Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Современные языки программирования

**Отчёт**

по выполнению лабораторной работы №1

на тему

«Использование языка программирования *Kotlin*»

Выполнил студент группы 310101

Сердюк У.В.

Проверил:

Усенко Ф. В.

Минск

2024

**Цель:** Выполнить разработку приложения с использованием языка программирования *Kotlin*.

*Kotlin* представляет современный, статически типизированный и один из самых быстроразвивающихся языков программирования, созданный и развиваемый компанией *JetBrains*. *Kotlin* можно использовать для создания самых разных приложений. Это и приложения для мобильных устройств - *Android*, *iOS*. Причем *Kotlin* позволяет писать кроссплатформенный код, который будет применяться на всех платформах. Это и веб-приложения, причем как серверные приложения, которые отрабатывают на стороне на стороне сервера − бекэнда, так и браузерные клиентские приложения − фронтенд. *Kotlin* также можно применять для создания десктопных приложений, для *Data Science* и так далее.

**Задание 1**

Добавить класс База пользователей, хранящий данные о Зарегистрированных пользователях. Обеспечить управление этим классом с помощью класса Администратор.

Листинг кода класса *main*:

import java.util.Scanner

fun main(){

val scanner = Scanner(System.`in`)

val forum = Forum("Любительский форум")

val userBase = UserBase()

println("Введите логин администратора")

val adminLogin = scanner.nextLine()

println("Введите пароль администратора")

val adminPassword = scanner.nextLine()

val admin = Administrator(adminLogin,adminPassword)

admin.login(adminLogin, adminPassword)

var exit = false

while(!exit){

println("Выберите действие:")

println("1. Создать сообщение")

println("2. Получить список сообщений")

println("3. Удалить сообщение")

println("4. Сменить логин")

println("5. Добавить пользователя")

println("6. Удалить пользователя

println("7. Выйти")

val choice = scanner.nextLine()

when(choice){

"1" -> {

// Создать сообщение

if (admin.isLogged()) { // Проверяем, что администратор залогинен

println("Введите текст сообщения:")

val messageText = scanner.nextLine()

println("Введите изображение для сообщения:")

val imagePath = scanner.nextLine()

admin.newMessage(messageText, imagePath, forum)

println("Сообщение создано.")

} else {

println("Администратор не залогинен!")

}

}

"2" -> {

// Получить список сообщений

val messages = forum.getMessages()

if (messages.isNotEmpty()) {

println("Список сообщений:")

messages.forEach { message ->

println("Сообщение: ${message.line}, Изображение: ${message.image}")

}

} else {

println("Сообщений нет.")

}

}

"3" -> {

// Удалить сообщение

println("Введите текст сообщения для удаления:")

val messageText = scanner.nextLine()

val messageToDelete = forum.getMessages().find { it.line == messageText }

if (messageToDelete != null) {

admin.deleteMessage(forum, messageToDelete)

println("Сообщение удалено.")

} else {

println("Сообщение не найдено.")

}

}

"4" -> {

// Сменить логин

println("Введите новый логин администратора:")

val newLogin = scanner.nextLine()

admin.changeUserLogin(admin, newLogin)

}

"5" -> {

// Добавить пользователя

println("Введите логин нового пользователя:")

val userLogin = scanner.nextLine()

println("Введите пароль нового пользователя:")

val userPassword = scanner.nextLine()

val newUser = RegisteredUser(userLogin, userPassword)

admin.addUserRequest(userBase, newUser)

}

// Удалить пользователя

"6"->{

println("Вы точно хотите удалить пользователя?\n1 - да\n2 - нет")

val ifdelUser = scanner.nextLine()

when(ifdelUser){

"1"-> {

println("Введите логин пользователя для удаления:")

val userLogin = scanner.nextLine()

val userToDelete =

userBase.getUsers().find { it.getLogin() == userLogin } // Поиск пользователя по логину

if (userToDelete != null) {

admin.deleteUserRequest(userBase, userToDelete) // Удаляем пользователя

} else {

println("Пользователь с логином '$userLogin' не найден.")

}

}

"2"->{

println("Пользователь не был удалён.")

}

}

}

"7" -> {

// Выйти

exit = true

println("Выход из программы.")

}

else -> {

println("Неверная команда.")

}

}

}

}

Листинг кода класса *Administrator*, *UserBase*:

class Administrator(

username: String,

password: String

) : RegisteredUser(

username,

password

) {

fun deleteMessage(forum: Forum, message: Message) {

val messages = forum.getMessages().toMutableList()

if (messages.contains(message)) {

messages.remove(message)

println("Сообщение '${message.line}' с изображением '${message.image}' было удалено с форума '${forum.getName()}' администратором ${getLogin()}.")

// Обновляем оригинальный список сообщений

forum.setMessages(messages) // Добавьте метод setMessages в класс Forum

} else {

println("Сообщение не найдено на форуме.")

}

}

fun deleteUserRequest(userBase: UserBase, user: RegisteredUser) {

userBase.receiveDeleteRequest(this, user)

}

fun addUserRequest(userBase: UserBase, user: RegisteredUser) {

userBase.receiveAddRequest(this, user)

println("Пользователь '${user.getLogin()}' был добавлен администратором ${getLogin()}.")

}

fun changeUserLogin(user: RegisteredUser, newUsername: String) {

user.changeLogin(newUsername)

println("Логин пользователя был изменён на $newUsername администратором ${getLogin()}.")

}

}

class UserBase(){

private val users: MutableList<User> = mutableListOf()

private fun addUser(user: RegisteredUser){

if (!users.contains(user)){

users.add(user)

}

else{

println("Пользователь ${user.getLogin()} уже есть в базе.")

}

}

private fun delUser(user: RegisteredUser){

if(users.contains(user)){

users.remove(user)

println("Пользователь ${user.getLogin()} был удалён из базы.")

}

else{

println("Пользователь ${user.getLogin()} не был найден в базе.")

}

}

fun getUsers():List<RegisteredUser>{

return users.filterIsInstance<RegisteredUser>() // Фильтруем только RegisteredUser

}

fun receiveAddRequest(requester: User, user: RegisteredUser) {

if (requester is Administrator) {

println("Администратор ${requester.getLogin()} сделал запрос на добавление пользователя ${user.getLogin()}.")

addUser(user)

} else {

println("Запрос отклонён. Только администратор может создавать запросы.")

}

}

fun receiveDeleteRequest(requester: User, user: RegisteredUser) {

if (requester is Administrator) {

println("Администратор ${requester.getLogin()} сделал запрос на удаление пользователя ${user.getLogin()}.")

delUser(user)

} else {

println("Запрос отклонён. Только администратор может создавать запросы.")

}

}

}

Скриншоты работающих программ:

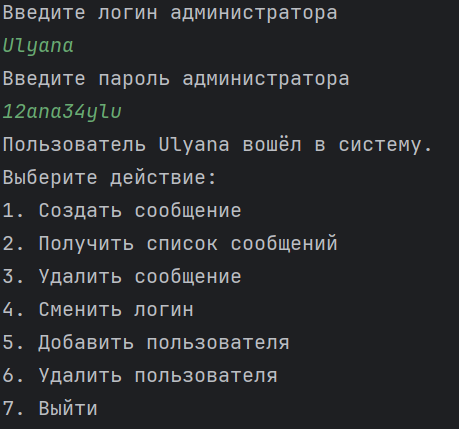


Рисунок 1 ‒ Администратор входит в систему

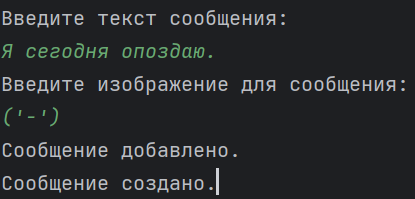


Рисунок 2 ‒ Создание сообщения

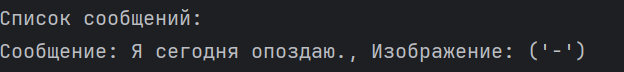


Рисунок 3 ‒ Вывод сообщений

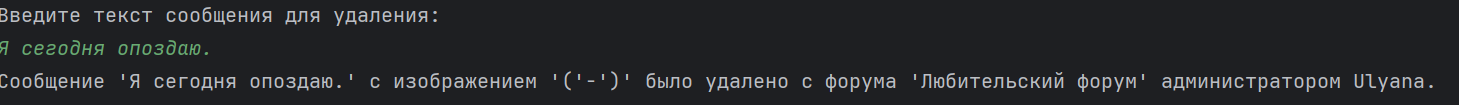


Рисунок 4 ‒ Удаление сообщения

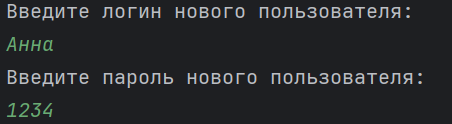


Рисунок 5 ‒ Добавление нового пользователя

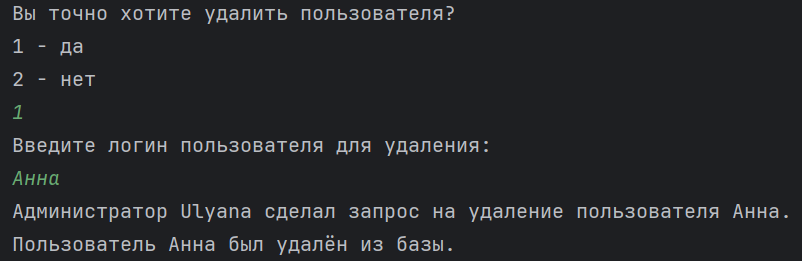


Рисунок 6 ‒ Удаление пользователя

**Ответы на вопросы**

1. Что такое JVM?

*Java Virtual Machine (JVM)* ‒ это виртуальная машина, которая исполняет байт-код. Она предоставляет среду выполнения для приложений, написанных на *Java* или других языках, совместимых с *JVM* (например, *Kotlin*). *JVM* отвечает за управление памятью, многопоточность, выполнение байт-кода и обеспечивает независимость от платформы.

2. Как объявить переменную в *Kotlin*? Различие между *var* и *val*.  
Переменные объявляются через ключевые слова *var* и *val*:

− *var* − изменяемая переменная.

− *val* − неизменяемая переменная (аналог константы).

3. Пакеты в *Kotlin*?

Пакеты используются для организации кода. Исходный файл может начинаться с объявления пакета (*package org.example*), что позволяет организовывать классы и функции в именованные пространства. При отсутствии явного пакета используется безымянный пакет по умолчанию.

**Вывод:** Была изучена основа программирования на языке Kotlin. Были рассмотрены ключевые аспекты языка, включая организацию пакетов, объявление переменных с использованием var и val, работу с классами, конструкторами, модификаторами доступа и механизмами наследования. Особое внимание уделено функциональным возможностям Kotlin, таким как лямбда-выражения, функциональные типы данных.