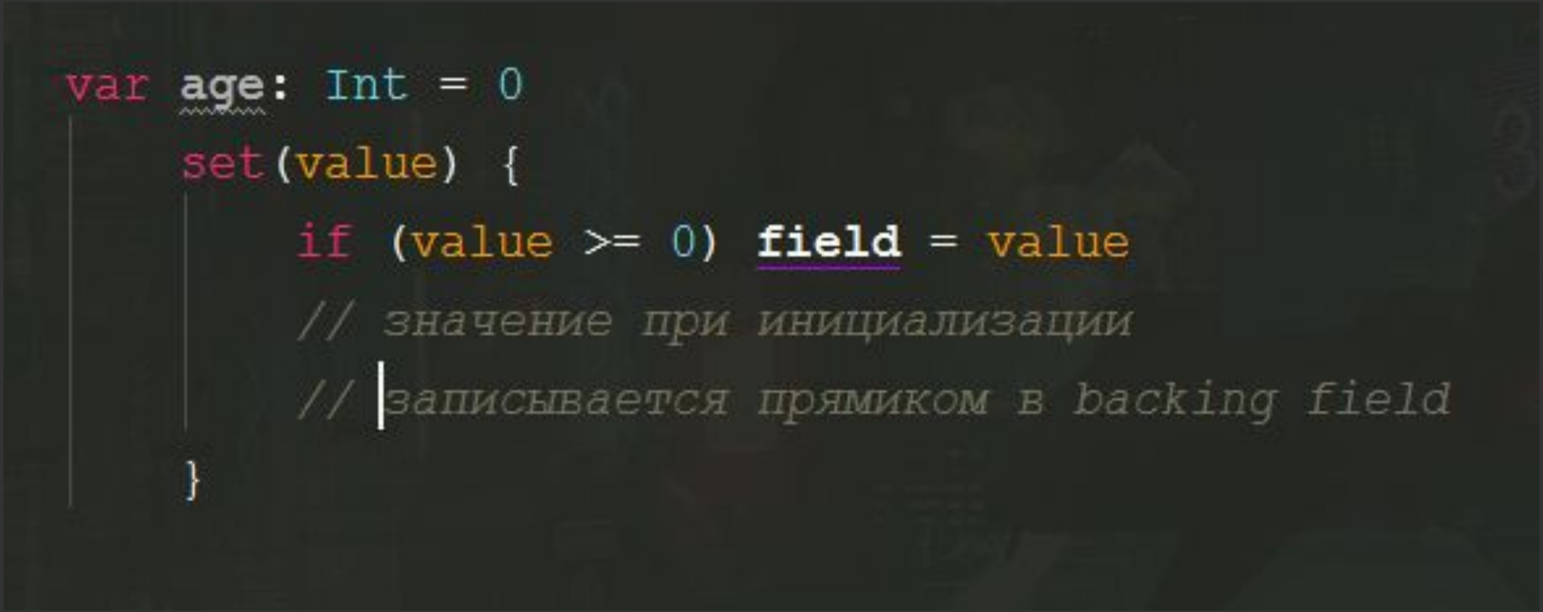
```
class Login(var email: String = "email",
            private var pass: String = "") : User(), MyCallback {

    var isAdult: Boolean
        get() = email.length > 18
        set(value) {
            email.length > 18
        }

    var age: Int = 3
        private set
```

Класс. Getter. Setter.



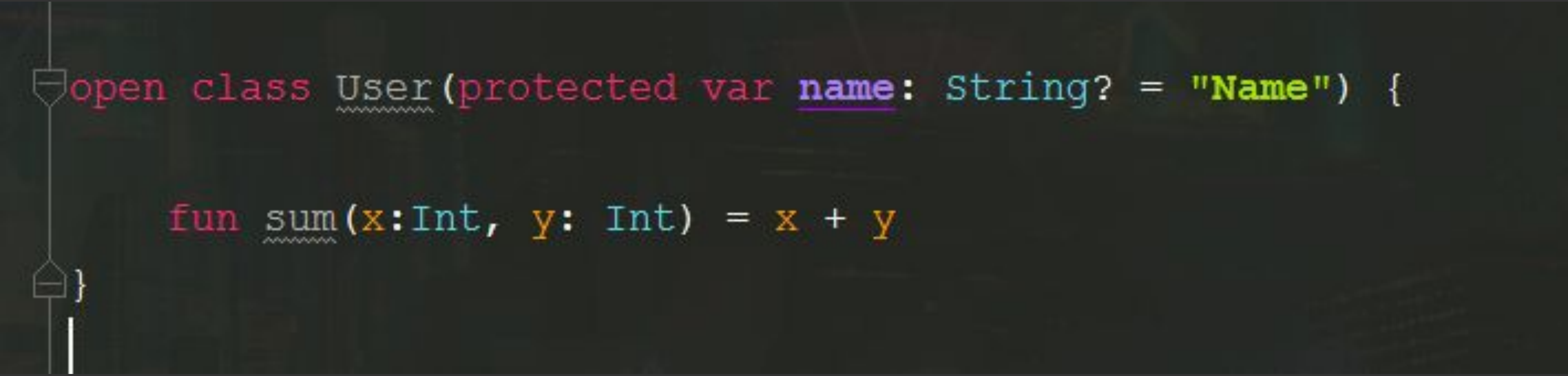
```
private var list: ArrayList<String>? = null
var emailList: ArrayList<String>? = null
    get() {
        val a = "one"
        val b = "two"
        if (list == null) {
            list = ArrayList()
        }
        list?.add(a)
        list?.add(b)
        return list ?: ArrayList()
    }
```

Класс. Getter. Setter.



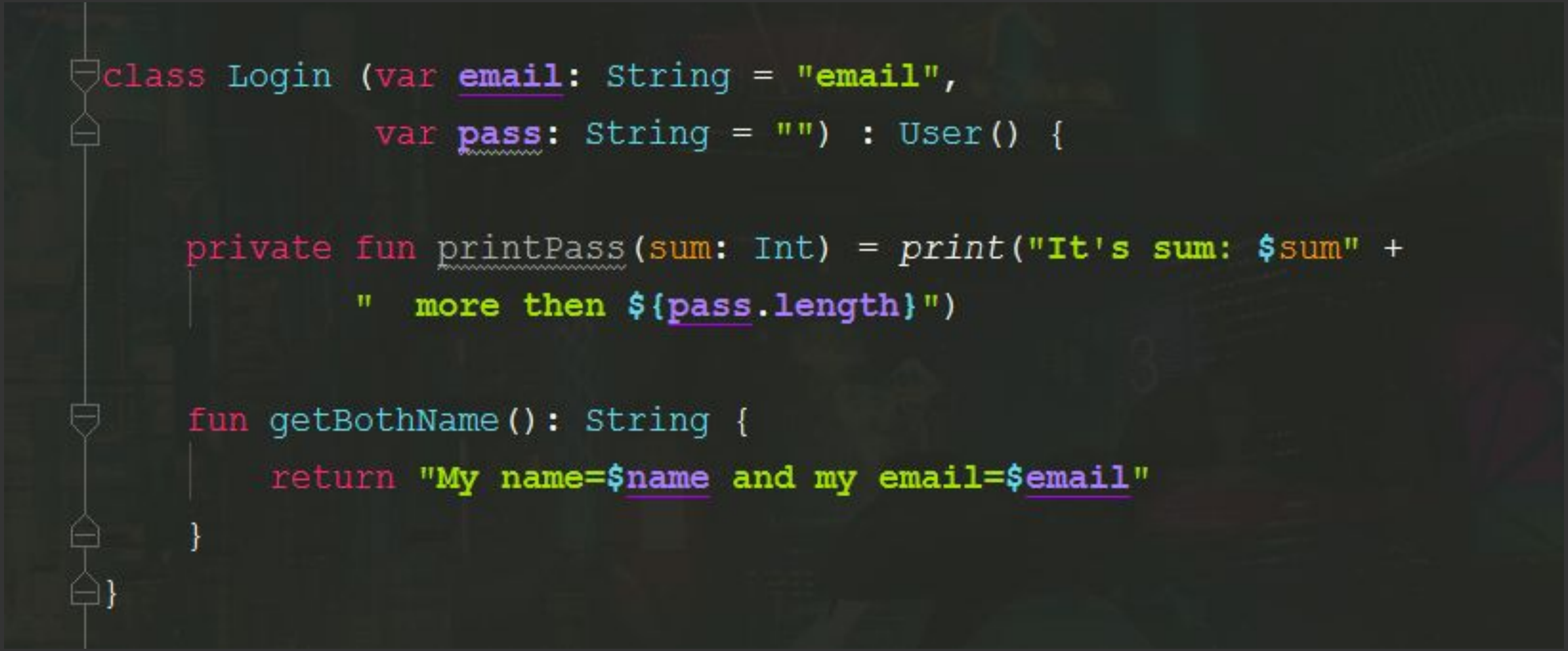
```
var age: Int = 0
    set(value) {
        if (value >= 0) field = value
        // значение при инициализации
        // записывается прямиком в backing field
    }
```

Класс. Наследование. Parent.



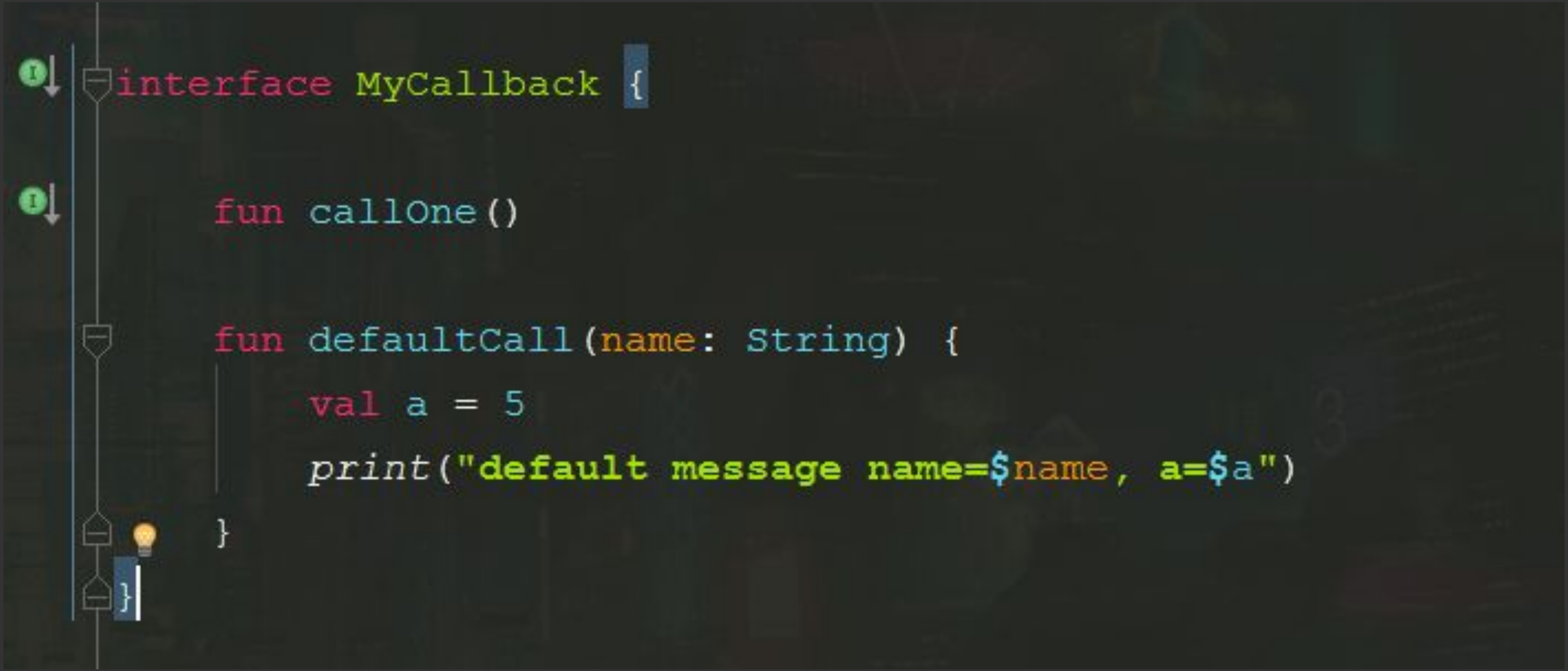
```
open class User(protected var name: String? = "Name") {

    fun sum(x:Int, y: Int) = x + y
}
```

Класс. Наследование. Child.



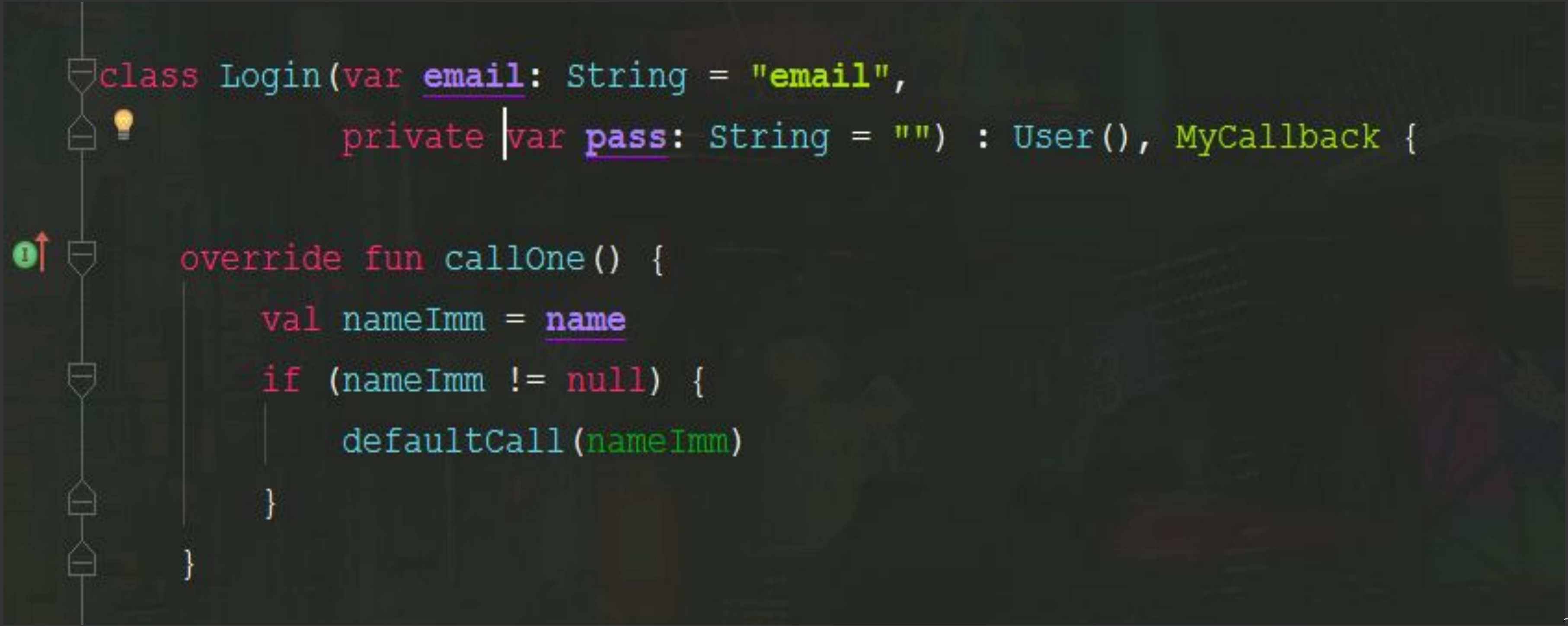
```
class Login (var email: String = "email",
             var pass: String = "") : User() {

    private fun printPass(sum: Int) = print("It's sum: $sum" +
            "  more then ${pass.length}")

    fun getBothName(): String {
        return "My name=$name and my email=$email"
    }
}
```

Интерфейс



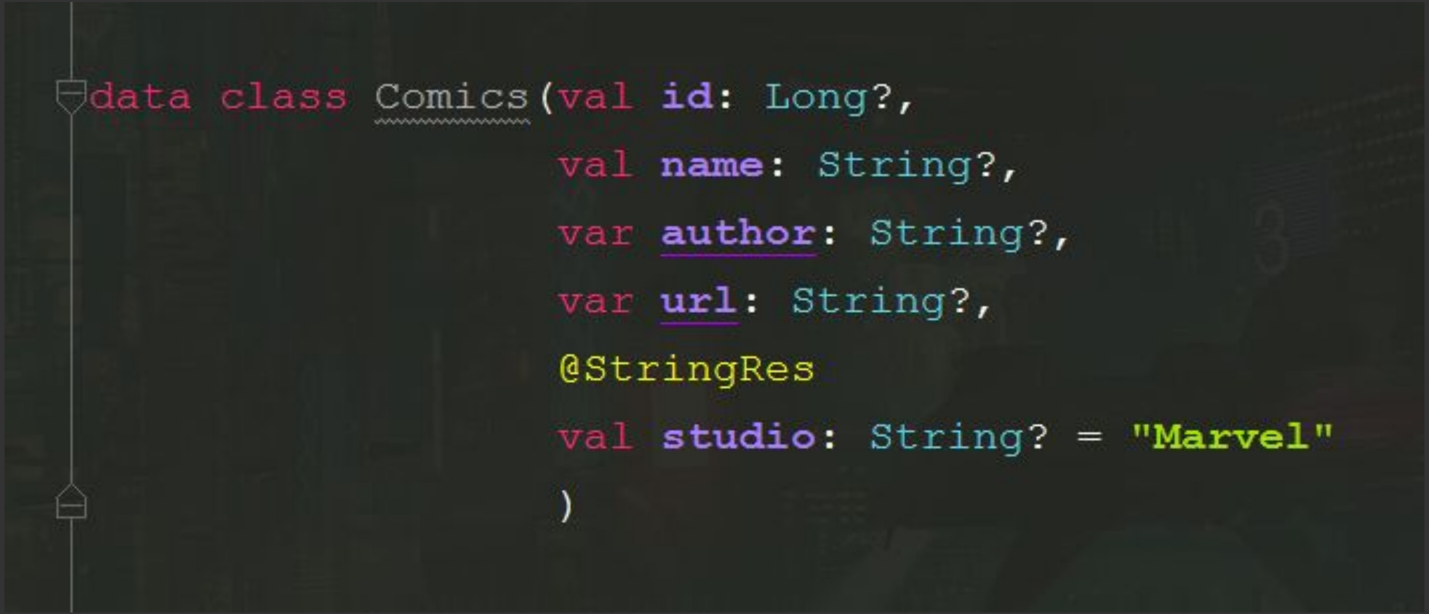
```kotlin
interface MyCallback {

    fun callOne()

    fun defaultCall(name: String) {
        val a = 5
        print("default message name=$name, a=$a")
    }
}
```

Интерфейс. Реализация.



```
class Login(var email: String = "email",
            private var pass: String = "") : User(), MyCallback {

    override fun callOne() {
        val nameImm = name
        if (nameImm != null) {
            defaultCall(nameImm)
        }
    }
```

Дата класс = POJO



```
data class Comics(val id: Long?,
                  val name: String?,
                  var author: String?,
                  var url: String?,
                  @StringRes
                  val studio: String? = "Marvel"
                  )
```

Java POJO Kotlin

```java
class Person {
    private String name;

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

    public String getName() {
        return name;
    }

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

    // toString…

    // hashCode…

    // equals…

    // copy…
```

```kotlin
data class Person(val name: String)
```

NULL.NULL.NULL.NULL.NULL.NULL.NULL!!NULL.NULL.NULL.

1. Сокращение для "Если не null"

    val name: String? = “Name”
    name?.length

1. Сокращение для "Если не null, иначе"

    name?.length ?: “default name”

1. Вызов оператора при равенстве null

    name?.length ?: throw IllegalStateException("name is missing!")

1. NULL!!(можно указать явно, что будет null)

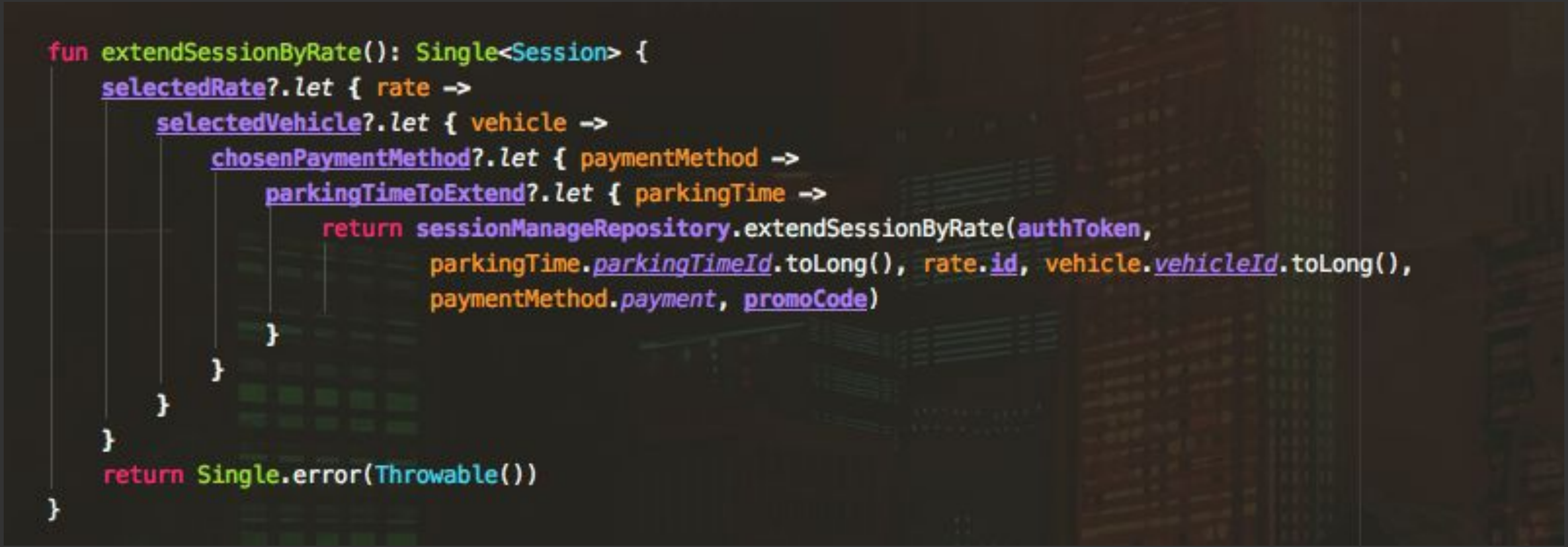
    name!!.length

NULL.NULL.NULL.NULL.NULL.NULL.NULL!!NULL.NULL.NULL.

Выполнение при неравенстве null:

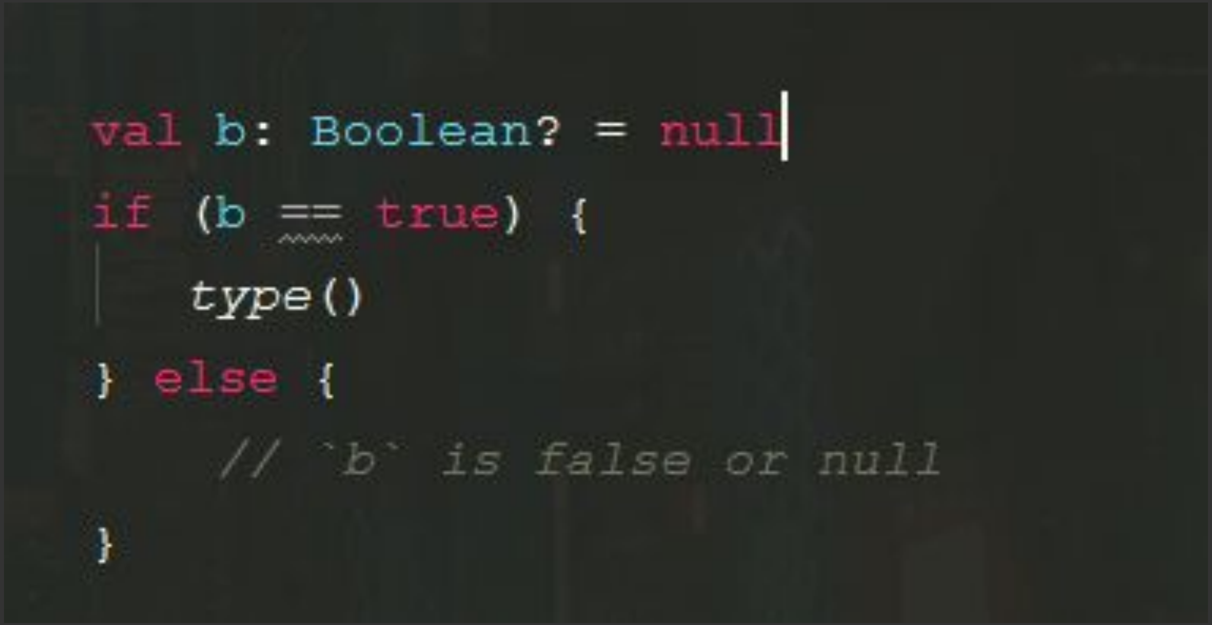
```
    activity?.let { activity ->
        name?.let {
        activity.setUserName(“My name $name”)
    }
}
```

NULL.NULL.NULL.NULL.NULL.NULL.NULL!!NULL.NULL.NULL.



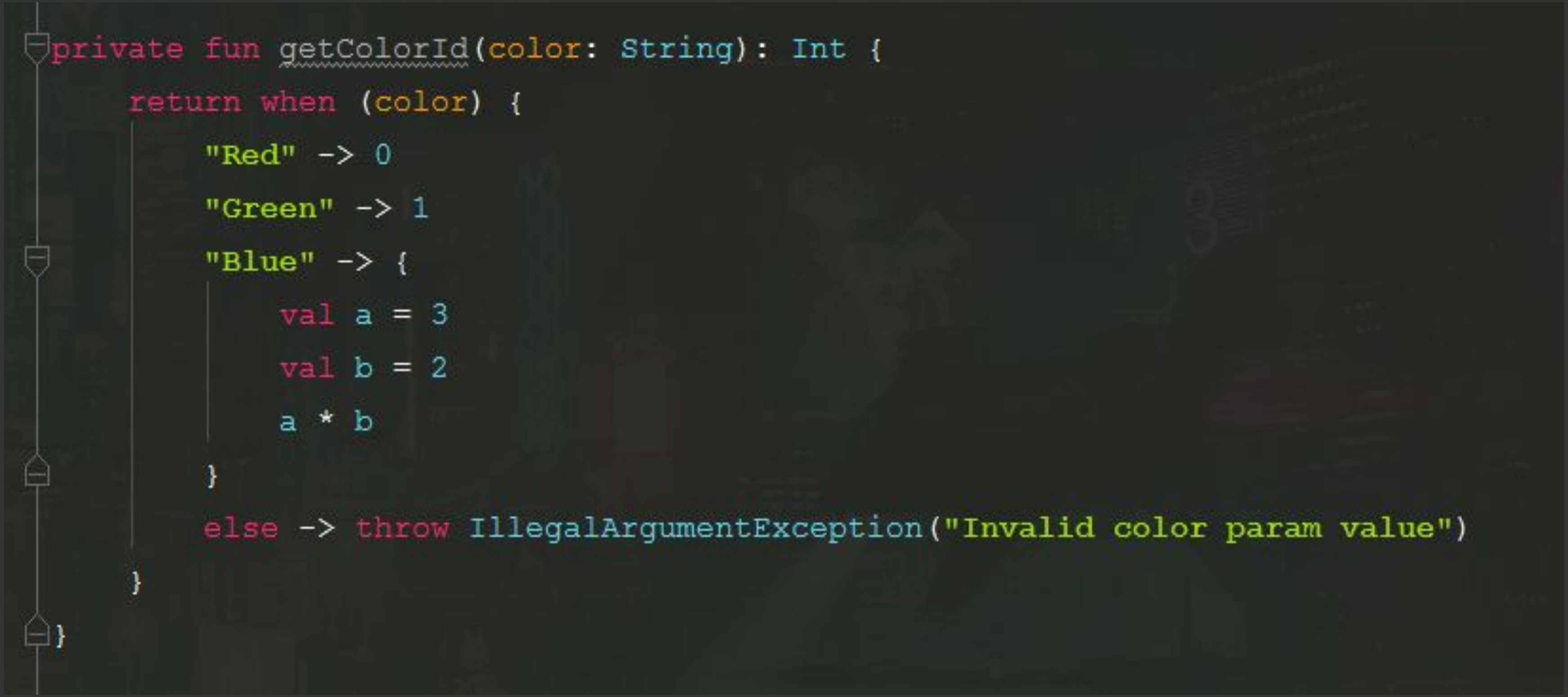
```kotlin
fun extendSessionByRate(): Single<Session> {
    selectedRate?.let { rate ->
        selectedVehicle?.let { vehicle ->
            chosenPaymentMethod?.let { paymentMethod ->
                parkingTimeToExtend?.let { parkingTime ->
                    return sessionManageRepository.extendSessionByRate(authToken,
                            parkingTime.parkingTimeId.toLong(), rate.id, vehicle.vehicleId.toLong(),
                            paymentMethod.payment, promoCode)
                }
            }
        }
    }
    return Single.error(Throwable())
}
```

Обработка nullable Boolean

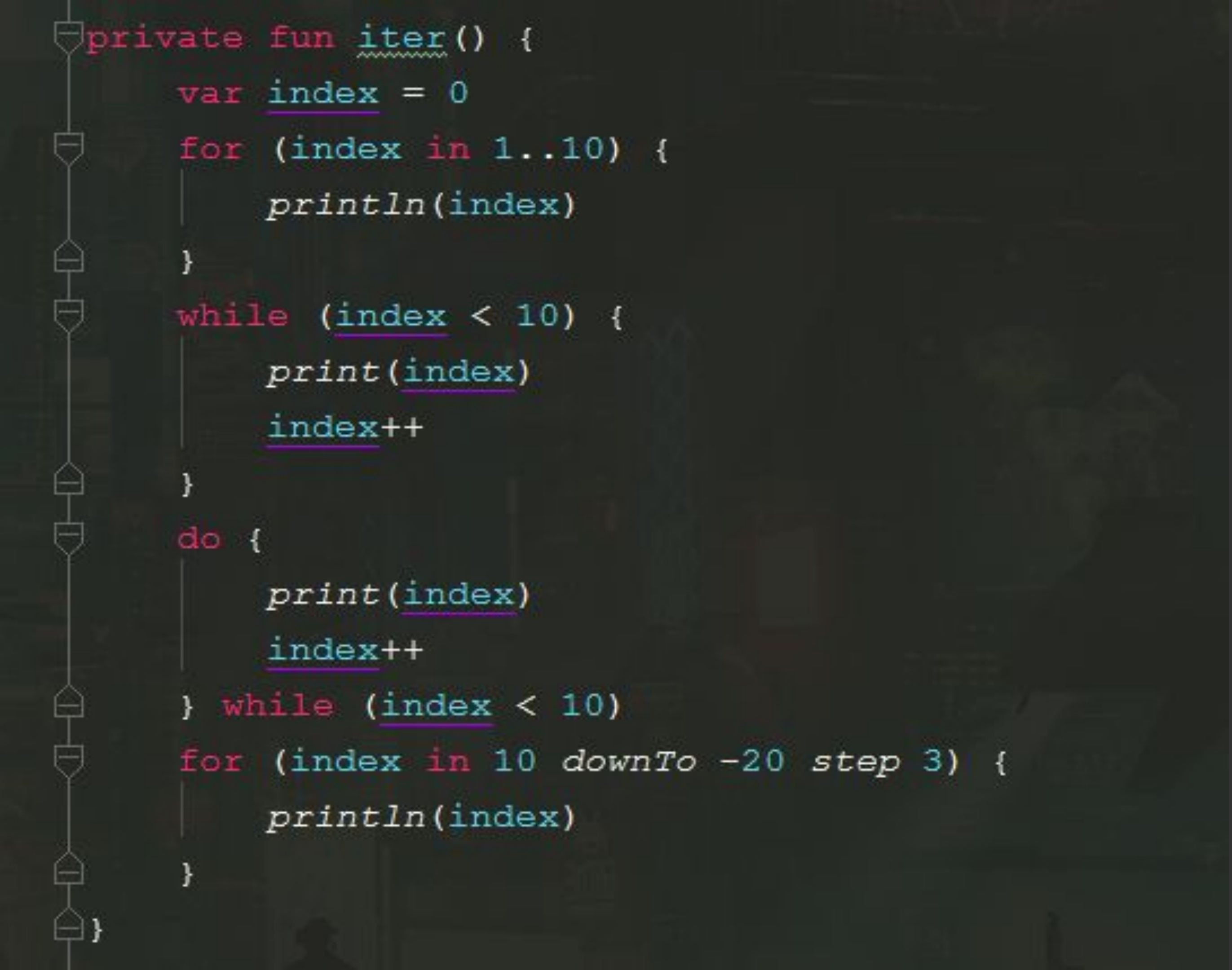


```
val b: Boolean? = null
if (b == true) {
    type()
} else {
    // `b` is false or null
}
```

Switch. Case.



```
private fun getColorId(color: String): Int {
    return when (color) {
        "Red" -> 0
        "Green" -> 1
        "Blue" -> {
            val a = 3
            val b = 2
            a * b
        }
        else -> throw IllegalArgumentException("Invalid color param value")
    }
}
```

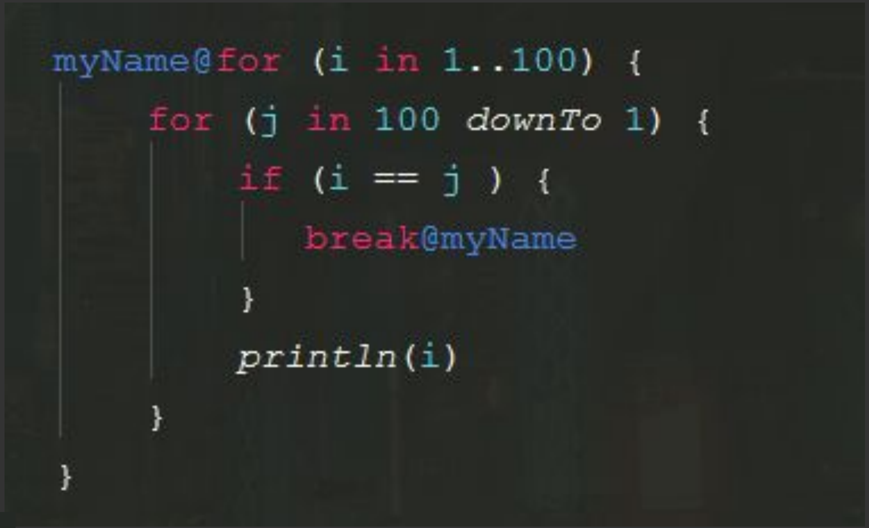


```
private fun iter() {
    var index = 0
    for (index in 1..10) {
        println(index)
    }
    while (index < 10) {
        print(index)
        index++
    }
    do {
        print(index)
        index++
    } while (index < 10)
    for (index in 10 downTo -20 step 3) {
        println(index)
    }
}
```

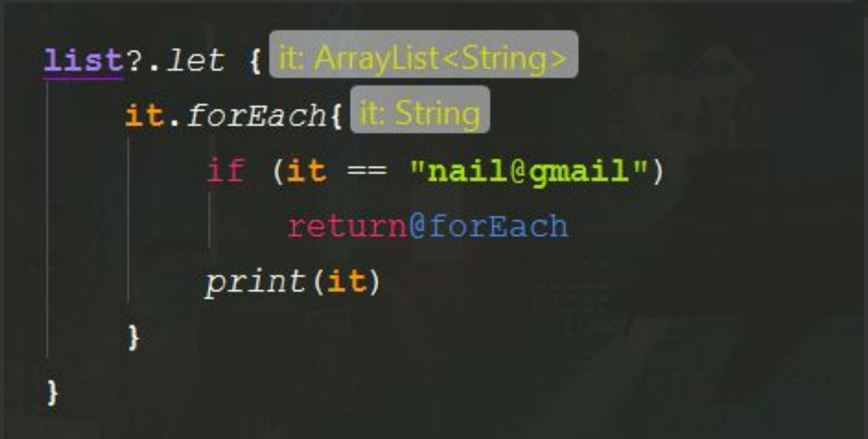
Цикл

Операторы перехода

1. return
2. break
3. continue



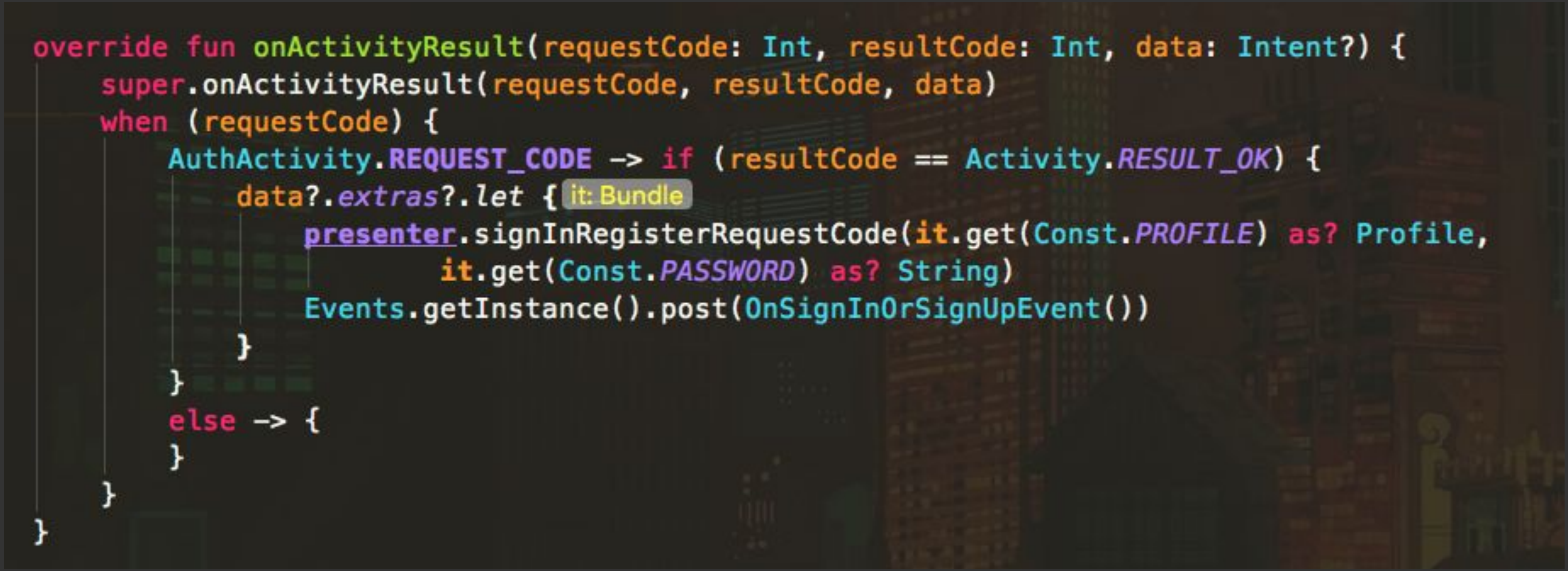
```
myName@for (i in 1..100) {
    for (j in 100 downTo 1) {
        if (i == j ) {
            break@myName
        }
        println(i)
    }
}
```



```
list?.let { it: ArrayList<String>
    it.forEach{ it: String
        if (it == "nail@gmail")
            return@forEach
        print(it)
    }
}
```

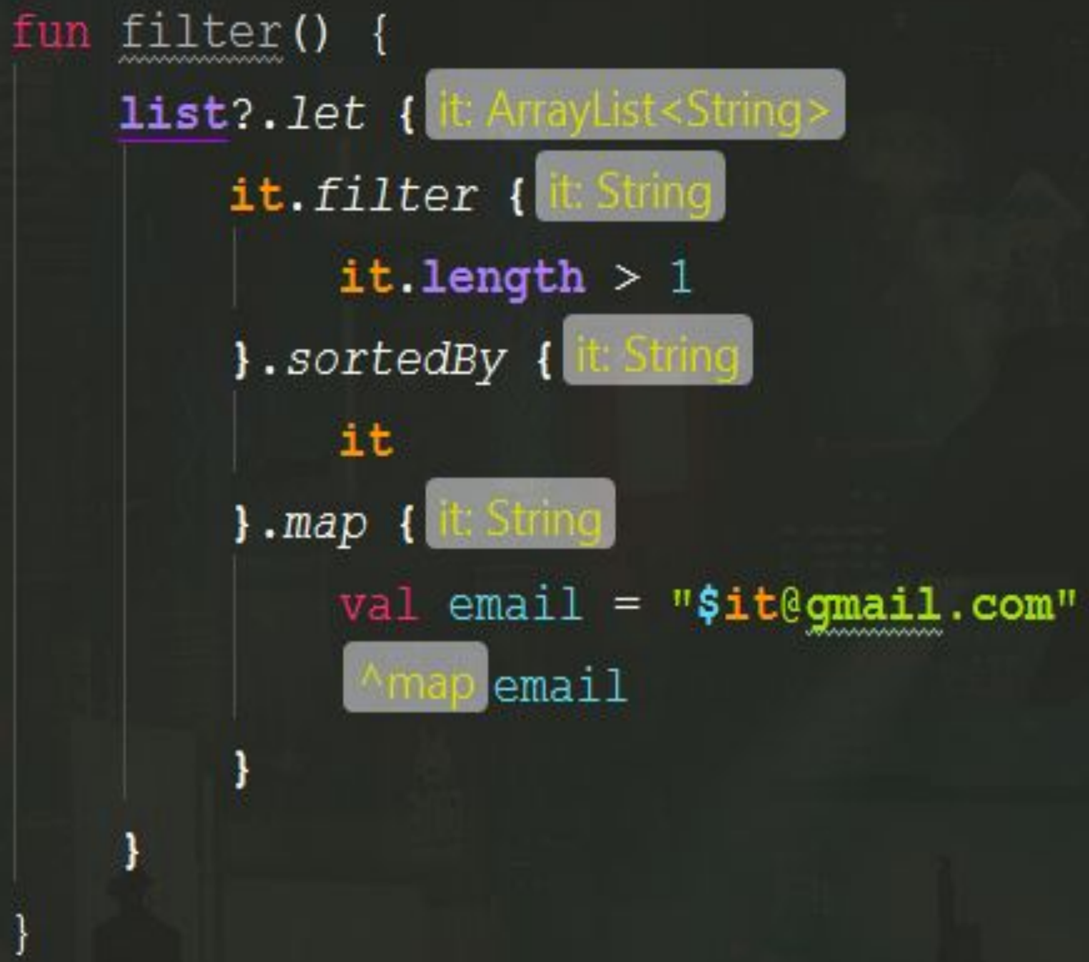
Любое выражение в **Kotlin** может быть помечено меткой **label**. Метки имеют идентификатор в виде знака **@**

Приведение типов

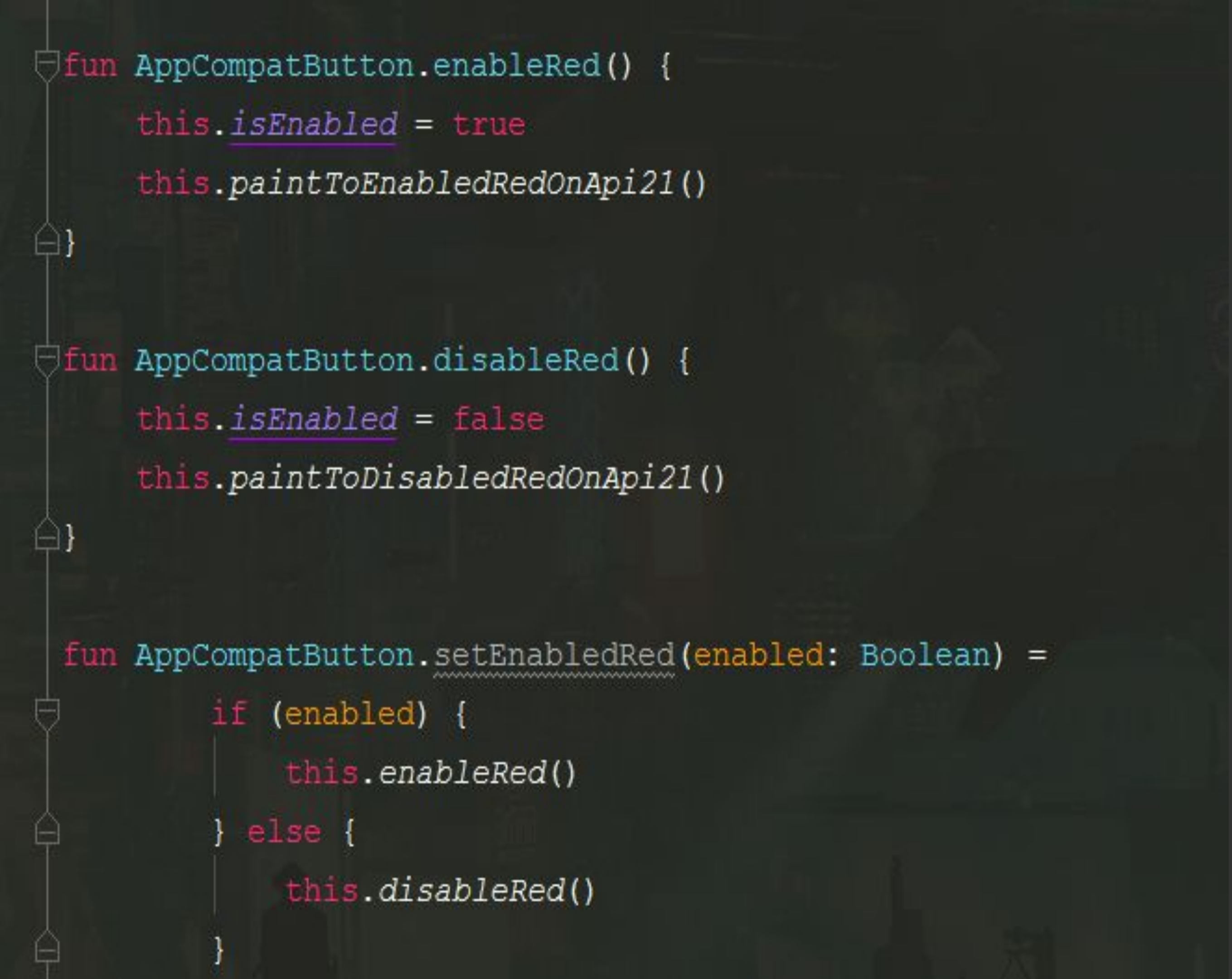


```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    when (requestCode) {
        AuthActivity.REQUEST_CODE -> if (resultCode == Activity.RESULT_OK) {
            data?.extras?.let { it: Bundle
                presenter.signInRegisterRequestCode(it.get(Const.PROFILE) as? Profile,
                        it.get(Const.PASSWORD) as? String)
                Events.getInstance().post(OnSignInOrSignUpEvent())
            }
        }
        else -> {
        }
    }
}
```

Лямбда.
It.



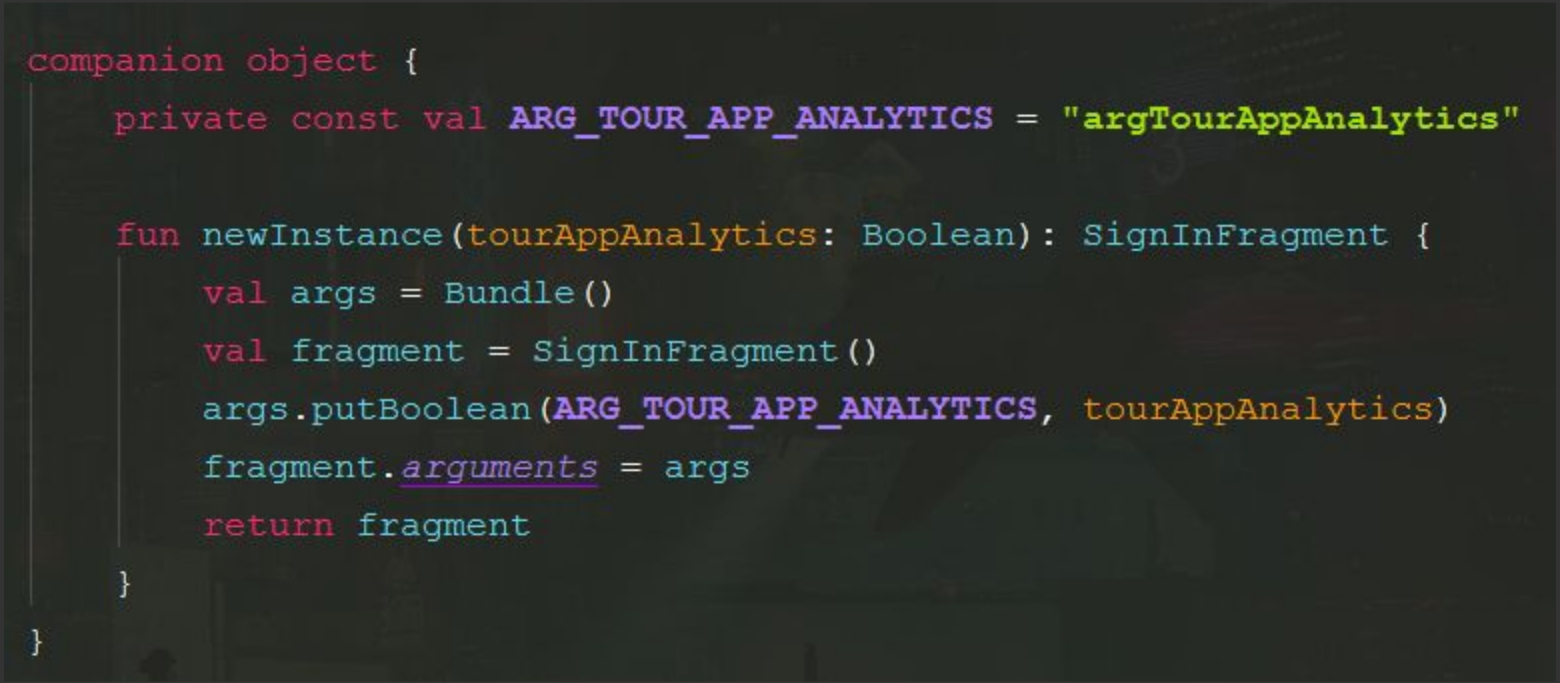
```
fun filter() {
    list?.let { it: ArrayList<String>
        it.filter { it: String
            it.length > 1
        }.sortedBy { it: String
            it
        }.map { it: String
            val email = "$it@gmail.com"
            ^map email
        }
    }
}
```



```
fun AppCompatButton.enableRed() {
    this.isEnabled = true
    this.paintToEnabledRedOnApi21()
}

fun AppCompatButton.disableRed() {
    this.isEnabled = false
    this.paintToDisabledRedOnApi21()
}

fun AppCompatButton.setEnabledRed(enabled: Boolean) =
        if (enabled) {
            this.enableRed()
        } else {
            this.disableRed()
        }
```

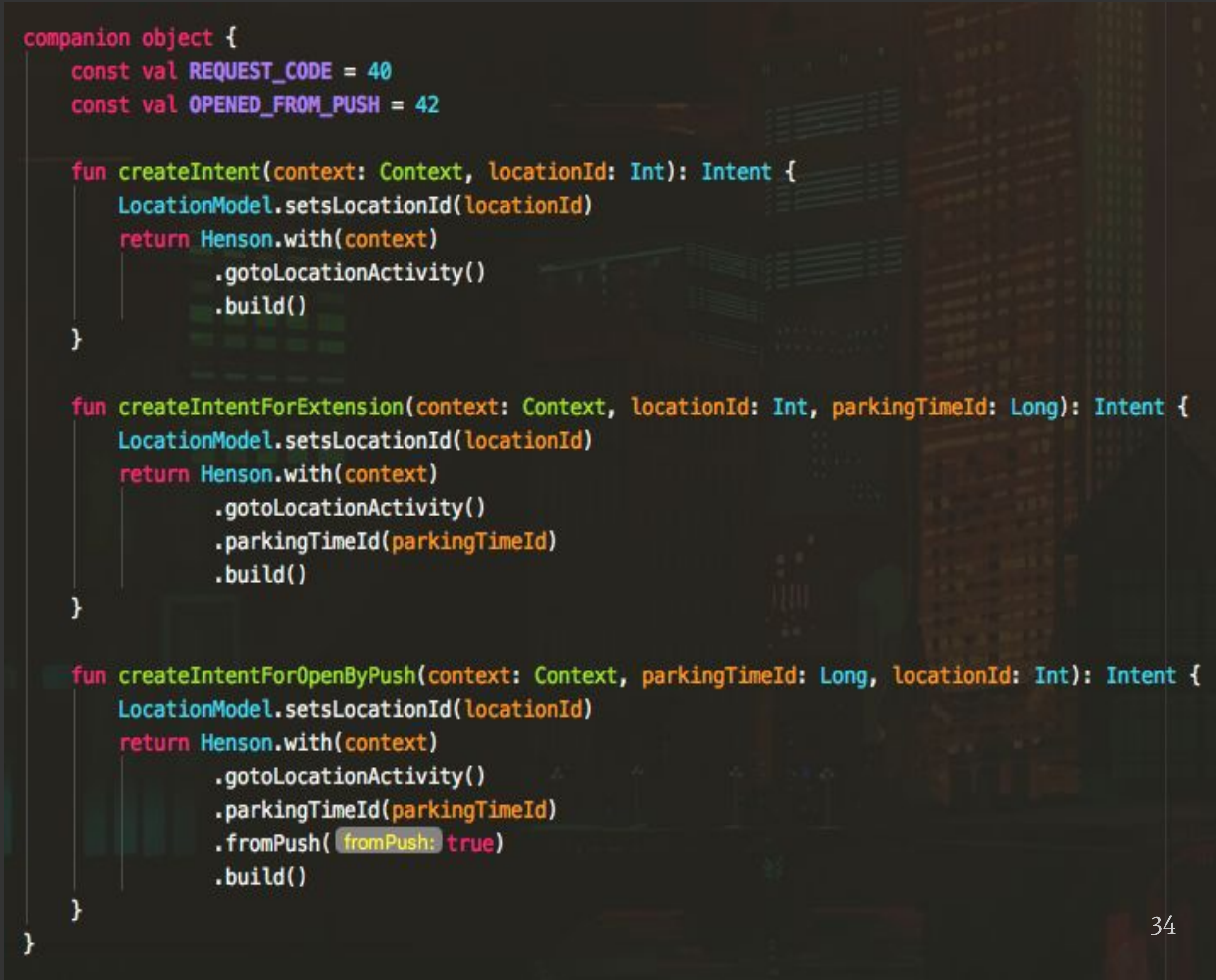
Расширение

Companion object



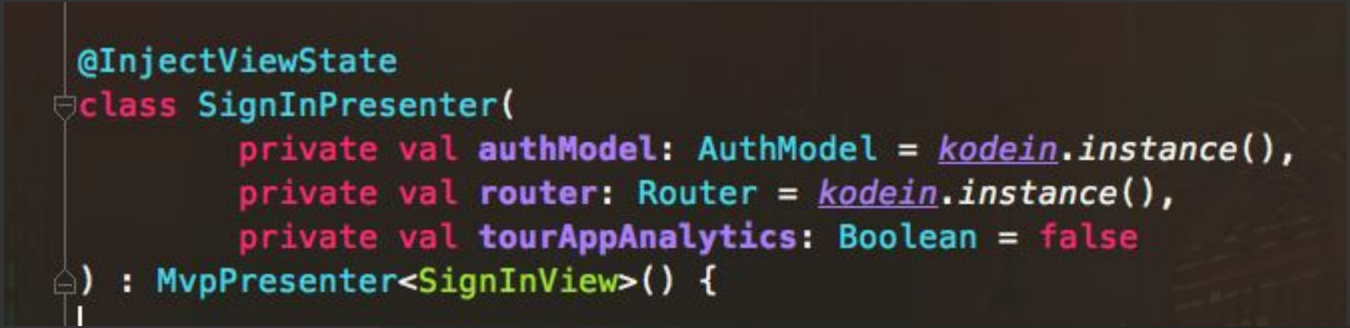
```
companion object {
    private const val ARG_TOUR_APP_ANALYTICS = "argTourAppAnalytics"

    fun newInstance(tourAppAnalytics: Boolean): SignInFragment {
        val args = Bundle()
        val fragment = SignInFragment()
        args.putBoolean(ARG_TOUR_APP_ANALYTICS, tourAppAnalytics)
        fragment.arguments = args
        return fragment
    }
}
```

Companion object

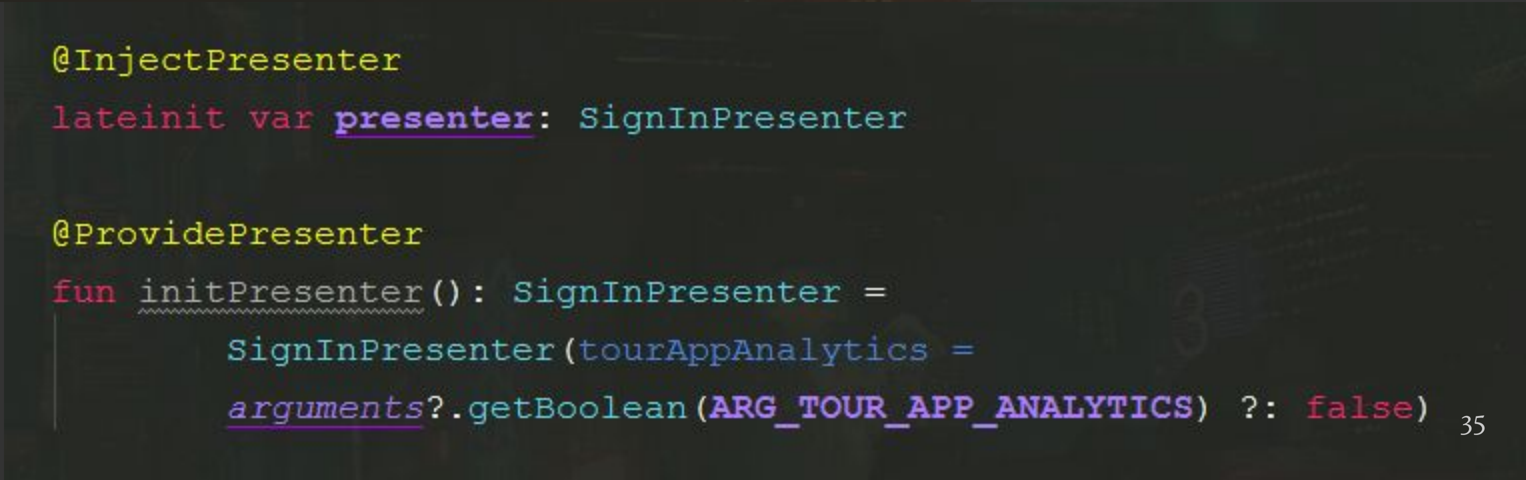


```kotlin
companion object {
    const val REQUEST_CODE = 40
    const val OPENED_FROM_PUSH = 42

    fun createIntent(context: Context, locationId: Int): Intent {
        LocationModel.setsLocationId(locationId)
        return Henson.with(context)
                .gotoLocationActivity()
                .build()
    }

    fun createIntentForExtension(context: Context, locationId: Int, parkingTimeId: Long): Intent {
        LocationModel.setsLocationId(locationId)
        return Henson.with(context)
                .gotoLocationActivity()
                .parkingTimeId(parkingTimeId)
                .build()
    }

    fun createIntentForOpenByPush(context: Context, parkingTimeId: Long, locationId: Int): Intent {
        LocationModel.setsLocationId(locationId)
        return Henson.with(context)
                .gotoLocationActivity()
                .parkingTimeId(parkingTimeId)
                .fromPush( fromPush: true)
                .build()
    }
}
```

Именованные аргументы

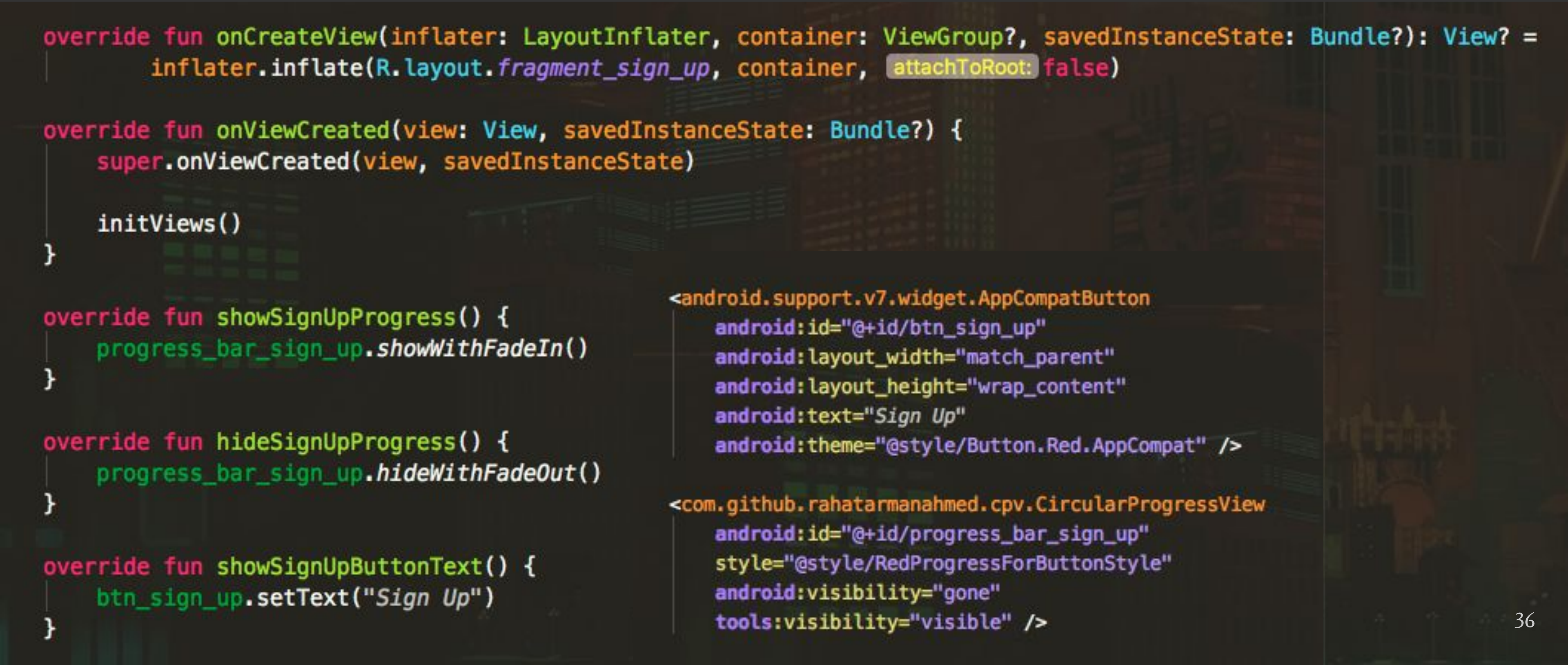


```
@InjectViewState
class SignInPresenter(
        private val authModel: AuthModel = kodein.instance(),
        private val router: Router = kodein.instance(),
        private val tourAppAnalytics: Boolean = false
) : MvpPresenter<SignInView>() {
```



```
@InjectPresenter
lateinit var presenter: SignInPresenter

@ProvidePresenter
fun initPresenter(): SignInPresenter =
        SignInPresenter(tourAppAnalytics =
        arguments?.getBoolean(ARG_TOUR_APP_ANALYTICS) ?: false)
```

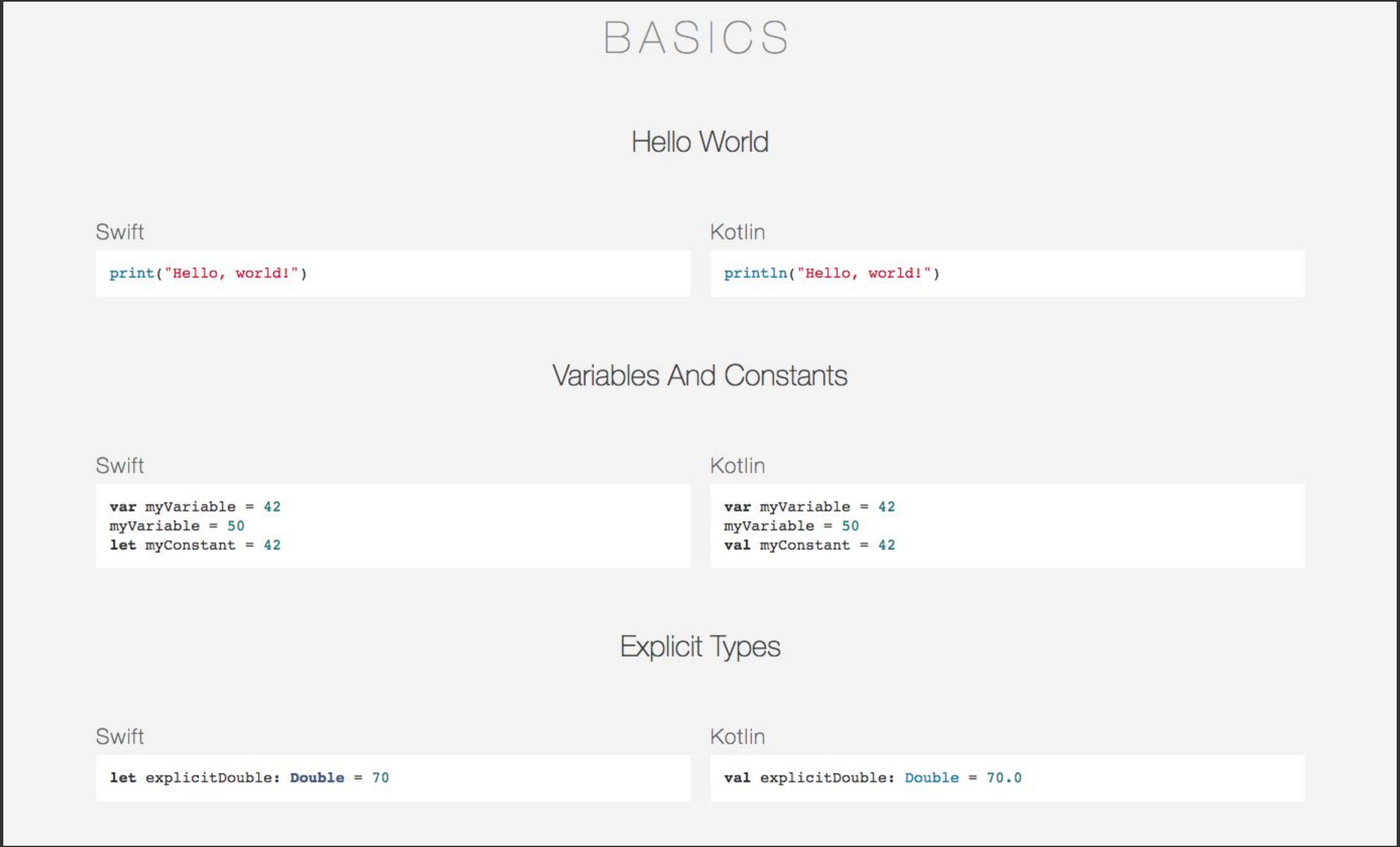
FindViewById(R.id.view_name)



```
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? =
        inflater.inflate(R.layout.fragment_sign_up, container, attachToRoot: false)

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    initViews()
}

override fun showSignUpProgress() {
    progress_bar_sign_up.showWithFadeIn()
}

override fun hideSignUpProgress() {
    progress_bar_sign_up.hideWithFadeOut()
}

override fun showSignUpButtonText() {
    btn_sign_up.setText("Sign Up")
}
```

```
<android.support.v7.widget.AppCompatButton
    android:id="@+id/btn_sign_up"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Sign Up"
    android:theme="@style/Button.Red.AppCompat" />

<com.github.rahatarmanahmed.cpv.CircularProgressView
    android:id="@+id/progress_bar_sign_up"
    style="@style/RedProgressForButtonStyle"
    android:visibility="gone"
    tools:visibility="visible" />
```

Orange County Choppers



BASICS

Hello World

Swift

```swift
print("Hello, world!")
```

Kotlin

```kotlin
println("Hello, world!")
```

Variables And Constants

Swift

```swift
var myVariable = 42
myVariable = 50
let myConstant = 42
```

Kotlin

```kotlin
var myVariable = 42
myVariable = 50
val myConstant = 42
```

Explicit Types

Swift

```swift
let explicitDouble: Double = 70
```

Kotlin

```kotlin
val explicitDouble: Double = 70.0
```

Type Coercion

Swift

```
let label = "The width is "
let width = 94
let widthLabel = label + String(width)
```

Kotlin

```
val label = "The width is "
val width = 94
val widthLabel = label + width
```

String Interpolation

Swift

```
let apples = 3
let oranges = 5
let fruitSummary = "I have \(apples + oranges) " +
                   "pieces of fruit."
```

Kotlin

```
val apples = 3
val oranges = 5
val fruitSummary = "I have ${apples + oranges} " +
                   "pieces of fruit."
```

Range Operator

Swift

```
let names = ["Anna", "Alex", "Brian", "Jack"]
let count = names.count
for i in 0..<count {
    print("Person \(i + 1) is called \(names[i])")
}
// Person 1 is called Anna
// Person 2 is called Alex
// Person 3 is called Brian
// Person 4 is called Jack
```

Kotlin

```
val names = arrayOf("Anna", "Alex", "Brian", "Jack")
val count = names.count()
for (i in 0..count - 1) {
    println("Person ${i + 1} is called ${names[i]}")
}
// Person 1 is called Anna
// Person 2 is called Alex
// Person 3 is called Brian
// Person 4 is called Jack
```

Ссылки

- https://kotlinlang.org/
- https://kotlinlang.ru/
- https://blog.mindorks.com/a-complete-guide-to-learn-kotlin-for-android-development-b1e5d23cc2d8
- https://antonioleiva.com/kotlin-android-extensions/
- @kotlin_lang