This document outlines the changes made to a payment processing system to incorporate the Factory Design Pattern. The goal was to decouple the instantiation of payment methods from the client code, enabling easier extension and maintenance.

**Changes Made**

1. **PaymentMethod and PaymentMethodFactory Interfaces**

Introduced two interfaces: PaymentMethod for defining a common payment method behavior, and PaymentMethodFactory for creating payment method objects.

java

*public interface PaymentMethod {*

*boolean pay(double amount);*

*}*

*public interface PaymentMethodFactory {*

*PaymentMethod createPaymentMethod();*

*}*

**2. Concrete Payment Method Implementations**

Implemented CreditCardPayment and PayPalPayment classes that adhere to the PaymentMethod interface, encapsulating the logic specific to each payment method.

java

*public class CreditCardPayment implements PaymentMethod { /\* Implementation \*/ }*

*public class PayPalPayment implements PaymentMethod { /\* Implementation \*/ }*

**3. Payment Method Factories**

Introduced CreditCardPaymentFactory and PayPalPaymentFactory to instantiate respective payment method objects, implementing the PaymentMethodFactory interface.

java

*public class CreditCardPaymentFactory implements PaymentMethodFactory { /\* Implementation \*/ }*

*public class PayPalPaymentFactory implements PaymentMethodFactory { /\* Implementation \*/ }*

**4. Refactoring Payment Processing**

Replaced direct instantiation of payment methods within the order processing methods with factory-created instances. Introduced a generic processOrderWithPaymentFactory method accepting a PaymentMethodFactory.

java

*public boolean processOrderWithPaymentFactory(PaymentMethodFactory paymentMethodFactory) { /\* Implementation \*/ }*

**Usage Example**

Refactored client code to use the factory for PayPal payment processing.

java

*PaymentMethodFactory payPalFactory = new PayPalPaymentFactory(customer.getEmail(), "password");*

*boolean isProcessed = order.processOrderWithPaymentFactory(payPalFactory);*

**Conclusion**

The refactoring introduces a flexible and extensible structure for payment processing. By applying the Factory Design Pattern, the system can now easily accommodate new payment methods without modifying the client code or the core processing logic.