

EECS1022 – Lab 6: Classes & Objects

Fall 2025

Due Date: Sunday, November 23(11:59pm)

Overview

Welcome to Lab 6!

This graded lab assignment focuses on object-oriented programming with emphasis on class design, association relationships, and exception handling. Students will implement a hotel reservation system consisting of two associated classes (Room and Booking) along with custom exception classes. This lab demonstrates the "has-a" relationship between objects and comprehensive error handling through custom exceptions.

Lab Policies

- **Academic Integrity:** Submit your own work. Do not copy code from classmates or online sources. All violations will be reported as academic misconduct.
- **Submission Format:** Submit only the file: **HotelReservationSystem.java**. Do not upload ZIP files or project folders. Do not change the name of the file as this will not allow us to test your code.
- **Deadline:** Submit your .java file to the eClass course page by **Sunday, November 23(11:59pm)**. No late submissions are accepted. Email submissions are not accepted.
- Your lab assignment is not graded during the weekly lab sessions scheduled.

Learning Outcomes

By the end of this lab, you will be able to:

- Import a starter project archive file.
- Design and implement multiple classes with association relationships
- Create and use custom exception classes for error handling
- Implement proper exception throwing and propagation in methods
- Use association (has-a relationship) between classes
- Implement comprehensive data validation with exception handling
- Apply encapsulation principles using private instance variables
- Use the given JUnit tests (calling the utility methods) to guide the development.

EECS1022 – Lab 6: Classes & Objects

Fall 2025

Due Date: Sunday, November 23(11:59pm)

Lab Requirements

- For the JUnit test class `HotelReservationSystemTest.java` given to you:
 - Do not modify the test methods given to you.
 - You are allowed to add new test methods.
- For each method in the `Loan` class that you are assigned to implement:
 - Javadoc has been generated for you. You can check the documentation of the project in your browser by clicking on `index.html` file.
 - No `System.out.println` statements should appear in each of the utility method.
 - No Scanner operations (e.g., `input.nextInt()`) should appear in each of the utility methods. Instead, refer to the input parameters of the method.
 - No main method.
 - NO RECURSION - No recursive method calls
 - Don't change the method signatures provided to you. Otherwise, your test cases will fail.
 - Follow the UML diagram specifications exactly
 - All instance variables must be declared as private
 - Implement all constructors as specified
 - Implement all getter and setter methods
 - Implement all computational methods
 - All validation must throw appropriate exceptions
 - Round all monetary values to 2 decimal places using: `Math.round(value × 100.0) / 100.0`
 - Complete the partial code in `HotelReservationSystem.java` file to achieve desired solution for the listed programming tasks.
 - A tester file `HotelReservationSystemTest.java` is provided to you with some of the test cases implemented.

Download and Import the Starter Project

- Download the Eclipse Java project archive file from eClass: `EECS1022_Lab6.zip`
- Launch Eclipse and browse to `EECS1022-workspace` (for instance or your own created workspace).

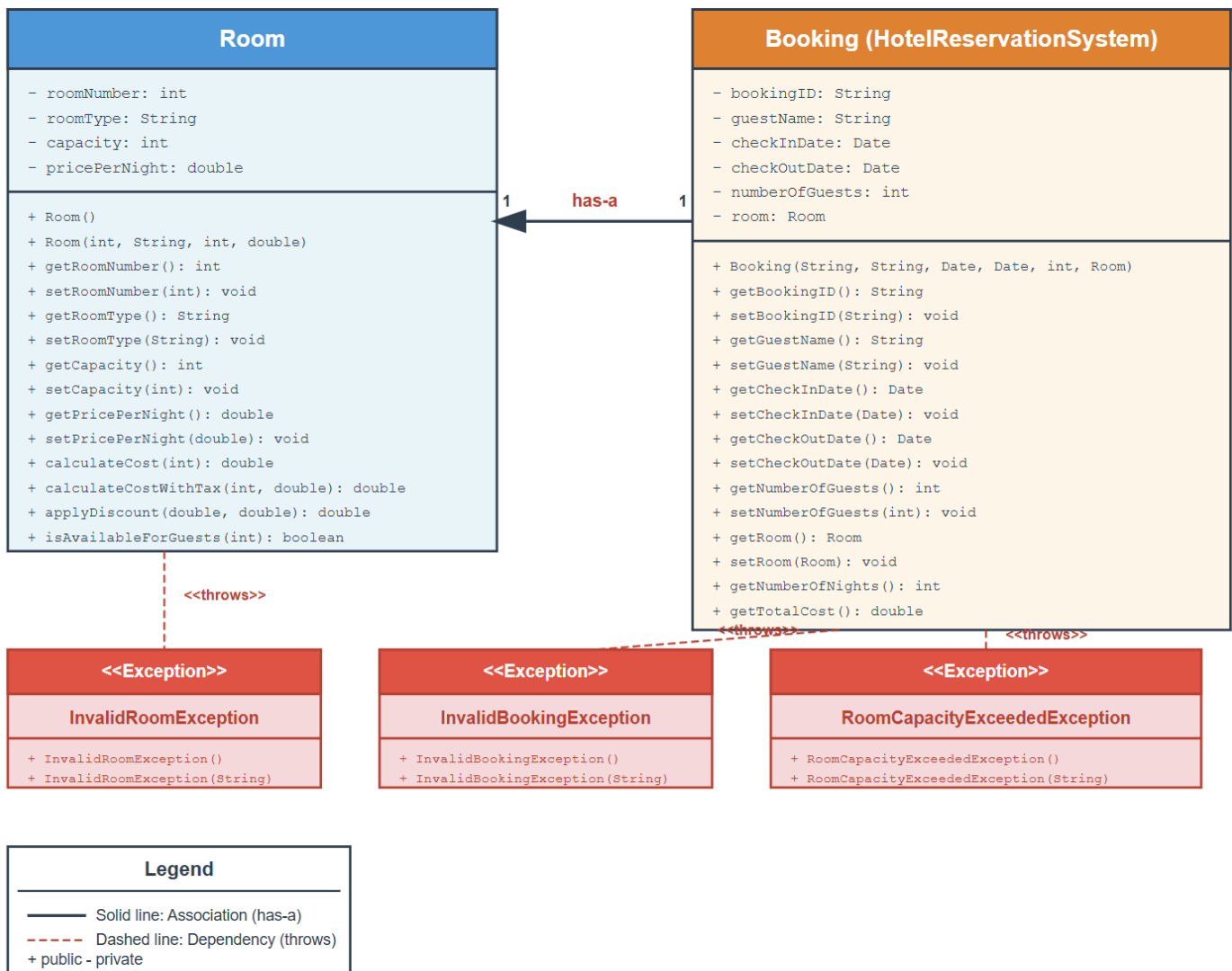
EECS1022 – Lab 6: Classes & Objects

Fall 2025

Due Date: Sunday, November 23(11:59pm)

- In Eclipse:
 - Choose File->Import
 - Under General, choose Existing Projects into workspace
 - Choose Select archive file. Browse your compressed zip folder and attach it.
 - Make sure that the EECS1022_Lab6 box is checked under Projects and you don't have the same project already in the workspace. Then Finish.

UML Diagram:



EECS1022 – Lab 6: Classes & Objects

Fall 2025

Due Date: Sunday, November 23(11:59pm)

System Architecture

- The system consists of five classes:
- **Room:** Represents a hotel room with properties and cost calculation methods
- **Booking:** Represents a reservation that is associated with a Room object
- **InvalidRoomException:** Custom exception for invalid room properties
- **InvalidBookingException:** Custom exception for invalid booking details
- **RoomCapacityExceededException:** Custom exception when guests exceed room capacity

Programming Tasks

You are required to implement the following tasks in the HotelReservationSystem.java class

Task 1: Custom Exception Classes

Three custom exception classes that extend the Exception class.

- InvalidRoomException:** It's already implemented for you. This exception is thrown when invalid room properties are provided (invalid room number, capacity, price, or room type)
- InvalidBookingException :** Thrown when invalid booking details are provided (invalid dates, guest information, or booking identifiers). Implement it (see the above exception implementation for assistance). Use default constructor and a Parameterized constructor.
- RoomCapacityExceededException:** Thrown when the number of guests exceeds the room's maximum capacity. Implement it (see the above exception implementation for assistance). Use default constructor and a Parameterized constructor.

Task 2: Room Class Implementation

- **Instance Variables (all private)**
 - roomNumber (int) - must be positive
 - roomType (String) - must not be null or empty
 - capacity (int) - maximum guests allowed, must be positive
 - pricePerNight (double) - must be non-negative
- **Constructors**
 - **Default Constructor:** Sets roomNumber = 101, Sets roomType = "Single", Sets capacity = 1, Sets pricePerNight = 100.0
 - **Parameterized Constructor:** Signature: Room(int roomNumber, String roomType, int capacity, double pricePerNight) throws InvalidRoomException. **Validates all parameters using setter methods**

EECS1022 – Lab 6: Classes & Objects

Fall 2025

Due Date: Sunday, November 23(11:59pm)

▪ Accessor and Mutator Methods

- Implement standard getters and setters for all instance variables. All setters must validate input and throw `InvalidRoomException` for invalid data:
 - `setRoomNumber`: Throw exception if `roomNumber ≤ 0`
 - `setRoomType`: Throw exception if `roomType` is null or empty
 - `setCapacity`: Throw exception if `capacity ≤ 0`
 - `setPricePerNight`: Throw exception if `pricePerNight < 0`

▪ Computational Methods

- **`calculateCost(int numberOfNights): double`**
 - Calculates total cost for given number of nights
 - Formula: $\text{pricePerNight} \times \text{numberOfNights}$
 - Return value rounded to 2 decimal places
- **`calculateCostWithTax(int numberOfNights, double taxRate): double`**
 - Calculates total cost including tax
 - `taxRate` is percentage (e.g., 13.0 for 13%)
 - Formula: $\text{calculateCost}(\text{numberOfNights}) \times (1 + \text{taxRate} / 100)$
 - Return value rounded to 2 decimal places
- **`applyDiscount(double cost, double discountPercent): double`**
 - Applies discount to given cost
 - `discountPercent` is percentage (e.g., 10.0 for 10% off)
 - Formula: $\text{cost} \times (1 - \text{discountPercent} / 100)$
 - Return value rounded to 2 decimal places
- **`isAvailableForGuests(int numberOfGuests): boolean`**
 - Checks if room can accommodate specified number of guests
 - Returns true if `numberOfGuests ≤ capacity`, false otherwise
- **`toString(): String` (Implemented for you)**

Task 3: Booking Class Implementation

▪ Instance Variables (all private)

- `bookingID` (String) - must not be null or empty
- `guestName` (String) - must not be null or empty
- `checkInDate` (Date) - must not be null
- `checkOutDate` (Date) - must be after `checkInDate`
- `numberOfGuests` (int) - must be positive and \leq room capacity
- `room` (Room) - *Association: Booking has-a Room*

EECS1022 – Lab 6: Classes & Objects

Fall 2025

Due Date: Sunday, November 23(11:59pm)

▪ Constructor

- Signature: Booking(String bookingID, String guestName, Date checkInDate, Date checkOutDate, int numberOfGuests, Room room) throws InvalidBookingException, RoomCapacityExceededException
- **Validates all parameters using setter methods**

▪ Accessor and Mutator Methods

- Implement standard getters and setters for all instance variables with appropriate validation:
 - setBookingID: Throw InvalidBookingException if null or empty
 - setGuestName: Throw InvalidBookingException if null or empty
 - setRoom: Throw InvalidBookingException if null (**Set room before dates and guests (needed for validation)**)
 - setCheckInDate: Throw InvalidBookingException if null or not before checkOutDate
 - setCheckOutDate: Throw InvalidBookingException if null or not after checkInDate
 - setNumberOfGuests: Throw InvalidBookingException if ≤ 0 , throw RoomCapacityExceededException if exceeds room capacity

▪ Computational Methods

- **getNumberOfNights(): int**
 - Calculates number of nights between check-in and check-out dates. Use getTime() method of Date class.
 - Use TimeUnit.MILLISECONDS.toDays() for calculation
- **getTotalCost(): double**
 - Calculates total cost using associated room's calculateCost method
 - Formula: room.calculateCost(getNumberOfNights())
- **getTotalCostWithTax(double taxRate): double**
 - Calculates total cost with tax using associated room's method
 - Formula: room.calculateCostWithTax(getNumberOfNights(), taxRate)
- **getCostPerGuest(): double**
 - Calculates cost per guest
 - Formula: getTotalCost() / numberOfGuests
 - Return value rounded to 2 decimal places
- **getDiscountedTotal(double discountPercent): double**
 - Applies discount using associated room's applyDiscount method
 - Formula: room.applyDiscount(getTotalCost(), discountPercent)
- **isWeekendBooking(): boolean**
 - Checks if check-in date falls on Friday or Saturday

EECS1022 – Lab 6: Classes & Objects

Fall 2025

Due Date: Sunday, November 23(11:59pm)

- Use `Calendar.getInstance()` and `getDayOfWeek()`
- Returns true if day is `Calendar.FRIDAY` or `Calendar.SATURDAY`
- **`toString(): String`(Implemented for you)**

Submission Instructions

- Submit only one file: **`HotelReservationSystem.java`**
- Upload it to the Lab 6 assignment on eClass.
- Ensure your file compiles and runs without error.
- Submitting files with compilation errors will receive zero marks. No partial marks.
- Your file will be tested with additional test cases.