**Exercise 4: Implementing the Adapter Pattern**

* **Create a New Java Project:**
* Begin by setting up a new Java project in your development environment. Name this project *AdapterPatternExample*.
* **Define Target Interface:**
* Create an interface named *PaymentProcessor* which represents the common interface that will be used by all payment processors. This interface should include a method *processPayment*(double amount) that allows processing a payment with a specified amount.
* **Implement *Adaptee* Classes:**
  + Implement classes for different payment gateways. Each class will have its own specific method for processing payments. These classes are:
* **PayPal:** This class should have a method *sendPayment*(double amount) which processes payments using PayPal.
* **Stripe:** This class should have a method *makePayment*(double amount) for processing payments through Stripe.
* **Square:** This class should have a method *pay*(double amount) for processing payments using Square.
* **Implement the Adapter Class:**
* Create adapter classes that implement the *PaymentProcessor* interface and translate the common *processPayment()* method into the specific methods of the corresponding payment gateway classes. Each adapter will wrap an instance of one of the adaped classes and call the appropriate method:
* ***PayPalAdapter*:** Adapts the PayPal class. Implements *PaymentProcessor* and translates *processPayment*(double amount) to *sendPayment*(double amount).
* ***StripeAdapter*:** Adapts the Stripe class. Implements *PaymentProcessor* and translates *processPayment*(double amount) to *makePayment*(double amount).
* ***SquareAdapter*:** Adapts the Square class. Implements *PaymentProcessor* and translates *processPayment*(double amount) to pay(double amount).
* **Test the Adapter Implementation:**
* Create a test class to demonstrate how the adapters can be used to process payments through different payment gateways. This involves:
* Instantiating each of the payment gateway classes (*PayPal, Stripe, Square*).
* Creating an adapter for each gateway. And using these adapters to process payments through the common *processPayment()* method.
* **Summary of Classes:**
* ***PaymentProcessor Interface:*** Defines a common method processPayment(double amount) that all adapters will implement.
* ***PayPal:*** Adaptee class with a method sendPayment(double amount).
* ***Stripe:*** Adaptee class with a method makePayment(double amount).
* Square: Adaptee class with a method pay(double amount).
* ***PayPalAdapter***: Adapter class that implements PaymentProcessor and translates processPayment() to sendPayment() of PayPal.
* ***StripeAdapter***: Adapter class that implements PaymentProcessor and translates processPayment() to makePayment() of Stripe.
* ***SquareAdapter***: Adapter class that implements PaymentProcessor and translates processPayment() to pay() of Square.

This setup allows the *PaymentProcessor* interface to be used uniformly to interact with different payment gateways, providing flexibility and simplifying integration.

Here is the code – [link](https://github.com/Hyperstrom/Aniket-Pal_5017587/tree/main/WEEK-1/2.Design%20Patterns%20and%20Principles/Excercise-4)

Here is the output of the code

