**ISE Q2**
**Name: Ritik Singh**
**2018130052**

```java
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;

interface PaymentStrategy {

    public void pay(int amount);
}
```

```java
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");
    }

}
```

```java
class PaytmStrategy implements PaymentStrategy {

    private String name;
    private String password;

    public PaytmStrategy(String name, String pwd){
        this.name=name;
        this.password=pwd;
    }

    @Override
    public void pay(int amount) {
        System.out.println(amount + " paid using Paytm.");
    }

}
```

```java
class Item {

    private int price;
    private String name;

    public Item(String name,int cost){
        this.price=cost;
        this.name=name;
    }

    public int getPrice() {
        return price;
    }

    public String getName() {
        return name;
    }

}

class ShoppingCart {

    List<Item> items;

    public ShoppingCart(){
        this.items=new ArrayList<Item>();
    }

    public void addItem(Item item){
        this.items.add(item);
    }

    public void removeItem(Item item){
        this.items.remove(item);
    }

    public int calculateTotal(){
        int sum = 0;
        for(Item item : items){
            sum += item.getPrice();
        }
        return sum;
    }

    public void pay(PaymentStrategy paymentMethod){
        int amount = calculateTotal();
        paymentMethod.pay(amount);
    }
}
```

```java
class Main {

    public static void main(String[] args) {
        ShoppingCart cart = new ShoppingCart();

        Item item1 = new Item("shirt",1500);
        Item item2 = new Item("telephone1",200);

        cart.addItem(item1);
        cart.addItem(item2);


        cart.pay(new PaytmStrategy("Ritik Singh", "mypwd"));

        cart.pay(new CreditCardStrategy("Raj Gorhekar", "224466883355", "786", "12/15"));
    }

}
```

**Strategy pattern** is one of the **behavioral design pattern**. Strategy pattern is used when we have multiple algorithm for a specific task and client decides the actual implementation to be used at runtime.

For our example, we will try to implement a simple Shopping Cart where we have two payment strategies – using Credit Card or using Paytm.

First of all we will create the interface for our strategy, in our case to pay the amount passed as argument.

Now we will have to create concrete implementations of algorithms for payment using credit/debit card or through paypal.

Now our algorithms are ready and we can implement Shopping Cart and payment method will require input as Payment strategy.

Notice that payment method of shopping cart requires payment algorithm as argument and doesn’t store it anywhere as instance variable.

Output of above program is:

```
1700 paid using Paytm.
17000 paid with credit/debit card
```