Payment Gateway

Description

This Usecase provides a payment processing system that supports multiple payment methods: Credit Card, PayPal, and Bitcoin. Users can choose a payment method and process payments securely. The application utilizes the Factory Method pattern to create instances of different payment processors based on the user's choice. The use of the Factory Method pattern allows for easy extension to support additional payment methods in the future.

Files:

1. **PaymentProcessor.java**
   Description: Abstract class defining the interface for payment processors. It includes methods for processing payments and getting the type of processor.
2. **CreditCardProcessor.java**
   Description: Concrete implementation of the PaymentProcessor interface for processing payments via Credit Card. It handles payment processing specific to Credit Card transactions.
3. **PayPalProcessor.java**
   Description: Concrete implementation of the PaymentProcessor interface for processing payments via PayPal. It manages payment processing for PayPal transactions.
4. **BitcoinProcessor.java**
   Description: Concrete implementation of the PaymentProcessor interface for processing payments via Bitcoin. It deals with payment processing for Bitcoin transactions.
5. **PaymentProcessorFactory.java**
   Description: Abstract factory class that declares the method for creating PaymentProcessor instances. It provides a common interface for the creation of different types of payment processors.
6. **SimplePaymentProcessorFactory.java**
   Description: Concrete factory class that implements PaymentProcessorFactory. It creates instances of the appropriate PaymentProcessor based on user input and logs operations.
7. **PaymentProcessorManager.java**
   Description: Main application class that handles user interaction. It prompts the user to select a payment method, enter the amount, and processes the payment using the selected payment processor. It also logs significant actions and errors.

Design Patterns Used:

- **Factory Method Pattern:** This pattern is used to create instances of different payment processors (CreditCardProcessor, PayPalProcessor, BitcoinProcessor) without specifying the exact class to be instantiated. It provides flexibility in adding new payment methods by creating new concrete processors and updating the factory class.

- **Logging:** The application uses the Logger class to record significant events and errors, such as the creation of payment processors and invalid input. This helps in debugging and monitoring the application's operation.