# ПРАКТИЧЕСКАЯ РАБОТА №1

## 1.1. Основы языка Kotlin

### *1.1.1. Сравнение Kotlin и Java как языков для мобильной разработки*

В современном цифровом контексте, где мобильные приложения занимают центральное место в жизни человека, определение оптимального языка программирования становится первостепенной задачей для разработчиков. В арсенале доступных языков для платформы Android, Java и Kotlin выступают в качестве главных кандидатов, каждый из которых обладает уникальным набором характеристик и преимуществ.

Java, которая на протяжении длительного времени оставалась предпочтительным выбором благодаря своей проверенной стабильности и возможности масштабирования, продолжает быть важным игроком в этой области.

Тем не менее, Kotlin, являющийся более новым и инновационным языком, быстро набирает популярность за счет своего современного синтаксиса и функциональных возможностей, которые направлены на упрощение и ускорение процесса разработки.

В условиях постоянного развития области мобильных технологий, выбор подходящего языка программирования для мобильных приложений приобретает решающее значение в контексте создания эффективных технологических решений. С учетом экспоненциального роста использования мобильных устройств и их влияния на различные аспекты жизни, разработка мобильных приложений становится ключевым элементом в современной бизнес-среде и инновационной деятельности в сфере технологий.

В сфере программирования, адекватный выбор языка, который соответствует как функциональным, так и техническим потребностям проекта, критичен как для эффективности отдельных компонентов приложения, так и для его общей производительности. В контексте разработки мобильных приложений для Android, профессионалы сталкиваются с необходимостью выбора между двумя главными языками программирования: Java и Kotlin, где выбор определяется рядом факторов.

Java, созданная Sun Microsystems и впоследствии приобретенная Oracle, длительное время оставалась основным языком для разработки Android-приложений. Этот язык выделяется своей устойчивостью и мультиплатформенностью, имея обширное применение в области корпоративных приложений. Он базируется на концепциях объектно-

ориентированного программирования и поддерживается обширным сообществом разработчиков, а также имеет богатую экосистему библиотек.

В то же время, Kotlin, разработанный JetBrains на основе Java Virtual Machine, является относительно новым языком, задуманным как средство для преодоления ограничений Java. Быстро получив статус официального языка для Android-разработки, Kotlin отличается современным синтаксисом, который способствует написанию более чистого и понятного кода. Он также поддерживает функциональное программирование и предлагает продвинутые инструменты для асинхронной разработки, что делает его важным выбором в современной программной инженерии.

Можно выделить следующие критерии для сравнения языков:

1. Синтаксис:

Kotlin отличается от Java своим более новым и компактным синтаксисом, обеспечивая повышенную читаемость и сжатость кода. Этот язык программирования сокращает необходимость в написании обширного шаблонного кода, демонстрируя это на примере определения классов данных, где достаточно всего одной строки кода.

В контрасте, Java применяет более консервативный подход в программировании, требуя более развернутого кода для реализации аналогичных функций. Это особенно заметно в случаях, когда разработчикам необходимо написать обширный шаблонный код для классов данных, что увеличивает общий объем кода.

В Листингах 1.1-1.2 можно заметить, как отличается код для класса данных в Kotlin и Java.

Листинг 1.1. Класс данных в Kotlin

```
class Book(val author: String, val name: String)
```

Листинг 1.2. Класс данных в Java

```
public class Book {
	private String author;
	private String name;

	public Book(String author, String name) {
		this.author = author;
		this.name = name;
	}

	public String getAuthor() {
```

```
            return author;
      }

      public void setAuthor(String author) {
            this.author = author;
      }

      public String getName() {
            return name;
      }

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

2. Расширенные функции:

Kotlin обладает функциональностью расширенных методов, позволяющих улучшить существующие классы, добавляя новые возможности без необходимости изменения исходного класса. Эта особенность языка обеспечивает гибкость в добавлении дополнительной функциональности.

В отличие от Kotlin, Java не предоставляет возможностей для реализации расширенных методов. Это создает ограничения в возможностях усовершенствования существующих классов, поскольку требуется использовать наследование или паттерны проектирования, такие как декоратор, для добавления новых функций.

Пример расширения системного класса String, представлен в Листинге 1.3.

Листинг 1.3. Расширение функциональности класса String

```
fun String.addPrefix(prefix: String) {
      println("$this $prefix")
}
```

3. Компиляция:

Kotlin применяет передовые методы компиляции, включая инкрементальную и кэшированную компиляцию, что способствует ускорению процесса разработки. Эта технология позволяет компилятору оптимизировать время обработки, обновляя только те части кода, которые были изменены.

В контрасте, Java использует более традиционный подход к компиляции, который может замедлять процесс разработки, особенно в случае больших

проектов. Эта методика может приводить к более длительному времени компиляции, поскольку необходимо перекомпилировать большие объемы кода, даже если изменения были минимальны.

В качестве примера разберем результаты десяти сборок инкрементальной компиляции.

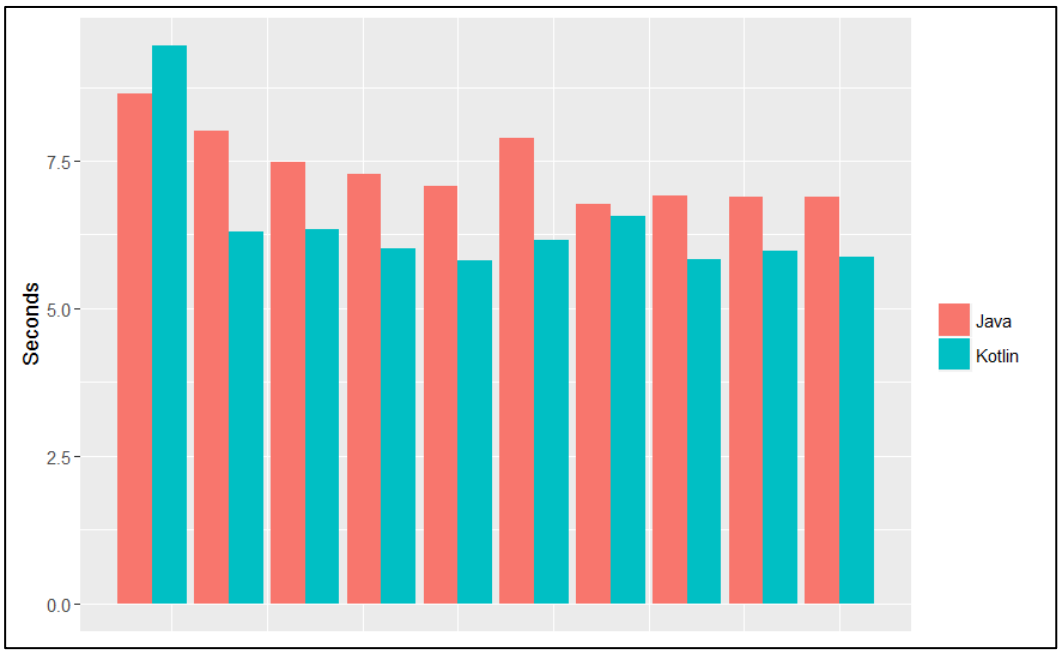


*Рисунок 1.1. Сравнение времени сборки десяти проектов*

Как видно из диаграммы Kotlin показывает немного большее время выполнения при первом прогоне, в остальных случаях Kotlin превосходит Java.

4. Интероперабельность:

Kotlin демонстрирует полную совместимость с Java, обеспечивая легкость в интеграции Java-библиотек и фреймворков, а также плавный переход от Java-кода к Kotlin. Это позволяет разработчикам эффективно использовать существующие ресурсы и постепенно адаптировать проекты к Kotlin.

Хотя Java может быть интегрирована с Kotlin, переход к последнему может предложить более упрощенные и мощные подходы в разработке. Java, будучи классическим и широко используемым языком, сохраняет свои преимущества благодаря обширному опыту, развитой экосистеме и строгой статической типизации, что делает его подходящим для крупных и сложных проектов. В то же время, Kotlin предлагает доступ к современным инструментам и фреймворкам, специфичным для мобильной разработки, включая Jetpack Compose, Koin, RxKotlin, Coroutines и другие.

Таким образом, при выборе между Java и Kotlin для мобильной разработки, Kotlin выделяется как более современный вариант. Его лаконичный синтаксис, безопасность типов и мощные инструменты обеспечивают идеальные условия

для создания качественных и читаемых мобильных приложений, особенно для платформы Android. Следовательно, Kotlin представляет собой предпочтительный выбор в контексте мобильной разработки.

### *1.1.2. Синтаксис Kotlin*

Синтаксис Kotlin отличается простотой и читаемостью, делая его привлекательным для разработчиков, уже знакомых с Java и другими современными языками программирования. Основные аспекты синтаксиса Kotlin включают:

1. Объявление переменных: В Kotlin используются ключевые слова val для объявления неизменяемых переменных (аналогично final в Java) и var для изменяемых. Одна из ключевых особенностей Kotlin — это возможность опускать явное указание типа данных при объявлении переменных.

Листинг 1.4. Объявление переменных в коде

```
val name1: String = "Kotlin" // Неизменяемая переменная
var version1: Int = 1 // Изменяемая переменная
val name2 = "Kotlin" // Тип String выдается автоматически
var version2 = 1 // Тип Int выдается автоматически
```

2. В Kotlin есть несколько видов циклов, среди которых наиболее часто используются for, while и do-while. Разберем каждый из них:

Цикл for в Kotlin часто используется для итерации по диапазонам, массивам, коллекциям и другим итерируемым объектам. Пример итерации по диапазону чисел от 1 до 100:

Листинг 1.5. Пример использования цикла for

```
for (i in 1..100) {
    println("Значение счетчика: $i")
}
```

Цикл while выполняется, пока условие истинно, например:

Листинг 1.6. Пример использования цикла for

```
var x = 5
while (x > 0) {
    println(x)
    x-- // Уменьшаем x на единицу после каждой итерации
}
```

Цикл do-while похож на while, но он гарантированно выполнится хотя бы один раз, поскольку условие проверяется после выполнения тела цикла. Пример:

Листинг 1.7. Пример использования цикла do-while

```
var y = 5
do {
    println(y)
    y--
} while (y > 0)
```

3. Функции объявляются с помощью ключевого слова fun. Kotlin поддерживает функции верхнего уровня, то есть не требует создания класса для определения функции.

Листинг 1.8. Пример создания функций

```
// Функция возвращающая значение
fun hello(name: String): String
// Явное указание возвращаемого типа данных
{
    return "Hello, $name!"
}

// Функция, которая ничего не возвращает
fun printMessageShort(message: String) {
    println(message)
}
```

4. Классы в Kotlin объявляются через ключевое слово class и могут содержать свойства, методы, конструкторы и многое другое.

Листинг 1.9. Пример создания класса

```
class Book {
    // Свойства класса
    var title: String
    var author: String
    var yearPublished: Int

    // Основной конструктор
    constructor(title: String, author: String, yearPublished: Int)
{
        this.title = title
        this.author = author
```

```
        this.yearPublished = yearPublished
    }

    // Метод для отображения информации о книге
    fun displayInfo() {
        println("Книга: \"$title\", Автор: $author, Год издания:
$yearPublished")
    }

    // Метод для обновления года издания
    fun updateYear(newYear: Int) {
        yearPublished = newYear
    }
}
```

Листинг 1.10. Создание объекта класса Book

```
fun main() {
    // Создаем объект класса Book
    val myBook = Book("Преступление и наказание", "Ф.М.
Достоевский", 1866)
    // Отображаем информацию о книге
    myBook.displayInfo()

    // Обновляем год издания с помощью метода updateYear
    myBook.updateYear(2021)
    myBook.displayInfo()
}
```

## Задание

Используя знания о переменных, циклах, условных операторах, классах, методах и конструкторах в Kotlin создать приложение, которое помогает пользователю отслеживать его личные расходы. Программа должна отвечать следующим требованиям:

1. Наличие класса, содержащего информацию о расходах (сумма расхода, категория, дата). Класс должен содержать метод, выводящий информацию о конкретном расходе.
2. Наличие класса, содержащего информацию о списке всех расходов. Класс должен содержать метод добавления нового расхода в список, метод вывода всех расходов, а также метод подсчета суммы всех расходов по каждой категории.