Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Программирование мобильных информационных систем

Отчёт

по лабораторной работе №4

на тему

**Объектно-ориентированное программирование (ООП)**

Выполнил: Проверил:

ст. гр. 214302 Усенко Ф.В.

Кирик Р.С.

Минск 2024

1. Задание: **Абстрактные классы и адаптеры:** Создайте абстрактный класс Shape и реализуйте несколько его наследников (Circle, Rectangle, Triangle). Реализуйте паттерн "Адаптер" для преобразования различных типов фигур в единый интерфейс работы с ними, например, для расчета площади и периметра.

Листинг кода:

import kotlin.math.PI

class Circle(private val radius: Double) : Shape() {

override fun calculateArea(): Double = PI \* radius \* radius

override fun calculatePerimeter(): Double = 2 \* PI \* radius

}

class Rectangle(private val width: Double, private val height: Double) : Shape() {

override fun calculateArea(): Double = width \* height

override fun calculatePerimeter(): Double = 2 \* (width + height)

}

abstract class Shape {

abstract fun calculateArea(): Double

abstract fun calculatePerimeter(): Double

}

interface ShapeAdapter {

fun getArea(): Double

fun getPerimeter(): Double

}

class ShapeAdapterImpl(private val shape: Shape) : ShapeAdapter {

override fun getArea(): Double = shape.calculateArea()

override fun getPerimeter(): Double = shape.calculatePerimeter()

}

import kotlin.math.sqrt

class Triangle(private val a: Double, private val b: Double, private val c: Double) : Shape() {

override fun calculateArea(): Double {

val s = (a + b + c) / 2 // Полупериметр

return sqrt(s \* (s - a) \* (s - b) \* (s - c)) // Формула Герона

}

override fun calculatePerimeter(): Double = a + b + c

}

fun main() {

val shapes: List<Shape> = listOf(

Circle(5.0),

Rectangle(4.0, 6.0),

Triangle(3.0, 4.0, 5.0)

)

val adapters: List<ShapeAdapter> = shapes.map { ShapeAdapterImpl(it) }

adapters.forEach { adapter ->

println("Area: ${adapter.getArea()}, Perimeter: ${adapter.getPerimeter()}")

}

}

Контрольные вопросы:

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

Класс — это шаблон для создания объектов. Объявляется с ключевым словом `class`:

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

val person = Person("Alice", 25)

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

Свойства — это переменные или значения, принадлежащие классу.

4. Как объявить и использовать метод класса? Приведите пример.

Методы — это функции внутри класса:

class Person(val name: String) {

fun greet() = "Hello, my name is $name"

}

val person = Person("Alice")

println(person.greet())

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

Первичный конструктор объявляется в заголовке класса.

class Person(val name: String, val age: Int)

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

Вторичный конструктор используется для дополнительной логики.

class Person {

var name: String

constructor(name: String) {

this.name = name

}

}

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

Наследование позволяет создавать подклассы от базового класса:

open class Animal(val name: String)

class Dog(name: String) : Animal(name)

8. Как переопределить метод базового класса в подклассе? Приведите пример.

open class Animal {

open fun sound() = "Generic animal sound"

}

class Dog : Animal() {

override fun sound() = "Bark"

}

9. В чем разница между интерфейсом и абстрактным классом в Kotlin?

Интерфейс описывает поведение, но не хранит состояния.

Абстрактный классможет содержать реализацию и состояние.

10. Как реализовать множественное наследование через интерфейсы? Приведите пример.

interface Flyable {

fun fly()

}

interface Walkable {

fun walk()

}

class Bird : Flyable, Walkable {

override fun fly() = println("Flying")

override fun walk() = println("Walking")

}

### 11. Что такое абстрактный класс, и как объявить абстрактный метод?

Абстрактный класс не может быть создан напрямую и может содержать абстрактные методы.

abstract class Animal {

abstract fun sound(): String

}

class Dog : Animal() {

override fun sound() = "Bark"

}

12. Как реализовать полиморфизм в Kotlin? Приведите пример использования.

Полиморфизм позволяет вызывать методы базового класса, используя объекты подклассов:

open class Animal {

open fun sound() = "Animal sound"

}

class Cat : Animal() {

override fun sound() = "Meow"

}

val animals: List<Animal> = listOf(Animal(), Cat())

animals.forEach { println(it.sound()) }

13. Что такое компаньон-объект, и как его использовать для создания единственного экземпляра класса?

Компаньон-объект предоставляет методы и свойства, общие для всех экземпляров.

class Utils {

companion object {

fun greet() = "Hello"

}

}

println(Utils.greet())

14. Как в Kotlin реализовать инкапсуляцию данных? Приведите пример с использованием приватных полей.

class Account(private var balance: Double) {

fun deposit(amount: Double) {

balance += amount

}

fun getBalance() = balance

}

15. Как объединить объекты разных классов в один объект? Приведите пример.

Использовать композицию.

class Engine(val type: String)

class Car(val engine: Engine, val brand: String)

val car = Car(Engine("V8"), "Ford")

16. Как переопределить метод `toString` в классе для предоставления настраиваемого строкового представления объекта?

class Person(val name: String) {

override fun toString() = "Person(name=$name)"

}

17. Как создать класс с несколькими конструкторами, использующимися в разных случаях?

class Person(val name: String, val age: Int) {

constructor(name: String) : this(name, 0)

}

18. Как реализовать событие в классе с использованием интерфейса в Kotlin?

interface EventListener {

fun onEvent()

}

class Button(val listener: EventListener) {

fun click() = listener.onEvent()

}

19. В чем разница между статическим методом в Java и методом компаньон-объекта в Kotlin?

Статический метод в Java доступен без создания объекта.

Компаньон-объект — это Kotlin-аналог, но допускает наследование.

20. Как использовать обобщенные классы для создания универсальных контейнеров? Приведите пример.

class Box<T>(val value: T)

val intBox = Box(10)

val stringBox = Box("Hello")