

OBJECT ORIENTED PROGRAMMING (CSE -2143) MINI PROJECT REPORT ON

**TITLE:AIRLINE RESERVATION SYSTEM**

*SUBMITTED TO*

**Department of Computer Science & Engineering**

*By*

1. Name - PRAKHAR ARYA

Registration No. - 220905458

Roll No. - 49

1. Name - NAMRATA RAJ

Registration No. - 220905550

Roll No. - 61

1. Name - AAROHI JADHAV

Registration No. - 220905472

Roll No. - 51

1. Name - ARYAN SINGH

Registration No. - 220905492

Roll No. - 53

SEMESTER – 3

Name & Signature of Evaluator 1 Name & Signature of Evaluator 2

**Table Of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| Serial No. | Chapter | Chapter Title | Page no |
|  |  |  |  |
| 1 | 1 | INTRODUCTION | 3 |
| 2 | 1.1 | KEY FEATURES | 3 |
| 3 | 2 | OBJECT ORIENTED PROGRAMMING | 3 |
| 4 | 2.1 | INTRODUCTION | 3 |
| 5 | 2.2 | USE OF OBJECTED ORIENTED PROGRAMMING IN AIRLINE TICKET SYSTEM | 3 |
|  |  |  |  |
| 6 | 2.3 | CONCEPTS USED IN DETAIL | 4 |
| 7 | 2.3.1 | ENCAPSULATION | 4 |
| 8 | 2.3.2 | INHERITENCE | 4 |
| 9 | 2.3.3 | POLYMORPHISM | 4 |
| 10  11  12  13 | 2.3.4  2.3.5    2.3.6  3  4 | ABSTRACTION  GENERICS  EXCEPTIONS  METHODOLOGY  RESULTS | 4  5  5  6  6 |
| 13  14 | 5  6 | ENHANCEMENTS  REFERENCES | 6  6 |
|  |  |  |  |

1.INTRODUCTION

The Simple Flight Reservation System is a Java project that focuses on basic object-oriented programming principles and functionalities. This project allows users to add, modify, and delete flight reservations, including flight date, time, venue, and meal choices, for different flights spanning different dates, destinations, and models. Additionally, it provides options to display records in different ways. This project serves as a practical implementation of OOP concepts like making different classes, exception handling, abstraction through interfaces, and creating objects while addressing a simplified flight reservation scenario.

* 1. **KEY FEATURES**

Flight Reservation Management: -Users can add new flight reservations, specifying flight date, time and venue.

Existing reservations can be modified, allowing users to update flight details or cancel their already made bookings.

Record Display Options: -The system offers multiple ways to display flight reservation records along with flight details.

**2. OBJECT ORIENTED PROGRAMMING**

**2.1 INTRODUCTION**

Object-Oriented Programming (OOP) plays a crucial role in a flight reservation system. It enables the representation of system entities as classes, such as "Model" "Passenger" and "FlightDetails." Inheritance allows the creation of different models of the Airbus aircrafts like “A380” and “A350” from our “Model” class. Polymorphism simplifies the handling of different flight descriptions by overriding, promoting code uniformity. Abstraction conceals unnecessary details, focusing on essential features like seat availability. OOP encourages modularity and code reusability, facilitating system maintenance. Data encapsulation enhances data security by using private methods and variables. OOP provides flexibility for accommodating changing requirements and extending system functionality. It promotes a structured and organized approach to system design.

**2.2 USE IN AIRLINE RESERVATION SYSTEM**

Object-Oriented Programming (OOP) plays a crucial role in a flight reservation system. It enables the representation of system entities/ Inheritance allows the creation of specialized classes inheriting common features from a general class. Polymorphism simplifies the handling of different flight types, promoting code uniformity. Abstraction conceals unnecessary details, focusing on essential features like seat availability and payment processing. OOP encourages modularity and code reusability, facilitating system maintenance. Data encapsulation enhances data security. OOP provides flexibility for accommodating changing requirements and extending system functionality. It promotes a structured and organized approach to system design.

**2.3 CONCEPTS OF OBJECT ORIENTED PROGRAMMING USED IN DETAIL**

**2.3.1 ENCAPSULATION**

In Java, encapsulation is a core concept in Object-Oriented Programming (OOP). It involves bundling data and methods within a class, controlling access to them using access modifiers. Encapsulation is achieved by making data attributes private or using restricted access modifiers, while exposing a controlled interface through public methods.

**USE IN AIRLINE RESERVATION SYSTEM -** In a flight reservation system, encapsulation is employed to enhance data security and code organization. Private access modifiers are used to safeguard sensitive data like passenger details. Controlled access through public methods ensures data is only manipulated according to defined rules and validations. Data validation and error handling are integrated into class methods. Encapsulation promotes well-structured and organized code, separating internal details from external interfaces.

**2.3.2 INHERITENCE**

In Java, inheritance where one class (the child or subclass) can inherit attributes and methods from another class (the parent or superclass). It promotes code reusability and a hierarchical class structure. The `extends` keyword is used to establish inheritance relationships in Java. The parent class defines common attributes and methods, and the child class can extend and specialize them.

**USE IN AIRLINE RESERVATION SYSTEM-**In an airline reservation system, inheritance is employed for efficient organization. A superclass captures shared attributes for passengers facilitating data reuse. The "FlightDetails" base class represents common flight information, subclasses handling specific details.

**2.3.3 POLYMORPHISM**

Polymorphism in Java allows different classes to be treated as instances of a common superclass, promoting code flexibility and extensibility. There are two types of polymorphism: compile-time, achieved through method overloading, and runtime, achieved through method overriding.

**USE IN AIRLIINE TICKET SYSTEM-**Polymorphism in an airline ticket system streamlines code and promotes flexibility. It's applied in multiple areas like getting flight details and modifying the seat matrix.

**2.3.4 ABSTRACTION**

Abstraction in Java is a fundamental OOP concept that simplifies complex systems by defining abstract classes and methods, serving as blueprints for other classes to follow. Abstract classes cannot be instantiated and are declared with the `abstract` keyword or by using Interface.

**USE IN AIRLINE TICKET SYSTEM-**Abstraction is a crucial organizational concept in an airline ticket system, simplifying code and enhancing efficiency. It is effectively applied to make different models like “A350” and “A380”. An abstract "Ticket" class defines common attributes and abstract methods, ensuring a consistent interface while accommodating differences.

**2.3.5 GENERICS**

Generics in Java serve as a foundational Object-Oriented Programming (OOP) concept, designed to enhance type safety and code reusability. They allow you to create classes, methods, and data structures that can operate on a variety of data types without compromising type safety.

**USE IN AIRLINE TICKET SYSTEM -** Generics play a crucial role in the organization and efficiency of an airline ticket system, much like abstraction. In the context of this airline ticket system, Generics are effectively applied when creating a ComboBox to select flights. This way, the ComboBox can work seamlessly with different flight models like "A320," "A350," and "A380" while maintaining type safety throughout the application.

**2.3.6 EXCEPTION HANDLING**

Exception handling in Java is an essential aspect of robust software development. It allows you to manage and gracefully recover from runtime errors, ensuring that your application doesn't crash unexpectedly. Exception handling in Java involves the use of try-catch blocks, which help identify and respond to exceptional conditions.

**USE IN AIRLINE TICKET SYSTEM** -In the airline ticket system, exception handling is a critical part of ensuring the application operates smoothly and handles unexpected errors.

**Age Validation**: When a passenger's age is entered, we can use a try-catch block to ensure that the input is a valid integer and that the age is 18 or older. This helps prevent runtime errors related to incorrect or out-of-range age values.

**Invalid Input**: We can use exception handling to catch and handle exceptions related to invalid or unexpected user input. For example, if a user enters non-numeric characters in the age field, the system can catch this exception and provide an informative error message.

**Index Out of Bounds:** Exception handling can be used when working with passenger lists to catch any index out-of-bounds exceptions when attempting to modify or delete passengers. This ensures that the application gracefully handles such scenarios without crashing.

**3-METHODOLOGY**

The project adopts an Agile development approach, divided into key phases. In the initiation phase, project objectives and requirements are defined. The planning phase encompasses creating a project plan, user stories, and technology stack selection. The design phase involves class diagrams and user interface design, considering polymorphism and abstraction implementation. During development, the system is built, employing Java and JavaFX, with a focus on exception handling. Testing phases include unit, integration, and system testing. Deployment is followed by user training and documentation. Maintenance ensures ongoing support, and project closure evaluates objectives and prepares for future enhancements. This methodology ensures structured development, user-focused design, and effective utilization of software principles.

**4-RESULT AND DISSCUSION**

The project successfully created a Java-based Airline Ticket System with a user-friendly interface. It effectively utilizes Object-Oriented Programming principles, such as polymorphism and abstraction, to manage different aircraft models. Exception handling ensures robustness. A ComboBox implements generics, providing flexibility. The project promotes efficient airline ticket management, allowing passengers to book, update, and delete tickets while displaying seat availability. It can easily expand to accommodate additional features, making it suitable for real-world airline operations. The system fulfils its primary objectives of reservation management, demonstrating the benefits of structured development and good coding practices.

**5-CONCLUSION and FUTURE ENHANCEMENTS**

In summary, our airline ticket system project, built using Java and OOP principles, has successfully demonstrated the benefits of modularity and maintainability. OOP concepts such as encapsulation, inheritance, and polymorphism have enabled a well-organized and flexible solution. The use of classes and objects has enhanced code readability and reusability.

To enhance the Airline Ticket System, we can implement multi-threading to efficiently handle concurrent user interactions, providing a seamless experience. Introducing additional fields like meal preferences (veg, non-veg, etc.) and flight classes (business, economy, first) will cater to diverse passenger needs. Improved formatting and design will enhance user interface aesthetics. Furthermore, integrating payment gateways for real bookings, incorporating user authentication, and expanding the database to include more flight options are key directions for future development.

**6-REFERENCES**

1- THE COMPLETE REFERENCE IN JAVA, TATA MCGRAW HILL, 11e

2- Object Oriented Programming Material, Manipal Institute of Technology

3- https://stackoverflow.com/

4 – https://www.geeksforgeeks.org/java/