Министерство образования

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Современные языки программирования

Отчет по лабораторной работе №1

***«Использование языка программирования Kotlin»***

Выполнил: Карпеченко М. В.

Студент группы 310902

Проверил: Усенко Ф. В.

Минск 2024

Цель работы: выполнить разработку приложения с использованием языка программирования Kotlin.

Индивидуальное задание:

Вариант 9: расширить взаимодействие классов. Организовать получение Работниками дать им возможность попросить увеличить зарплату. Результат просьбы увеличить зарплату должен зависеть от количества выполненных заказов.

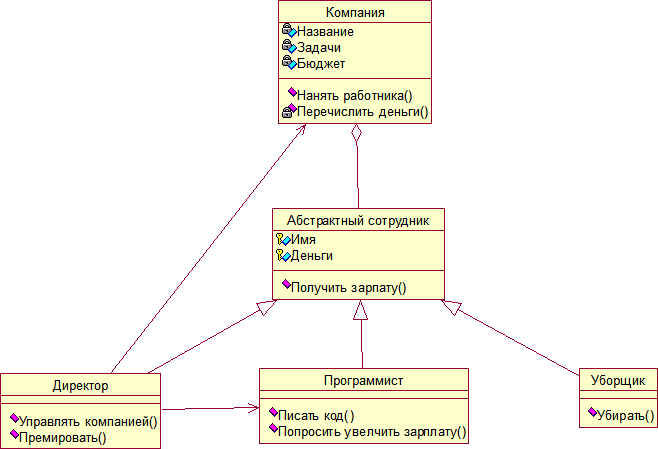


Рисунок 1 – Диаграмма классов

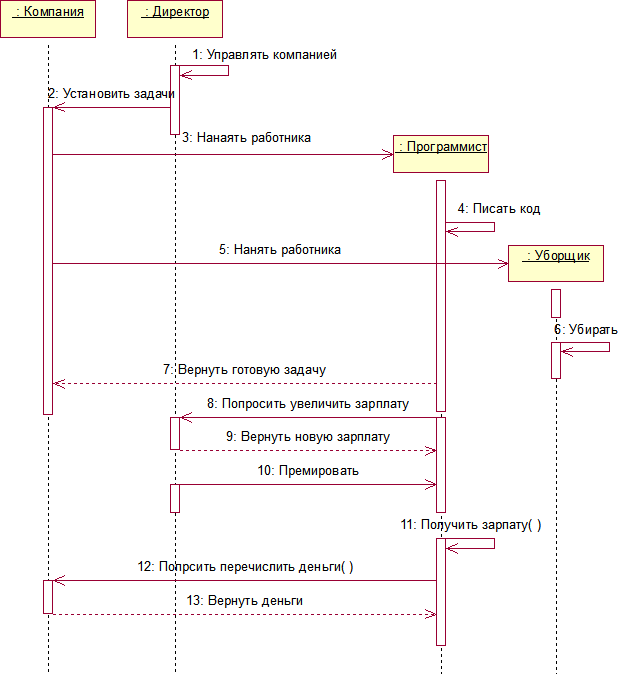


Рисунок 2 – Диаграмма последовательности

Код Main.kt

package org.example

fun main() {

val company = Company("Intel", 10, 15000000)

val director = Director("Joshua", 2000, 1900)

val a: MutableList<Worker> = mutableListOf()

a.addFirst(Programmer("Joshua", 12000, 1200))

a.addFirst(Janitor("Leron", 9000, 900))

while (true) {

println("1. Установить задачи ")

println("2. Нанять программиста")

println("3. Нанять уборщика")

println("4. Писать код")

println("5. Убраться")

println("6. Попросить увеличить зарплату")

println("7. Премировать")

println("8. Получить зарплату")

println("9. Вывести данные всех сотрудников")

println("10. Выйти из программы")

var x: Int = -1

do {

try {

x = readln().toInt()

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (x < 0 || x > 10)

when (x) {

1 -> {

director.controlCompany(company)

}

2 -> {

a.add(company.hireWorker(true))

}

3 -> {

a.add(company.hireWorker(false))

}

4 -> {

println("Кто из сотрудников (index) будет писать код?")

for (i in 0..<a.size) {

if (a[i] is Programmer)

println(i.toString() + "." + a[i].toString())

}

var j: Int

do {

try {

j = readln().toInt()

if(a[j] !is Programmer)

continue

break

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (true)

val prog: Programmer = a[j] as Programmer

prog.doTask(company)

}

5 ->{

println("Кто из сотрудников (index) будет писать код?")

for (i in 0..<a.size) {

if (a[i] is Janitor)

println(i.toString() + "." + a[i].toString())

}

var j: Int

do {

try {

j = readln().toInt()

if(a[j] !is Janitor)

continue

break

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (true)

val jan: Janitor = a[j] as Janitor

jan.clean()

}

6 -> {

println("Кто из сотрудников (index) попросит увеличить зарплату?")

for (i in 0..<a.size) {

if (a[i] is Programmer)

println(i.toString() + "." + a[i].toString())

}

var j:Int

do {

try {

j = readln().toInt()

if(a[j] !is Programmer)

continue

break

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (true)

val prog = a[j] as Programmer

prog.askForRiseSalary(director)

}

7 -> {

println("Кто из сотрудников (index) получает премию?")

for (i in 0..<a.size) {

if (a[i] is Programmer)

println(i.toString() + "." + a[i].toString())

}

var j : Int

do {

try {

j = readln().toInt()

if(a[j] !is Programmer)

continue

break

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (true)

val prog = a[j] as Programmer

prog.askGiveBonus(director)

}

8 -> {

println("Кто из сотрудников (index) получает зарплату?")

for (i in 0..<a.size) {

println(i.toString() + "." + a[i].toString())

}

var j = -1

do {

try {

j = readln().toInt()

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (j < 0 || j > a.size - 1)

a[j].getPaid(company)

}

9 -> {

for (i in 0..< a.size) {

println(i.toString() + "." + a[i].toString())

}

}

10->{

break

}

}

}

}

Код Class.kt

package org.example

class Company(val title: String, var goals: Int, var budget: Long) {

val taskCost = 2000

fun hireWorker(x: Boolean): Worker {

println("Введите имя сотрудника: ")

val name: String = readln()

println("Введите кол-во денег сотрудника (0..15000): ")

var money = 0

do {

try {

money = readln().toInt()

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (money <= 0 || money > 15000)

println("Введите зарплату сотрудника(900..1500): ")

var workerSalary = 0

do {

try {

workerSalary = readln().toInt()

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (workerSalary <= 900 || workerSalary > 1500)

return when (x) {

true -> {

Programmer(name, money, workerSalary)

}

else -> {

Janitor(name, money, workerSalary)

}

}

}

fun payMoney(salary: Int): Int {

println("Заработал ли сотрудник деньги?(true/false)")

var b: Boolean

do {

try {

b = readln().toBoolean()

break

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (true)

when (b) {

true -> {

budget -= salary; return salary

}

else -> {

return 0

}

}

}

}

abstract class Worker(val name: String, var money: Int, var mySalary: Int) {

abstract fun getPaid(c: Company)

override fun toString(): String {

return "Имя: $name, деньги: $money, зарплата: $mySalary"

}

}

class Programmer(name: String, money: Int, mySalary: Int) : Worker(name, money, mySalary) {

override fun getPaid(c: Company) {

money += c.payMoney(mySalary)

}

fun doTask(c: Company) {

if (c.goals > 0) {

c.goals--

println("Код пишется...")

c.budget += c.taskCost // константная прибыль за 1 выполненную задачу

println("Задача выполнена")

} else {

println("Нет задач")

}

}

fun askGiveBonus(d: Director) {

d.giveBonus(this)

}

fun askForRiseSalary(d: Director) {

d.riseSalary(this)

}

}

class Director(name: String, money: Int, mySalary: Int) : Worker(name, money, mySalary) {

override fun getPaid(c: Company) {

money += c.payMoney(mySalary)

}

fun controlCompany(c: Company) {

var a = 0

println("Введите кол-во новых задач(1..100)")

do {

try {

a = readln().toInt()

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (a <= 0 || a > 100)

c.goals += a

}

fun giveBonus(a: Programmer) {

println("Введите кол-во денег в премии(0..200)")

var amount = 0

do {

try {

amount = readln().toInt()

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (amount <= 0 || amount > 200)

a.money += amount

}

fun riseSalary(

p: Programmer

) {

println("Увеличить зарплату для " + p.name + "? (true/false)")

var b: Boolean

do {

try {

b = readln().toBoolean()

break

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (true)

if (b) {

println("Насколько увеличить зарплату для " + p.name + "? (50..300)")

var addSalary = 0

do {

try {

addSalary = readln().toInt()

} catch (e: NumberFormatException) {

println("Неверный ввод")

}

} while (addSalary < 50 || addSalary > 300)

p.mySalary += addSalary

println("Зарплата увеличена")

} else {

println("Просьба не одобрена")

}

}

}

class Janitor(name: String, money: Int, mySalary: Int) : Worker(name, money, mySalary) {

override fun getPaid(c: Company) {

money += c.payMoney(mySalary)

}

fun clean() {

println("Уборка помещений...")

println("Уборка завершена")

}

}

Результат выполнения изображен на рисунках:

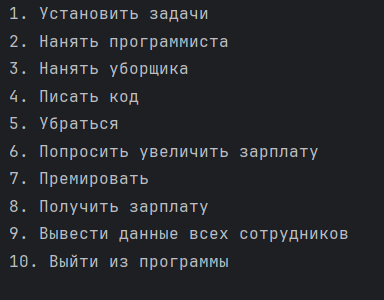


Рисунок 1 – Главное меню

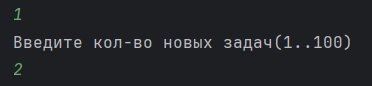


Рисунок 2 – Выбран 1 пункт

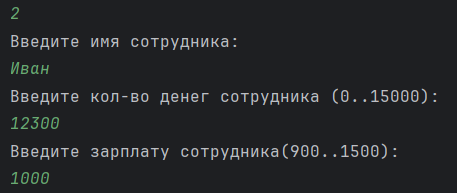


Рисунок 3 – «Нанят» сотрудник-программист

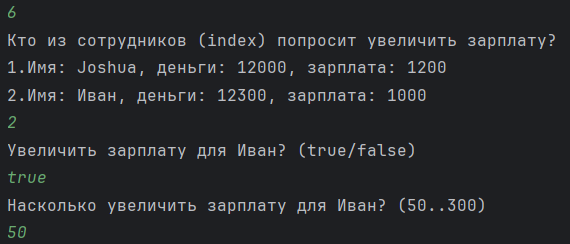


Рисунок 4 – Увеличение зарплаты сотруднику «Иван»

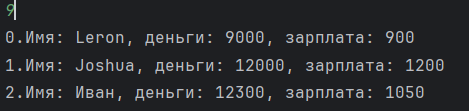


Рисунок 5 – Все сотрудники

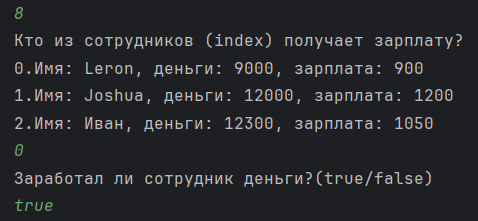


Рисунок 6 – Получение зарплаты

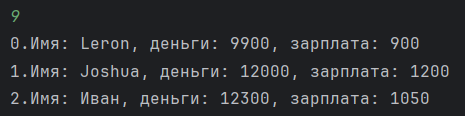


Рисунок 7 – Leron получил зарплату (+900)

Вывод: лабораторная работа №1 по Kotlin позволила нам освоить основы языка, применив на практике базовые конструкции. Для лучшего понимания задания мы также изучили UML диаграммы, что помогло нам структурировать решение.