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**

Выполнил: Проверил:

ст. гр. 214302 Усенко Ф.В.

Карлюк П.А.

Минск 2024

Вариант 8.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.

Листинг кода:

class Reservation(  
 val id: Int,  
 val guest: Guest,  
 val room: Room,  
 var checkInDate: LocalDate,  
 var checkOutDate: LocalDate,  
 var additionalServices: List<Service> = listOf()  
) {  
 fun addService(service: Service) {  
 additionalServices += service  
 }  
  
 fun calculateTotalCost(): Double {  
 val daysStayed = ChronoUnit.DAYS.between(checkInDate, checkOutDate)  
 val serviceCost = additionalServices.sumOf { it.price }  
 return (daysStayed \* room.pricePerNight) + serviceCost  
 }  
  
 fun changeDates(newCheckInDate: LocalDate, newCheckOutDate: LocalDate) {  
 checkInDate = newCheckInDate  
 checkOutDate = newCheckOutDate  
 }  
}

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

class HotelBookingSystem {  
  
 val rooms = mutableListOf<Room>()  
 private val reservations = mutableListOf<Reservation>()  
 private var nextReservationId = 1  
  
 fun addRoom(room: Room) {  
 rooms.add(room)  
 }  
  
 fun createReservation(guest: Guest, roomNumber: Int, checkIn: String, checkOut: String, selectedServices: List<Int> = listOf()): Reservation? {  
 val room = rooms.find { it.roomNumber == roomNumber }  
 val checkInDate = localDateConvert(checkIn)  
 val checkOutDate = localDateConvert(checkOut)  
  
 if (room != null && isRoomAvailable(room, checkInDate, checkOutDate)) {  
 val reservation = Reservation(nextReservationId++, guest, room, checkInDate, checkOutDate)  
 selectedServices.forEach { serviceId ->  
 val service = ServiceCatalog.services.getOrNull(serviceId)  
 if (service != null) {  
 reservation.addService(service)  
 }  
 }  
 reservations.add(reservation)  
 return reservation  
 } else {  
 println("Комната $roomNumber занята на указанные даты.")  
 return null  
 }  
 }  
  
 private fun localDateConvert(date: String): LocalDate {  
 return try {  
 val formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy")  
 LocalDate.parse(date, formatter)  
 } catch (e: Exception) {  
 e.printStackTrace()  
 LocalDate.now()  
 }  
 }  
  
 fun isRoomAvailable(room: Room, checkInDate: LocalDate, checkOutDate: LocalDate): Boolean {  
 return reservations.none { reservation ->  
 reservation.room == room &&  
 (checkInDate < reservation.checkOutDate && checkOutDate > reservation.checkInDate)  
 }  
 }  
 fun cancelReservationById(reservationId: Int) {  
 val reservation = reservations.find { it.id == reservationId }  
 if (reservation != null) {  
 reservation.room.changeAvailability(true)  
 reservations.remove(reservation)  
 println("Бронирование с ID $reservationId отменено.")  
 } else {  
 println("Бронирование с ID $reservationId не найдено.")  
 }  
 }  
 fun modifyReservationById(reservationId: Int, newCheckIn: String, newCheckOut: String) {  
 val reservation = reservations.find { it.id == reservationId }  
 if (reservation != null) {  
 val newCheckInDate = localDateConvert(newCheckIn)  
 val newCheckOutDate = localDateConvert(newCheckOut)  
  
 if (newCheckInDate.isBefore(newCheckOutDate)) {  
 reservation.changeDates(newCheckInDate, newCheckOutDate)  
 println("Бронирование с ID $reservationId изменено.")  
 } else {  
 println("Ошибка: дата заселения не может быть позже даты выселения.")  
 }  
 } else {  
 println("Бронирование с ID $reservationId не найдено.")  
 }  
 }  
 fun calculateTotalCostForReservationId(reservationId: Int): Double {  
 val reservation = reservations.find { it.id == reservationId }  
 return reservation?.calculateTotalCost() ?: 0.0  
 }  
 fun showCurrentReservations() {  
 if (reservations.isEmpty()) {  
 println("Нет текущих бронирований.")  
 } else {  
 println("Текущие бронирования:")  
 reservations.forEach { reservation ->  
 val servicesList = reservation.additionalServices.joinToString(", ") { it.name }  
 println("ID: ${reservation.id}, Гость: ${reservation.guest.name}, Номер комнаты: ${reservation.room.roomNumber}, Даты: ${reservation.checkInDate}, ${reservation.checkOutDate}, Доп. услуги: $servicesList")  
 }  
 }  
 }  
 fun validateRoomNumber(num: Int): Boolean {  
 rooms.forEach { room ->  
 if (room.roomNumber == num) {  
 return true  
 }  
 }  
 return false  
 }  
 override fun toString(): String {  
 rooms.forEach { println(it.toString()) }  
 return ""  
 }  
}

Контрольные вопросы:

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

Класс в *Kotlin* – это шаблон для создания объектов, который может содержать свойства (данные) и методы (функции). Класс объявляется с использованием ключевого слова *class*.

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

Объекты создаются с помощью вызова конструктора класса.

Пример:

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

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

Свойства класса – это переменные, которые хранят состояние объекта. Свойства класса могут быть изменяемыми (*var*) или неизменяемыми (*val*) и объявляются внутри класса. Они могут быть инициализированы сразу или в конструкторе.