BLM2012 Object Oriented Programming
2025-2026 Fall Semester Project Document

Subject: Airline Reservation and Management System

Main Objective: The main goal of this project is to bring a complex real-world business system (Airline Management) to life using the Java programming language. Students are expected to effectively apply Object-Oriented Programming (OOP) principles (Encapsulation, Inheritance, Polymorphism, Abstraction), modern software development practices (Unit Testing), and concurrent processing (Multithreading).

1. **Detailed Project Modules**
   The project consists of 3 main modules.

   a. **Flight Management Module**
   The minimum requirements for this module are given in the table below.

Class Name	Explanation
Plane	planeID, planeModel, capacity (int), **seatMatrix** (2D array or Map structure)
Flight	flightNum, departurePlace, arrivalPlace, date, hour, duration
Seat	seatNum (String, exp. "15A"), Class (Enum: ECONOMY, BUSINESS), price, **reserveStatus** (boolean).
Route	Departure/Arrival information

   b. **Reservation and Ticketing Module**
   The minimum requirements for this module are given in the table below.

Class Name	Explanation
Passenger	passengerID, name, surname, contactInfo
Reservation	reservationCode, **Flight** object, **Passenger** object, **Seat** object, dateOfReservation
Ticket	ticketID, Reservation object, price, baggageAllowance
Baggage	weight. It is related to the ticket class.

c. Services and Managers

The minimum requirements for this module are given in the table below.

Class Name	Explanation
FlightManager	Creating new flights, updating/deleting existing flights.
SeatManager	It creates the seating arrangements on the plane and calculates the number of available seats.
ReservationManager	Making and canceling reservations. **Concurrency** will apply here.
CalculatePrice	Includes the business logic for calculating ticket prices (mandatory for JUnit).

2. System Requirements

a. Multithreading Structure

Students must solve the following two scenarios using Java's concurrency mechanisms (multithreading).

Scenario 1: Simultaneous Seat Reservation (Concurrency Control)

- In this scenario, let's assume there are initially 180 empty seats on the plane (30 rows of 6 seats) and 90 passengers will be seated in these seats.
- Each passenger should select their seat via a thread, and their chosen seat should be randomly selected from among the available seats.
- The distribution of empty and occupied seats on the plane should be visualized on a panel, and the number of occupied seats resulting from this process should be written below the panel.
- This scenario should be able to be run again and again by pressing a button whenever the user wants to repeat the simulation.
- A checkbox on the screen should allow you to determine whether threads will run synchronously or not. When synchronized, it should show 90 seats occupied and 90 seats empty; when not synchronized, it should show incorrect seat placements.

Scenario 2: Asynchronous Report Generation (Asynchronous Task)

- A lengthy reporting process (e.g., calculating the occupancy rate for all flights) must be initiated in the user interface (GUI).
- This calculation should be run in a separate ReportGenerator Thread to avoid blocking the main GUI Thread.
- During the reporting process, the GUI should display the message "Preparing report..." to the user and asynchronously report the result to the GUI when the process is complete.

b. Unit Test Application (JUnit 5)

It is mandatory to write a **minimum of 5 JUnit tests** for the following classes and methods.

PriceCalculation Test:

- Verify that the prices are calculated correctly for different classes (Economy, Business).

FlightSearchEngine Test:

- Test the system to filter and retrieve the correct flights for the given departure/arrival cities.
- Test the process of eliminating flights whose departure time has passed.

SeatManager Test:

- Test that the emptySeatsCount method accurately decreases after a seat has been reserved.
- Test the Exception thrown when trying to book a non-existent seat number.

3. **Graphical User Interface (GUI)**
   The project **must** use a graphical user interface (JavaFX or Swing) and include the following screens:

- **Login Screen:** User/Staff login**.**
- **Flight Search and Booking Screen:** The main screen where users can search for flights, select seats, and make reservations.
- **Reservation Management Screen:** Query, view, and cancel existing reservations**.**
- **Admin/Staff Screen:** Add new flights, edit existing flights (departure times, etc.), manage staff information.

4. **Project Rules and Submissions**

- Use file operations for permanent data storage in the project. Do not use data storage systems such as databases or XML.
  - *You can use multiple files for multiple tasks.*
- Identify the classes mentioned above, as well as any additional classes that may be used, and determine what kind of attributes and methods they possess.
- Draw a UML class diagram to show the interaction between classes.
- Prepare a PDF file containing the UML and its textual explanation, and name it "*report_yourGroupNumber*".
  - **Exp:** *report_12.zip*
- Rename your source code's src folder by adding an underscore and your group number to the end, then compress it in ZIP format to obtain a compressed file as shown in the example below.
  - **Exp:** *src_12.zip*
- If you have libraries that you use and add externally to your project, place them in a folder, name the folder "*lib_yourGroupNumber*" and compress the file in ZIP format.
  - **Exp:** *lib_22010017.zip*
  - ***Note:*** *If you are not using any additional libraries, you do not need to submit this!*
- Create an executable file (.jar extension) of your project. Run it on your own computer (and on other computers if possible) to ensure it works correctly. Name this .jar file with your group number.
  - **Exp:** *12.jar*
- **Create a video (maximum 3 minutes) in .mp4 format explaining how your project worked.** This video must include a screenshot of your computer screen and, importantly, a video recording of yourself on your camera! Name the video "*video_yourGroupNumber*".
  - **Exp:** *video_12.mp4*

- Place the files you created as described above (***report_12.zip***, ***src_12.zip***, ***lib_12.zip***, ***12.jar***, ***video_12.mp4***) into a folder named after your group number, and compress this folder in .zip format to obtain the final submission document.
    - **Exp:** *12.zip*
    - **Note:** Please submit only one file for your project. Do not submit any additional files!
    - **Note:** *Each group should only make one submission.*
- The project will be carried out in **groups of two**.
- ***Projects that are identical or significantly similar to each other will be considered plagiarism and will be punished severely according to the relevant regulations.***
- You will submit your projects through the " **Proje Teslimi**" class activity on Google Classroom, code **rhql3fpr**, titled " **NYP Lab 2526 Güz**". Therefore, be sure to join this class.

- **Deadline:** 09.01.2026 23:59 (with Turkish Timezone)
    - **Submissions made after this date will not be considered.**
    - The system will automatically shut down when the scheduled time arrives, so please complete your project uploads in time.
    - **Submit your .zip file only ones. Do not resubmit for correction purposes!**

- **Project controls will be conducted online by our research assistants during the final weeks, between January 13-22, 2026. Detailed explanations regarding this will be provided after project submissions are completed.**