

SOLID Principles

Single Responsibility Principle (SRP)

A class should have only one responsibility or one reason to change.

SRP Problem Statement – Order Management System

You are tasked with building an Order Management System for an e-commerce platform. For each order, the system needs to:

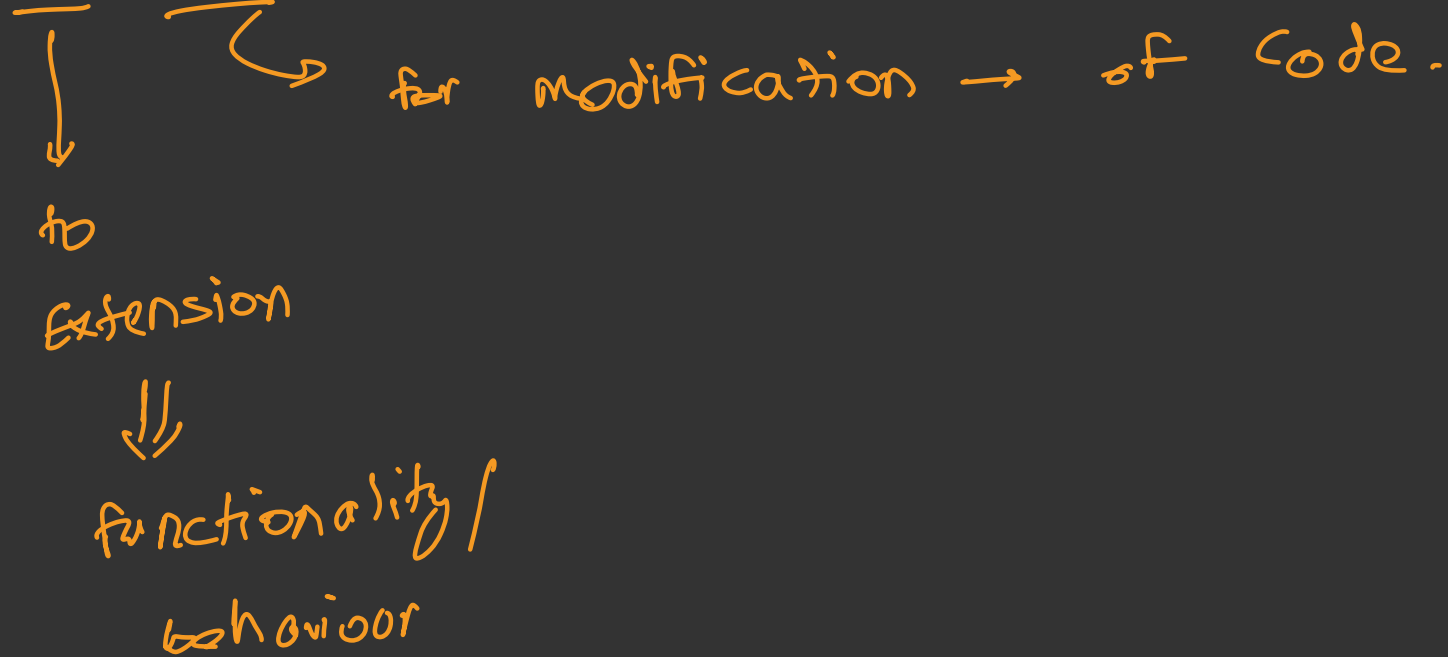
- Calculate the total price of the order (including discounts, taxes, etc.).

- Save the order details into the database.

- Send a confirmation email to the buyer.

- Update the inventory to reflect the purchased items.

Open/Closed Principle (OCP)



OCP – Notification System

You are building a **Notification Service** for an e-commerce platform. The requirements are:

Send notifications to users via different channels (e.g., **Email**, **SMS**).

The system should be extensible to support new channels in the future (e.g., **Push Notifications**, **WhatsApp**) without modifying existing code.

```
class NotificationService {
```

```
  send(type: string, message: string): void {  
    if (type === "email") {  
      console.log(`Sending EMAIL: ${message}`);  
    } else if (type === "sms") {  
      console.log(`Sending SMS: ${message}`);  
    }  
  }  
}
```

obj.

→ notify.

interface not f email.
&

```
class Email {  
  .notify()  
}
```

```
class SMS {  
  .notify()  
}
```

.....

```
class notifications {
```

```
    send() {
```

```
    }
```

```
interface {
```

```
    notify()
```

```
}
```

```
class E
```

```
{
```

```
    notify()
```

```
}
```

```
}
```

```
CS
```

```
{
```

```
    notify()
```

```
}
```

```
}
```

```
CPN
```

```
{
```

```
    N
```

```
}
```

```
CLWP
```

```
{
```

```
    w
```

```
}
```


Liskov Substitution Principle (LSP)

- Subtypes must be substitutable for their base types without altering the correctness of the program.
- A derived class should be able to replace its base class without breaking functionality.
- Is the subclass truly a “is-a” relationship?