Railway Management System (RRMS)

Project Overview

The **Railway Management System (RRMS)** is a Java-based project designed to simulate essential functionalities in a railway reservation and management system. This project adheres strictly to the **SOLID principles** of software design to ensure scalability, flexibility, and maintainability. The system includes features like ticket booking, ticket cancellation, passenger management, and report generation, demonstrating the effective use of object-oriented principles.

Features

1. **Booking Service**: Enables passengers to book and cancel tickets.
2. **Passenger Management**: Manages and stores information about passengers.
3. **Payment Processing**: Allows for secure payment processing for bookings.
4. **Report Generation**: Generates various reports such as booking reports and revenue reports.

Class Overview

Core Classes

- **Main**: The entry point of the application, used to demonstrate booking a ticket, adding passengers, and processing payments.
- **RailwayService**: Coordinates ticket booking and payment processing by integrating multiple services.
- **PassengerRepository**: Manages a list of passengers and allows adding and retrieving passenger data.
- **PassengerTrain**: Represents a passenger train, implementing the `ITrain` interface with schedule and route details.

Services and Interfaces

- **BookingService** (Interface): Defines the contract for booking and cancellation services.
  - `TicketBooking`: Implements ticket booking functionality.
  - `TicketCancellation`: Implements ticket cancellation functionality.
- **PassengerService** (Interface): Defines the methods for adding and retrieving passenger data.
- **PaymentProcessor** (Interface): Outlines payment processing methods.
  - `CreditCardPayment`: Simulates credit card payment processing.
- **ReportGenerator** (Interface): Outlines methods for generating reports.
  - `BookingReport`: Generates reports on ticket bookings.

- `RevenueReport`: Generates reports on revenue collected.

SOLID Principles Used

1. **Single Responsibility Principle (SRP)**: Each class in this project has a single responsibility, such as managing bookings or handling payments.
2. **Open/Closed Principle (OCP)**: New functionality, such as adding a new type of report, can be implemented without modifying existing code.
3. **Liskov Substitution Principle (LSP)**: Classes that implement interfaces, like `ITrain`, can replace each other without affecting functionality.
4. **Interface Segregation Principle (ISP)**: Specific interfaces like `BookingService` and `PaymentProcessor` ensure that classes only implement necessary methods.
5. **Dependency Inversion Principle (DIP)**: The `RailwayService` class depends on abstractions rather than concrete implementations for flexibility.

Installation and Usage

1. **Prerequisites**: Ensure that Java (JDK 8 or above) is installed on your system.
2. **Clone Repository**:

   ```
   bash
   Copy code
   git clone https://github.com/your-username/RailwayManagementSystem.git
   ```

3. **Compile and Run**:
   - Compile all Java files:

     ```
     bash
     Copy code
     javac RailwayManagement/*.java
     ```

   - Run the main program:

     ```
     bash
     Copy code
     java RailwayManagement.Main
     ```

Example Usage

The `Main` class demonstrates a sample workflow:

1. A new passenger is added to the system.
2. A train is defined with a route and schedule.
3. A ticket is booked, and payment is processed for the booking.

Folder Structure

```
RailwayManagement/

├── Main.java                  # Entry point of the application
├── RailwayService.java        # Core service class for railway operations
├── PassengerRepository.java   # Manages passenger data
├── PassengerTrain.java        # Represents a train
├── interfaces/
│   ├── BookingService.java    # Defines methods for booking operations
│   ├── PassengerService.java  # Defines methods for passenger management
│   ├── PaymentProcessor.java  # Defines methods for payment processing
│   └── ReportGenerator.java   # Defines methods for generating reports
└── implementations/
    ├── TicketBooking.java     # Handles ticket booking
    ├── TicketCancellation.java # Handles ticket cancellation
    ├── CreditCardPayment.java # Simulates credit card payments
    ├── BookingReport.java     # Generates booking reports
    └── RevenueReport.java     # Generates revenue reports
```

Future Enhancements

- Add support for different payment methods (e.g., PayPal, bank transfer).
- Expand report generation to include detailed passenger and train statistics.
- Implement a graphical user interface (GUI) for enhanced user interaction.