

# **Final Assignment: Development of an Online Travel Agency System**

- **Overview**

In this assignment, you will develop a complete software system for an online travel agency, similar to platforms like Booking or Airbnb. This system will manage hotel reservations based on various search filters and availability. You are required to implement both the client and server sides of the application, utilizing a wide range of object-oriented programming concepts and Java-specific features.

- **System Requirements**

1. **Database Initialization:** Your system should include a database of at least 10 hotels. Each hotel should have attributes like number of rooms, amenities (e.g., breakfast, lunch, dinner options, room types, balconies, hot tubs, swimming pool availability, star ranking), and a calendar of availability for the next year.
2. **Search and Reservation:** Users should be able to search for hotels based on dates and various filters (amenities and star rankings). The system should allow users to make reservations based on available data.
3. **Server and Client Code:** Implement both the server-side logic that handles data management and client-side functionality that allows for user interaction.

- **Assignment Tasks**

1. **Data Structures and Modifiers:**
  - a. Implement classes with appropriate use of modifiers: ``private``, ``public``, ``protected``, and ``final``.
  - b. Utilize ``static`` fields and methods effectively.
2. **Design Patterns:** Integrate at least 6 of the following design patterns:
  - a. Singleton: For database creation or configuration settings.
  - b. Factory: To create objects of different room types or amenities.
  - c. Strategy: For different search algorithms.
  - d. Facade: To simplify interactions between the client and server sides.
  - e. Iterator: To navigate through collections of hotels and rooms.
  - f. Observer: To update users about changes in reservation status or availability.
  - g. Decorator: To dynamically add features to rooms.

- h. Model-View-Controller (MVC): Organize code structure into models, views, and controllers.
3. **Generics and Collections:**
- a. Use collections like ``List``, ``Map``, and ``Set`` to manage your data effectively.
  - b. Implement generic methods or classes to increase code reusability and safety.
4. **Abstract Classes, Interfaces, Inheritance, and Polymorphism:**
- a. Design an Interface and inheritance hierarchy for different types of rooms or amenities.
  - b. Use polymorphism to handle different types of accommodations or services dynamically.
5. **Exception Handling:** Implement robust exception handling to deal with input errors, and data retrieval issues.
6. **Practical Implementation and System Testing:**
- a. Write a main class that initializes the system and populates it with random data for availability, one year ahead.
  - b. Implement a simple menu-driven interface in the main class to allow users to select search filters, view results, and make reservations.
  - c. **Example Searches (add your own searches!):**
    - i. Search for accommodations with a swimming pool and breakfast included.
    - ii. Hotels with at least 4 stars that offer both a hot tub in the room and a balcony.
    - iii. Budget-friendly accommodations with dinner options.
    - iv. Family suites that include lunch and have access to a private balcony.
    - v. Luxurious accommodations with all amenities included for a specific date range.

## Theoretical Questions

Alongside the practical implementation, the following theoretical questions are designed to assess your understanding of key concepts in software design, particularly focusing on design patterns, code implementation, and system maintenance. These questions should be answered in detail in your final submission.

**1. General Explanation of your system:**

- a. Provide a 3-pages at most explanation of your system:
  - i. How does it work?
  - ii. What did you consider while designing and implementing your system?
  - iii. What is the relationship between the classes, design patterns, etc?

**2. Explain the Use of Modifiers in Java:**

- a. Describe how different modifiers (``private``, ``public``, ``protected``, ``final``, and ``static``) affect class and member accessibility.
- b. Provide examples from your project where you have used these modifiers and explain why.

**3. Abstract Classes, Interfaces, Inheritance, and Polymorphism:**

- a. Explain how you have used abstract classes, inheritance, and polymorphism in your project.
- b. Discuss the benefits these concepts bring to object-oriented programming, particularly in your application.

**4. Design Patterns:**

- a. For each design pattern you implemented, describe the following:
  - i. Definition and Role: What is the purpose of the pattern and what problem does it solve?
  - ii. Implementation: How did you implement this pattern in your project? Provide code snippets or UML diagrams to support your explanation.
  - iii. Benefits and Disadvantages: What are the benefits of using this pattern? Are there any disadvantages or situations where this pattern might not be the best choice?
  - iv. Impact on Code Reuse and Maintenance: Discuss how the use of this pattern affects code reuse and maintainability. Are there any patterns that particularly helped improve the system's maintainability?

**5. Generics and Collections:**

- a. Discuss the advantages of using Generics in Java. How has Generics helped you implement collections in your system?
- b. Provide examples of where Generics were essential in your application.

**6. Exception Handling Strategies:**

- a. Describe the exception-handling strategy used in your project. How does effective exception handling contribute to robust application development?
- b. Provide examples from your project where exception handling was crucial.

## 7. Code Optimization and Efficiency:

- a. Discuss any methods or strategies you employed to optimize the code for better performance and efficiency.
- b. How do you assess the trade-offs between code readability, ease of maintenance, and performance?

## 8. Testing and Debugging:

- a. Explain your approach to testing and debugging the application.
- b. What challenges did you face, and how did you overcome them?
- c. How do design patterns affect the testing process?

## Submission Guidelines

### 1. System Source Code

- a. Provide all source code files and any additional documentation necessary to understand and run the application.
- b. Ensure that your code is well-commented and adheres to standard coding conventions.

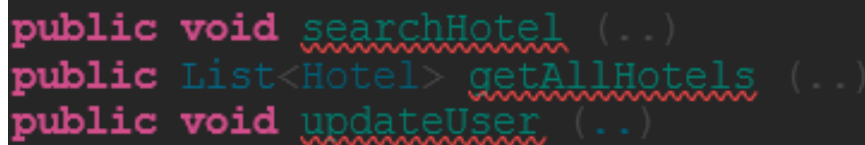
### 2. Theoretical Questions

- a. Provide a PDF document for the theoretical questions so that they are accompanied by TEXTUAL shots of your code. That is:
- b. **DO NOT use screenshots or any format that is textually uncopyable!**

The following is a legal code example assimilated in your PDF:

```
public void searchHotel (..)
public List<Hotel> getAllHotels (..)
public void updateUser (..)
```

The following is an illegal code example assimilated in your PDF, since it contains a screenshot:



```
public void searchHotel (..)
public List<Hotel> getAllHotels (..)
public void updateUser (..)
```

3. In case any of the submission guidelines or system requirements are violated, a point decrease will be held.
4. All of the theoretical questions and code must be developed and answered in JAVA only!

5. Submission is by individuals only (כלומר ההגשה היא ביחידים בלבד).

**a. Do NOT copy!**

6. Submission is due date June 30, on 23:59 PM at most.

a. Submit your files to [orhaim@ariel.ac.il](mailto:orhaim@ariel.ac.il)

b. For the source code submit a ZIP file that contains all your code. Do not submit a Github link!

Good-Luck!

Or