**SOLID PRINCIPLES**

1. **Single Responsibility principle**

***Every java class should perform only one functionality.***

Implementation of multiple functionalities in a single class mashup the code and if any modification is required may affect the whole class.

**public class** Student {  
 **public void** printDetails() {  
*//functionality of the method* }  
  
 **public void** calculatePercentage() {  
*//functionality of the method* }  
  
 **public void** addStudent() {  
*//functionality of the method* }  
}

The above code snippet violates the single responsibility principle. To achieve the goal of the principle, we should implement a separate class that performs a single functionality only.

**public** **class** Student

{

**public** **void** addStudent();

{

//functionality of the method

}

}

}

**public** **class** PrintStudentDetails

{

**public** **void** printDetails();

{

//functionality of the method

}

}

**public** **class** Percentage

{

**public** **void** calculatePercentage();

{

//functionality of the method

}

}

1. **Open Closed principle**

The application or module contains the methods, functions, variables, etc. The open-closed principle states that **according to new requirements **the module should be open for extension but *closed for modification*.****

The extension allows us to implement new functionality to the module.

For ex.-

In below code, we are returning vehicle number based on vehicle type like **car** or ***bike***. If we have to return vehicle number of truck, then we can add one if condition. But according to rules it will modify our code which is against rules.

We can achieve it by overriding the method. Here for each type we will take new class and extend super class called vehicle.

**public class** VehicleInfo {  
 **double number**;  
 **public double** vehicleNumber(Vehicle vcl) {  
 **if** (vcl **instanceof** Car) {  
 **return** vcl.getNumber();  
 } **else if** (vcl **instanceof** Bike) {  
 **return** vcl.getNumber();  
 }  
 **return** vcl.getNumber();*//default* }  
}

Solution

**Super class**

**public class** VehicleInfo {  
 **double number**;  
 **public double** vehicleNumber(Vehicle vcl) {  
 **return** vcl.getNumber();*//default* }  
}

**SubClasses**

**class** Car **extends** VehicleInfo {  
 @Override  
 **public double** getVehicleNumber() {  
 **return super**.getVehicleNumber();  
 }  
}  
  
**class** Bike **extends** VehicleInfo {  
 @Override  
 **public double** getVehicleNumber() {  
 **return super**.getVehicleNumber();  
 }  
}  
  
**class** Truck **extends** VehicleInfo {  
 @Override  
 **public double** getVehicleNumber() {  
 **return super**.getVehicleNumber();  
 }  
}

3.