Министерство образования Республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина «Современные языки программирования»

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема «Использование языка программирования Kotlin»

Вариант № 21

Выполнила: Сикаченко Л.М., гр.310901

Проверил: Усенко Ф.В.

Минск 2024

Задание: Добавить класс Квартира, хранящий в себе класс Адрес и организовать взаимодействие с ним: вселение жильцов в комнату и выселение. Согласно диаграмме классов и диаграмме последовательности, представленных на рисунках 1 и 2.

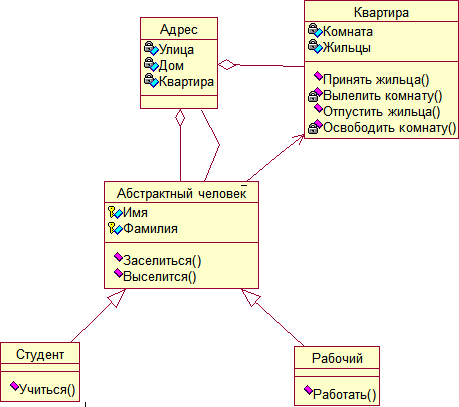


Рисунок 1 – Диаграмма классов

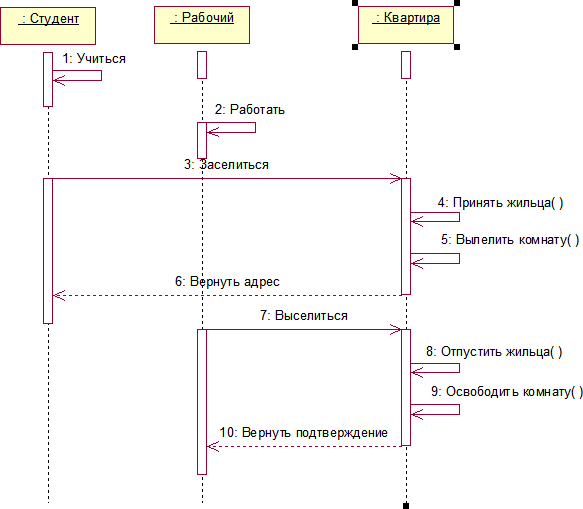


Рисунок 2 – Диаграмма последовательности

package person  
  
import housing.Apartment  
import housing.Address  
  
abstract class Person(protected val firstName: String, protected val lastName: String) {  
  
 abstract fun settle(address: Address, apartment: Apartment): Address?  
  
 abstract fun evict(address: Address, apartment: Apartment, roomNumber: Int): Boolean  
  
 fun getFullName() = "$firstName $lastName"  
}

package person  
  
import housing.Address  
import housing.Apartment  
  
class Student(firstName: String, lastName: String) : Person(firstName, lastName) {  
  
 override fun settle(address: Address, apartment: Apartment): Address? {  
 return apartment.acceptResident(this, address)  
 }  
  
 override fun evict(address: Address, apartment: Apartment, roomNumber: Int): Boolean {  
 return apartment.releaseResident(this, address, roomNumber)  
 }  
  
 fun study() {  
 val range = 1..3  
 for ( i in range) {  
 *println*("${getFullName()} учится...")  
 Thread.sleep(500)  
 }  
 *println*("${getFullName()} завершил изучение.")  
 }  
}

package person  
  
import housing.Address  
import housing.Apartment  
  
class Worker(firstName: String, lastName: String) : Person(firstName, lastName) {  
  
 override fun settle(address: Address, apartment: Apartment): Address? {  
 return apartment.acceptResident(this, address)  
 }  
  
 override fun evict(address: Address, apartment: Apartment, roomNumber: Int): Boolean {  
 return apartment.releaseResident(this, address, roomNumber)  
 }  
  
 fun work() {  
 val range = 1..3  
 for ( i in range) {  
 *println*("${getFullName()} работает...")  
 Thread.sleep(500)  
 }  
 *println*("${getFullName()} завершил работу.")  
 }  
}

package housing  
import person.Person  
  
class Apartment {  
  
 private val residents = *mutableMapOf*<Address, Array<Person?>>()  
 private val rooms: Array<Boolean> = *arrayOf*(false, false, false)  
  
 // Принять жильца  
 fun acceptResident(person: Person, address: Address): Address? {  
 val currentResidents = residents.*getOrPut*(address) **{** *arrayOfNulls*(3) **}** val roomNumber = allocateRoom(currentResidents, person)  
 if (roomNumber != -1 ) {  
 *print*("${person.getFullName()} был заселен в №$roomNumber комнату.")  
 returnAddress(address)  
 return address  
 } else {  
 *println*("Нет свободных комнат по адресу ${address.getFullAddress()}.")  
 return null  
 }  
 }  
  
 // Отпустить жильца  
 fun releaseResident(person: Person, address: Address, roomNumber: Int): Boolean {  
 val currentResidents = residents[address]  
 if (currentResidents != null && roomNumber in 1..3 && currentResidents[roomNumber - 1] == person) {  
 freeRoom(currentResidents, roomNumber - 1)  
 *println*("Жилец ${person.getFullName()} был отпущен из №$roomNumber комнаты по адресу ${address.getFullAddress()}.")  
 return true  
 }  
 *println*("Такого жильца нет.")  
 return false  
 }  
  
 // Выделить комнату  
 private fun allocateRoom(currentResidents: Array<Person?>, person: Person): Int {  
 for (i in currentResidents.*indices*) {  
 if (currentResidents[i] == null) {  
 currentResidents[i] = person  
 *println*("Выделена №${i+1} комната.")  
 rooms[i]=true;  
 return i + 1  
 }  
 }  
 return -1  
 }  
  
 // Освободить комнату  
 private fun freeRoom(currentResidents: Array<Person?>, roomIndex: Int) {  
 currentResidents[roomIndex] = null  
 rooms[roomIndex] = false;  
 getConfirmation(roomIndex);  
 }  
  
 // вернуть адрес  
 fun returnAddress(address: Address) {  
 *println*("по адресу ${address.getFullAddress()}");  
 }  
  
 // Вернуть подтверждение  
 fun getConfirmation(roomIndex: Int){  
 if (!rooms[roomIndex]){  
 *println*("Освобождена №${roomIndex+1} комната.")  
 }  
 }  
}

package housing  
  
data class Address( private val street: String,  
 private val house: Int,  
 private val apartment: Int){  
  
 fun getFullAddress() = "ул.${street}, д.${house}, кв.${apartment}"  
}

import housing.Address  
import housing.Apartment  
import person.Student  
import person.Worker  
  
fun main() {  
 val apartments = *mutableMapOf*<Address, Apartment>()  
 val students = *mutableListOf*<Student>()  
 val workers = *mutableListOf*<Worker>()  
  
 while (true) {  
 *println*("1. Заселиться");  
 *println*("2. Выселиться");  
 *println*("3. Учиться");  
 *println*("4. Работать");  
 *println*("5. Отобразить студентов");  
 *println*("6. Отобразить работников");  
 *println*("7. Выйти");  
 var choice:Number;  
 try{  
 choice= *readln*().*toInt*()  
 }catch(e:Exception){  
 *println*("Неверный ввод")  
 continue  
 }  
 when (choice) {  
 1 -> {  
 *println*("Введите имя:")  
 val firstName = *readln*()  
 *println*("Введите фамилию:")  
 val lastName = *readln*()  
 *println*("Кто вселяется? 1 - Студент, 2 - Работник:")  
 val type = *readln*().*toInt*()  
 *println*("Введите улицу:")  
 val street = *readln*()  
 *println*("Введите дом:")  
 val house = *readln*().*toInt*()  
 *println*("Введите квартиру:")  
 val apartmentNum = *readln*().*toInt*()  
  
 val address = Address(street, house, apartmentNum)  
 val apartment = apartments.*getOrPut*(address) **{** Apartment() **}** when (type) {  
 1 -> {  
 val student = Student(firstName, lastName)  
 student.settle(address, apartment)  
 students.add(student)  
 }  
  
 2 -> {  
 val worker = Worker(firstName, lastName)  
 worker.settle(address, apartment)  
 workers.add(worker)  
 }  
 }  
 }  
  
 2 -> {  
 *println*("Введите улицу:")  
 val street = *readln*()  
 *println*("Введите дом:")  
 val house = *readln*().*toInt*()  
 *println*("Введите квартиру:")  
 val apartmentNum = *readln*().*toInt*()  
 *println*("Введите номер комнаты:")  
 val roomNumber = *readln*().*toInt*()  
  
 val address = Address(street, house, apartmentNum)  
 val apartment = apartments[address]  
  
 if (apartment == null) {  
 *println*("Такого адреса не существует.")  
 continue  
 }  
  
 *println*("Введите имя:")  
 val firstName = *readln*()  
 *println*("Введите фамилию:")  
 val lastName = *readln*()  
  
 val person = (students + workers).*find* **{ it**.getFullName() == "$firstName $lastName" **}** if (person != null) {  
 if (person.evict(address, apartment, roomNumber)) {  
 students.*remove*(person)  
 workers.*remove*(person)  
 }  
 } else {  
 *println*("Жилец не найден.")  
 }  
 }  
  
 3 -> {  
 *println*("Введите имя:")  
 val firstName = *readln*()  
 *println*("Введите фамилию:")  
 val lastName = *readln*()  
  
 val student = students.*find* **{ it**.getFullName() == "$firstName $lastName" **}** student?.study() ?: *println*("Студент с таким именем не найден.")  
 }  
  
 4 -> {  
 *println*("Введите имя:")  
 val firstName = *readln*()  
 *println*("Введите фамилию:")  
 val lastName = *readln*()  
  
 val worker = workers.*find* **{ it**.getFullName() == "$firstName $lastName" **}** worker?.work() ?: *println*("Работник с таким именем не найден.")  
 }  
  
 5 -> {  
 if(students.isEmpty()){  
 *println*("Среди жителей нет ни одного студента.")  
 } else {  
 *println*("Список студентов:")  
 students.*forEach* **{** *println*(**it**.getFullName()) **}** }  
 }  
  
 6 -> {  
 if(workers.isEmpty()){  
 *println*("Среди жителей нет ни одного работника.")  
 } else {  
 *println*("Список работников:")  
 workers.*forEach* **{** *println*(**it**.getFullName()) **}** }  
 }  
  
 7 -> return  
 }  
 }  
}

Результат работы программы представлен на рисунках 4-7.

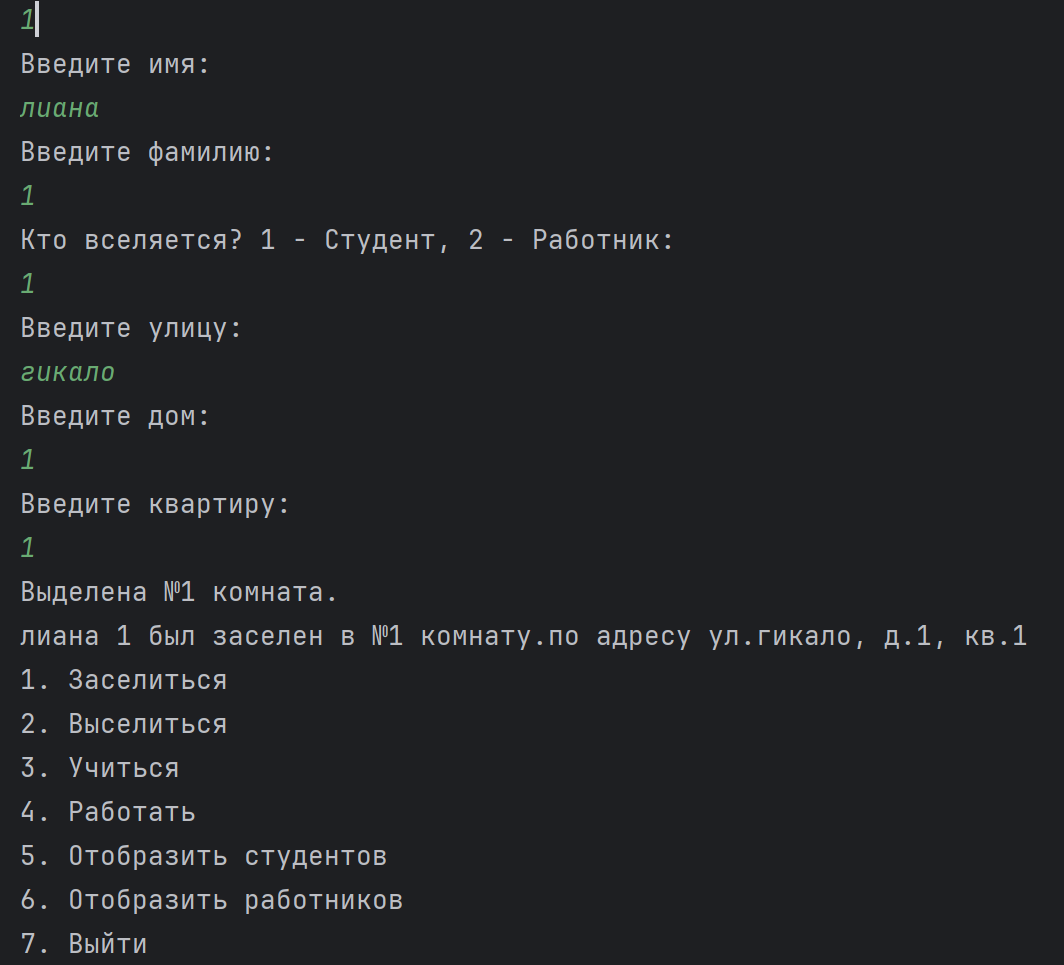
****

Рисунок 4 – Результат работы программы.

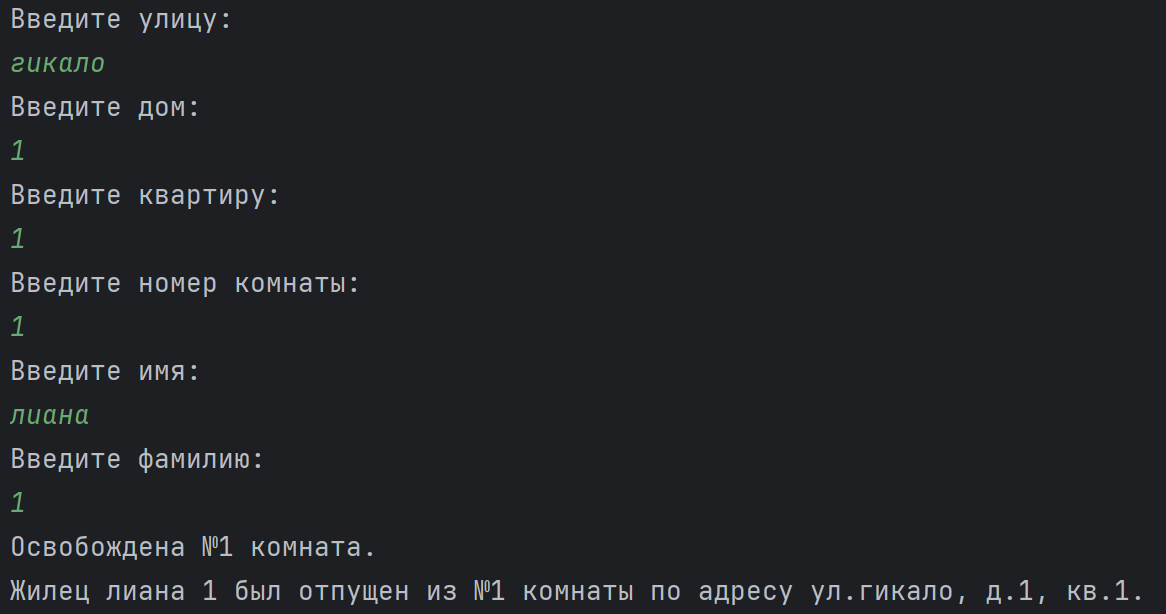
****

Рисунок 5 – Результат работы программы.

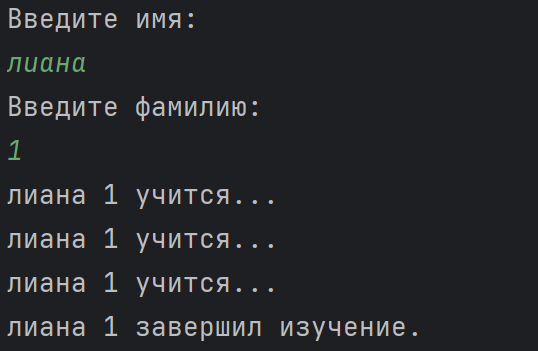
****

Рисунок 6 – Результат работы программы.

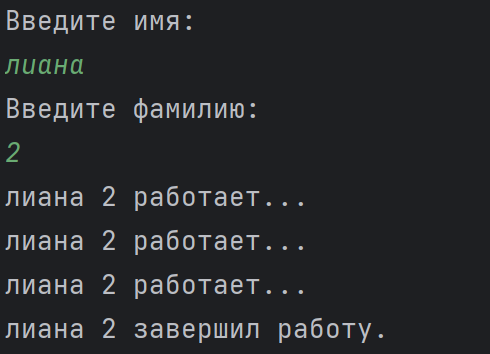
****

Рисунок 7 – Результат работы программы.

Вывод: в ходе лабораторной работы была выполнена разработка приложения с использованием языка программирования Kotlin.