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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Кушнер

(подпись) гр. 214301

Минск, 2024

**Цель:** Изучить использование объектно-ориентированного программирования на практике. Разработать систему для бронирования мест в отеле. Создать классы *Room*, *Guest*, *Reservation*, и реализовать методы для управления бронированием, обработки запросов, изменения брони, и расчета стоимости проживания с учетом дополнительных услуг.

**Код**

import java.time.LocalDate  
  
data class Room(  
 val number: Int,  
 val type: String,  
 val pricePerNight: Double,  
 val hasWiFi: Boolean = true,  
 val hasBreakfast: Boolean = false  
)  
  
data class Guest(  
 val name: String,  
 val age: Int  
)  
  
data class Reservation(  
 val guest: Guest,  
 val room: Room,  
 val startDate: LocalDate,  
 val endDate: LocalDate,  
 val additionalServices: List<String> = *emptyList*()  
) {  
 fun totalNights(): Int {  
 return (endDate.toEpochDay() - startDate.toEpochDay()).toInt()  
 }  
  
 fun calculateTotalCost(): Double {  
 var baseCost = room.pricePerNight \* totalNights()  
   
 additionalServices.*forEach* **{** service **->** when (service) {  
 "WiFi" -> if (!room.hasWiFi) baseCost += 5 \* totalNights() // Доплата за WiFi  
 "Breakfast" -> if (!room.hasBreakfast) baseCost += 10 \* totalNights() // Доплата за завтрак  
 }  
 **}** return baseCost  
 }  
  
 fun modifyReservation(newStartDate: LocalDate, newEndDate: LocalDate) {  
 *println*("Изменение брони для гостя ${guest.name}:")  
 *println*("С ${startDate} на $newStartDate и с ${endDate} на $newEndDate")  
 }  
}  
  
class HotelBookingSystem {  
 private val reservations = *mutableListOf*<Reservation>()  
  
 fun addReservation(reservation: Reservation) {  
 reservations.add(reservation)  
 *println*("Бронирование для ${reservation.guest.name} добавлено.")  
 }  
  
 fun findReservationByGuestName(guestName: String): Reservation? {  
 return reservations.*find* **{ it**.guest.name == guestName **}** }  
  
 fun cancelReservation(guestName: String) {  
 val reservation = findReservationByGuestName(guestName)  
 if (reservation != null) {  
 reservations.remove(reservation)  
 *println*("Бронирование для $guestName отменено.")  
 } else {  
 *println*("Бронирование для $guestName не найдено.")  
 }  
 }  
  
 fun showAllReservations() {  
 if (reservations.isEmpty()) {  
 *println*("Нет активных бронирований.")  
 } else {  
 reservations.forEach **{** reservation **->** println("Гость: ${reservation.guest.name}, Номер: ${reservation.room.number}, Даты: ${reservation.startDate} - ${reservation.endDate}")  
 **}** }  
 }  
}  
  
fun main() {  
 val room1 = Room(101, "Стандарт", 50.0, hasWiFi = true, hasBreakfast = false)  
 val room2 = Room(102, "Люкс", 120.0, hasWiFi = true, hasBreakfast = true)  
  
 val guest1 = Guest("Иван Иванов", 30)  
 val guest2 = Guest("Мария Смирнова", 28)  
  
 val reservation1 = Reservation(guest1, room1, LocalDate.of(2024, 9, 10), LocalDate.of(2024, 9, 15), listOf("WiFi", "Breakfast"))  
 val reservation2 = Reservation(guest2, room2, LocalDate.of(2024, 9, 12), LocalDate.of(2024, 9, 14))  
  
 val bookingSystem = HotelBookingSystem()  
  
 bookingSystem.addReservation(reservation1)  
 bookingSystem.addReservation(reservation2)  
  
 bookingSystem.showAllReservations()  
  
 val foundReservation = bookingSystem.findReservationByGuestName("Иван Иванов")  
 if (foundReservation != null) {  
 println("Найдено бронирование для ${foundReservation.guest.name}")  
 println("Общая стоимость проживания: ${foundReservation.calculateTotalCost()}")  
 }  
  
 foundReservation?.modifyReservation(LocalDate.of(2024, 9, 11), LocalDate.of(2024, 9, 16))  
  
 bookingSystem.cancelReservation("Мария Смирнова")  
   
 bookingSystem.showAllReservations()  
}

**Вывод**: Была изучена парадигма объектно-ориентированная программирования и применена на практике при разработке системы для бронирования мест в отеле. Созданы были классы *Room*, *Guest*, *Reservation*, и реализованы методы для управления бронированием, обработки запросов, изменения брони, и расчета стоимости проживания с учетом дополнительных услуг.