SOLID PRINCIPLE ASSIGNMENTS OUTPUTS

BY KOMMOJU NAGA RAJESWAR ROHIT

NOTE: For Detailed codes , Please check on my repository.

ASSIGNMENT -1: Single Responsibility Principle

```java
package com.Solid;
//Login Page
class username{
    String name;
    username(String name) {
        this.name=name;
    }
    void display() {
        System.out.println("Single Responsibility of username "+this.name);
    }
}
class password{
    int pwd;
    password(int pwd) {
        this.pwd=pwd;
    }
    void display() {
        System.out.println("Single Responsibility of password "+this.pwd);
    }
}

public class Single_Responsibilty_Principle {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        username u=new username("Rohit");
        u.display();
        password p=new password(1234);
        p.display();
    }
}
```

Problems Javadoc Declaration Console Progress SonarQube Rule Description

<terminated> Single_Responsibilty_Principle [Java Application] C:\Users\91798\Downloads\spring-tools-for-eclips

```
Single Responsibility of username Rohit
Single Responsibility of password 1234
```

SOLID PRINCIPLE ASSIGNMENTS OUTPUTS

BY KOMMOJU NAGA RAJESWAR ROHIT

ASSIGNMENT -2: Open – Close Principle

```java
package com.Solid;

interface laptop{
    void ram();
    void processor();
    void price();
    //Not modifying
}

 class HP implements laptop{
     int ram_spec;
     String proc_spec;
     int price_spec;
     HP(int ram, String processor,int price){
         this.ram_spec=ram;
         this.proc_spec=processor;
         this.price_spec=price;
     }
    @Override
    public void ram() {
        // TODO Auto-generated method stub
        System.out.println("Extensible "+ this.ram_spec+" GB ram");
    }

    @Override
    public void processor() {
        // TODO Auto-generated method stub
        System.out.println("Extensible processor "+ this.proc_spec);
    }

    @Override
```

Problems Javadoc Declaration Console Progress SonarQube Rule Descrip

<terminated> open_close_principle [Java Application] C:\Users\91798\Downloads\spring-tools-for-eclips

```
Extensible 8 GB ram
Extensible processor AMD
Extensible price 50000
```

SOLID PRINCIPLE ASSIGNMENTS OUTPUTS

BY KOMMOJU NAGA RAJESWAR ROHIT

ASSIGNMENT -3: Liskov Substitution Principle

```java
package com.Solid;

class Bike {
    void run() {
        System.out.println("Bike runs");
    }
}

class Motor_Bike extends Bike{
    void sub_part() {
        super.run();
        System.out.println("Motor Bike is part of Bike");
    }
}

class Honda extends Motor_Bike{
    int mil;
    void mileage(int mil) {
        this.mil=mil;
        System.out.println("Honda Mileage is "+this.mil +" KMPL");
    }
}
public class Liskov_Substitution_Principle {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Motor_Bike mb= new Honda();
        mb.sub_part();
    }

}
```

Problems Javadoc Declaration Console Progress SonarQube Rule Description

<terminated> Liskov_Substitution_Principle [Java Application] C:\Users\91798\Downloads\spring-tools-for-eclip

```
Bike runs
Motor Bike is part of Bike
```

SOLID PRINCIPLE ASSIGNMENTS OUTPUTS

BY KOMMOJU NAGA RAJESWAR ROHIT

ASSIGNMENT -4: Interface Segregation Principle

```java
package com.Solid;

interface Callable {
    void call();
}

interface InternetEnabled {
    void internet();
}

interface CameraEnabled {
    void photo();
}

class ButtonPhone implements Callable {

    @Override
    public void call() {
        System.out.println("Supports call");
    }
}

class SmartPhone implements Callable, InternetEnabled, CameraEnabled {

    @Override
    public void call() {
        System.out.println("Supports call");
    }
```

Problems | Javadoc | Declaration | Console | Progress | SonarQube Rule Description

<terminated> Interface_Segregation_Principle [Java Application] C:\Users\91798\Downloads\spring-tools-for-e

```
Button phone supports only:
Supports call

Smart phone supports:
Supports call
Supports internet browsing
Supports taking photos
```

SOLID PRINCIPLE ASSIGNMENTS OUTPUTS

BY KOMMOJU NAGA RAJESWAR ROHIT

ASSIGNMENT -5: Dependency Inversion Principle

```java
package com.Solid;
//Abstraction (interface)
interface MessageService {
 void sendMessage(String msg, String rec);
}

//Low-level module implementing abstraction
class EmailService implements MessageService {
 @Override
 public void sendMessage(String msg, String rec) {
     System.out.println("Sending email to " + rec + ": " + msg);
 }
}

//Another low-level module
class SmsService implements MessageService {
 @Override
 public void sendMessage(String msg, String rec) {
     System.out.println("Sending SMS to " + rec + ": " + msg);
 }
}

//High-level module depending on abstraction
class NotificationManager {
 private MessageService msgser;

 // Dependency injected through constructor
 public NotificationManager(MessageService msgser) {
```

Problems | Javadoc | Declaration | Console | Progress | SonarQube Rule Description

<terminated> Dependency_Inversion_Principle [Java Application] C:\Users\91798\Downloads\spring-tools-for-

```
Sending email to user@example.com: Hello via email
Sending SMS to +1234567890: Hello via SMS
```