Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Дисциплина: современные языки программирования

**Лабораторная работа №1**

**«Использование языка программирования Kotlin»**

Вариант 9

Выполнила: Кудрявская В.С.

студентка гр. 310102

Проверил: Усенко Ф.В.

Минск 2024

Цель: Выполнить разработку приложения с использованием языка программирования Kotlin.

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

Листинг кода:

package main  
  
import main.models.\*  
  
fun main() {  
 val director = Director(name = "Валерия")  
 val company = Company(name = "TechCorp", tasks = 0, budget = 10000.0)  
  
 // 1. Директор -> управлять компанией  
 director.manageCompany()  
 *println*("Введите кол-во задач на сегодня: ")  
 var tasks:Int  
 while (true) {  
 try {  
 tasks = *readln*().*toInt*()  
 if( tasks < 0 ){  
 *println*("Введите натуральное число!")  
 }  
 break  
 } catch (e: Exception) {  
 *println*("Введите натуральное число!")  
 }  
 }  
 // 2. Директор -> установить задачи  
 director.setTasks(company, tasks)  
  
 // 3. Компания -> нанять работника (программиста)  
 val programmer = Programmer(name = "Иван")  
 company.hireEmployee(programmer)  
  
 // 4. Программист -> писать код  
 val tasksCompleted = programmer.writeCode(company)  
  
 // 5. Компания -> нанять работника (уборщика)  
 val cleaner = Cleaner(name = "Сергей")  
 company.hireEmployee(cleaner)  
  
 // 6. Уборщик -> убирать  
 cleaner.clean()  
  
 // 7. Программист -> вернуть готовую задачу  
 *println*("${programmer.name} выполнил $tasksCompleted из ${company.getTasks() + tasksCompleted} задач")  
  
 // 8. Программист -> попросить увеличить зарплату  
 var requestedMoney = programmer.askForRaise(tasksCompleted, tasks)  
  
 // 9. Директор -> Вернуть новую зарплату  
 director.returnNewSalary(requestedMoney)  
  
 // 10. Директор -> премировать  
 var bonusAmount: Double  
 *println*("На какую сумму вы хотите премировать сотрудника: ${programmer.name}")  
 while (true) {  
 try {  
 bonusAmount = *readln*().*toDouble*()  
 if(bonusAmount > 0 && bonusAmount <= company.getBudget() - requestedMoney){  
 break  
 }else{  
 *println*("Введите натуральное число не больше ${company.getBudget() - requestedMoney}!")  
 }  
 } catch (e: Exception) {  
 *println*("Введите натуральное число!")  
 }  
 }  
  
 if(director.awardBonus(company, programmer, bonusAmount)){  
 requestedMoney += bonusAmount  
 }  
  
 // 11. Программист -> Получить зарплату()  
 // 12. Программист -> Попросить перечислить деньги()  
 *println*("${programmer.name} попросил перечислить зарплату: $requestedMoney")  
  
 // 13. Компания -> Вернуть деньги  
 company.transferMoney(programmer, requestedMoney)  
}

package main.models  
  
abstract class Employee(  
 val name: String,  
 var money: Double = 0.0  
) {  
  
 fun receiveSalary(amount: Double) {  
 money += amount  
 *println*("$name получил: $amount. Баланс: $money")  
 }  
  
}

class Cleaner(name: String) : Employee(name) {  
  
  
 fun clean() {  
 *println*("$name убирает помещение")  
 }  
}

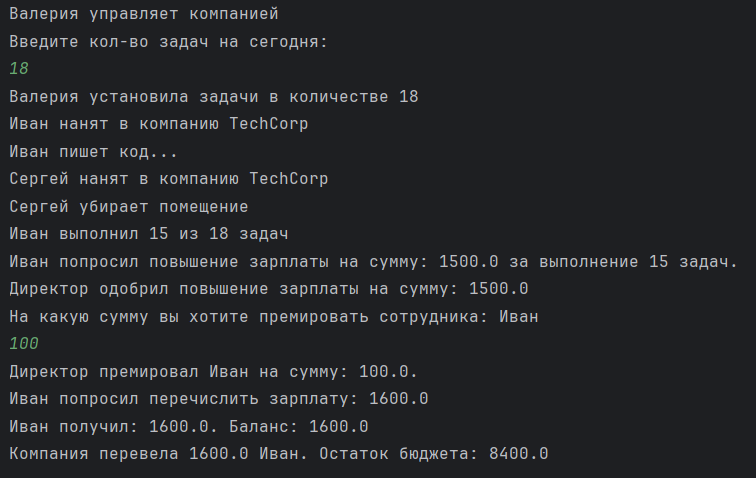
class Director(name: String) : Employee(name) {  
  
 fun manageCompany() {  
 *println*("$name управляет компанией")  
 }  
  
 fun awardBonus(company: Company, employee: Employee, bonus: Double):Boolean {  
 if (company.getBudget() >= bonus) {  
 *println*("Директор премировал ${employee.name} на сумму: $bonus.")  
 return true  
 }  
 *println*("У компании недостаточно бюджета для данной премии")  
 return false  
 }  
  
 fun setTasks(company: Company, tasks: Int) {  
 company.setTasks(tasks)  
 *println*("$name установила задачи в количестве $tasks")  
 }  
  
 fun returnNewSalary(raise: Double) {  
 if (raise > 100.0) {  
 *println*("Директор одобрил повышение зарплаты на сумму: $raise")  
 } else {  
 *println*("Директор отклонил просьбу о повышении зарплаты.")  
 }  
 }  
}

import kotlin.random.Random  
  
class Programmer(name: String) : Employee(name) {  
  
 fun writeCode(company: Company): Int{  
 *println*("$name пишет код...")  
 if (company.getTasks() > 0) {  
 val tasksCompleted = Random.nextInt(1, company.getTasks() + 1)  
 company.decreaseTasks(tasksCompleted)  
 return tasksCompleted  
 }  
 *println*("Нет задач для выполнения")  
 return 0  
  
 }  
  
 fun askForRaise(tasksCompleted: Int, tasksToComplete:Int): Double {  
 val baseRaiseAmount = 100.0  
 *println*("$name попросил повышение зарплаты на сумму: ${baseRaiseAmount \* tasksCompleted} за выполнение $tasksCompleted задач.")  
 if (tasksCompleted >= tasksToComplete/2){  
 return tasksCompleted \* baseRaiseAmount  
 }  
 return baseRaiseAmount  
 }  
}

class Cleaner(name: String) : Employee(name) {  
  
  
 fun clean() {  
 *println*("$name убирает помещение")  
 }  
}

class Company(  
 private val name: String,  
 private var tasks: Int,  
 private var budget: Double  
) {  
 private val employees = *mutableListOf*<Employee>()  
  
 fun decreaseTasks(completedTasks: Int) {  
 tasks -= completedTasks  
 if (tasks < 0) tasks = 0  
 }  
  
 fun getTasks(): Int {  
 return tasks  
 }  
  
 fun getBudget(): Double{  
 return budget  
 }  
  
 fun hireEmployee(employee: Employee) {  
 employees.add(employee)  
 *println*("${employee.name} нанят в компанию $name")  
 }  
  
 fun transferMoney(employee: Employee, amount: Double): Boolean {  
 return if (budget >= amount) {  
 budget -= amount  
 employee.receiveSalary(amount)  
 *println*("Компания перевела $amount ${employee.name}. Остаток бюджета: $budget")  
 true  
 } else {  
 *println*("Недостаточно средств в бюджете компании для перевода $amount. Бюджет: $budget")  
 false  
 }  
 }  
  
 fun setTasks(newTasks: Int) {  
 tasks = newTasks  
 }  
}

Результат выполнения кода:



Вывод: разработана программа на языке программирования Kotlin с помощью двух диаграмм (диаграмма классов и диаграмма последовательности). В результате разработки программы на языке Kotlin была успешно реализована функциональность, позволяющая директору компании ставить количество задач, нанимать сотрудников, выдавать зарплату, начислять премию и др. Была проделана работа с функциями, классами, полями и методами.