**SOLID Principles**

Single Responsibility Principle

Single_R_P.java

```java
package assignment;

//one class=one job
class Student{
    String name;

    Student(String name){
        this.name=name;
    }
}

class Print{
    void print(Student s) {
        System.out.println("Name is : "+s.name);
    }
}

public class Single_R_P {

    public static void main(String[] args) {

        Student s=new Student("Archana");
        Print p=new Print();
        p.print(s);

    }

}
```

Problems | Javadoc | Declaration | Console | Progress

```
<terminated> Single_R_P [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_6
Name is : Archana
```

Open/Closed Principle

Open_Close_P.java

```java
package assignment;

//Add new feature without changing old cod

interface PaymentMethod{
    void pay();
}

class credit implements PaymentMethod{
    public void pay() {
        System.out.println("process CreditCard Payment");
    }
}
//this is new feature
class debit implements PaymentMethod{
    public void pay() {
        System.out.println("process DebitCard Payment");
    }
}
class Processor{
    void Process(PaymentMethod paymentMethod) {
        paymentMethod.pay();

    }
}

public class Open_Close_P {

    public static void main(String[] args) {
        Processor p=new Processor();
        p.Process(new credit());
        p.Process(new debit());
    }

}
```

Problems | Javadoc | Declaration | Console

```
<terminated> Single_R_P [Java Application] D:\sts\sts-4.31.0.RE
Name is : Archana
```

## Liskov Substitution Principle

Liskov_Substitution_P.java

```java
package assignment;

//child can replace parents

class Bird{
    void fly(){
        System.out.println("Bird can fly");
    }
}
class Sparrow extends Bird{
    void fly() {
        System.out.println("These fly fast");
    }
}


public class Liskov_Substitution_P {
  public static void main(String[] args) {
        Bird b=new Sparrow();
        b.fly();
    }
}
```

Problems Javadoc Declaration Console

```
<terminated> Liskov_Substitution_P [Java Application] D:\sts\st
These fly fast
```

## Interface Segregation Principle

Interface_Segregation_P.java

```java
package assignment;

interface Work{
    void work();
}
interface Eat{
    void eat();
}
class Humans implements Work,Eat{
    public void work() {
        System.out.println("Humans are working");
    }
    public void eat() {
        System.out.println("Humans are eating");
    }
}
class Animals implements Eat{
    public void eat() {
        System.out.println("Animals are eating");
    }

}

public class Interface_Segregation_P {
    public static void main(String[] args) {
        Humans h=new Humans();
        h.work();
        h.eat();

        Animals a=new Animals();
        a.eat();

    }

}
```

Problems Javadoc Declaration Console

```
<terminated> Interface_Segregation_P [Java Application] D:\s
Humans are working
Humans are eating
Animals are eating
```

Dependency Inversion Principle