Software Development

(Hatly project)

Team Members :

Name	code	department
1- Mohamed Yasser	210381	MM
2- Arwa Alaa Abd-ELHamied	200066	IT
3-Basma Adel Hassan	200116	IT
4-Yasmien Mohamed Abd-ELGayed	200432	IT
5- Wafaa Mostafa Gaber	200429	IT
6- Muhap Ayman Mahmoud	200397	IT
7-Sandra Rafla William	200166	IT
8-Menna Sayed Fayez	200394	IT
9- ALi Atef Ahmed	200247	MM
10-Lobna Khaled Koraney	200278	IT

Design Pattern

A software design pattern : is a general repeatable solution to a commonly occurring problem in software design. It represents a best practice solution to a specific design problem. These patterns are tried and tested solutions that have been developed over time by experienced software developers to help in creating more robust and efficient software systems. Design patterns can be applied to different programming languages and platforms and are meant to improve software development practices and make the development process more streamlined. Examples of software design patterns include Singleton, Factory Method, Observer, and Visitor, among others.

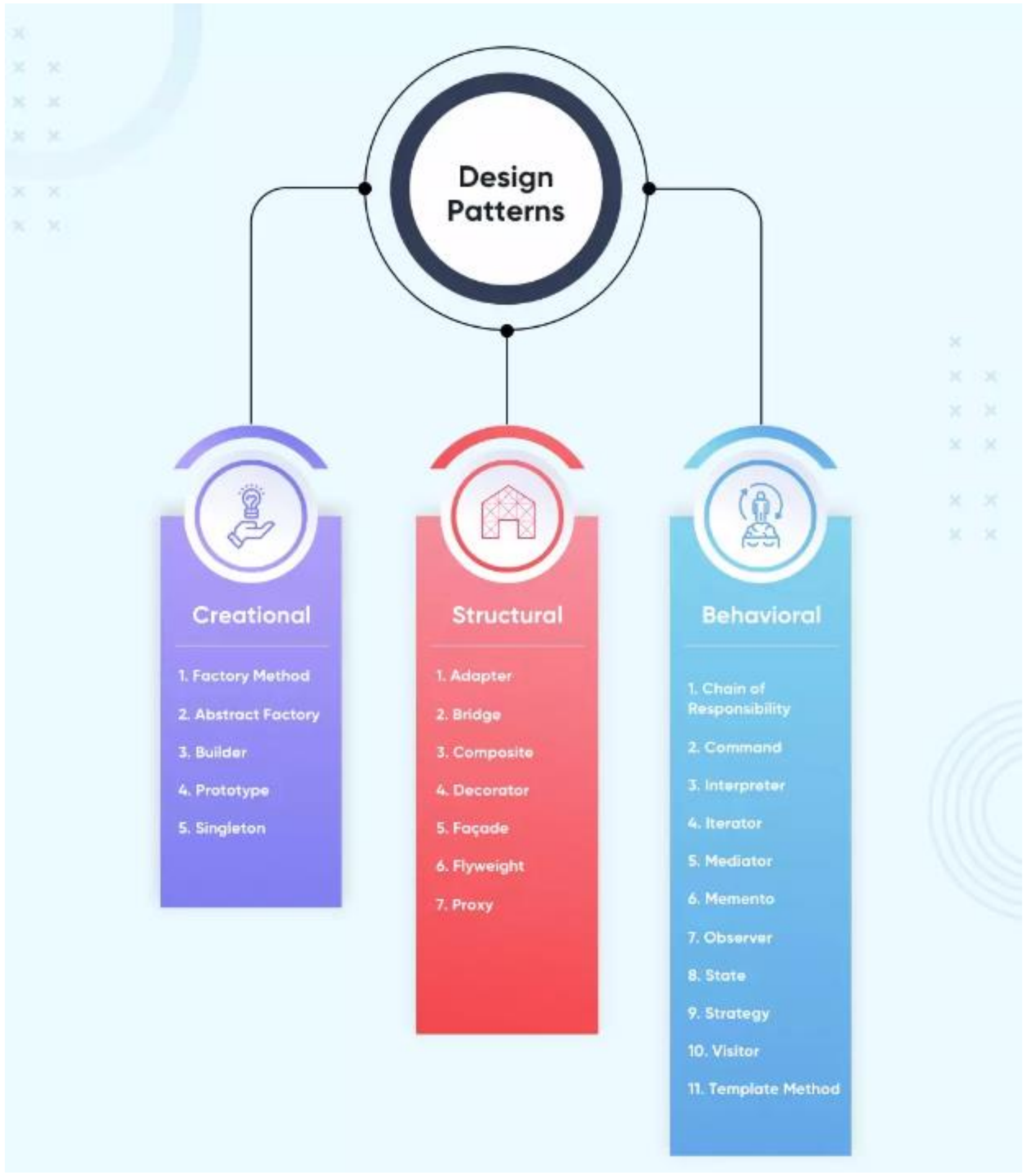
Types of software design pattern

Design patterns can be broken down into three types, organized by their intent into :

1-creational design patterns
2-structural design patterns
3-behavioral design patterns

And we will take about behavioral design patterns named **Visitor Pattern**

Types Of Software Design Pattern



Visitor Pattern Definition :

The pattern allows separate algorithms to be defined for elements of an object structure without modifying the classes on which the algorithms operate. This is achieved by defining a separate object, called a visitor, that can visit each object in the hierarchy and perform the required operation on that object.

Visitor Pattern Implement :

We have 3 actors at Hatly system :

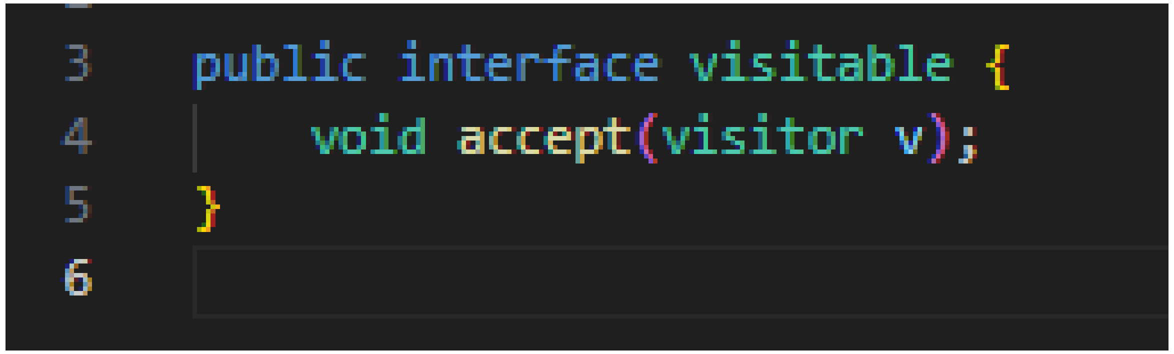
1- Customer

2- Delivery

3- Admin

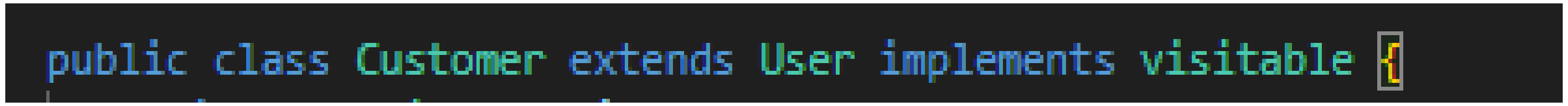
All of them we should have an access inside them so we create an interface called **visitable** and make 3 actors implements from it

>> Interface

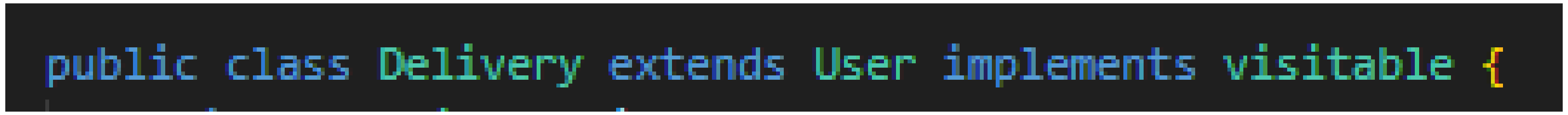


```
public interface visitable {
    void accept(visitor v);
}
```

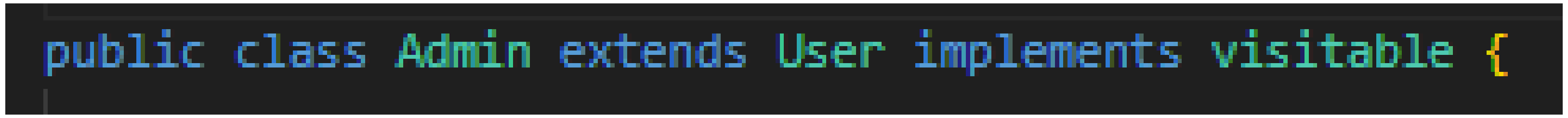
- The override of 3 functions at 3 actors in their classes
- Customer
- Delivery
- Admin



```
public class Customer extends User implements visitable {
```

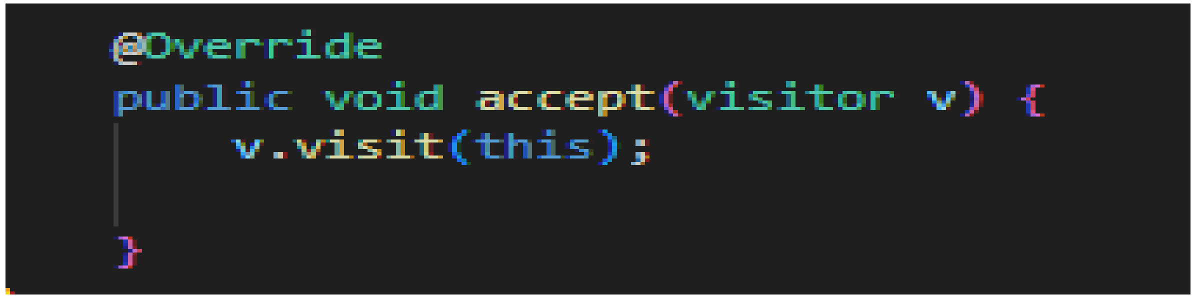


```
public class Delivery extends User implements visitable {
```



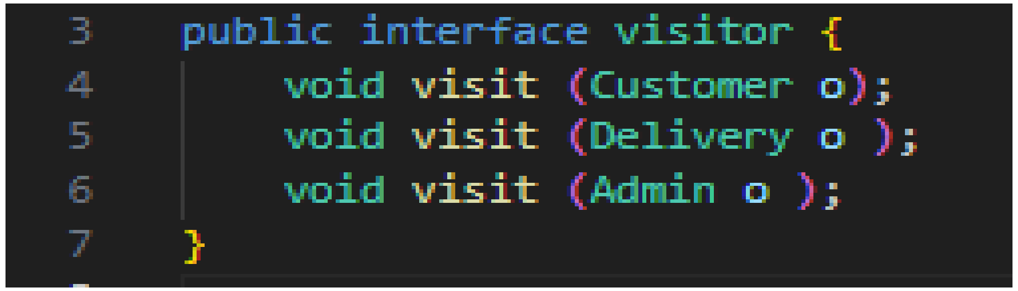
```
public class Admin extends User implements visitable {
```

And override visit function at all of them to have an access at them



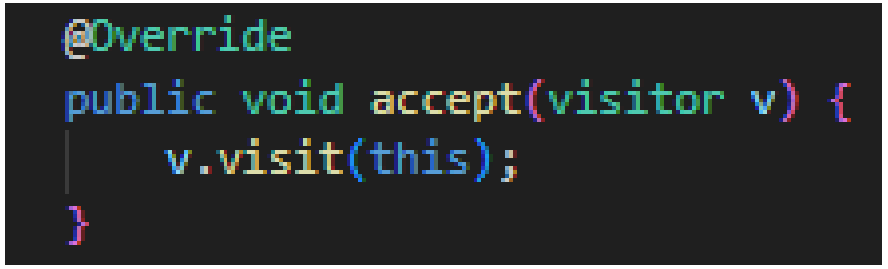
```
@Override
public void accept(visitor v) {
    v.visit(this);

}
```

- At override we take a parameter of new interface called **visitor** and declare on it 3 to take an instance of one of 3 actors
- >> Visitor interface



```
public interface visitor {
    void visit (Customer o);
    void visit (Delivery o );
    void visit (Admin o );
}
```

- And use visit function on it to have an access inside all of 3 classes



```
@Override
public void accept(visitor v) {
    v.visit(this);
}
```

Conclusion

- We implement this pattern in our design to solve the problem of updating data in specific classes that we add to visitor interface to have an access on it and controlled on it from one place only

The End

Thanks ☺