Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчет
по лабораторной работе №4
на тему:

«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ (ООП)»

Проверил _____ Ф.В. Усенко
(подпись)

Выполнил _____ В.И. Голушко
(подпись) Группа 214301

Минск 2024

Цель работы:

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.

Задание:

Реализация системы бронирования: Разработайте систему для бронирования мест в отеле. Создайте классы Room, Guest, Reservation, и реализуйте методы для управления бронированием, обработки запросов, изменения брони, и расчета стоимости проживания с учетом дополнительных услуг.

Код программы:

```
data class Room(
    val roomNumber: Int,
    val type: String,
    val pricePerNight: Double,
    var isBooked: Boolean = false
)

data class Guest(
    val name: String,
    val email: String
)

class Reservation(
    val guest: Guest,
    val room: Room,
    val startDate: String,
    val endDate: String,
    var additionalServices: List<String> = listOf()
) {
    fun calculateTotalCost(): Double {
        val nights = calculateNights(startDate, endDate)
        val serviceCost = additionalServices.size * 50.0
        return (nights * room.pricePerNight) + serviceCost
    }

    private fun calculateNights(start: String, end: String): Int {
        val startDate = java.time.LocalDate.parse(start)
        val endDate = java.time.LocalDate.parse(end)
        return              java.time.Duration.between(startDate.atStartOfDay(),
endDate.atStartOfDay()).toDays().toInt()
    }
}

class HotelBookingSystem {
    private val rooms = mutableListOf<Room>()
    private val reservations = mutableListOf<Reservation>()
```

```
    fun addRoom(room: Room) {
        rooms.add(room)
    }

    fun bookRoom(guest: Guest, roomNumber: Int, startDate: String, endDate:
String): String {
        val room = rooms.find { it.roomNumber == roomNumber && !it.isBooked }
        return if (room != null) {
            val reservation = Reservation(guest, room, startDate, endDate)
            reservations.add(reservation)
            room.isBooked = true
            "Номер успешно забронирован!"
        } else {
            "Номер занят"
        }
    }

    fun modifyReservation(reservation: Reservation, newStartDate: String,
newEndDate: String): String {
        cancelReservation(reservation)
        val newReservation = Reservation(reservation.guest, reservation.room,
newStartDate, newEndDate)
        reservations.add(newReservation)
        return "Бронирование успешно изменено!"
    }

    fun cancelReservation(reservation: Reservation) {
        reservations.remove(reservation)
        reservation.room.isBooked = false
    }

    fun listReservations(): List<Reservation> {
        return reservations
    }
}

fun main() {
    val hotel = HotelBookingSystem()
    hotel.addRoom(Room(101, "Одиночная", 100.0))
    hotel.addRoom(Room(102, "Двойная", 150.0))
    val guest = Guest("Джон Марстон", "john@example.com")
    println(hotel.bookRoom(guest, 101, "2024-10-01", "2024-10-05"))
    hotel.listReservations().forEach {
        println("Бронировани: ${it.guest.name} в номере ${it.room.roomNumber} с
${it.startDate} до ${it.endDate} с общей стоимостью: ${it.calculateTotalCost()}")
    }
}
```

Ответ на контрольные вопросы

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

Вторичный конструктор объявляется с использованием ключевого слова constructor. Он нужен, если необходимо предоставить дополнительные способы создания объекта.

```
class Person(val name: String) {
    var age: Int = 0
    constructor(name: String, age: Int) : this(name) {
        this.age = age
    }
}
```

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

Для переопределения метода используют ключевое слово override.

```
open class Animal {
    open fun sound() = "Generic sound"
}
class Dog : Animal() {
    override fun sound() = "Bark"
}
```

3. Что такое абстрактный класс, и как объявить абстрактный метод?

Абстрактный класс задаёт шаблон для других классов. Абстрактные методы не имеют реализации.

```
abstract class Shape {
    abstract fun area(): Double
}
class Circle(val radius: Double) : Shape() {
    override fun area() = Math.PI * radius * radius
}
```

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

Метод toString переопределяется с использованием override.

```
class Person(val name: String, val age: Int) {
    override fun toString() = "Person(name=$name, age=$age)"
}
```

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

В Kotlin нет статических методов, вместо них используют компаньон-объекты. Они позволяют объявлять функции и свойства, которые относятся к классу, но не требуют экземпляра.

```
class Utils {
    companion object {
        fun greet() = "Hello!"
    }
}
```

```
println(Utils.greet())
```

Вывод:

Изучил принципы объектно-ориентированного программирования в Kotlin, включая классы, объекты, наследование, полиморфизм, интерфейсы и абстрактные классы. Научился создавать и использовать собственные классы, выполняя задание по варианту лабораторной работы.