Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

Современные языки программирования

Отчет по лабораторной работе №1

«Использование языка программирования Kotlin»

Выполнил: Богуш А. В.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

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

**Вариант 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.

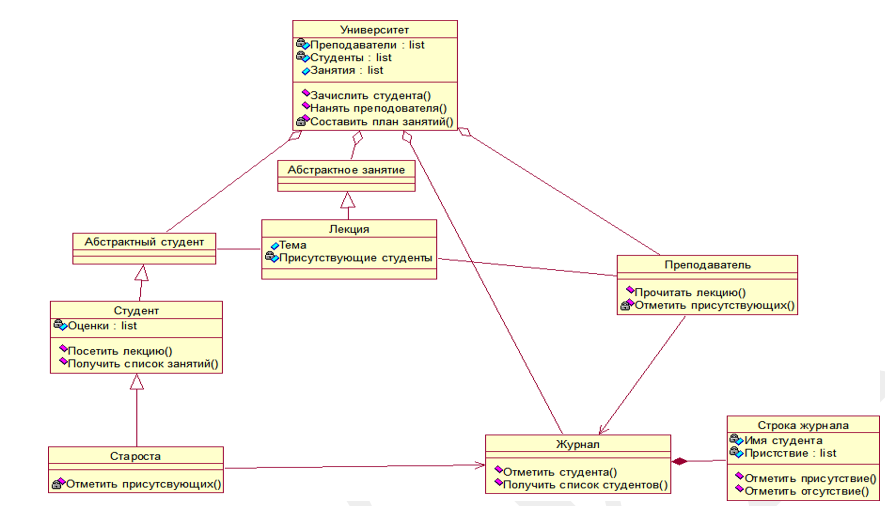


Рисунок 1 – Диаграмма классов

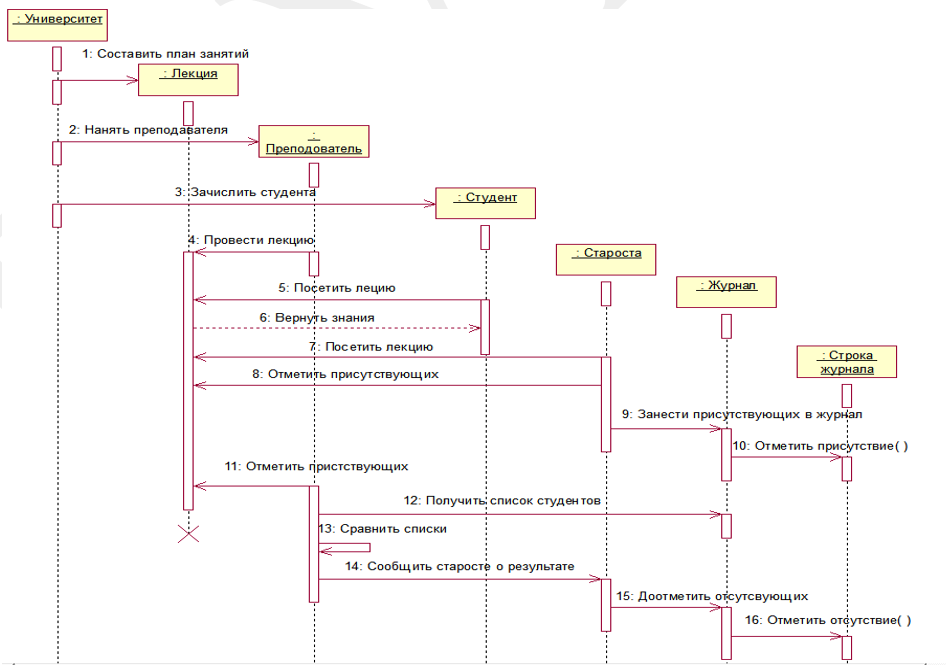


Рисунок 2 – Диаграмма последовательности

**Текст программы:**

class University {

private var professorslist : MutableList<String> = mutableListOf()

private var studentslist : MutableList<String> = mutableListOf()

public var lessonslist : MutableList<String> = mutableListOf()

init {

addlesson("ОАИП")

addlesson("СЯП")

addlesson("КПО")

addlesson("ППД")

addlesson("ТВиМС")

}

fun addstudent(studentname : String) {

studentslist.add(studentname)

println("Студент зачислен успешно.")

}

fun addprofessor(professorname : String) {

professorslist.add(professorname)

println("Преподователь нанят успешно.")

}

private fun addlesson(lessontopic : String) {

lessonslist.add(lessontopic)

println("План занятий обновлен.")

}

fun getstudentslist() : MutableList<String> {

return studentslist

}

fun getprofessorslist() : MutableList<String> {

return professorslist

}

}

open abstract class AbstractLesson {

}

class Lecture(val univ : University) : AbstractLesson() {

private var presentstudents : MutableList<String> = mutableListOf()

public var topic : String = ""

private var mapoflessons : MutableMap<String, MutableList<String>> = mutableMapOf()

init {

for (lessonname in univ.lessonslist) {

mapoflessons[lessonname] = mutableListOf()

}

}

fun getpresents() : MutableList<String> {

val default : MutableList<String> = mutableListOf()

presentstudents = mapoflessons[topic] ?: default

return presentstudents

}

fun addpresents(nameofstudent : String) {

mapoflessons[topic]?.add(nameofstudent)

}

}

open abstract class AbstractStudent {

}

open class Student(val univ : University, val lect: Lecture) : AbstractStudent() {

private var marks : MutableList<Int> = mutableListOf()

private var mapofstudents : MutableMap<String, String> = mutableMapOf()

init {

for (studs in univ.getstudentslist()) {

mapofstudents[studs] = "Знания получены по предметам:"

}

}

fun visitlecture(topic: String, name : String) {

lect.topic = topic

if (lect.getpresents().contains(name)) {

println("Вы уже посетили данную лекцию!")

} else {

lect.addpresents(name)

mapofstudents[name] += " $topic"

println("Вы успешно посетили лекцию!")

}

}

fun getlessons() {

println("Ваши занятия: " + univ.lessonslist)

}

fun getknowledge(name: String) {

println(mapofstudents[name])

}

}

class Headman(univ: University, lect : Lecture, val journal: Journal) : Student(univ, lect) {

private fun markpresentsjournal(presentstudent: MutableList<String>, topic: String) {

for (stud in presentstudent) {

if (univ.getstudentslist().contains(stud) == true) {

journal.markstudent(presentstudent, topic)

} else {

println("Студента $stud не найдено.")

}

}

println("Журнал обновлен.")

}

fun addnewmark(presentstudent: MutableList<String>, topic: String) {

for (stud in presentstudent) {

if (univ.getstudentslist().contains(stud) == true) {

journal.addmarkstudent(presentstudent, topic)

}

}

}

fun getpres(presentstudent: MutableList<String>, topic: String) {

markpresentsjournal(presentstudent, topic)

}

}

class Professor(val univ: University, val lect : Lecture, val journal : Journal) {

fun readlecture(topic : String) {

univ.lessonslist.remove(topic)

println("Лекция прочитана!")

}

private fun markpresents(presentstudent: MutableList<String>,topic: String) {

for (stud in presentstudent) {

if (stud in univ.getstudentslist()) {

journal.markstudent(presentstudent, topic)

} else {

println("Студента $stud не найдено.")

}

}

println("Журнал обновлен.")

}

private fun getlist(topic: String) : MutableList<String> {

return journal.getlistofstudents()

}

fun usemark(presentstudent: MutableList<String>,topic: String) {

markpresents(presentstudent, topic)

}

fun check(topic: String) {

println("Общий список группы: " + getlist(topic))

println("Присутствующие на лекции: " + lect.getpresents())

println("Отмеченные студенты на лекции: " + journal.presentsjourn(topic))

}

fun addnewmark(presentstudent: MutableList<String>, topic: String) {

for (stud in presentstudent) {

if (univ.getstudentslist().contains(stud) == true) {

journal.addmarkstudent(presentstudent, topic)

}

}

}

}

class Journal(val uni : University) {

private var listofstudents : MutableList<String> = uni.getstudentslist()

public var mapjournal : MutableMap<String, StringOfJournal> = mutableMapOf()

val lessons :MutableList<String> = mutableListOf("СЯП", "ОАИП", "КПО", "ППД", "ТВиМС")

init {

for (lessonname in lessons) {

val stringj = StringOfJournal(this)

mapjournal[lessonname] = stringj

}

}

fun getlistofstudents() : MutableList<String> {

return listofstudents

}

fun markstudent(presentstudent: MutableList<String>, topic : String) {

for (stud in presentstudent) {

mapjournal[topic]?.markgood(stud)

}

var templist = uni.getstudentslist().filter { it !in presentstudent }.toMutableList()

for (stud in templist) {

mapjournal[topic]?.markbad(stud)

}

}

fun addmarkstudent(presentstudent: MutableList<String>, topic : String) {

for (stud in presentstudent) {

mapjournal[topic]?.markgood(stud)

}

}

fun presentsjourn(topic: String) : MutableList<String> {

val listofpresence : MutableList<String> = mutableListOf()

for (stud in listofstudents) {

if (mapjournal[topic]!!.mapstringjournal[stud] == "присутствовал") {

listofpresence.add(stud)

}

}

return listofpresence

}

}

class StringOfJournal(val journal: Journal) {

private val presence : MutableList<String> = mutableListOf()

public var mapstringjournal : MutableMap<String, String> = mutableMapOf()

private var nameofstudent : String = ""

init {

for (person in journal.getlistofstudents()) {

mapstringjournal[person] = ""

}

}

fun markgood(name: String) {

mapstringjournal[name] = "присутствовал"

}

fun markbad(name: String) {

mapstringjournal[name] = "отсутствовал"

}

fun check(topic : String) {

println("Журнал предмета $topic: ")

for ((key,value) in mapstringjournal) {

println("$key: $value")

}

}

}

fun main() {

val univer = University()

univer.addstudent("Богуш")

univer.addstudent("Дашкевич")

univer.addstudent("Осталовски")

univer.addstudent("Воскресенский")

univer.addstudent("Навросюк")

univer.addprofessor("Василькова")

univer.addprofessor("Дорошевич")

univer.addprofessor("Давыдович")

univer.addprofessor("Усенко")

univer.addprofessor("Хмелев")

val lecture = Lecture(univer)

val student = Student(univer,lecture)

val journal = Journal(univer)

while (true) {

println("Кто вы?")

println("1. Староста")

println("2. Преподаватель")

println("3. Студент")

println("4. Выход")

var choice: Number

try {

choice = readln().toInt()

} catch (e: Exception) {

println("Некорректный ввод")

Thread.sleep(500)

continue

}

when (choice) {

1->{

println("Представьтесь, пожалуйста.")

val name : String

name = readln()

var isStudExist : Boolean = false

if (univer.getstudentslist().contains(name)) {

isStudExist = true

} else {

println("Такого старосты не найдено.")

Thread.sleep(500)

}

while (isStudExist) {

println("Выберите действие:")

println("1. Отметить присутствующих")

println("2. Посетить лекцию")

println("3. Просмотреть свое расписание")

println("4. Просмотреть свои знания")

println("5. Просмотреть журнал")

println("6. Доотметить присутствующих")

println("7. Выход")

var choice2 : Number

try {

choice2 = readln().toInt()

} catch (e2: Exception) {

println("Некорректный ввод")

Thread.sleep(500)

continue

}

when (choice2) {

1->{

val starosta = Headman(univer, lecture, journal)

println("Введите предмет: ")

val predmet : String

predmet = readln()

if (univer.lessonslist.contains(predmet)) {

lecture.topic = predmet

print("Общий список студентов: ")

println(univer.getstudentslist())

println("Список присутствующих на лекции: " + lecture.getpresents())

println("Введите фамилии студентов через пробел, которых хотите отметить, как присутствующих: ")

val input = readLine()

if (input != null) {

val list = input.split(" ").toMutableList()

starosta.getpres(list, predmet)

} else {

println("Ошибка: вы ничего не ввели.")

}

Thread.sleep(1000)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

2->{

val starosta = Headman(univer, lecture, journal)

println("Введите предмет: ")

val predmet : String

predmet = readln()

if (univer.lessonslist.contains(predmet)) {

starosta.visitlecture(predmet,name)

Thread.sleep(1000)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

3->{

val starosta = Headman(univer, lecture, journal)

starosta.getlessons()

Thread.sleep(1000)

}

4->{

student.getknowledge(name)

Thread.sleep(1000)

}

5->{

println("Введите предмет: ")

val predmet : String

predmet = readln()

if (journal.lessons.contains(predmet)) {

journal.mapjournal[predmet]?.check(predmet)

Thread.sleep(5000)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

6->{

val starosta = Headman(univer, lecture, journal)

println("Введите предмет: ")

val predmet : String

predmet = readln()

if (univer.lessonslist.contains(predmet)) {

lecture.topic = predmet

print("Общий список студентов: ")

println(univer.getstudentslist())

println("Список присутствующих на лекции: " + lecture.getpresents())

journal.mapjournal[predmet]?.check(predmet)

println("Введите фамилии студентов через пробел, которых хотите доотметить, как присутствующих: ")

val input = readLine()

if (input != null) {

val list = input.split(" ").toMutableList()

starosta.addnewmark(list, predmet)

} else {

println("Ошибка: вы ничего не ввели.")

}

Thread.sleep(1000)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

7->{

break

}

}

}

}

2->{

println("Представьтесь, пожалуйста.")

val name : String

name = readln()

var isPrepodExist : Boolean = false

if (univer.getprofessorslist().contains(name)) {

isPrepodExist = true

} else {

println("Такого преподавателя не найдено.")

Thread.sleep(500)

}

while(isPrepodExist) {

println("Выберите действие:")

println("1. Прочитать лекцию")

println("2. Отметить присутствующих")

println("3. Сверить присутствие студентов на лекции с отметками в журнале")

println("4. Просмотреть журнал")

println("5. Доотметить присутствующих")

println("6. Выход")

var choice3 : Number

try {

choice3 = readln().toInt()

} catch (e3: Exception) {

println("Некорректный ввод")

Thread.sleep(500)

continue

}

when (choice3) {

1->{

val prepod = Professor(univer, lecture, journal)

print("Все лекции: ")

println(univer.lessonslist)

println("Введите предмет, который хотите прочитать: ")

val predmet : String

predmet = readln()

if (univer.lessonslist.contains(predmet)) {

prepod.readlecture(predmet)

Thread.sleep(1000)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

2->{

val prepod = Professor(univer, lecture, journal)

println("Введите предмет: ")

val predmet : String

predmet = readln()

if (univer.lessonslist.contains(predmet)) {

lecture.topic = predmet

print("Общий список студентов: ")

println(univer.getstudentslist())

println("Список присутствующих на лекции: " + lecture.getpresents())

println("Введите через пробел фамилии студентов, которых хотите отметить, как присутствующих: ")

val input = readLine()

if (input != null) {

val list = input.split(" ").toMutableList()

prepod.usemark(list, predmet)

} else {

println("Ошибка: вы ничего не ввели.")

}

Thread.sleep(1000)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

3->{

val prepod = Professor(univer, lecture, journal)

println("Введите предмет: ")

val predmet : String

predmet = readln()

if (journal.lessons.contains(predmet)) {

lecture.topic = predmet

prepod.check(predmet)

Thread.sleep(500)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

4->{

println("Введите предмет: ")

val predmet : String

predmet = readln()

if (journal.lessons.contains(predmet)) {

journal.mapjournal[predmet]?.check(predmet)

Thread.sleep(5000)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

5->{

val prepod = Professor(univer, lecture, journal)

println("Введите предмет: ")

val predmet : String

predmet = readln()

if (univer.lessonslist.contains(predmet)) {

lecture.topic = predmet

print("Общий список студентов: ")

println(univer.getstudentslist())

println("Список присутствующих на лекции: " + lecture.getpresents())

journal.mapjournal[predmet]?.check(predmet)

println("Введите фамилии студентов через пробел, которых хотите доотметить, как присутствующих: ")

val input = readLine()

if (input != null) {

val list = input.split(" ").toMutableList()

prepod.addnewmark(list, predmet)

} else {

println("Ошибка: вы ничего не ввели.")

}

Thread.sleep(1000)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

6->{

break

}

}

}

}

3->{

println("Представьтесь, пожалуйста.")

val name : String

name = readln()

var isStuddExist : Boolean = false

if (univer.getstudentslist().contains(name)) {

isStuddExist = true

} else {

println("Такого студента не найдено.")

Thread.sleep(500)

}

while(isStuddExist) {

println("Выберите действие:")

println("1. Посетить лекцию")

println("2. Просмотреть свое расписание")

println("3. Просмотреть свои знания")

println("4. Выход")

var choice4 : Number

try {

choice4 = readln().toInt()

} catch (e4: Exception) {

println("Некорректный ввод")

Thread.sleep(500)

continue

}

when (choice4) {

1->{

println("Введите предмет: ")

val predmet : String

predmet = readln()

if (univer.lessonslist.contains(predmet)) {

student.visitlecture(predmet,name)

Thread.sleep(1000)

} else {

println("Такого предмета не существует.")

Thread.sleep(500)

}

}

2->{

student.getlessons()

Thread.sleep(1000)

}

3->{

student.getknowledge(name)

Thread.sleep(1000)

}

4->{

break

}

}

}

}

4->{

break

}

}

}

}

Результат работы программы представлен на рисунках 1, 2.

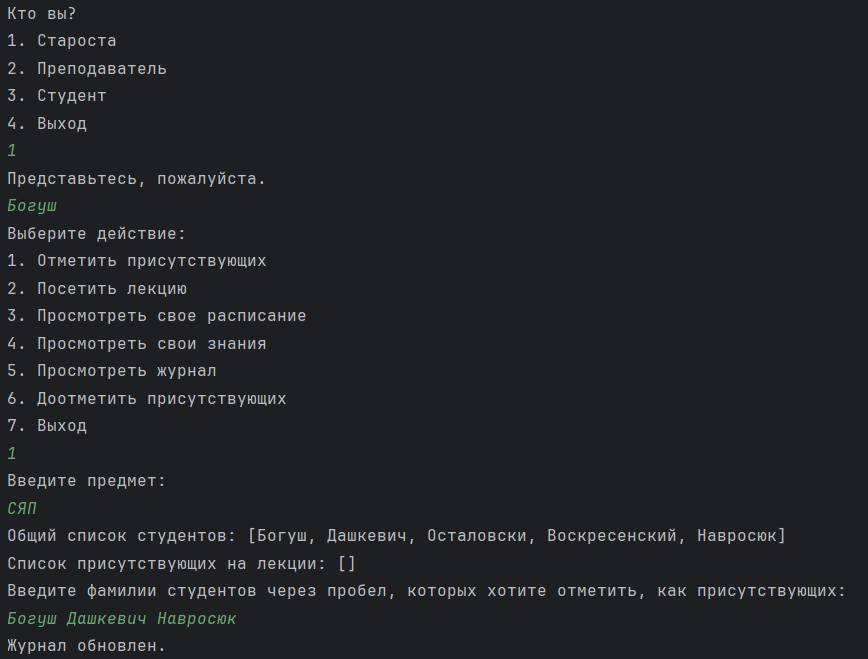


Рисунок 1 – Результат работы программы.

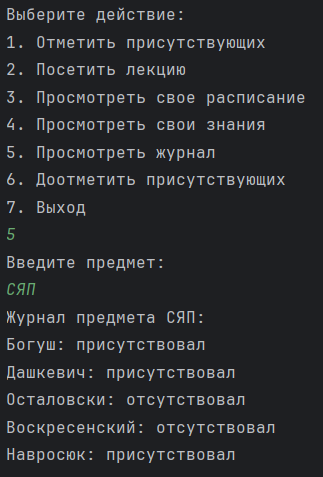


Рисунок 2 – Результат работы программы.

**Выводы**: Изучены основы языка Kotlin и выполнена разработка приложения на данном языке.