The Strategy pattern is a behavioral design pattern that allows a set of behaviors to be turned into objects and made interchangeable inside an original context object.

This pattern is used when there are multiple algorithms for a specific task, and the client decides the actual implementation to be used at runtime.

First, we create a Strategy interface defining an action and concrete strategy classes implementing the Strategy interface.

Next, we create a Context class that uses a Strategy. The Context class has a reference to a Strategy object, and delegates executing the behavior to the linked strategy object. The Context class can change the way it performs its work by replacing the currently linked strategy object with another one.

The Strategy pattern is commonly used especially in various frameworks to provide users a way to change the behavior of a class without extending it.

One of the best example of strategy pattern is Collections.sort() method that takes Comparator parameter. Based on the different implementations of Comparator interfaces, the Objects are getting sorted in different ways.

Example :

We will try to implement a simple Shopping Cart where we have two payment strategies - using Credit Card or using PayPal.

interface PaymentStrategy {  
 public void pay(int amount);  
}  
  
class CreditCardStrategy implements PaymentStrategy {  
  
 private String name;  
 private String cardNumber;  
 private String cvv;  
 private String dateOfExpiry;  
  
 public CreditCardStrategy(String nm, String ccNum, String cvv, String expiryDate){  
 this.name=nm;  
 this.cardNumber=ccNum;  
 this.cvv=cvv;  
 this.dateOfExpiry=expiryDate;  
 }  
 @Override  
 public void pay(int amount) {  
 System.*out*.println(amount +" paid with credit/debit card");  
 }  
  
}  
  
class PaypalStrategy implements PaymentStrategy {  
  
 private String emailId;  
 private String password;  
  
 public PaypalStrategy(String email, String pwd){  
 this.emailId=email;  
 this.password=pwd;  
 }  
  
 @Override  
 public void pay(int amount) {  
 System.*out*.println(amount + " paid using Paypal.");  
 }  
  
}  
  
class PaymentContext {  
 private PaymentStrategy paymentStrategy ;  
  
 PaymentContext(PaymentStrategy paymentStrategy){  
 this.paymentStrategy = paymentStrategy ;  
 }  
  
 void makePayment( int amount ){  
 paymentStrategy.pay(amount);  
 }  
}  
  
public class StrategyDesignPattern {  
 public static void main(String[] args) {  
 PaymentStrategy paypalStrategy = new PaypalStrategy("paypal\_email@gmail.com" , "paypal\_pwd" ) ;  
  
  
 PaymentStrategy creditCardStrategy =  
 new CreditCardStrategy("Pankaj Kumar", "1234567890123456", "786", "12/15") ;  
  
 PaymentContext paymentContext = new PaymentContext(paypalStrategy) ;  
  
 paymentContext.makePayment(120);  
  
 paymentContext = new PaymentContext(creditCardStrategy) ;  
  
 paymentContext.makePayment(150);  
 }  
}

Output :

120 paid using Paypal.

150 paid with credit/debit card

Process finished with exit code 0