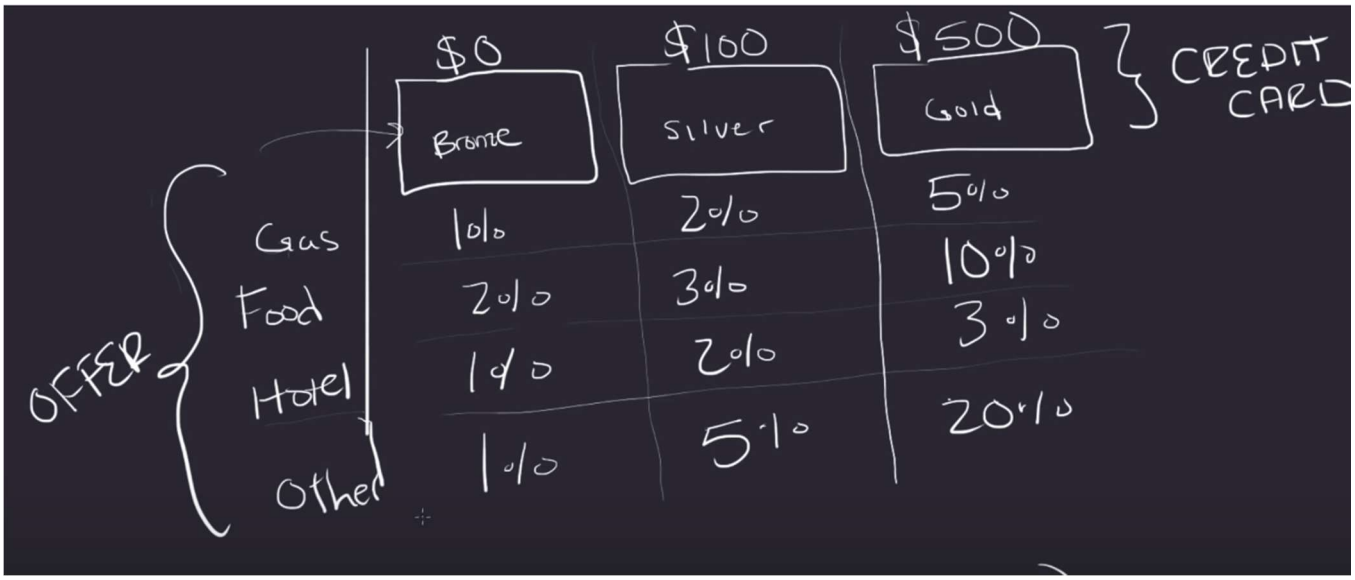
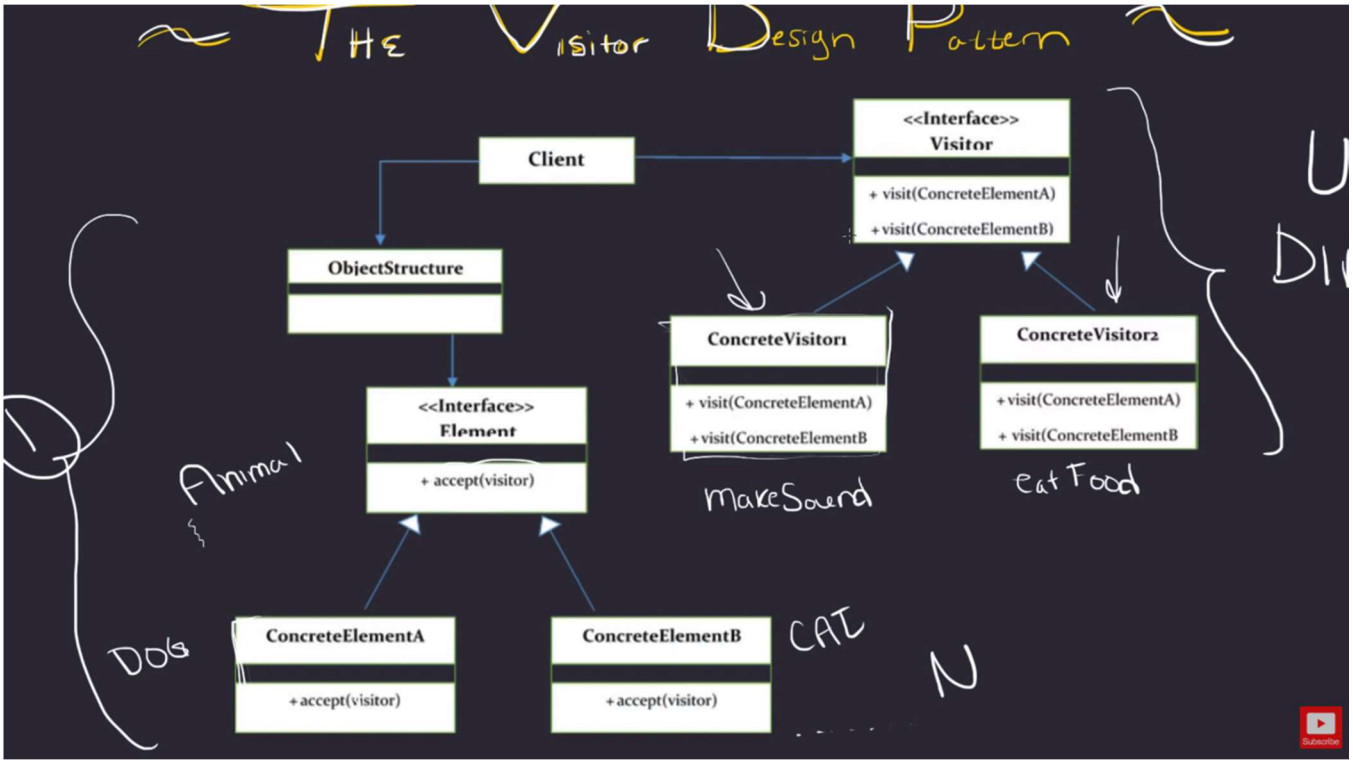
Behavioral - Visitor Design Pattern

Aslında basit bir amacı var, o da şu; var olan sınıfların hiyerarşik yapılarını ve mevcut yapılarını değiştirmeden yeni metotlar eklemek. Yeni metotlarımız visitor sınıfı üzerinde tanımlanır ve mevcut sınıflar kendilerini bu visitor sınıfa parametre olarak aktarıp gerekli işlemleri yaparlar



Offer bir nesne, Credit card ayrı bir nesnedir. Bronze,Silver ve Gold card'lar için alışverişlerde geri alınacak değerler farklıdır



Visitor'lar burada offerType'lar olacak.Sağ tarafta yer alan memeli parantezin orası

ConcreteVisitor1 (Gas Offer)

ConcreteVisitor2(Food Offer)

ConcreteVisitor3(Hotel Offer)

ConcreteVisitor4(Other Offer)

Element

```
//Interface element
interface CreditCard{
   String getName();
   void accept(OfferVisitor v);
}

//Interface visitor
interface OfferVisitor{
   void visitBronzeCreditCard(BronzeCard bronzeCard);
   void visitSilverCreditCard(SilverCard silverCard);
   void visitGoldCreditCard(GoldCard goldCard);
}
```

Element'in concrete class'ları

```
class BronzeCard implements CreditCard{
   @Override
   public String getName() {
      return "Bronze Card";
   }
   @Override
   public void accept(OfferVisitor v) {
      v.visitBronzeCreditCard(this);
   }
}
class SilverCard implements CreditCard{
   public String getName(){
      return "Silver Card";
   }
   public void accept(OfferVisitor v){
      v.visitSilverCreditCard(this);
   }
}
class GoldCard implements CreditCard{
   public String getName(){
      return "Gold Card";
   }
   @Override
   public void accept(OfferVisitor v) {
      v.visitGoldCreditCard(this);
   }
}
```

Visitor'ın Concrete Class'ları

```
class GasOfferVisitor implements OfferVisitor{
    @Override
    public void visitBronzeCreditCard(BronzeCard bronzeCard) {
        System.out.println("We are computing the cashback for a bronze card buying gas");
    }
    @Override
    public void visitSilverCreditCard(SilverCard silverCard) {
        System.out.println("We are computing the cashback for a silver card buying gas");
    }
    @Override
    public void visitGoldCreditCard(GoldCard goldCard) {
        System.out.println("We are computing the cashback for a gold card buying gas");
    }
}

class HotelOfferVisitor implements OfferVisitor{
    @Override
    public void visitBronzeCreditCard(BronzeCard bronzeCard) {
        System.out.println("We are computing the cashback for a bronze card buying hotel reservation");
    }

    @Override
    public void visitSilverCreditCard(SilverCard silverCard) {
        System.out.println("We are computing the cashback for a silver card buying hotel reservation");
    }

    @Override
    public void visitGoldCreditCard(GoldCard goldCard) {
        System.out.println("We are computing the cashback for a gold card buying hotel reservation");
    }
}
```

Main

```
public static void main(String[] args) {
    OfferVisitor hotelOfferVisitor = new HotelOfferVisitor();
    OfferVisitor gasOfferVisitor = new GasOfferVisitor();

    CreditCard bronzeCard = new BronzeCard();
    CreditCard silverCard = new SilverCard();
    CreditCard goldCard = new GoldCard();
    bronzeCard.accept(hotelOfferVisitor);
    silverCard.accept(hotelOfferVisitor);
    goldCard.accept(hotelOfferVisitor);
    bronzeCard.accept(gasOfferVisitor);
    silverCard.accept(gasOfferVisitor);
    goldCard.accept(gasOfferVisitor);
}
```