**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное

учреждение высшего образования

**Национальный исследовательский**

**Томский политехнический университет**

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Отчет по лабораторной работе №2 по дисциплине

**«Язык Kotlin и основы разработки»**

Имитатор телефонной станции

****

Выполнил:

Студент группы 1А22 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.К. Кравцов

Проверил:

Ст. преп. ОИТ ИШИТР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Дорофеев

Томск 2024

# Задание

Напишите программу, которая имитирует работу телефонной станции. В программе должно быть два класса:

* Abonent – содержит поля с именем абонента и его номером, а также список входящих и исходящих звонков
* Station – содержит список абонентов и два метода:
* call(from: String, to: String) – ищет абонентов с именами from и to, и если они найдены, то первому добавляет в журнал вызовов строку «Исходящий к <…>», где вместо <…> подставляется имя абонента to, а второму добавляет в журнал вызовов строку «Входящий от <…>», где вместо <…> подставляется имя абонента from.
* showStat() – выводит полные журналы звонков каждого абонента в формате:

Журнал звонков абонента <…>:

Входящий от <…>

Входящий от <…>

Исходящий от <…>

Журнал звонков абонента <…>:

Исходящий от <…>

Исходящий от <…>

В функции main следует создать объект Station, добавить в него несколько абонентов, несколько раз вызвать функцию call() для имитации звонков между разными абонентами, а затем вызвать функцию showStat() для отображения журнала звонков.

# Текст программы

class Abonent(val name: String, val number: String) {

    val callLogs: MutableList<String> = mutableListOf()

}

class Station {

    val abonents: MutableList<Abonent> = mutableListOf()

    fun addAbonent(abonent: Abonent) {

        abonents.add(abonent)

    }

    fun call(from: String, to: String) {

        val fromAbonent = abonents.find { it.name == from }

        val toAbonent = abonents.find { it.name == to }

        if (fromAbonent != null && toAbonent != null) {

            fromAbonent.callLogs.add("Исходящий к ${toAbonent.name}")

            toAbonent.callLogs.add("Входящий от ${fromAbonent.name}")

        }

    }

    fun showStat() {

        for (abonent in abonents) {

            println("Журнал звонков абонента ${abonent.name}:")

            abonent.callLogs.forEach { println("    $it") }

            println()

        }

    }

}

fun main() {

    val station = Station()

    station.addAbonent(Abonent("Иван", "001"))

    station.addAbonent(Abonent("Ольга", "002"))

    station.addAbonent(Abonent("Сергей", "003"))

    station.call("Иван", "Ольга")

    station.call("Ольга", "Сергей")

    station.call("Сергей", "Иван")

    station.call("Иван", "Сергей")

    station.call("Ольга", "Иван")

    station.showStat()

}

# Результаты работы

Журнал звонков абонента Иван:  
 Исходящий к Ольга  
 Входящий от Сергей  
 Исходящий к Сергей  
 Входящий от Ольга  
  
Журнал звонков абонента Ольга:  
 Входящий от Иван  
 Исходящий к Сергей  
 Исходящий к Иван  
  
Журнал звонков абонента Сергей:  
 Входящий от Ольга  
 Исходящий к Иван  
 Входящий от Иван

# Выводы

В ходе выполнения лабораторной работы была разработана программа, имитирующая работу телефонной станции на языке Kotlin. Программа успешно реализует основные функции телефонной станции, такие как регистрация абонентов, осуществление звонков между ними и ведение журнала вызовов для каждого абонента.

В процессе разработки были использованы основные концепции объектно-ориентированного программирования, такие как классы и объекты. Были созданы классы Abonent и Station с необходимыми полями и методами. Класс Abonent хранит информацию об имени и номере абонента, а также список его звонков. Класс Station управляет списком абонентов и реализует логику осуществления звонков и отображения статистики.

Лабораторная работа позволила закрепить навыки работы с коллекциями в Kotlin, такими как MutableList, а также практически применить механизмы поиска элементов в коллекции с помощью функции find.

Разработанная программа демонстрирует эффективное использование ООП для моделирования реальных систем и подтверждает возможности языка Kotlin для создания прикладных приложений. Полученный опыт будет полезен для дальнейшего изучения разработки программного обеспечения на Kotlin.

# Ссылка на код для проверки

<https://pl.kotl.in/92c0vmqtJ>