Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

Информатики и радиоэлектроники»

Учебная дисциплина: «Программирование мобильных информационных систем»

Отчет

по выполнению лабораторной работы «Объектно-ориентированное программирование (ООП)»

Выполнил студент: гр. 210101 Лисовский О.Ю.

Проверил: Усенко Ф.В.

Минск 2024

**1. Цель работы:** изучить принципы объектно-ориентированного программирования (ООП) в Kotlin, включая классы, объекты, наследование, полиморфизм, интерфейсы и абстрактные классы. Научиться создавать и использовать собственные классы, а также применять принципы ООП на практике.

**2. Задание. Комплексная система уведомлений:** Напишите систему уведомлений, которая поддерживает несколько каналов связи (например, Email, SMS, Push-уведомления). Создайте классы Notification, EmailNotification, SMSNotification, и PushNotification, и реализуйте методы для отправки уведомлений по разным каналам в зависимости от предпочтений пользователя.

**3. Оснащение работы:** техническое задание, технический проект, ЭВМ, IntelliJ IDEA.

import java.util.Scanner

interface Notification {

fun send(message: String, recipient: String)

}

class EmailNotification : Notification {

override fun send(message: String, recipient: String) {

println("Отправка Email на $recipient: $message")

}

}

class SMSNotification : Notification {

override fun send(message: String, recipient: String) {

println("Отправка SMS на $recipient: $message")

}

}

class PushNotification : Notification {

override fun send(message: String, recipient: String) {

println("Отправка Push уведомления на $recipient: $message")

}

}

data class UserPreferences(

val prefersEmail: Boolean = false,

val prefersSMS: Boolean = false,

val prefersPush: Boolean = false

)

data class User(

val name: String,

val email: String,

val phoneNumber: String,

val pushId: String,

val preferences: UserPreferences

)

class NotificationSystem {

private val emailNotification = EmailNotification()

private val smsNotification = SMSNotification()

private val pushNotification = PushNotification()

fun sendNotification(user: User, message: String) {

when {

user.preferences.prefersEmail -> emailNotification.send(message, user.email)

user.preferences.prefersSMS -> smsNotification.send(message, user.phoneNumber)

user.preferences.prefersPush -> pushNotification.send(message, user.pushId)

else -> println("Нет предпочитаемого метода уведомлений для ${user.name}. Уведомление не отправлено.")

}

}

fun sendNotificationManually(user: User, message: String, method: String) {

when (method.lowercase()) {

"email" -> {

if (user.preferences.prefersEmail) {

emailNotification.send(message, user.email)

} else {

println("Пользователь ${user.name} не предпочитает уведомления по Email.")

}

}

"sms" -> {

if (user.preferences.prefersSMS) {

smsNotification.send(message, user.phoneNumber)

} else {

println("Пользователь ${user.name} не предпочитает уведомления по SMS.")

}

}

"push" -> {

if (user.preferences.prefersPush) {

pushNotification.send(message, user.pushId)

} else {

println("Пользователь ${user.name} не предпочитает уведомления по Push.")

}

}

else -> println("Неверный выбранный метод.")

}

}

}

fun main() {

val scanner = Scanner(System.`in`)

val notificationSystem = NotificationSystem()

val user1 = User(

name = "Александр",

email = "alex@example.com",

phoneNumber = "+375293172538",

pushId = "alex\_push\_id",

preferences = UserPreferences(prefersEmail = true)

)

val user2 = User(

name = "Василий",

email = "vas@example.com",

phoneNumber = "+375296266348",

pushId = "vas\_push\_id",

preferences = UserPreferences(prefersSMS = true)

)

println("Введите сообщение для уведомления:")

val message = scanner.nextLine()

while (true) {

println("\nВыберите опцию:")

println("1. Отправить уведомление Александру автоматически")

println("2. Отправить уведомление Василию автоматически")

println("3. Отправить уведомление Александру вручную (выбор метода)")

println("4. Отправить уведомление Василию вручную (выбор метода)")

println("5. Выйти")

when (scanner.nextLine()) {

"1" -> notificationSystem.sendNotification(user1, message)

"2" -> notificationSystem.sendNotification(user2, message)

"3" -> {

println("Выберите метод для Александра (email, sms, push):")

val method = scanner.nextLine()

notificationSystem.sendNotificationManually(user1, message, method)

}

"4" -> {

println("Выберите метод для Василия (email, sms, push):")

val method = scanner.nextLine()

notificationSystem.sendNotificationManually(user2, message, method)

}

"5" -> {

println("Выход из программы.")

break

}

else -> println("Неверная опция. Пожалуйста, попробуйте снова.")

}

}

}

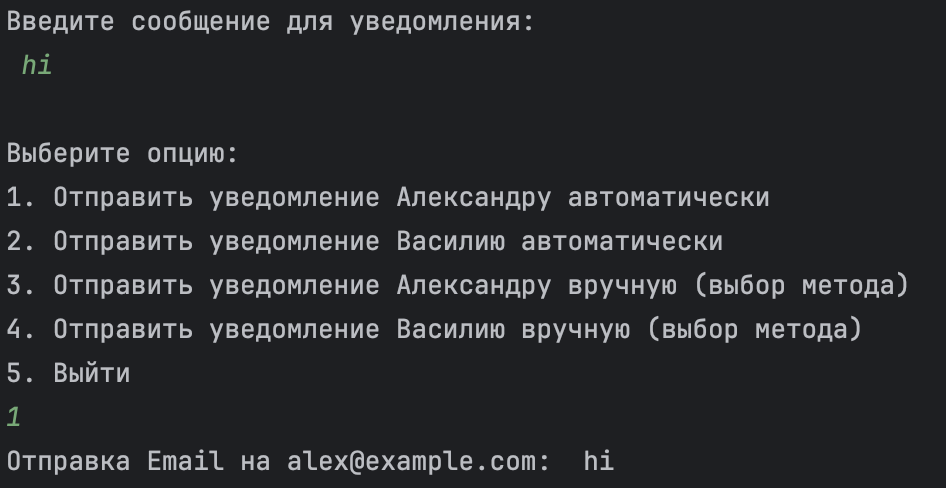


Рисунок 1 – Результат исполнения кода программы

**Вывод:** в ходе выполнения данной лабораторной работы была разработана многомодульная система управления пользователями на языке программирования Kotlin. В системе были реализованы классы для различных ролей: User, Admin, Guest, Moderator. Каждый класс обладает различными уровнями доступа и правами, которые зависят от роли пользователя. Администратор (Admin) имеет расширенные права, в том числе возможность динамически изменять роли других пользователей.