Министерство образования

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Отчет по лабораторной работе №1

«Использование языка программирования Kotlin»

Выполнил: Бельский М.Д.

Студент группы 310902

Проверил: Усенко Ф. В.

Минск 2024

Цель работы – Выполнить разработку приложения с использованием языка программирования Kotlin

Индивидуальное задание

1. Добавить класс журнал и организовать взаимодействие с ним. Чтобы при вызове метода Отметить присутствующих класса Староста в классе Строка журнала обозналось присутствие студента на лекции. А при вызове метода Отметить присутствующих класса Преподаватель эти данные считывались и сравнивались с данными объекта Преподаватель.

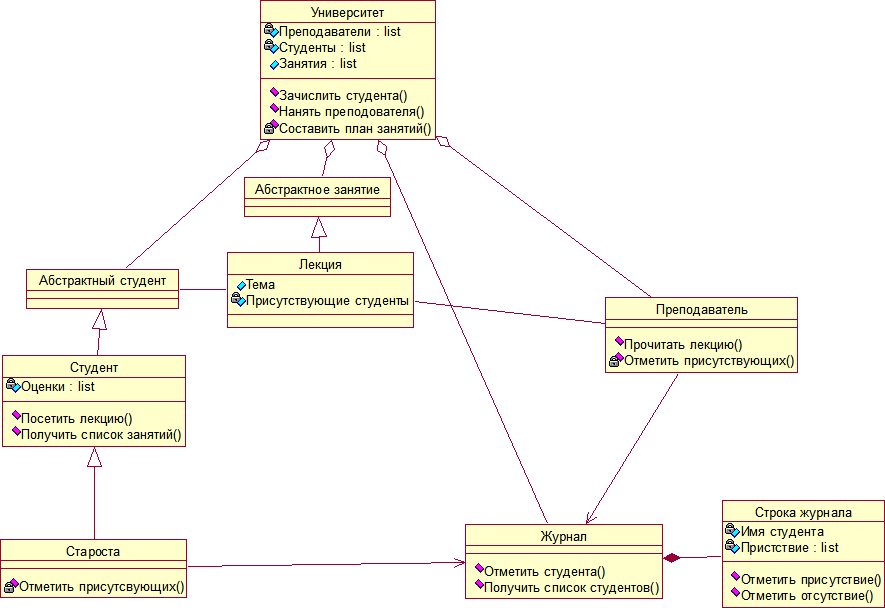


Рисунок 1 – Диаграмма классов

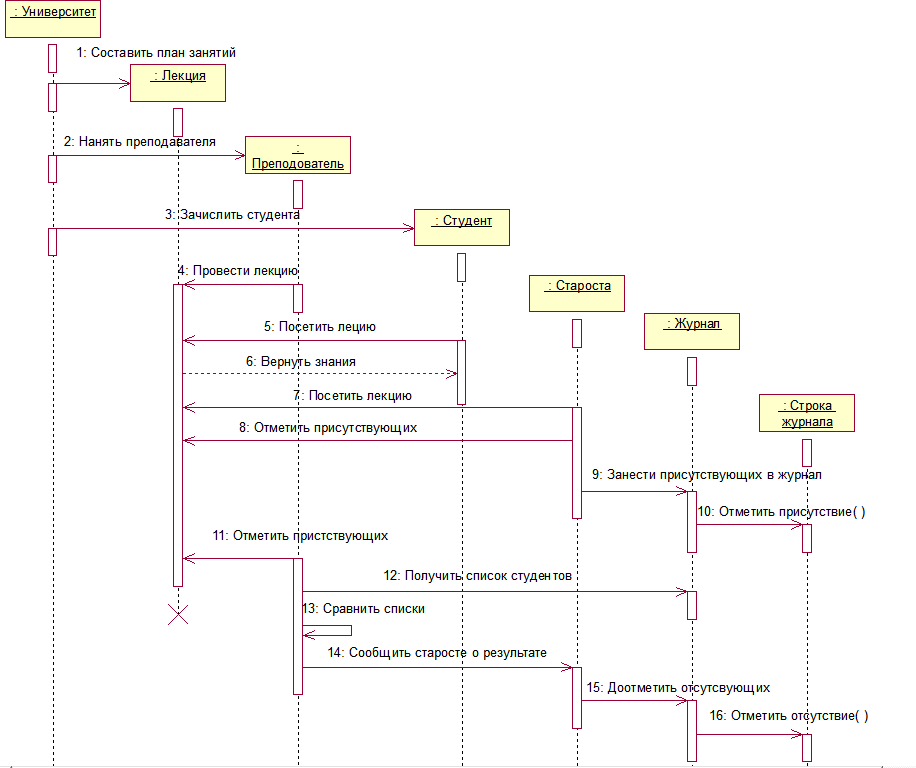


Рисунок 2 – Диаграмма последовательности

Листинг кода:

Class University

package Laba1

class University {

val teachers: MutableList<Teacher> = mutableListOf();

val students: MutableList<Student> = mutableListOf();

val lessons: MutableList<Lesson> = mutableListOf();

fun addStudent() {

for (i in 0..32) {

if (i == 0) {

students.add(Starasta())

continue

}

if (i % 2 == 0) {

students.add(Student("Студент ${i+1}", true))

} else {

students.add(Student("Студент ${i+1}", false))

}

}

println("Студенты зачислены")

}

fun getListLessons() {

println("Список предметов")

for (el in lessons) {

println(el.topic)

}

}

fun addLessons() {

lessons.add(Lecture("Современные языки программирования"))

lessons.add(Lecture("Алгоритмы и структура данных"))

lessons.add(Lecture("Математический анализ"))

lessons.add(Lecture("Конструирование программного обеспечения"))

lessons.add(Lecture("Политэкономия"))

println("Уроки добавлены")

}

fun addTeachers(tech: Teacher) {

teachers.add(tech)

println("Учитель нанят")

}

}

Class Student

package Laba1

open class Student {

open var name: String = ""

open var isPresence: Boolean = false

val marks: MutableList<Int> = mutableListOf()

constructor(\_name: String, \_isPresence: Boolean){

name = \_name

isPresence = \_isPresence

}

constructor()

open fun comeLesson(less: Lesson, university: University) {

for (item in university.students) {

if (item.isPresence) {

println("Студент ${item.name} посетил лекцию на тему ${less.topic}")

}

}

}

}

internal class Starasta : Student {

override var name: String = "Староста"

override var isPresence: Boolean = true

constructor(\_name: String = "Староста", \_isPresence: Boolean = true) : super(\_name, \_isPresence)

fun absent(less: Lesson, stud: List<Student>, journal: Jornal) {

println("Староста $name отметил присутствующих")

stud.forEachIndexed{index, value ->

if(index % 2 ==0) journal.markStudent(stud[index], true)

else journal.markStudent(stud[index], false)

}

}

}

Class Lesson

package Laba1

abstract class Lesson{

internal var topic: String? = null;

constructor(\_topic:String){

topic = \_topic

}

}

internal class Lecture(topic: String) : Lesson(topic) {

val presentStudents: MutableList<Student> = mutableListOf()

constructor() : this("")

fun markPresence(university: University) {

for (student in university.students) {

if (student.isPresence) {

presentStudents.add(student)

}

}

}

fun getPresentStudents() {

for (student in presentStudents) {

println("Присутствующий студент: ${student.name}")

}

}

}

Class Teacher

package Laba1

import kotlin.random.Random

class Teacher {

var name: String = ""

fun readLesson(less: Lesson) {

println("Преподаватель $name прочитал лекцию на тему ${less.topic}")

}

fun markAbsent(less: Lesson, stud: List<Student>, journal: Jornal) {

println("Преподаватель $name отметил отсутствующих на лекции ${less.topic}")

stud.forEachIndexed{index, value ->

if(index % 2 == 0 ) journal.markStudent(stud[index], true)

else journal.markStudent(stud[index], false)

}

}

private fun generateOddNumber(stud: List<Student>): Int {

val maxIndex = stud.size - 1

return 2 \* Random.nextInt(0, maxIndex / 2) + 1

}

fun markUp(less: Lesson, stud: List<Student>, journal: Jornal){

println("Преподаватель $name доотметил отсутствующих на лекции ${less.topic}")

for(i in 0..5){

journal.markStudent(stud[generateOddNumber(stud)], true)

}

}

}

Class Journal

package Laba1

open class StringJournal {

var nameOfStudent: String = ""

val presence: MutableList<Boolean> = mutableListOf()

fun markPresence() {

presence.add(true)

}

fun markAbsent() {

presence.add(false)

}

}

class Jornal : StringJournal() {

val journalEntries: MutableList<StringJournal> = mutableListOf()

fun markStudent(student: Student, isPresent: Boolean) {

val entry = StringJournal().apply {

nameOfStudent = student.name

}

if (isPresent) {

entry.markPresence()

} else {

entry.markAbsent()

}

journalEntries.add(entry)

}

fun getListStudents() {

for (entry in journalEntries) {

println("Студент: ${entry.nameOfStudent}")

}

}

}

Main

package Laba1

import kotlin.random.Random

object StateMenu {

var value: Int = 0

}

fun main(args: Array<String>) {

val university = University()

val teacher = Teacher()

val journal = Jornal()

val lecture = Lecture()

val student = Student()

var Ispresence = false

menu()

while (true) {

when (StateMenu.value) {

0 -> {

return

}

1 -> {

university.addLessons()

println("Нажмите Enter для продолжения")

readLine()

menu()

}

2 -> {

teacher.name = "Иванов И.И."

university.addTeachers(teacher)

println("Нажмите Enter для продолжения")

readLine()

menu()

}

3 -> {

university.addStudent()

println("Нажмите Enter для продолжения")

readLine()

menu()

}

4 -> {

student.comeLesson(lecture, university)

println("Нажмите Enter для продолжения")

readLine()

menu()

}

5 -> {

lecture.topic = university.lessons[Random.nextInt(university.lessons.size)].topic;

teacher.readLesson(lecture)

println("Нажмите Enter для продолжения")

readLine()

menu()

}

6 -> {

println("Преподаватель делится знаниями на лекции...")

println("Нажмите Enter для продолжения")

readLine()

menu()

}

7 -> {

if(!Ispresence){

teacher.markAbsent(lecture, university.students, journal)

println("Нажмите Enter для продолжения")

Ispresence = true;

readLine()

menu(true)

}

else{

teacher.markUp(lecture,university.students,journal)

println("Нажмите Enter для продолжения")

readLine()

menu(true)

}

}

8 -> {

journal.getListStudents()

println("Нажмите Enter для продолжения")

readLine()

menu(true)

}

9 -> {

university.getListLessons()

readLine()

menu(true)

}

10 -> {

val presentStudents = journal.journalEntries.filter { it.presence.any { p -> p } }

println("Сравнение списков студентов:")

println("Всего студентов: ${university.students.size}, Присутствующих: ${presentStudents.size}")

println("Нажмите Enter для продолжения")

readLine()

menu(true)

}

11 -> {

lecture.markPresence(university)

lecture.getPresentStudents()

println("Нажмите Enter для продолжения")

readLine()

menu(true)

}

12 -> {

val starosta =Starasta()

starosta.absent(lecture, university.students, journal)

println("Нажмите Enter для продолжения")

readLine()

menu(true)

}

else -> {

println("Неверный вариант")

println("Нажмите Enter для продолжения")

readLine()

menu()

}

}

}

}

fun menu(isPres: Boolean = false) {

println("Меню:")

println("0. Выйти")

println("1. Добавить лекции")

println("2. Добавить преподавателей")

println("3. Добавить студентов")

println("4. Студент пришел на лекцию")

println("5. Преподаватель читает лекцию")

println("6. Преподаватель делится знаниями")

if(isPres) println("7. Доотметить присутствующих")

else println("7. Отметить студентов как отсутствующих")

println("8. Получить список студентов")

println("9. Получить список предметов")

println("10. Сравнить списки студентов")

println("11. Список отсутствующих")

println("12. Староста отмечает отсутствующих")

print("Выберите действие: ")

StateMenu.value = readLine()?.toIntOrNull() ?: 0

}

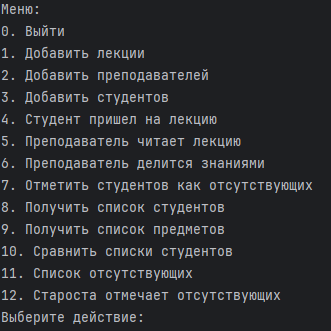


Рисунок 1 – Результат выполнения программы

Вывод: в результате выполнения лабораторной работы я выполнил разработку приложения с использованием языка программирования Kotlin