

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем

Отчёт
по лабораторной работе №4
на тему:
**Объектно-ориентированное программирование
(ООП)**

Проверил _____ Усенко Ф.В.
(подпись)

Выполнил _____ Ахрамович С.П.
(подпись) гр. 214301

Минск, 2024

Цель: Изучить принципы объектно-ориентированного программирования в *Kotlin*. Научиться создавать и использовать собственные классы, а также применять принципы ООП на практике.

ВАРИАНТ 2. Сложная система учета сотрудников

КОД ПРОГРАММЫ

```
fun main() {
    val employees = listOf(
        Employee("Иван Петров", "Разработчик", 3500.0),
        Manager("Андрей Федоров ", 4200.0, 5),
        Director("Мария Смирнова", 6000.0, 50000.0)
    )

    employees.forEach { employee ->
        val netSalary = employee.calculateSalary(bonus = 300.0, deductions =
150.0)
        println("$employee, Чистая зарплата: $netSalary")
    }

    val forecast = forecastCompanyExpenses(employees, 12)
    println("\nПрогнозируемые затраты компании на 12 месяцев: $forecast
руб.")
}

open class Employee(
    val name: String,
    val position: String,
    val baseSalary: Double
) {

    open fun calculateSalary(bonus: Double = 0.0, deductions: Double = 0.0,
taxRate: Double = 0.14): Double {
        val grossSalary = baseSalary + bonus - deductions
        val netSalary = grossSalary - (grossSalary * taxRate)
        return netSalary
    }

    override fun toString(): String {
        return "Сотрудник: $name, Должность: $position, Базовая зарплата:
$baseSalary"
    }
}

class Director(
    name: String,
    baseSalary: Double,
    val departmentBudget: Double
) : Employee(name, "Директор", baseSalary) {

    override fun calculateSalary(bonus: Double, deductions: Double, taxRate:
Double): Double {
        val budgetBonus = departmentBudget * 0.03
        return super.calculateSalary(bonus + budgetBonus, deductions,
taxRate)
    }
}

fun forecastCompanyExpenses(employees: List<Employee>, months: Int): Double {
```

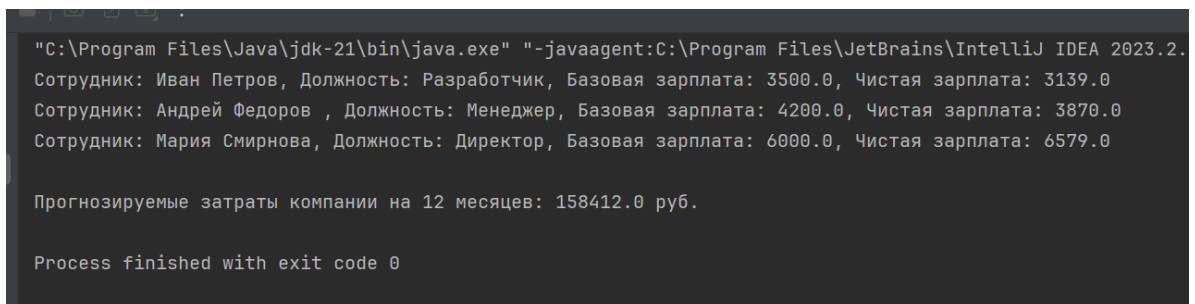
```

        val totalMonthlyExpense = employees.sumOf { it.calculateSalary() }
        return totalMonthlyExpense * months
    }

    class Manager(
        name: String,
        baseSalary: Double,
        val teamSize: Int
    ) : Employee(name, "Менеджер", baseSalary) {
        override fun calculateSalary(bonus: Double, deductions: Double, taxRate: Double): Double {
            val teamBonus = teamSize * 30
            return super.calculateSalary(bonus + teamBonus, deductions, taxRate)
        }
    }
}

```

Результат работы программы представлен на рисунке 1.



```

"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.
Сотрудник: Иван Петров, Должность: Разработчик, Базовая зарплата: 3500.0, Чистая зарплата: 3139.0
Сотрудник: Андрей Федоров , Должность: Менеджер, Базовая зарплата: 4200.0, Чистая зарплата: 3870.0
Сотрудник: Мария Смирнова, Должность: Директор, Базовая зарплата: 6000.0, Чистая зарплата: 6579.0

Прогнозируемые затраты компании на 12 месяцев: 158412.0 руб.

Process finished with exit code 0

```

Рисунок 1 – Результат работы программы

КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Что такое класс в *Kotlin*, и как он объявляется?

Класс в *Kotlin* представляет собой шаблон для создания объектов, который может содержать свойства и методы. Объявляется с помощью ключевого слова *class*.

2 Как создать объект класса в *Kotlin*? Приведите пример.

Объект класса создается путем вызова конструктора без использования ключевого слова *new*. Пример:

```
val person = Person("Alice", 25)
```

3 Что такое свойства класса, и как их объявить в *Kotlin*?

Свойства класса представляют собой переменные, принадлежащие классу. Они могут быть изменяемыми (*var*) или неизменяемыми (*val*). Пример:

```
class Person {
    var name: String = "John"
    val age: Int = 30}

```

4 Как объявить и использовать метод класса? Приведите пример.

Метод класса объявляется внутри класса и представляет собой функцию, которая может манипулировать свойствами класса. Пример:

```
class Person(val name: String) {  
    fun greet() {  
        println("Hello, my name is $name")  
    }  
}  
  
val person = Person("Alice")  
person.greet() // Выводит "Hello, my name is Alice"
```

5 Что такое первичный конструктор, и как он используется для инициализации свойств класса?

Первичный конструктор объявляется в заголовке класса и используется для инициализации свойств класса.

6 Как в *Kotlin* создать вторичный конструктор, и зачем он может понадобиться?

Вторичный конструктор объявляется с помощью ключевого слова *constructor* и может предоставлять альтернативные способы инициализации объекта.

7 Что такое наследование, и как его реализовать в *Kotlin*? Приведите пример.

Наследование позволяет классу-потомку получать свойства и методы родительского класса. В *Kotlin* родительский класс должен быть отмечен ключевым словом *open*. Пример:

```
open class Animal(val name: String) {  
    fun eat() {  
        println("$name is eating")  
    }  
}  
  
class Dog(name: String) : Animal(name) {  
    fun bark() {  
        println("$name is barking")  
    }  
}
```

Вывод: Проведено ознакомление с основами функционального программирования в *Kotlin*, изучение лямбда-выражений, анонимных функций и замыканий. Научились использовать эти концепции для написания более гибкого и читаемого кода.