Programming introduction

Fundamentals to programming I

Contreras Huamani, Paul Michaell
Flores Silva, David

[1]System Engineering School
System Engineering and Informatic Department
Production and Services Faculty
San Agustin National University of Arequipa

2020-04-01

Content

Class design

symptoms of a class design

- Rigidity.
- Fragidity.
- Inmovility.
- Viscosity.
- Innesesary Complexity.
- Innesesary Repetition.
- Opacity.

1.- Rigidity

classes are difficult to change

2.- Fragidity

classes stop working

3.- Inmovility

classes are difficult to reuse

4.- Viscosity

classes are difficult to use

5.- Innesesary Complexity

overdesigned

6.- Innesesary Repetition

copy and paste

7.- Opacity

messy

principles of class designs

Eliminate the symptoms of a Class Design

- Sole responsability.
- Open and closed.
- Liskov substitution.
- Independence investment.
- Interface segregation.

1.- Sole responsability

only reason to change

2.- Open and closed

open extended and closed modify

3.- Liskov subtitution

is replaced by subtypes...

4.- Independence investment

details depend on abstractions

5.- Interface segregation

a class does not depend on interfaces that do not use

BASIC EXAMPLE

Campo.java

```
1  package ejemplo;
2
3  public interface Campo {
4      String name();
5      double price();
6  }
```

*Lapto.java

```
1  package ejemplo;
2
3  public class Lapto implements Campo {
4
5      public String name() {
6          return "Laptop: " +price();
7      }
8
9      public double price() {
10         return 400.00;
11     }
12
13 }
14
```

Componentes.java

Problems @ Java

<terminated> Funcion

```
Laptop: 400.0
400.0
```

ADVANCED EXAMPLE

Campo.java | *Lapto.java

```java
package ejemplo;

public interface Campo {
    String name();
    double price();
}
```

Campo.java | *Lapto.java | Componentes.java

```java
package ejemplo;

public class Lapto implements Campo {

    public String name() {
        return "Laptop: " +price();
    }

    public double price() {
        return 400.00;
    }
```

Campo.java | Lapto.java | Componentes.java

```java
package ejemplo;

public class Ventilador extends Componentes {
    public Ventilador(Campo componente) {
        super (componente);

        namecomp = "Ventilador";
        pricecomp = 60.00;
    }

}
```

Campo.java | Lapto.java | Componentes.java

```java
package ejemplo;

public class Soporte extends Componentes {
    public Soporte(Campo componente) {
        super (componente);

        namecomp = "Soporte";
        pricecomp = 40.00;
    }

}
```

```
-----------
Laptop: 400.0  Soporte=40.0  Ventilador=60.0
500.0
-----------
```

classes abstract and interface

An abstract class has two main functions which are:

- has no instance.
- subclasses are defined.

And an interface is declared with the keyword **INTERFACE** and it works

- method statements.
- cannot instantiate.
- appear on packages.