Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Kotlin Coding Conventions

Доповідь виконав
студент III курсу
групи ПЗПІ-22-2
Синенко Іван Костянтинович

Перевірив
Доц. кафедри ПІ
Лещинський Володимир Олександрович

2024

Мета роботи

Дослідити основні рекомендації щодо написання коду мовою Kotlin і представити їх у вигляді презентації

Хід роботи:

Я прочитав Kotlin Coding Conventions з офіційного сайту мови Kotlin та кілька інших статей на цьому сайті, і створив презентацію.

Загалом Kotlin створювався як «покращена версія» Java.

Статична типізація, проте можливо і не вказувати тип явно, якщо змінна ініціалізована при створенні.

Можливість перевіряти тип змінної (спрощений smart cast через слово is).

Можливість призначати типи nullable доданням «?» після типу, що значно зменшує ризики посилання на null (Java страждала від NullPointerException і тому у Java 8 був введений клас Optional, але він дещо громіздкий).

Конвенції цієї мови направлені на створення легко читаємого об'єктно-орієнтованого коду.

Висновки:

Синтаксис Kotlin дозволяє писати компактніший і легше читаємий код, ніж Java і конвенції підтримують його у цьому.

ДОДАТОК А
Слайди презентації

Kotlin coding conventions

Мова Kotlin

Причина створення

Мова Kotlin створена компанією JetBrains як покращена версія Java: більш лаконічна, і типобезпечна, ніж Java, і простіша, ніж Scala. Працює поверх віртуальної машини JVM, може компілюватися в JavaScript. Можна викликати код Kotlin з коду Java і навпаки.
Мова мультиплатформенна, влаштована в Android Studio.

Синтаксис

Мова статично типізована. Але якщо змінній одразу ж призначене значення, компілятор визначає тип автоматично, проте у іншому випадку треба написати тип.
Крапки з комами не обов'язкові.
Змінні можуть бути nullable, для цього їх треба помітити знаком питання.
Також перевірка типа змінної присутня у вигляді слова “is”.

01

Чому Kotlin?

- компактніший синтаксис
- не страждає від небезпеки NullPointerException
- є додаткові можливості, як функції розширення чи спрощений smart cast
- може викликати Java код і навпаки

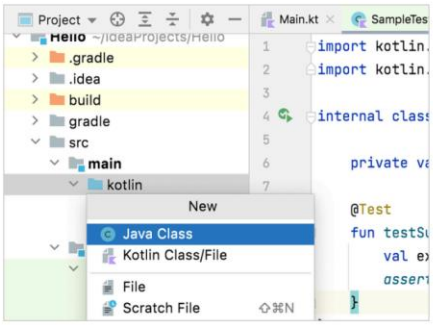
02

03

Файли і пакети

Найменування файлів

Файли мають мати розширення .kt. Як і в Java, окремі класи мають бути винесені в окремі файли, де назви класів мають співпадати з назвами файлів.



04

Найменування пакетів

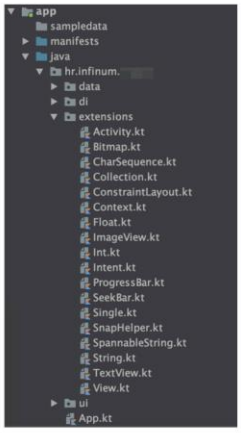
Всі пакети мають бути названі з маленької літери.

У мультиплатформенних проектах файли, пристосовані для різних платформ мають бути винесені в окремі пакети.

jvmMain/kotlin/Platform.jvm.kt

androidMain/kotlin/Platform.android.kt

iosMain/kotlin/Platform.ios.kt



05

Найменування

06

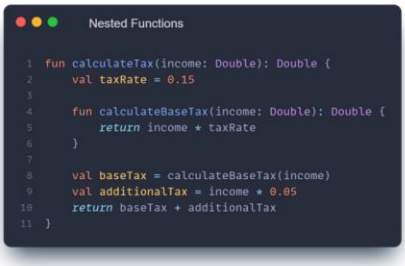
Найменування класів

```
// The class
data class User(val name: String, val age: Int)

// The equivalent copy() implementation in Kotlin
fun copy(name: String = this.name, age: Int = this.age) = User(name, age)

// The usage
val lam = User(name = "LamPham", age = 19)
val olderLam = lam.copy(age = 25)
```

Як і всюди, краще за все використовувати найменування класів, які описують їх функціонал словом.
Назви класів - з великої літери.

07

Найменування функцій



Функції - camelCase, з маленької літери.
Виключення - фабричні функції, що можуть мати ту ж назву, що і абстрактний тип, що вони повертають.

```
interface Foo { /*...*/ }

class FooImpl : Foo { /*...*/ }

fun Foo(): Foo { return FooImpl() }
```

08

Найменування тестів

Тільки у тестах дозволяється використання одинарних лапок.

Треба відзначити, що це не працює на багатьох версіях андроіду, і тому найменування може змінюватися.

```
class MyTestCase {
    @Test fun `ensure everything works`() { /*...*/ }

    @Test fun ensureEverythingWorks_onAndroid() { /*...*/ }
}
```

09

Найменування властивостей

```
class C {
    private val _elementList = mutableListOf<Element>()

    val elementList: List<Element>
        get() = _elementList
}
```

Константи чи незмінні val властивості - великими літерами з _.

Звичайні властивості - camelCase-ом.

Приватні властивості , які лише для внутрішнього користування - починати з _.

Форматування

Форматування

Прийнято використовувати пробіли замість табуляцій. Чотири пробіли на відступ.
Дужки принято відділяти пробілами.

```
if (elements != null) {
    for (element in elements) {
        // ...
    }
}
```

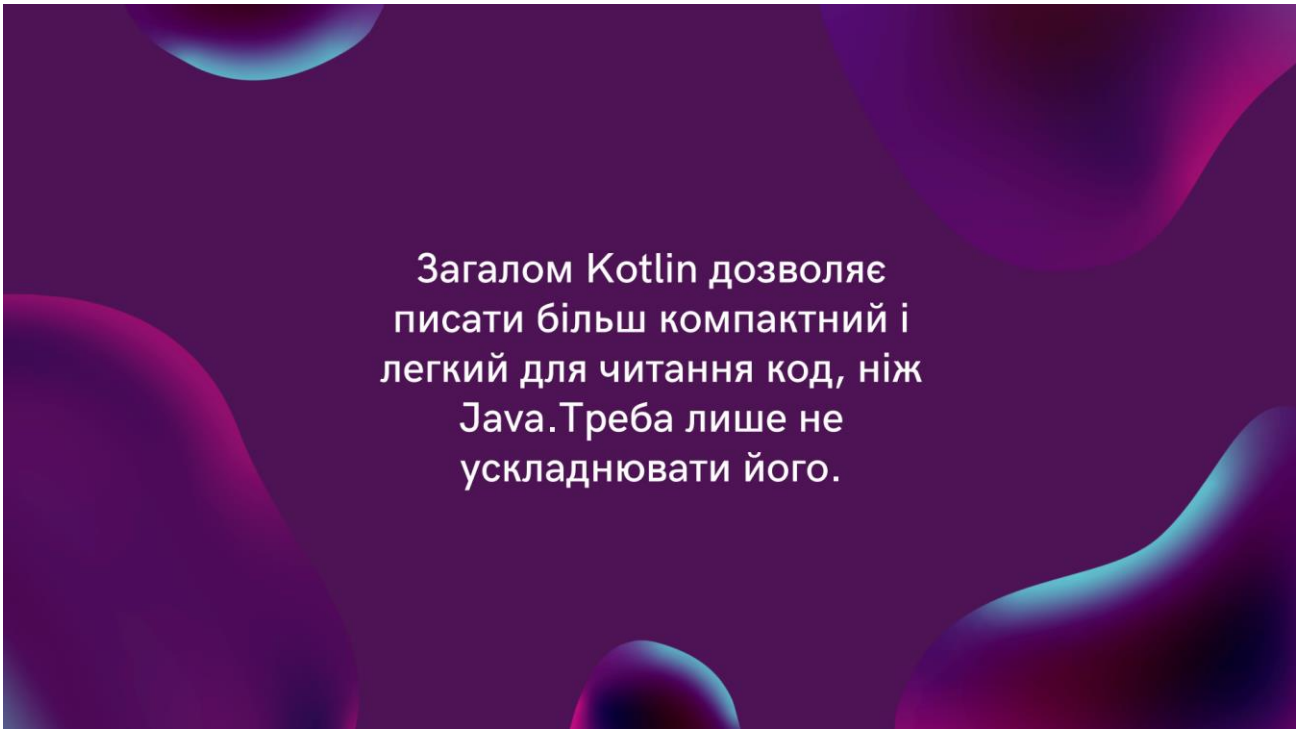
Точки з комами не обов'язкові, тож їх не треба використовувати.

Intellij Idea та Android Studio можуть передивлятися стиль вашого коду.

Для цього треба встановити Settings/Preferences | Editor | Code Style | Kotlin. Set from Kotlin Style Guide.

Порядок модифікаторів

```
public / protected / private / internal
expect / actual
final / open / abstract / sealed / const
external
override
lateinit
tailrec
vararg
suspend
inner
enum / annotation / fun // as a modifier in `fun interface`
companion
inline / value
infix
operator
data
```


Загалом Kotlin дозволяє
писати більш компактний і
легкий для читання код, ніж
Java.Треба лише не
ускладнювати його.