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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.В. Усенко

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сердюк К.В.

(подпись) гр. 214301

Минск, 2024

**Цель: Композитный объект с динамическими компонентами:** Напишите класс Computer, который состоит из объектов CPU, GPU, Memory, Storage, и PowerSupply. Реализуйте методы для динамического изменения конфигурации компьютера (например, замена процессора или увеличение объема памяти) и оценки его производительности.

**КОД**

class CPU(var name: String, var cores: Int, var frequencyGHz: Double) {

override fun toString(): String = "CPU(name='$name', cores=$cores, frequencyGHz=$frequencyGHz)"

}

class GPU(var name: String, var memoryGB: Int) {

override fun toString(): String = "GPU(name='$name', memoryGB=$memoryGB)"

}

class Memory(var sizeGB: Int) {

override fun toString(): String = "Memory(sizeGB=$sizeGB)"

}

class Storage(var capacityGB: Int, var type: String) {

override fun toString(): String = "Storage(capacityGB=$capacityGB, type='$type')"

}

class PowerSupply(var powerWatt: Int) {

override fun toString(): String = "PowerSupply(powerWatt=$powerWatt)"

}

class Computer(

private var cpu: CPU,

private var gpu: GPU,

private var memory: Memory,

private var storage: Storage,

private var powerSupply: PowerSupply

) {

fun replaceCPU(newCPU: CPU) {

println("Замена CPU: ${cpu.name} -> ${newCPU.name}")

cpu = newCPU

}

fun upgradeMemory(additionalGB: Int) {

println("Увеличение памяти: ${memory.sizeGB}GB + ${additionalGB}GB")

memory.sizeGB += additionalGB

}

fun replaceGPU(newGPU: GPU) {

println("Замена GPU: ${gpu.name} -> ${newGPU.name}")

gpu = newGPU

}

fun replacePowerSupply(newPowerSupply: PowerSupply) {

println("Замена блока питания: ${powerSupply.powerWatt}W -> ${newPowerSupply.powerWatt}W")

powerSupply = newPowerSupply

}

fun evaluatePerformance(): Double {

val cpuScore = cpu.cores \* cpu.frequencyGHz \* 100

val gpuScore = gpu.memoryGB \* 50

val memoryScore = memory.sizeGB \* 10

val storageScore = if (storage.type == "SSD") 200.0 else 100.0

val powerScore = powerSupply.powerWatt / 10.0

val totalScore = cpuScore + gpuScore + memoryScore + storageScore + powerScore

println("Производительность системы: $totalScore баллов")

return totalScore

}

fun showConfiguration() {

println(

"""

Текущая конфигурация компьютера:

CPU: ${cpu.name}, ${cpu.cores} ядер, ${cpu.frequencyGHz} ГГц

GPU: ${gpu.name}, ${gpu.memoryGB} ГБ

Memory: ${memory.sizeGB} ГБ

Storage: ${storage.capacityGB} ГБ, ${storage.type}

Power Supply: ${powerSupply.powerWatt} Вт

""".trimIndent()

)

}

}

fun main() {

val initialCPU = CPU("Intel Core i5", 6, 3.5)

val initialGPU = GPU("NVIDIA GTX 1650", 4)

val initialMemory = Memory(16)

val initialStorage = Storage(512, "SSD")

val initialPowerSupply = PowerSupply(500)

val computer = Computer(initialCPU, initialGPU, initialMemory, initialStorage, initialPowerSupply)

computer.showConfiguration()

computer.evaluatePerformance()

println("---------------------------------------------------")

computer.replaceCPU(CPU("Intel Core i7", 8, 4.0))

computer.upgradeMemory(16)

computer.replaceGPU(GPU("NVIDIA RTX 3060", 12))

computer.replacePowerSupply(PowerSupply(750))

println("---------------------------------------------------")

computer.showConfiguration()

computer.evaluatePerformance()

}

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. **Что такое первичный конструктор, и как он используется для инициализации свойств класса?**

В Kotlin первичный конструктор объявляется прямо в заголовке класса и используется для инициализации свойств класса.

Пример:

class Car(val brand: String, val model: String, var year: Int)

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

Вторичные конструкторы используются для предоставления дополнительных способов инициализации объектов. Они объявляются с использованием ключевого слова constructor.

Пример:

class Car {

val brand: String

val model: String

var year: Int

constructor(brand: String, model: String) {

this.brand = brand

this.model = model

this.year = 2021 // Значение по умолчанию

}

}

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

В Kotlin классы могут наследоваться с использованием ключевого слова open перед базовым классом и : для указания наследования.

Пример:

open class Vehicle(val brand: String) {

fun honk() {

println("Vehicle is honking")

}

}

class Car(brand: String, val model: String) : Vehicle(brand) {

fun drive() {

println("Driving a $brand $model")

}

}

**ВЫВОД**

В данной лабораторной работе были изучены основные принципы ООП, такие как инкапсуляция, наследование и полиморфизм. Были созданы несколько классов для выполнения цели. Задачи, выполненные в ходе работы, позволили глубже понять, как использовать возможности Kotlin для создания гибкой и расширяемой архитектуры программного обеспечения, а также улучшили навыки работы с объектно-ориентированным подходом в разработке.