Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Отчет

по практической работе №4

на тему:

**ОБЪЕКТО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ**

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Усенко Ф.В.

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Папко К.А.

(подпись)  гр. 214301

Минск, 2024

**Задание:** Напишите многомодульную систему, которая управляет пользователями в рамках крупного проекта. Реализуйте классы для различных ролей (User, Admin, Guest, Moderator) с разными уровнями доступа и возможностью динамически изменять права.

1. ВЫПОЛНЕНИЕ ЗАДАНИЯ

import user\_management.User  
import user\_management.UserManager  
import roles.\*  
  
fun main() {  
 val userManager = UserManager()  
  
 val admin = User(id = 1, name = "Alice", role = Admin())  
 val moderator = User(id = 2, name = "Bob", role = Moderator())  
 val guest = User(id = 3, name = "Charlie", role = Guest())  
 val user = User(id = 4, name = "David", role = User())  
  
 userManager.addUser(admin)  
 userManager.addUser(moderator)  
 userManager.addUser(guest)  
 userManager.addUser(user)  
  
 while (true) {  
 println("Выберите пользователя для входа:")  
 userManager.getAllUsers().forEach **{** println("${it.id}. ${it.name} - ${it.role.roleName}") **}** val selectedUserId = readLine()?.toIntOrNull()  
 val selectedUser = selectedUserId?.let **{** userManager.getUserById(it) **}** if (selectedUser != null) {  
 println("Вы вошли как ${selectedUser.name} с ролью ${selectedUser.role.roleName}")  
  
 when (selectedUser.role) {  
 is Admin -> adminMenu(selectedUser, userManager)  
 is Moderator -> moderatorMenu(selectedUser, userManager)  
 is Guest -> guestMenu(selectedUser)  
 is roles.User -> userMenu(selectedUser)  
 }  
 } else {  
 println("Пользователь не найден!")  
 }  
 }  
}  
  
fun adminMenu(user: User, userManager: UserManager) {  
 while (true) {  
 println("Меню администратора:")  
 println("1. Создать пользователя")  
 println("2. Удалить пользователя")  
 println("3. Изменить роль пользователя")  
 println("4. Просмотреть всех пользователей")  
 println("5. Выйти")  
  
 when (readLine()?.toIntOrNull()) {  
 1 -> {  
 print("Введите имя нового пользователя: ")  
 val name = readLine()  
 val newUser = User(id = userManager.getAllUsers().size + 1, name = name ?: "Unnamed", role = User())  
 userManager.addUser(newUser)  
 println("Пользователь ${newUser.name} создан.")  
 }  
 2 -> {  
 print("Введите ID пользователя для удаления: ")  
 val userId = readLine()?.toIntOrNull()  
 val userToDelete = userManager.getUserById(userId ?: -1)  
 if (userToDelete != null) {  
 userManager.removeUser(userToDelete)  
 println("Пользователь ${userToDelete.name} удален.")  
 } else {  
 println("Пользователь с таким ID не найден.")  
 }  
 }  
 3 -> {  
 print("Введите ID пользователя для изменения роли: ")  
 val userId = readLine()?.toIntOrNull()  
 val userToModify = userManager.getUserById(userId ?: -1)  
 if (userToModify != null) {  
 println("Выберите новую роль: 1. Admin 2. Moderator 3. User")  
 val roleChoice = readLine()?.toIntOrNull()  
 val newRole = when (roleChoice) {  
 1 -> Admin()  
 2 -> Moderator()  
 3 -> User()  
 else -> {  
 println("Неверный выбор.")  
 return  
 }  
 }  
 userManager.changeUserRole(userToModify, newRole)  
 println("Роль пользователя ${userToModify.name} изменена на ${newRole.roleName}.")  
 } else {  
 println("Пользователь с таким ID не найден.")  
 }  
 }  
 4 -> {  
 println("Список всех пользователей:")  
 userManager.getAllUsers().forEach **{** println("${it.id}. ${it.name} - ${it.role.roleName}") **}** }  
 5 -> {  
 println("Выход из меню администратора.")  
 return  
 }  
 else -> println("Неверный выбор!")  
 }  
 }  
}  
  
fun moderatorMenu(user: User, userManager: UserManager) {  
 while (true) {  
 println("Меню модератора:")  
 println("1. Просмотреть всех пользователей")  
 println("2. Изменить данные пользователя")  
 println("3. Выйти")  
  
 when (readLine()?.toIntOrNull()) {  
 1 -> {  
 println("Список всех пользователей:")  
 userManager.getAllUsers().forEach **{** println("${it.id}. ${it.name} - ${it.role.roleName}") **}** }  
 2 -> {  
 print("Введите ID пользователя для изменения данных: ")  
 val userId = readLine()?.toIntOrNull()  
 val userToModify = userManager.getUserById(userId ?: -1)  
 if (userToModify != null) {  
 println("Выберите новое имя для пользователя ${userToModify.name}:")  
 val newName = readLine()  
 if (!newName.isNullOrEmpty()) {  
 userToModify.name = newName  
 println("Имя пользователя изменено на ${newName}.")  
 }  
 } else {  
 println("Пользователь с таким ID не найден.")  
 }  
 }  
 3 -> {  
 println("Выход из меню модератора.")  
 return  
 }  
 else -> println("Неверный выбор!")  
 }  
 }  
}  
  
fun guestMenu(user: User) {  
 while (true) {  
 println("Меню гостя:")  
 println("1. Просмотреть всех пользователей")  
 println("2. Выйти")  
  
 when (readLine()?.toIntOrNull()) {  
 1 -> {  
 println("Список всех пользователей:")  
 println("${user.name} имеет доступ только для просмотра пользователей.")  
 }  
 2 -> {  
 println("Выход из меню гостя.")  
 return  
 }  
 else -> println("Неверный выбор!")  
 }  
 }  
}  
  
fun userMenu(user: User) {  
 while (true) {  
 println("Меню пользователя:")  
 println("1. Просмотреть свои данные")  
 println("2. Выйти")  
  
 when (readLine()?.toIntOrNull()) {  
 1 -> {  
 println("Ваши данные: ${user.name} - ${user.role.roleName}")  
 }  
 2 -> {  
 println("Выход из меню пользователя.")  
 return  
 }  
 else -> println("Неверный выбор!")  
 }  
 }  
}

1. ОТВЕТЫ НА ВОПРОСЫ
2. Класс в Kotlin — это шаблон для создания объектов, определяющий их свойства и поведение. Он объявляется с использованием ключевого слова class.
3. Объект класса создаётся с использованием оператора () после имени класса.
4. Свойства класса — это переменные, определяющие состояние объекта. Они могут быть изменяемыми (var) или неизменяемыми (val).
5. Метод класса — это функция, определённая внутри класса, которая описывает его поведение. Методы объявляются с использованием ключевого слова fun.
6. Первичный конструктор — это встроенный механизм для определения параметров, необходимых для создания объекта, и инициализации его свойств. Он объявляется в круглых скобках после имени класса.
7. ВЫВОД

В ходе выполнения практической работы №4 была реализована многомодульная система для управления пользователями с различными ролями: администратор, модератор, гость и пользователь.