# Design Principles

The SOLID Principles are a set of Object-Oriented design principles that have revolutionized how we write software. They are a collection of guidelines and best practices for writing clean, flexible, reusable, scalable, and maintainable code.

1. Single Responsibility Principle (SRP)
2. Open/Closed Principle (OCP)
3. Liskov Substitution Principle (LSP)
4. Interface Segregation Principle (ISP)
5. Dependency Inversion Principle (DIP)

## **Single Responsibility Principle (SRP):**

SRP states:

“A class should have one and only one reason to change, meaning that a class should have only one job.”

So with statement we says that one class should do only one job. As an example we—

class userInfo{

public function getUserName() {

return 'name';

}

public function sendMailToUser() {

return 'success';

}

}

Here userInfo class have two function which do two works, get user name and send email. So it is violate the concept of SRP. So we divided the function into two separate class. Here is also a problem. If we divided each class with each function then, there will be a lot of class which not suitable for maintainable. SRP do not says that each class only one function. There will be a lot of function. But each function do exact same work.

class userInfo{

public function getUserName() {

return 'name';

}

public function getUserAddress() {

return 'address';

}

public function getUserRole(){

return 'role';

}

}

Here userInfo class have 3 method but they all supply user information. So it is maintains SRP states.

## **Open-Closed Principle (OCP):**

OCP state:

“classes, modules, and functions should be open for extension but closed for modification.”

It means that these entities should be implemented in such a way that their functionalities can be extended to other entities without modifying the code of the original entity.