Airport Simulation – Design Patterns Implementation

Resources Used

This project was developed using the following references:

"Design Patterns: Elements of Reusable Object-Oriented Software" (Gamma et al.)

- Provided foundational knowledge on **Factory, Observer, and Strategy Patterns**.
- Helped structure the implementation based on best practices in object-oriented design.

"Head First Design Patterns" (Freeman & Freeman)

- Offered practical Java-based examples of the **Observer and Strategy Patterns**.
- The **Strategy Pattern example** helped refine the **gate allocation switching mechanism** in Airport.java.

Oracle Java Documentation

- Used for **best practices** in Java class design, interfaces, and exception handling.
- Guided the implementation of **abstract classes (Airplane.java) and interfaces (ResourceAllocationStrategy.java)**.

JUnit 5 Documentation

- Provided best practices for **assertions, exception handling, and unit testing**.
- Guided the creation of test cases for AirplaneFactoryTest.java, ObserverPatternTest.java, and StrategyPatternTest.java.

Gradle Documentation

- Assisted in configuring **build automation, dependency management, and quality checks**.
- Helped resolve initial Gradle wrapper setup issues.

GitHub Actions Documentation

- Used to configure the **CI/CD pipeline**, ensuring automatic testing and code quality checks.

Implementation Choices

The choice of design patterns was based on their suitability for specific aspects of airport operations:

Factory Pattern

- **Why Chosen:** The Factory Pattern allows flexible and scalable object creation.
- **Adaptation:** Used to create different airplane types dynamically, preventing direct instantiation of subclasses (CommercialAirplane.java, CargoAirplane.java).

Observer Pattern

- **Why Chosen:** The event-driven nature of flight status updates aligns well with the Observer Pattern.
- **Adaptation:** Implemented in FlightEvent.java, where multiple control towers (ControlTower.java) receive notifications when flight statuses change.

Strategy Pattern

- **Why Chosen:** The Strategy Pattern allows runtime flexibility in how gates are assigned.
- **Adaptation:** Implemented in Airport.java, allowing gate allocation strategies to be switched dynamically between BasicAllocation.java and PeakHourAllocation.java.

Challenges Faced

Gradle Wrapper Issues

- **Problem:** Initial Gradle builds failed due to missing wrapper files.
- **Solution:** Regenerated the Gradle wrapper and committed necessary files to the repository.

GitHub Actions Build Errors

- **Problem:** CI/CD pipeline initially failed due to Checkstyle violations.
- **Solution:** Adjusted Checkstyle configuration and reformatted code to comply with project standards.

Dynamic Strategy Switching

- **Problem:** Ensuring that the airport could dynamically switch between different gate allocation strategies.
- **Solution:** Implemented a setter method in Airport.java to allow runtime switching of gate allocation strategies.

Design Pattern Usage in Code

Factory Pattern

- **Implemented In:** AirplaneFactory.java, CommercialAirplane.java, CargoAirplane.java
- **How It Works:**
    - Uses a static factory method to create airplane objects dynamically.
    - Allows new airplane types to be added without modifying existing logic.

Observer Pattern

- **Implemented In:** FlightEvent.java, ControlTower.java
- **How It Works:**
    - FlightEvent acts as the subject, while ControlTower instances are observers.
    - When a flight status changes, all registered control towers receive updates automatically.

Strategy Pattern

- **Implemented In:** Airport.java, BasicAllocation.java, PeakHourAllocation.java
- **How It Works:**
    - Defines a common interface ResourceAllocationStrategy.java.
    - Airport.java allows runtime selection of gate allocation strategies.
    - BasicAllocation.java follows a first-come, first-served model, while PeakHourAllocation.java prioritizes commercial flights.

Screenshots

SpotBugs

SpotBugs Report

Project Information

Project: ser316assign5 (spotbugsMain)

SpotBugs version: 4.7.3

Code analyzed:

- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\airport\Airport.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\factory\Airplane.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\factory\AirplaneFactory.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\factory\CargoAirplane.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\factory\CommercialAirplane.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\Main.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\observer\ControlTower.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\observer\FlightEvent.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\observer\FlightObserver.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\strategy\BasicAllocation.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\strategy\PeakHourAllocation.class
- D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\build\classes\java\main\strategy\ResourceAllocationStrategy.class

Metrics

133 lines of code analyzed, in 12 classes, in 5 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

Checkstyle

CheckStyle Audit

Designed for use with CheckStyle and Ant.

Summary	
Files	**Errors**
12	0

Files	
Name	**Errors**
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\airport\Airport.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\factory\Airplane.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\factory\AirplaneFactory.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\factory\CargoAirplane.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\factory\CommercialAirplane.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\Main.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\observer\ControlTower.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\observer\FlightEvent.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\observer\FlightObserver.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\strategy\BasicAllocation.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\strategy\PeakHourAllocation.java	0
D:\Documents\school work\ASU\Fall 25\Session A\SER 316\Assignment 5\ser316assign5\src\main\java\strategy\ResourceAllocationStrategy.java	0

JUnit Test Execution

Test Summary

15	0	0	0.044s
tests	failures	ignored	duration

100% successful

Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
factory	3	0	0	0.021s	100%
observer	3	0	0	0.010s	100%
strategy	3	0	0	0.007s	100%
test	6	0	0	0.006s	100%

JaCoCo Code Coverage

ser316assign5

ser316assign5

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
factory		62%		100%	5	12	7	22	5	10	0	4
observer		100%		100%	0	8	0	20	0	7	0	2
strategy		100%		100%	0	5	0	8	0	4	0	2
airport		100%		n/a	0	3	0	7	0	3	0	1
Total	29 of 183	84%	0 of 8	100%	5	28	7	57	5	24	0	9