Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



Курсова робота

з дисципліни

**«Прикладне програмування»**

на тему

**«Програмне рішення для адміністрування готелю»**

**Виконав:** студент ОІ-21

Онищенко Ілля

**Керівник:** доцент каф. АСУ

Шпак З.Я.

Львів – 2024

**Зміст**

Вступ…………………………………….…………………………………3

1. Постановка задачі………………………………………………………5

2. Огляд літератури…………………………..……….…..…….………....7

3. Опис етапу проектування……………………………………...….……9

4. Програмне рішення…………………………………………................12

5. Опис проведених експериментів.………………………………….....15

Висновок…………………………………………………………………21

Список використаної літератури…………………………....…..….…...22

Додатки…………………………………………………………………...23

**Вступ**

Сучасна готельна індустрія зазнає значних змін завдяки технологічному прогресу та все більш вимогливим очікуванням клієнтів. Адміністрування готелю – це багатогранний процес, що включає управління бронюванням, розміщенням гостей, обліком фінансів, контролем запасів, управління персоналом, аналізом даних та багатьма іншими аспектами. Зростаюча складність цих завдань створює необхідність у застосуванні сучасних програмних рішень для оптимізації та автоматизації цих процесів.

Метою даної роботи є розробка програмного рішення, яке забезпечить комплексний підхід до адміністрування готелю, дозволяючи ефективно управляти всіма аспектами готельного бізнесу. Це рішення має спростити повсякденні операції, зменшити витрати, підвищити продуктивність та, найголовніше, покращити якість обслуговування гостей, що є ключовим фактором успіху у цій сфері.

Актуальність цієї роботи обумовлена кількома факторами. По-перше, сучасні клієнти готелів очікують високого рівня сервісу та індивідуального підходу. По-друге, конкуренція у готельному бізнесі зростає, що змушує готелі шукати шляхи для підвищення ефективності та скорочення витрат. По-третє, технологічні інновації швидко розвиваються, і ті компанії, які не встигають за цими змінами, ризикують залишитися позаду. Сучасні програмні засоби пропонують безліч можливостей для вирішення цих завдань, включаючи використання хмарних технологій, інтеграцію з системами онлайн-бронювання, аналіз великих обсягів даних, застосування штучного інтелекту для прогнозування потреб клієнтів та управління лояльністю.

Майбутнє застосування результатів цієї роботи на практиці може мати значний вплив на готельний бізнес. Запропоноване програмне рішення може бути впроваджене у готелях різного масштабу – від невеликих бутік-готелів до великих міжнародних готельних мереж. Воно дозволить автоматизувати рутинні завдання, такі як управління бронюваннями, облік фінансів, контроль записів, що зменшить навантаження на співробітників і дозволить їм зосередитися на покращенні взаємодії з гостями.

1. **Постановка задачі**

**Функціональні можливості:**

1. Перегляд інформації про номери:
   * Показ списку всіх номерів готелю з інформацією про ціну, тип та доступність.
   * Відображення лише доступних для бронювання номерів.
2. Фільтрація номерів:
   * Можливість фільтрації номерів за ціною та доступністю.
3. Бронювання номерів:
   * Можливість забронювати номер для певного клієнта на певну кількість днів.
   * Збереження даних про клієнта (ім'я, контактний номер, емейл, гендер).
4. Замовлення їжі:
   * Можливість замовити їжу для заброньованого номера.
   * Відображення списку доступних страв з цінами.
   * Збереження замовлень їжі для кожного номера.
5. Генерація чеку:
   * Автоматичне виставлення чеку для кожного заброньованого номера.
   * Розрахунок загальної суми за проживання та замовлену їжу.
6. Скидання даних:
   * Можливість скинути стан номерів, замовлень їжі та інформацію про клієнтів до початкового стану.

**Технічні вимоги:**

* Використання бази даних для збереження інформації про номери, клієнтів, замовлення їжі.
* Взаємодія з базою даних через мову запитів SQL.
* Реалізація консольного інтерфейсу користувача для зручного взаємодії з програмою.
* Обробка винятків та відлагодження для забезпечення надійності та стабільності програми.

1. **Огляд літератури**

Управління готелем є складним завданням, яке вимагає ефективного контролю над численними процесами, такими як бронювання номерів, обслуговування гостей, замовлення їжі, управління персоналом та багато іншого. Для полегшення цього процесу було створено різноманітні програмні рішення для адміністрування готелів.

Одним із популярних підходів до вирішення цієї проблеми є використання систем управління готелем (Hotel Management System, HMS). Ці системи, як правило, надають широкий спектр функцій, включаючи бронювання номерів, управління доступністю номерів, обробку платежів, управління персоналом та звітність. Прикладами таких систем є Hotelogix, Cloudbeds, WebRezPro та Opera від Oracle. Ці системи часто є комплексними та потужними, але вони також можуть бути дорогими та складними у впровадженні для невеликих готелів.

Альтернативою є використання спеціалізованих програмних рішень, розроблених для конкретних завдань в адмініструванні готелю. Наприклад, програми для бронювання номерів, такі як Booking.com та Expedia, дозволяють гостям легко бронювати номери онлайн. Системи замовлення їжі, такі як GrubHub та UberEats, можуть бути інтегровані для надання послуг харчування гостям. Крім того, існують окремі програми для управління номерним фондом, управління персоналом, бухгалтерського обліку та звітності.

Ще одним підходом є розробка власного спеціалізованого програмного рішення для адміністрування готелю. Це дозволяє створити систему, адаптовану до конкретних потреб готелю, із можливістю інтеграції з існуючими системами та процесами. Такий підхід може бути більш гнучким та економічно вигідним у довгостроковій перспективі, але вимагає значних зусиль на розробку та підтримку.

У контексті цього проекту, було прийнято рішення розробити спеціалізоване програмне рішення для адміністрування готелю з акцентом на базові функції, такі як управління номерами, бронювання номерів, замовлення їжі та виставлення рахунків. Це рішення буде орієнтоване на невеликі готелі та пропонуватиме простий та інтуїтивно зрозумілий інтерфейс для адміністраторів.

На відміну від комплексних систем управління готелем, це рішення буде менш функціональним, але більш доступним та легким у впровадженні. Порівняно з окремими спеціалізованими програмами, воно забезпечить інтегроване середовище для управління основними аспектами роботи готелю.

Для розробки цього рішення буде використана мова програмування Java та база даних MySQL. Java обрана завдяки її перевагам у розробці кросплатформених додатків, підтримці об'єктно-орієнтованого програмування та наявності великої кількості бібліотек і фреймворків. MySQL, у свою чергу, є потужною та надійною системою управління базами даних, що забезпечить ефективне зберігання та обробку даних готелю.

Підсумовуючи, це програмне рішення буде зосереджене на основних функціях адміністрування готелю, пропонуючи простий та інтуїтивно зрозумілий інтерфейс для невеликих готелів. Воно буде розроблене з використанням Java та MySQL, що забезпечить кросплатформеність, об'єктно-орієнтоване програмування та ефективне управління даними.

1. **Опис етапу проектування**

Етап проектування програмного рішення для адміністрування готелю є критичним, оскільки він визначає загальну архітектуру та структуру системи. Цей етап включає в себе детальний аналіз вимог, розбиття задачі на підзадачі та розробку об'єктної моделі системи за допомогою UML-діаграм.

**Use Case:**

Дійова особа:

Користувач - особа, яка використовує програму для бронювання номерів та замовлення послуг у готелі.

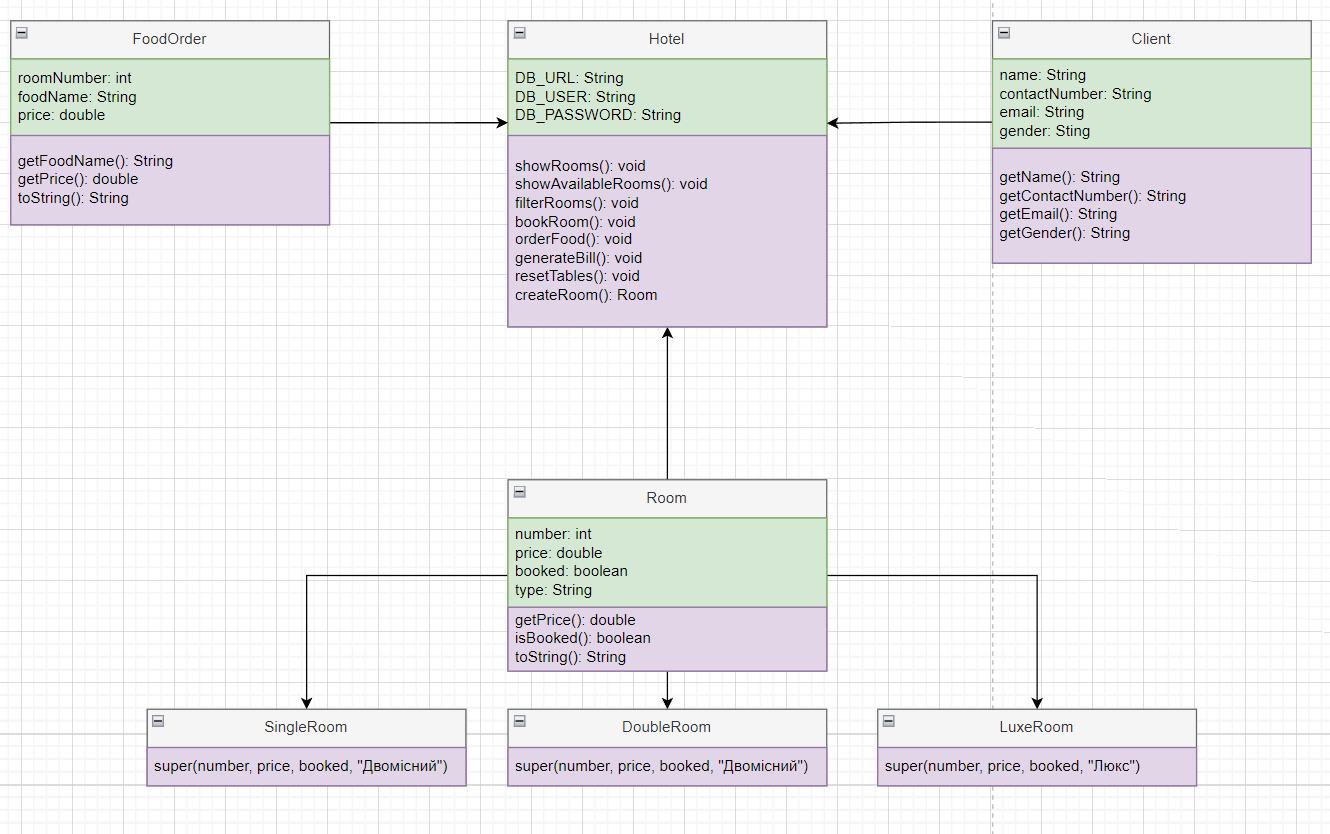
**Основний сценарій:**

1. Користувач відкриває програму.
2. Система пропонує користувачеві наступні опції:
   * Показати всі номери.
   * Показати доступні номери.
   * Фільтрація номерів.
   * Забронювати номер.
   * Замовити їжу.
   * Виставити чек.
   * Вийти.
3. Користувач обирає потрібну опцію.
   * Якщо обрана опція "Показати всі номери", система виводить на екран інформацію про всі доступні номери у готелі.
   * Якщо обрана опція "Показати доступні номери", система виводить на екран інформацію про ті номери, які не заброньовані.
   * Якщо обрана опція "Фільтрація номерів", система пропонує користувачеві обрати критерій фільтрації та виводить відсортовані за обраним критерієм номери.
   * Якщо обрана опція "Забронювати номер", система проводить процес бронювання номера, запитуючи необхідні дані від користувача та оновлюючи базу даних.
   * Якщо обрана опція "Замовити їжу", система дозволяє користувачеві вибрати страви та їх кількість для замовлення, а потім оновлює базу даних.
   * Якщо обрана опція "Виставити чек", система розраховує загальну суму чеку за проживання та послуги, виводить цю суму на екран і завершує роботу.
   * Якщо обрана опція "Вийти", програма завершує свою роботу.
4. Користувач продовжує користуватися програмою або завершує роботу.

Післяумова:

Програма готелю завершує свою роботу або продовжує працювати, очікуючи на наступні команди користувача.

**Class діаграма:**



1. **Програмне рішення**

**Загальна характеристика програми**

Код написаний в програмі Intellij idea. Збережений в папці під назвою project

Вміст папки:

* Папка «.idea» (35,3 КБ)
* Папка «lib» (2,41 МБ):
  + mysql-connector-j-8.4.0.jar (2,41 МБ)
* Папка «src» (21,5 КБ):
  + Папка «clients» (666 Б):
    - Client.java (666 Б)
  + Папка «hotelmanagement» (17 КБ):
    - Hotel.java (17.0 КБ)
  + Папка «rooms» (1.21 КБ):
    - Room.java (655 Б)
    - SingleRoom.java (202 Б)
    - DoubleRoom.java (200 Б)
    - LuxeRoom.java (186 Б)
  + Папка «services» (564 Б):
    - FoodOrder.java (564 Б)
  + Main.java (2,06 КБ)
* Папка «out» (23,5 КБ)
* .gitignore (344 Б)
* project.iml(687 Б)

Мова програмування: Java

**Структура програми**

**Заголовні файли:**

1. Hotel.java: Основний клас програми, що містить точку входу та основний функціонал для управління готелем.
2. Client.java: Клас, що представляє клієнта готелю.
3. Room.java: Клас, що представляє кімнату готелю та його підкласи - SingleRoom.java, DoubleRoom.java, LuxeRoom.java.
4. FoodOrder.java: Клас, що представляє замовлення їжі для кімнати.

**Основні бібліотечні функції:**

1. java.sql: Для роботи з базою даних MySQL.
2. java.util: Для використання списків, сканерів та інших утиліт.

**Середовище реалізації програми:**

* Операційна система: Windows 11.
* Середовище програмування: IntelliJ IDEA або Eclipse.
* Компілятор: Java Development Kit (JDK) для компіляції та виконання програм на Java.
* Інструментальні засоби:
  + MySQL: Для створення та управління базою даних готелю.
  + MySQL Connector/J: Для забезпечення зв'язку між Java-програмою та базою даних MySQL.
* **Текст програми:**
* package hotel;  
    
  import hotel.hotelmanagement.Hotel;  
    
  import java.util.Scanner;  
    
  public class Main {  
   public static void main(String[] args) {  
   Hotel hotel = new Hotel();  
   Scanner scanner = new Scanner(System.*in*);  
   int choice = -1;  
    
   Runtime.*getRuntime*().addShutdownHook(new Thread(Hotel::*resetTables*));  
    
   while (true) {  
   System.*out*.println("Виберіть опцію:");  
   System.*out*.println("1. Показати номери");  
   System.*out*.println("2. Показати доступні номери");  
   System.*out*.println("3. Фільтрація номерів");  
   System.*out*.println("4. Забронювати");  
   System.*out*.println("5. Замовити їжу");  
   System.*out*.println("6. Виставити чек");  
   System.*out*.println("0. Вийти");  
    
   try {  
   choice = Integer.*parseInt*(scanner.nextLine());  
   } catch (NumberFormatException e) {  
   System.*out*.println("Будь ласка, введіть числове значення.");  
   continue;  
   }  
    
   switch (choice) {  
   case 1:  
   hotel.showRooms();  
   break;  
   case 2:  
   hotel.showAvailableRooms();  
   break;  
   case 3:  
   hotel.filterRooms();  
   break;  
   case 4:  
   hotel.bookRoom();  
   break;  
   case 5:  
   hotel.orderFood();  
   break;  
   case 6:  
   hotel.generateBill();  
   break;  
   case 0:  
   System.*out*.println("Вихід...");  
   scanner.close();  
   return;  
   default:  
   System.*out*.println("Невідома опція. Спробуйте ще раз.");  
   }  
   }  
   }  
  }

**5.** **Опис проведених експериментів**

**Приклад №1:**

В першому прикладі я ввів значення «1» для демострації роботи програми, завершив введення «0». Всі дані в базі даних було скинуто (рис.5.1).

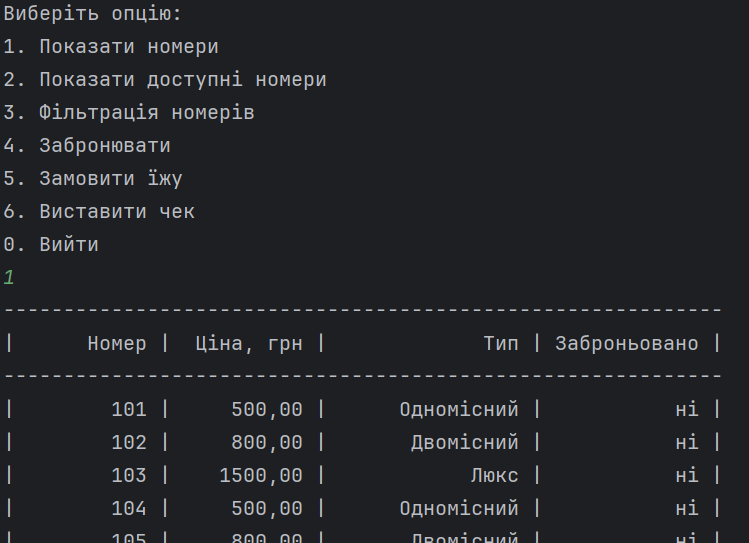
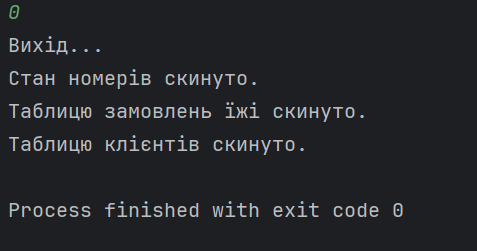
 

рис.5.1-результат прикладу №1

**Приклад №2:**

В другому прикладі я забронював номер на 4 дні командою «4» та ввів інформацію про клієнта (рис.5.2.1).

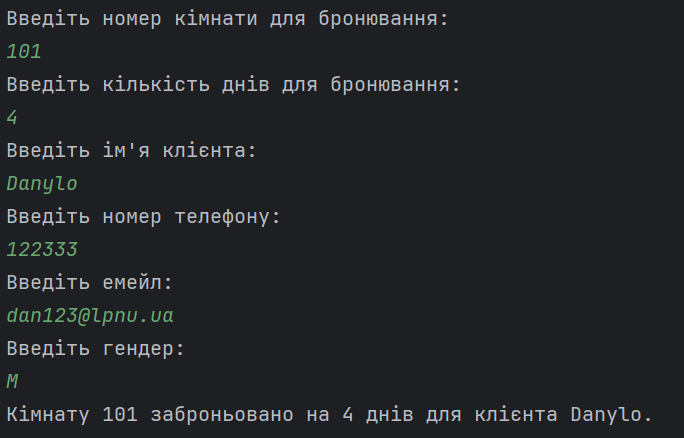


рис.5.2.1-результат прикладу №2

Як бачимо дані було занесино в базу даних (рис.5.2.2).

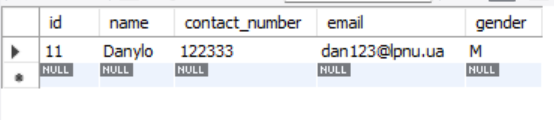


рис.5.2.2-результат прикладу №2

**Приклад №3:**

В третьому прикладі я скористався фільтрацією командою «3» та переглянув номери за спаданням ціни (рис.5.3).

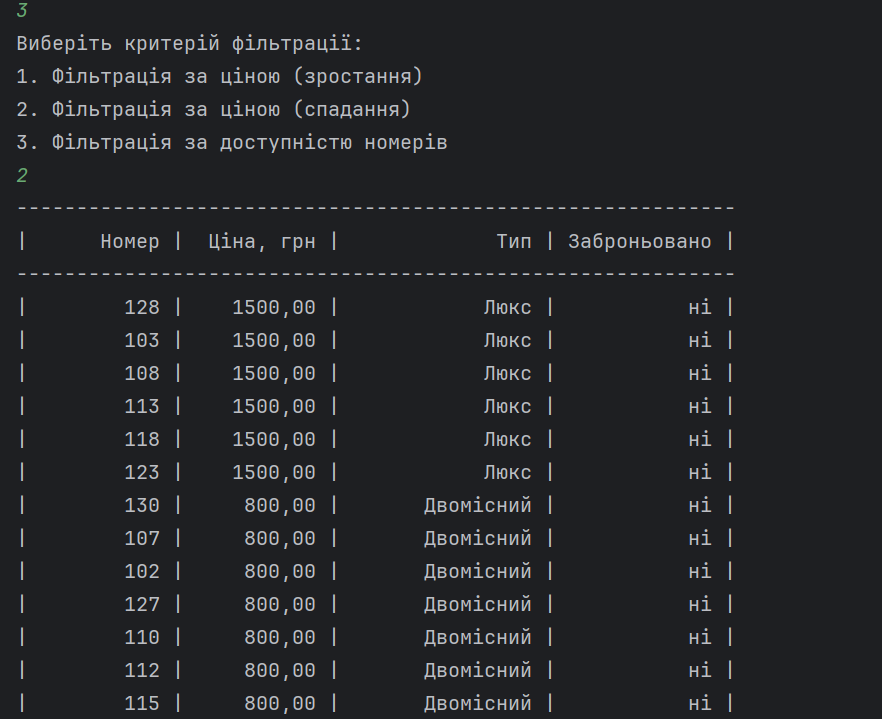


рис.5.3-результат прикладу №3

**Приклад №4:**

В четвертому прикладі я замовив їжу на вже попередньо заброньований номер. Відповідні змінни в базі даних відбулися (рис.5.4).

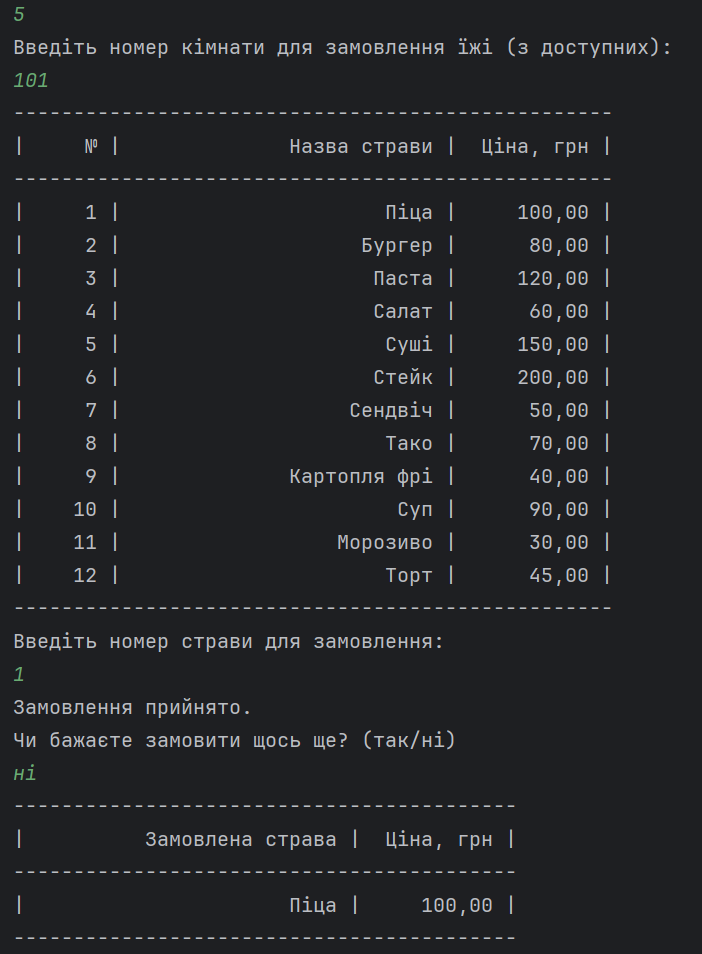


рис.5.4-результат прикладу №4

**Приклад №5:**

В п’ятому прикладі я вистаив чек для кімнати «101». Після цього вийшов з програми, всі дані в базі даних були скинуті до початкових. (рис.5.5).

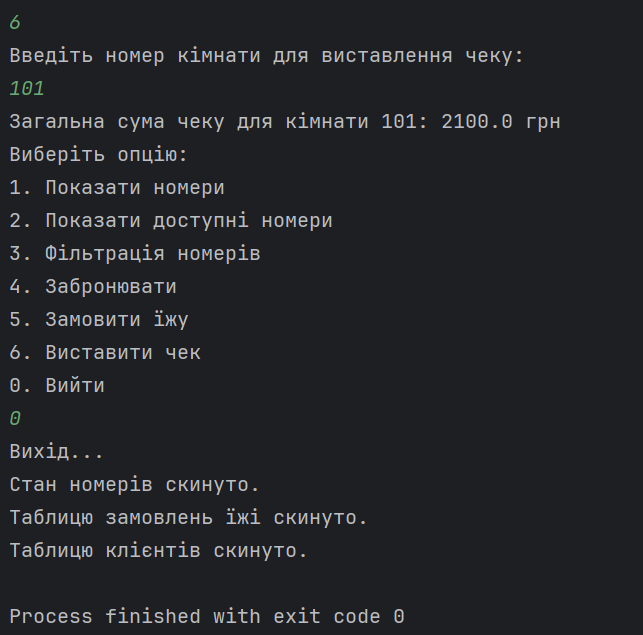
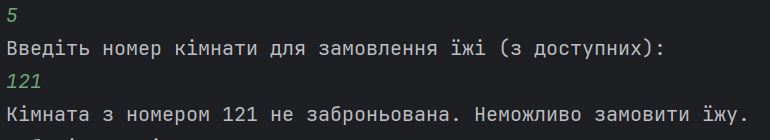


рис.5.5-результат прикладу №5

**Приклад №6:**

В шостому прикладі я навів декілька прикладів введення помилкових значень та відповідних дій до них. (рис.6.6.1) та (рис.6.6.2).

рис.6.6.1-результат прикладу №6

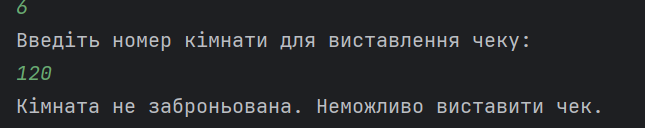


рис.6.6.2-результат прикладу №6

**Висновок**

У результаті роботи над програмним рішенням для адміністрування готелю була вирішена задача автоматизації та оптимізації управління готельним бізнесом. Програма надає зручний та ефективний інструмент для керування кімнатами, бронюванням, замовленням їжі та виставленням чеків.

Перспективи розвитку програмної системи включають розширення функціоналу, таке як додаткові опції фільтрації кімнат, підтримка різних типів оплати, статистичний аналіз даних для покращення стратегій управління готелем та розвиток мобільного додатку для зручності клієнтів.

Під час розробки цього програмного рішення я здобув багато цінних знань і вмінь у сфері розробки програмного забезпечення. Це включає роботу з базами даних, використання бібліотек, вдосконалення навичок в роботі з Java та вирішення проблем через аналіз та реалізацію оптимальних алгоритмів. Крім того, робота над цим проектом дозволила мені поглибити розуміння процесів управління бізнесом та важливості автоматизації для підвищення ефективності підприємства.

**Список використаної літератури**

1. "Effective Java" by Joshua Bloch, 2001 – 413 с.
2. "MySQL Cookbook" by Paul DuBois, 2002 – 1022 с.
3. "Database Systems: A Practical Approach to Design, Implementation, and Management" by Thomas Connolly and Carolyn Begg, 2015 – 1442 с.

**Додатки**

**hotel database:**

**USE hotel;**

**CREATE TABLE rooms (**

**id INT PRIMARY KEY AUTO\_INCREMENT,**

**number INT NOT NULL,**

**type VARCHAR(255) NOT NULL,**

**price DOUBLE NOT NULL,**

**booked BOOLEAN NOT NULL DEFAULT 0,**

**days\_booked INT DEFAULT 0**

**);**

**CREATE TABLE food\_orders (**

**id INT PRIMARY KEY AUTO\_INCREMENT,**

**order\_name VARCHAR(255) NOT NULL,**

**price DOUBLE NOT NULL**

**);**

**CREATE TABLE room\_food\_orders (**

**id INT PRIMARY KEY AUTO\_INCREMENT,**

**room\_id INT NOT NULL,**

**food\_order\_id INT NOT NULL,**

**order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**FOREIGN KEY (room\_id) REFERENCES rooms(id),**

**FOREIGN KEY (food\_order\_id) REFERENCES food\_orders(id)**

**);**

**CREATE TABLE customers (**

**id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(255),**

**contact\_number VARCHAR(20),**

**email VARCHAR(255),**

**gender VARCHAR(10)**

**);**

**INSERT INTO rooms (number, type, price, booked, days\_booked) VALUES**

**(101, 'Одномісний', 500.0, FALSE, 0),**

**(102, 'Двомісний', 800.0, FALSE, 0),**

**(103, 'Люкс', 1500.0, FALSE, 0),**

**(104, 'Одномісний', 500.0, FALSE, 0),**

**(105, 'Двомісний', 800.0, FALSE, 0),**

**(106, 'Одномісний', 500.0, FALSE, 0),**

**(107, 'Двомісний', 800.0, FALSE, 0),**

**(108, 'Люкс', 1500.0, FALSE, 0),**

**(109, 'Одномісний', 500.0, FALSE, 0),**

**(110, 'Двомісний', 800.0, FALSE, 0),**

**(111, 'Одномісний', 500.0, FALSE, 0),**

**(112, 'Двомісний', 800.0, FALSE, 0),**

**(113, 'Люкс', 1500.0, FALSE, 0),**

**(114, 'Одномісний', 500.0, FALSE, 0),**

**(115, 'Двомісний', 800.0, FALSE, 0),**

**(116, 'Одномісний', 500.0, FALSE, 0),**

**(117, 'Двомісний', 800.0, FALSE, 0),**

**(118, 'Люкс', 1500.0, FALSE, 0),**

**(119, 'Одномісний', 500.0, FALSE, 0),**

**(120, 'Двомісний', 800.0, FALSE, 0),**

**(121, 'Одномісний', 500.0, FALSE, 0),**

**(122, 'Двомісний', 800.0, FALSE, 0),**

**(123, 'Люкс', 1500.0, FALSE, 0),**

**(124, 'Одномісний', 500.0, FALSE, 0),**

**(125, 'Двомісний', 800.0, FALSE, 0),**

**(126, 'Одномісний', 500.0, FALSE, 0),**

**(127, 'Двомісний', 800.0, FALSE, 0),**

**(128, 'Люкс', 1500.0, FALSE, 0),**

**(129, 'Одномісний', 500.0, FALSE, 0),**

**(130, 'Двомісний', 800.0, FALSE, 0);**

**INSERT INTO food\_orders (order\_name, price) VALUES**

**('Піца', 100.0),**

**('Бургер', 80.0),**

**('Паста', 120.0),**

**('Салат', 60.0),**

**('Суші', 150.0),**

**('Стейк', 200.0),**

**('Сендвіч', 50.0),**

**('Тако', 70.0),**

**('Картопля фрі', 40.0),**

**('Суп', 90.0),**

**('Морозиво', 30.0),**

**('Торт', 45.0);**

**SELECT \* FROM rooms;**

**SELECT \* FROM food\_orders;**

**SELECT \* FROM room\_food\_orders;**

**SELECT \* FROM customers;**

**Main.java:**

package hotel;  
  
import hotel.hotelmanagement.Hotel;  
  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Hotel hotel = new Hotel();  
 Scanner scanner = new Scanner(System.*in*);  
 int choice = -1;  
  
 Runtime.*getRuntime*().addShutdownHook(new Thread(Hotel::*resetTables*));  
  
 while (true) {  
 System.*out*.println("Виберіть опцію:");  
 System.*out*.println("1. Показати номери");  
 System.*out*.println("2. Показати доступні номери");  
 System.*out*.println("3. Фільтрація номерів");  
 System.*out*.println("4. Забронювати");  
 System.*out*.println("5. Замовити їжу");  
 System.*out*.println("6. Виставити чек");  
 System.*out*.println("0. Вийти");  
  
 try {  
 choice = Integer.*parseInt*(scanner.nextLine());  
 } catch (NumberFormatException e) {  
 System.*out*.println("Будь ласка, введіть числове значення.");  
 continue;  
 }  
  
 switch (choice) {  
 case 1:  
 hotel.showRooms();  
 break;  
 case 2:  
 hotel.showAvailableRooms();  
 break;  
 case 3:  
 hotel.filterRooms();  
 break;  
 case 4:  
 hotel.bookRoom();  
 break;  
 case 5:  
 hotel.orderFood();  
 break;  
 case 6:  
 hotel.generateBill();  
 break;  
 case 0:  
 System.*out*.println("Вихід...");  
 scanner.close();  
 return;  
 default:  
 System.*out*.println("Невідома опція. Спробуйте ще раз.");  
 }  
 }  
 }  
}

**Hotel.java:**

package hotel.hotelmanagement;  
  
import hotel.clients.Client;  
import hotel.rooms.Room;  
import hotel.rooms.SingleRoom;  
import hotel.rooms.DoubleRoom;  
import hotel.rooms.LuxeRoom;  
import hotel.services.FoodOrder;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.sql.PreparedStatement;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
public class Hotel {  
private static final String DB\_URL = "jdbc:mysql://<host>:<port>/hotel";

private static final String DB\_USER = "<username>";

private static final String DB\_PASSWORD = "<password>";  
  
 public void showRooms() {  
 try (Connection connection = DriverManager.*getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery("SELECT \* FROM rooms")) {  
  
 System.*out*.println("------------------------------------------------------------");  
 System.*out*.printf("| %10s | %10s | %15s | %12s |\n", "Номер", "Ціна, грн", "Тип", "Заброньовано");  
 System.*out*.println("------------------------------------------------------------");  
  
 while (resultSet.next()) {  
 int number = resultSet.getInt("number");  
 double price = resultSet.getDouble("price");  
 boolean booked = resultSet.getBoolean("booked");  
 String type = resultSet.getString("type");  
  
 Room room = new Room(number, price, booked, type);  
 System.*out*.println(room);  
 }  
  
 System.*out*.println("------------------------------------------------------------");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void showAvailableRooms() {  
 try (Connection connection = DriverManager.*getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery("SELECT \* FROM rooms WHERE booked = 0")) {  
  
 System.*out*.println("------------------------------------------------------------");  
 System.*out*.printf("| %10s | %10s | %15s | %10s |\n", "Номер", "Ціна, грн", "Тип", "Заброньовано");  
 System.*out*.println("------------------------------------------------------------");  
  
 while (resultSet.next()) {  
 int number = resultSet.getInt("number");  
 double price = resultSet.getDouble("price");  
 boolean booked = resultSet.getBoolean("booked");  
 String type = resultSet.getString("type");  
  
 Room room = new Room(number, price, booked, type);  
 System.*out*.println(room);  
 }  
  
 System.*out*.println("------------------------------------------------------------");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void filterRooms() {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Виберіть критерій фільтрації:");  
 System.*out*.println("1. Фільтрація за ціною (зростання)");  
 System.*out*.println("2. Фільтрація за ціною (спадання)");  
 System.*out*.println("3. Фільтрація за доступністю номерів");  
 int filterChoice = scanner.nextInt();  
  
 List<Room> rooms = new ArrayList<>();  
 try (Connection connection = DriverManager.*getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery("SELECT \* FROM rooms")) {  
 System.*out*.println("------------------------------------------------------------");  
 System.*out*.printf("| %10s | %10s | %15s | %10s |\n", "Номер", "Ціна, грн", "Тип", "Заброньовано");  
 System.*out*.println("------------------------------------------------------------");  
 while (resultSet.next()) {  
 int number = resultSet.getInt("number");  
 double price = resultSet.getDouble("price");  
 boolean booked = resultSet.getBoolean("booked");  
 String type = resultSet.getString("type");  
  
 Room room = createRoom(number, price, booked, type);  
 rooms.add(room);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 switch (filterChoice) {  
 case 1:  
 quicksort(rooms, 0, rooms.size() - 1, true);  
 break;  
 case 2:  
 quicksort(rooms, 0, rooms.size() - 1, false);  
 break;  
 case 3:  
 sortByAvailabilityAndPrice(rooms);  
 break;  
 default:  
 System.*out*.println("Невідомий критерій фільтрації.");  
 return;  
 }  
  
 for (Room room : rooms) {  
 System.*out*.println(room);  
 }  
 System.*out*.println("------------------------------------------------------------");  
 }  
  
 private void quicksort(List<Room> rooms, int low, int high, boolean ascending) {  
 if (low < high) {  
 int partitionIndex = partition(rooms, low, high, ascending);  
 quicksort(rooms, low, partitionIndex - 1, ascending);  
 quicksort(rooms, partitionIndex + 1, high, ascending);  
 }  
 }  
  
 private int partition(List<Room> rooms, int low, int high, boolean ascending) {  
 double pivot = rooms.get(high).getPrice();  
 int i = (low - 1);  
 for (int j = low; j < high; j++) {  
 if (ascending ? rooms.get(j).getPrice() < pivot : rooms.get(j).getPrice() > pivot) {  
 i++;  
 swapRooms(rooms, i, j);  
 }  
 }  
 swapRooms(rooms, i + 1, high);  
 return i + 1;  
 }  
  
 private void sortByAvailabilityAndPrice(List<Room> rooms) {  
 for (int i = 0; i < rooms.size() - 1; i++) {  
 for (int j = 0; j < rooms.size() - i - 1; j++) {  
 Room room1 = rooms.get(j);  
 Room room2 = rooms.get(j + 1);  
 if (room1.isBooked() && !room2.isBooked()) {  
 swapRooms(rooms, j, j + 1);  
 } else if (room1.isBooked() == room2.isBooked() && room1.getPrice() > room2.getPrice()) {  
 swapRooms(rooms, j, j + 1);  
 }  
 }  
 }  
 }  
  
 private void swapRooms(List<Room> rooms, int index1, int index2) {  
 Room temp = rooms.get(index1);  
 rooms.set(index1, rooms.get(index2));  
 rooms.set(index2, temp);  
 }  
  
 public void bookRoom() {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введіть номер кімнати для бронювання:");  
 int roomNumber = scanner.nextInt();  
 System.*out*.println("Введіть кількість днів для бронювання:");  
 int days = scanner.nextInt();  
 scanner.nextLine();  
  
 System.*out*.println("Введіть ім'я клієнта:");  
 String name = scanner.nextLine();  
 System.*out*.println("Введіть номер телефону:");  
 String contactNumber = scanner.nextLine();  
 System.*out*.println("Введіть емейл:");  
 String email = scanner.nextLine();  
 System.*out*.println("Введіть гендер:");  
 String gender = scanner.nextLine();  
  
 Client client = new Client(name, contactNumber, email, gender);  
  
 try (Connection connection = DriverManager.*getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement()) {  
  
 ResultSet resultSet = statement.executeQuery("SELECT \* FROM rooms WHERE number = " + roomNumber);  
 if (!resultSet.next()) {  
 System.*out*.println("Кімнати " + roomNumber + " не існує.");  
 return;  
 }  
  
 if (resultSet.getBoolean("booked")) {  
 System.*out*.println("Кімната " + roomNumber + " вже заброньована.");  
 return;  
 }  
  
 String updateQuery = "UPDATE rooms SET booked = 1, days\_booked = " + days + " WHERE number = " + roomNumber;  
 statement.executeUpdate(updateQuery);  
  
 String insertCustomerQuery = "INSERT INTO customers (name, contact\_number, email, gender) VALUES (?, ?, ?, ?)";  
 PreparedStatement preparedStatement = connection.prepareStatement(insertCustomerQuery);  
 preparedStatement.setString(1, client.getName());  
 preparedStatement.setString(2, client.getContactNumber());  
 preparedStatement.setString(3, client.getEmail());  
 preparedStatement.setString(4, client.getGender());  
 preparedStatement.executeUpdate();  
  
 System.*out*.println("Кімнату " + roomNumber + " заброньовано на " + days + " днів для клієнта " + name + ".");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void orderFood() {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введіть номер кімнати для замовлення їжі (з доступних):");  
 int roomNumber = scanner.nextInt();  
 scanner.nextLine();  
  
 boolean roomExists = false;  
 boolean roomBooked = false;  
 try (Connection connection = DriverManager.*getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement();  
 ResultSet roomResultSet = statement.executeQuery("SELECT \* FROM rooms WHERE number = " + roomNumber)) {  
  
 if (roomResultSet.next()) {  
 roomExists = true;  
 roomBooked = roomResultSet.getBoolean("booked");  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 if (!roomExists) {  
 System.*out*.println("Кімната з номером " + roomNumber + " не існує.");  
 return;  
 }  
  
 if (!roomBooked) {  
 System.*out*.println("Кімната з номером " + roomNumber + " не заброньована. Неможливо замовити їжу.");  
 return;  
 }  
  
 List<FoodOrder> foodOrders = new ArrayList<>();  
 try (Connection connection = DriverManager.*getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery("SELECT \* FROM food\_orders")) {  
  
 List<String> foodNames = new ArrayList<>();  
 List<Double> foodPrices = new ArrayList<>();  
 int index = 1;  
  
 System.*out*.println("--------------------------------------------------");  
 System.*out*.printf("| %5s | %25s | %10s |\n", "№", "Назва страви", "Ціна, грн");  
 System.*out*.println("--------------------------------------------------");  
  
 while (resultSet.next()) {  
 String foodName = resultSet.getString("order\_name");  
 double foodPrice = resultSet.getDouble("price");  
 foodNames.add(foodName);  
 foodPrices.add(foodPrice);  
 System.*out*.printf("| %5d | %25s | %10.2f |\n", index++, foodName, foodPrice);  
 }  
  
 System.*out*.println("--------------------------------------------------");  
  
 boolean continueOrdering = true;  
 while (continueOrdering) {  
 System.*out*.println("Введіть номер страви для замовлення:");  
 int foodChoiceIndex = scanner.nextInt() - 1;  
 scanner.nextLine();  
  
 if (foodChoiceIndex < 0 || foodChoiceIndex >= foodNames.size()) {  
 System.*out*.println("Невірний вибір. Будь ласка, спробуйте ще раз.");  
 continue;  
 }  
  
 String selectedFoodName = foodNames.get(foodChoiceIndex);  
 double selectedFoodPrice = foodPrices.get(foodChoiceIndex);  
  
 FoodOrder foodOrder = new FoodOrder(roomNumber, selectedFoodName, selectedFoodPrice);  
 foodOrders.add(foodOrder);  
  
 String insertQuery = "INSERT INTO room\_food\_orders (room\_id, food\_order\_id) VALUES ((SELECT id FROM rooms WHERE number = " + roomNumber + "), (SELECT id FROM food\_orders WHERE order\_name = '" + selectedFoodName + "'))";  
 statement.executeUpdate(insertQuery);  
  
 System.*out*.println("Замовлення прийнято.");  
  
 System.*out*.println("Чи бажаєте замовити щось ще? (так/ні)");  
 String continueResponse = scanner.nextLine().trim().toLowerCase();  
 continueOrdering = continueResponse.equals("так");  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 System.*out*.println("------------------------------------------");  
 System.*out*.printf("| %25s | %10s |\n", "Замовлена страва", "Ціна, грн");  
 System.*out*.println("------------------------------------------");  
 for (FoodOrder order : foodOrders) {  
 System.*out*.printf("| %25s | %10.2f |\n", order.getFoodName(), order.getPrice());  
 }  
 System.*out*.println("------------------------------------------");  
 }  
  
 public void generateBill() {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введіть номер кімнати для виставлення чеку:");  
 int roomNumber = scanner.nextInt();  
  
 try (Connection connection = DriverManager.*getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement();  
 ResultSet roomResultSet = statement.executeQuery("SELECT \* FROM rooms WHERE number = " + roomNumber)) {  
  
 if (roomResultSet.next() && roomResultSet.getBoolean("booked")) {  
 double roomPrice = roomResultSet.getDouble("price");  
 int daysBooked = roomResultSet.getInt("days\_booked");  
  
 double totalRoomCost = roomPrice \* daysBooked;  
 double totalFoodCost = 0;  
  
 String foodQuery = "SELECT f.price FROM room\_food\_orders rfo JOIN food\_orders f ON rfo.food\_order\_id = f.id JOIN rooms r ON rfo.room\_id = r.id WHERE r.number = " + roomNumber;  
 try (ResultSet foodResultSet = statement.executeQuery(foodQuery)) {  
 while (foodResultSet.next()) {  
 totalFoodCost += foodResultSet.getDouble("price");  
 }  
 }  
  
 double totalBill = totalRoomCost + totalFoodCost;  
 System.*out*.println("Загальна сума чеку для кімнати " + roomNumber + ": " + totalBill + " грн");  
 } else {  
 System.*out*.println("Кімната не заброньована. Неможливо виставити чек.");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void resetTables() {  
 try (Connection connection = DriverManager.*getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement()) {  
 String resetBookingsQuery = "UPDATE rooms SET booked = 0, days\_booked = 0";  
 statement.executeUpdate(resetBookingsQuery);  
 System.*out*.println("Стан номерів скинуто.");  
  
 String resetFoodOrdersQuery = "DELETE FROM room\_food\_orders";  
 statement.executeUpdate(resetFoodOrdersQuery);  
 System.*out*.println("Таблицю замовлень їжі скинуто.");  
  
 String resetCustomersQuery = "DELETE FROM customers";  
 statement.executeUpdate(resetCustomersQuery);  
 System.*out*.println("Таблицю клієнтів скинуто.");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private Room createRoom(int number, double price, boolean booked, String type) {  
 switch (type) {  
 case "Одномісний":  
 return new SingleRoom(number, price, booked);  
 case "Двомісний":  
 return new DoubleRoom(number, price, booked);  
 case "Люкс":  
 return new LuxeRoom(number, price, booked);  
 default:  
 throw new IllegalArgumentException("Невідомий тип кімнати: " + type);  
 }  
 }  
}

**Room.java:**

package hotel.rooms;  
  
public class Room {  
 private int number;  
 private double price;  
 private boolean booked;  
 private String type;  
  
 public Room(int number, double price, boolean booked, String type) {  
 this.number = number;  
 this.price = price;  
 this.booked = booked;  
 this.type = type;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public boolean isBooked() {  
 return booked;  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("| %10d | %10.2f | %15s | %12s |", number, price, type, booked ? "так" : "ні");  
 }  
}

**SingleRoom.java:**

package hotel.rooms;  
  
public class SingleRoom extends Room {  
 public SingleRoom(int number, double price, boolean booked) {  
 super(number, price, booked, "Одномісний");  
 }  
}

**DoubleRoom.java:**

package hotel.rooms;  
  
public class DoubleRoom extends Room {  
 public DoubleRoom(int number, double price, boolean booked) {  
 super(number, price, booked, "Двомісний");  
 }  
}

**LuxeRoom.java:**

package hotel.rooms;  
  
public class LuxeRoom extends Room {  
 public LuxeRoom(int number, double price, boolean booked) {  
 super(number, price, booked, "Люкс");  
 }  
}

**FoodOrder.java:**

package hotel.services;  
  
public class FoodOrder {  
 private int roomNumber;  
 private String foodName;  
 private double price;  
  
 public FoodOrder(int roomNumber, String foodName, double price) {  
 this.roomNumber = roomNumber;  
 this.foodName = foodName;  
 this.price = price;  
 }  
  
 public String getFoodName() {  
 return foodName;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 @Override  
 public String toString() {  
 return foodName + " - " + price;  
 }  
}

**Client.java:**

package hotel.clients;  
  
public class Client {  
 private String name;  
 private String contactNumber;  
 private String email;  
 private String gender;  
  
 public Client(String name, String contactNumber, String email, String gender) {  
 this.name = name;  
 this.contactNumber = contactNumber;  
 this.email = email;  
 this.gender = gender;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getContactNumber() {  
 return contactNumber;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public String getGender() {  
 return gender;  
 }  
}