Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Реферат

«Null-безопасность в Kotlin»

Группа: P3131

Выполнил:
Воронин И.А.

Проверил:
Гаврилов А.В.

Санкт-Петербург
2022г

Вкратце про Kotlin

Kotlin — статически сильно типизированный, объектно-ориентированный язык программирования, работающий поверх JVM и разрабатываемый компанией JetBrains. Официально признан Google как приоритетный язык под Android разработку.

Null-безопасность

В Java (предок Котлина) одной из наиболее часто встречаемых ошибок является ссылка на несуществующий(неинициализированный) объект aka NullPointerException.

Для примера возьмем класс Guy:

```
public class Guy {
    private String name;
    private Guy friend;

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

    public void printFriend(){
        System.out.println( name + " has a friend, his name is "
        + friend.name );
    }

    public void setFriend(Guy friend){
        this.friend = friend;
    }

}
```

В данном классе у нас есть поле name, которое мы объявляем через конструктор при создании объекта, а так же поле friend для которого создан Setter. Логика класса такова, что имя задается при рождении, а друг может появиться позже, а может и не появиться вообще. Из-за этого можно попасть в следующую ситуацию:

```
Guy guy = new Guy("Name");
guy.printFriend();
```

Так как поле friend не инициализировано, мы получаем NPE:

```
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Cannot read field "name" because "this.friend" is null
    at Guy.printFriend(Guy.java:10)
    at Main.main(Main.java:8)
```

Решается это элементарной проверкой на null, но есть проблема:

Можно забыть проверку на null и узнать о ее необходимости в очень неподходящий момент (на защите лабораторной, например)

И тут в дело вступает Котлин!

Котлин различает ссылки, которые могут указывать на null и которые не могут. Именно благодаря этому получается предотвратить NPE еще на этапе компиляции.

```
class KotlinGuy(
    var name: String //Конструктор
){
    var friend: KotlinGuy //Не скомпилируется! Нужно инициализировать.

    fun printFriend(){
        println("$name has a friend, his name is ${friend.name}");
    }

}
```

Оператор **?**

В случае с переменной указывает что она может быть null

В случае с методом происходит safe-call (вызывает метод если объект не null, иначе возвращает null без NPE).

Так же используется чтобы указать что метод может вернуть null

Перепишем наш класс:

Вызов name не требует оператора, так как точно будет инициализирована в конструкторе не null значением, friend же из-за оператора в декларации может ссылаться на null, из-за чего нуждается в safe-call.

```
class KotlinGuy(
    var name: String //Точно не будет null
){
    var friend: KotlinGuy? = null //Из-за `?` может быть null

    fun printFriend(){
        println("$name has a friend, his name is ${friend?.name}");
    }

}
```

Если поле friend является null, метод выведет:

«Name has a friend, his name is **null**»

Но что делать, если очень хочется поймать NullPointerException?

Для этого существует оператор **!!** , используя его программист как бы говорит: «Я уверен, что этот объект не null! Даже не проверяй!» и перекладывает всю ответственность на себя.

Используется следующим образом:

```
class KotlinGuy(
    var name: String
){
    var friend: KotlinGuy? = null

    fun printFriend(){
        println("$name has a friend, his name is ${friend!!.name}");
    }

}
```

Результат:

```
Exception in thread "main" java.lang.NullPointerException Create breakpoint
    at KotlinGuy.printFriend(KotlinGuy.kt:7)
    at Main.main(Main.java:9)
```

Заключение

Kotlin – это здорово! Null-безопасность позволяет программисту сразу писать рабочий код, минуя(почти) этап выявления ошибок в ходе работы программы. Советую всем, кто имел дело с Java попробовать его и испытать вышеизложенные функции!

При написании реферата использовалась документация Kotlin:
https://kotlinlang.org/docs