VISITOR DESIGN PATTERN

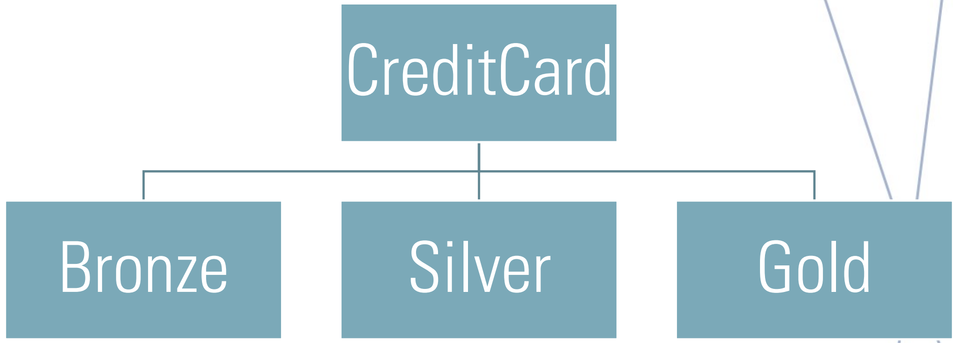
- Visitor design pattern is one of the behavioral design patterns. It is used when we have to perform an operation on a group of similar kind of Objects. With the help of visitor pattern, we can move the operational logic from the objects to another class.
- The visitor pattern consists of two parts:
  - a method called Visit() which is implemented by the visitor and is called for every element in the data structure
  - visitable classes providing Accept() methods that accept a visitor

CREDIT CARD CASE STUDY

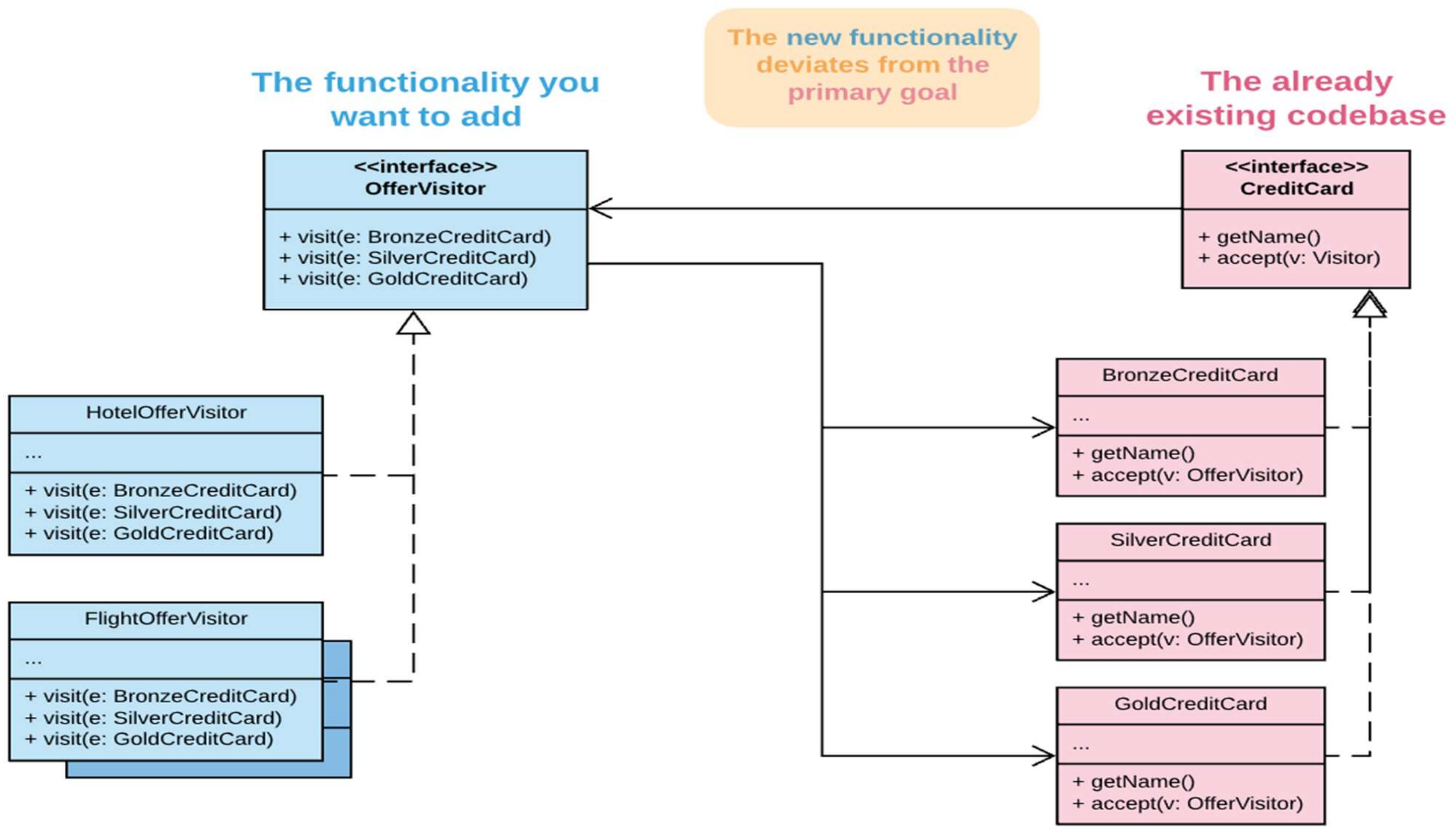
Offers

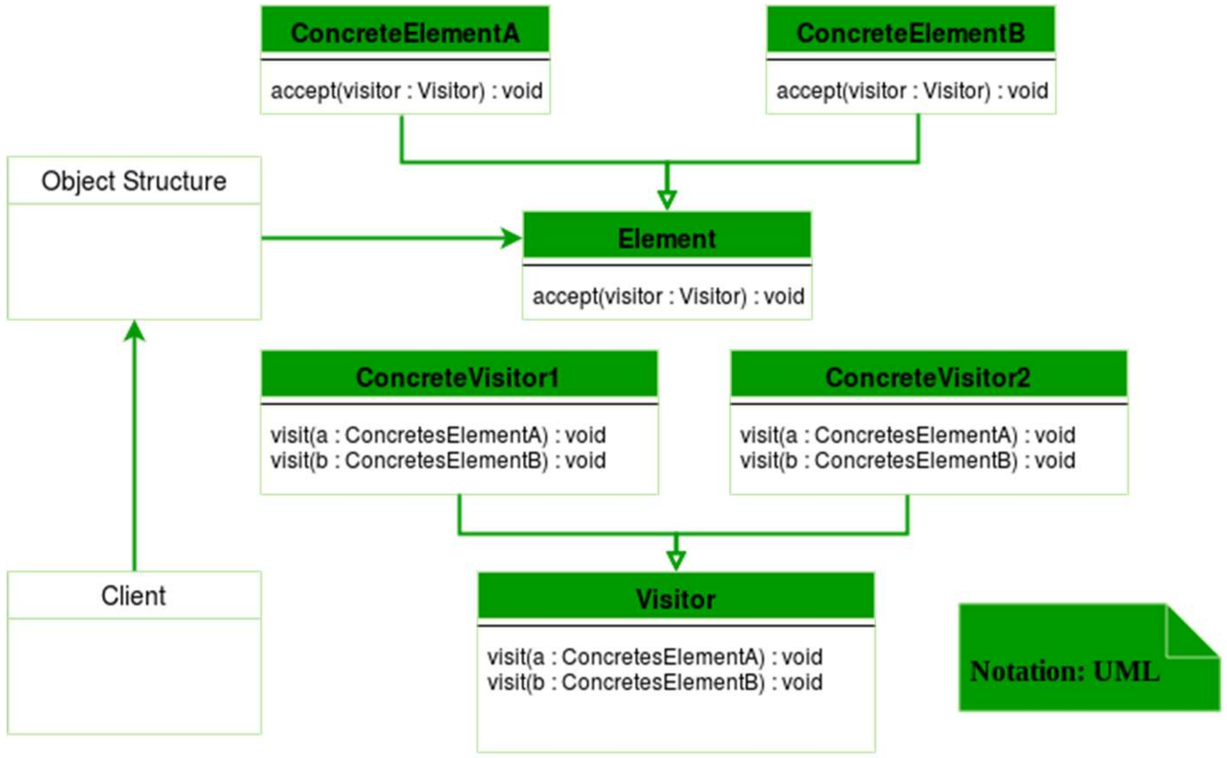
	Bronze	Silver	Gold
Food	1%	2%	5%
Gas	2%	3%	6%
Hotel	1%	2%	4%

- Credit card – Abstract class
- Methods –
    - getName()
    - computeGasOffer()
    - cmputeFoodOffer()
    - computeHotelOffer()
- All concrete classes inheriting from credit card will implement these methods.
- Consider the business requirement change wherein the bank wants to introduce offer on flight bookings.
- We will add computeFlightOffer in CreditCard class.
- With this we will have to make changes in all the child classes. This breaks the open close principle



VISITOR IMPLEMENTATION


The functionality you want to add
The new functionality deviates from the primary goal
The already existing codebase
<<interface>> OfferVisitor
+ visit(e: BronzeCreditCard)
+ visit(e: SilverCreditCard)
+ visit(e: GoldCreditCard)
<<interface>> CreditCard
+ getName()
+ accept(v: Visitor)
HotelOfferVisitor
...
+ visit(e: BronzeCreditCard)
+ visit(e: SilverCreditCard)
+ visit(e: GoldCreditCard)
FlightOfferVisitor
...
+ visit(e: BronzeCreditCard)
+ visit(e: SilverCreditCard)
+ visit(e: GoldCreditCard)
BronzeCreditCard
...
+ getName()
+ accept(v: OfferVisitor)
SilverCreditCard
...
+ getName()
+ accept(v: OfferVisitor)
GoldCreditCard
...
+ getName()
+ accept(v: OfferVisitor)


ConcreteElementA
accept(visitor : Visitor) : void
ConcreteElementB
accept(visitor : Visitor) : void
Object Structure
Element
accept(visitor : Visitor) : void
ConcreteVisitor1
visit(a : ConcretesElementA) : void
visit(b : ConcretesElementB) : void
ConcreteVisitor2
visit(a : ConcretesElementA) : void
visit(b : ConcretesElementB) : void
Client
Visitor
visit(a : ConcretesElementA) : void
visit(b : ConcretesElementB) : void
Notation: UML

VISITOR DESIGN PATTERN

WHEN TO USE VISITOR DESIGN PATTERN

- When there is a complex object structure with many classes and different interface and client needs to perform several operations on these objects depending on their concrete classes.
- When the operations are distinct and unrelated with one another it's easy to keep organized the classes and operations using this pattern.
- When the classes in the object structure rarely change but there is a need to define new operations over the structure.
- When there is a need for adding a new operation to the visitor hierarchy or new component to the object structure we can achieve it by not polluting the existing design