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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Косович П.В.

(подпись) гр. 214302

Минск, 2024

**Цель: Реализация многопоточной системы обработки данных.** Изучить принципы объектно-ориентированного программирования (ООП) в Kotlin, включая классы, объекты, наследование, полиморфизм, интерфейсы и абстрактные классы. Научиться создавать и использовать собственные классы, а также применять принципы ООП на практике.

**Код**

fun main() {  
 val dataProcessor = DataProcessor(numberOfWorkers = 4)  
 println("Вводите задачи (для выхода 't'):")  
  
 while (true) {  
 val input = readLine() ?: continue  
  
 if (input == "t") {  
 break  
 }  
  
 dataProcessor.addTask {  
 println("Выполняем ввод: $input")  
 Thread.sleep(2000)  
 }  
 }  
  
 dataProcessor.stopAllWorkers()  
 println("Все задачи выполнены.")  
}

class DataProcessor(private val numberOfWorkers: Int) {  
 private val taskQueue = TaskQueue<() -> Unit>()  
 private val workers = mutableListOf<WorkerThread>()  
  
 init {  
 repeat(numberOfWorkers) {  
 val worker = WorkerThread(taskQueue)  
 workers.add(worker)  
 worker.start()  
 }  
 }  
 fun addTask(task: () -> Unit) {  
 taskQueue.addTask(task)  
 println("Задание добавлено ${Thread.currentThread().name}")  
 }  
  
 fun stopAllWorkers() {  
 workers.forEach { it.stopWorker() }  
 println("Все работники остановлены")  
 }  
}

class WorkerThread(private val taskQueue: TaskQueue<() -> Unit>) : Thread() {  
  
 @Volatile  
 private var isRunning = true  
  
 override fun run() {  
 while (isRunning) {  
 try {  
 val task = taskQueue.getTask()  
 task?.*let* {  
 *println*("Задача выполняется в ${Thread.currentThread().*name*}")  
 it.invoke()  
 }  
 } catch (e: InterruptedException) {  
 *println*("${Thread.currentThread().*name*} закончил выполнение.")  
 if (!isRunning) {  
 break  
 }  
 }  
 }  
 }  
  
 fun stopWorker() {  
 isRunning = false  
 this.interrupt()  
 }  
}

import java.util.concurrent.BlockingQueue  
import java.util.concurrent.LinkedBlockingQueue  
  
class TaskQueue<T> {  
 private val queue: BlockingQueue<T> = LinkedBlockingQueue()  
  
 fun addTask(task: T) {  
 queue.put(task)  
 }  
  
 fun getTask(): T? {  
 return queue.take()  
 }  
  
}

**Контрольные вопросы**

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

Класс в Kotlin объявляется с использованием ключевого слова class. Класс может содержать свойства и методы, а также быть шаблоном для создания объектов.

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

fun startEngine() {

println("Engine started for $brand $model")

}

}

Здесь класс Car имеет свойства brand, model и year, а также метод startEngine.

1. Как создать объект класса в Kotlin? Приведите пример.

Объекты создаются с помощью вызова конструктора класса. В Kotlin ключевое слово new не требуется.

val myCar = Car("Toyota", "Corolla", 2020)

myCar.startEngine() // Выводит "Engine started for Toyota Corolla"

1. Что такое свойства класса, и как их объявить в Kotlin?

Свойства класса могут быть изменяемыми (var) или неизменяемыми (val). Они могут быть инициализированы сразу или в конструкторе.

class Person {

var name: String = "John"

val age: Int = 30

}

**Вывод**: Было изучено функциональное программирование. Была выявлена разница обычных функций от однострочной, а также познакомилась с лямбда-функцией.