ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

DISEÑO DE SOFTWARE

TALLER DE REFACTORIZACIÓN

INTEGRANTES

CARLIER DURANGO CÉSAR

CARRASCO MORÁN ENRIQUE

CASTRO ASCENCIO KARLA

I TÉRMINO 2020

Table of Contents

Code Smells Refactorizados

Speculative Generality: Clase Profesor

Explicación, Asunciones y Consecuencias

Existe un parámetro inutilizado, <String Facultad> en el constructor de la clase Profesor. Es posible que este parámetro haya sido creado para una futura implementación de facultades para agrupar profesores y/o materias, además de que este dato se guarda por su cuenta en la clase InformacionAdicionalProfesor. Si esta continua en el código podría llegar a confundir y hacer más difícil el soporte si en algún momento se desea llamar a este constructor.

Captura del código Inicial

```
5  public class Profesor {
6      public String codigo;
7      public String nombre;
8      public String apellido;
9      public int edad;
10     public String direccion;
11     public String telefono;
12     public InformacionAdicionalProfesor info;
13     public ArrayList<Paralelo> paralelos;
14
15     public Profesor(String codigo, String nombre, String apellido, String facultad, int edad, String direccion, String telefono) {
16         this.codigo = codigo;
17         this.nombre = nombre;
18         this.apellido = apellido;
19         this.edad = edad;
20         this.direccion = direccion;
21         this.telefono = telefono;
22         paralelos= new ArrayList<>();
23     }
```

Constructor en la línea 15, parámetro ”String facultad”, nunca es usado en el constructor ni es atributo de la clase.

Técnica de Refactorización

Remove Parameter: Eliminamos el parámetro inutilizado del método original, y buscamos en el código cualquier referencia a este constructor y lo modificamos

Código refactorizado

```
5  public class Profesor {
6      public String codigo;
7      public String nombre;
8      public String apellido;
9      public int edad;
10     public String direccion;
11     public String telefono;
12     public InformacionAdicionalProfesor info;
13     public ArrayList<Paralelo> paralelos;
14
15     public Profesor(String codigo, String nombre, String apellido, int edad, String direccion, String telefono) {
16         this.codigo = codigo;
17         this.nombre = nombre;
18         this.apellido = apellido;
19         this.edad = edad;
20         this.direccion = direccion;
21         this.telefono = telefono;
22         paralelos= new ArrayList<>();
23     }
```

Lazy Class: Clase calcularSueldoProfesor

Explicación, Asunciones y Consecuencias

La clase calcularSueldoProfesor no tiene ninguna responsabilidad real más que un método para calcular un valor consiguiendo datos de otra clase. Dejarla dentro del código solo causará mayor cantidad de clases y un mantenimiento más difícil.

Captura del código Inicial

```java
package modelos;

public class calcularSueldoProfesor {

    public double calcularSueldo(Profesor prof){
        double sueldo=0;
        sueldo= prof.info.añosdeTrabajo*600 + prof.info.BonoFijo;
        return sueldo;
    }
}

```

Técnica de Refactorización

Inline Class. Se moverán todos los atributos y métodos a una clase que pueda tener esa responsabilidad, en este caso, lo pasaremos a la clase Profesor, y se eliminará esta clase, refactorizando los lugares donde haya sido referenciada.

Código refactorizado

```java
package modelos;

import java.util.ArrayList;

public class Profesor {
    public String codigo;
    public String nombre;
    public String apellido;
    public int edad;
    public String direccion;
    public String telefono;
    public InformacionAdicionalProfesor info;
    public ArrayList<Paralelo> paralelos;

    public Profesor(String codigo, String nombre, String apellido, int edad, String direccion, String telefono) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
        this.direccion = direccion;
        this.telefono = telefono;
        paralelos= new ArrayList<>();
    }

    public void anadirParalelos(Paralelo p){
        paralelos.add(p);
    }

    public double calcularSueldo(){
        double sueldo=0;
        sueldo= info.añosdeTrabajo*600 + info.BonoFijo;
        return sueldo;
    }


}
```

Duplicated Code

Explicación, Asunciones y Consecuencias

Existen 2 métodos dentro de la clase Estudiante, CalcularNotaInicial y CalcularNotaFinal , tienen exactamente la misma funcionalidad y reciben la misma cantidad de parámetros, esto genera una duplicación de código y aumento innecesario de líneas en la clase.

Captura del código Inicial

```java
//Calcula y devuelve la nota inicial contando examen, deberes, lecciones y talleres. El teorico y el practico se
//calcula por parcial.
public double CalcularNotaInicial(Paralelo p, double nexamen,double ndeberes, double nlecciones, double ntalleres){
    double notaInicial=0;
    for(Paralelo par:paralelos){
        if(p.equals(par)){
            double notaTeorico=(nexamen+ndeberes+nlecciones)*0.80;
            double notaPractico=(ntalleres)*0.20;
            notaInicial=notaTeorico+notaPractico;
        }
    }
    return notaInicial;
}

//Calcula y devuelve la nota final contando examen, deberes, lecciones y talleres. El teorico y el practico se
// calcula por parcial.

public double CalcularNotaFinal(Paralelo p, double nexamen,double ndeberes, double nlecciones, double ntalleres){
    double notaFinal=0;
    for(Paralelo par:paralelos){
        if(p.equals(par)){
            double notaTeorico=(nexamen+ndeberes+nlecciones)*0.80;
            double notaPractico=(ntalleres)*0.20;
            notaFinal=notaTeorico+notaPractico;
        }
    }
    return notaFinal;
}
```

Técnica de Refactorización

Extract Method: Mover este código a un nuevo método independiente y reemplace el código antiguo con una llamada al método.

Código refactorizado

```java
// Realiza en cálculo de las notas inicial y final
public double CalcularNota(Paralelo p, double nexamen,double ndeberes, double nlecciones, double ntalleres){
    double nota=0;
    for(Paralelo par:paralelos){
        if(p.equals(par)){
            double notaTeorico=(nexamen+ndeberes+nlecciones)*0.80;
            double notaPractico=(ntalleres)*0.20;
            nota=notaTeorico+notaPractico;
        }
    }
    return nota;
}
```

Innappropiade Intimacy

Explicación, Asunciones y Consecuencias

La clase Profesor usa atributos de la clase InformacionAdicionalProfesor generando este mal olor.

Captura del código Inicial

```
public double calcularSueldo(){
    double sueldo=0;
    sueldo= info.añosdeTrabajo*600 + info.BonoFijo;
    return sueldo;
}
```

Técnica de Refactorización

Move Method: Crear un nuevo método en la clase que más usa el método, luego mover el código del método anterior allí. Convertir el código del método original en una referencia al nuevo método en la otra clase o elimínelo por completo y en nuestro caso lo eliminamos.

Código refactorizado

```
public class InformacionAdicionalProfesor {
    public int añosdeTrabajo;
    public String facultad;
    public double BonoFijo;

    public double calcularSueldo(){
        double sueldo=0;
        sueldo= añosdeTrabajo*600 + BonoFijo;
        return sueldo;
    }

}
```

Middle Man

Explicación, Asunciones y Consecuencias

La clase InformacionAdicionalProfesor genera un único cálculo, el del sueldo, si la mayortía o total de informción general y base del Profesor se encuentra en una clase diferent esta no tiene razón de existencia.

Captura del código Inicial

```java
package modelos;

import java.util.ArrayList;

public class Profesor {
    public String codigo;
    public String nombre;
    public String apellido;
    public int edad;
    public String direccion;
    public String telefono;
    public ArrayList<Paralelo> paralelos;

    public Profesor(String codigo, String nombre, String
        this.codigo = codigo;
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
        this.direccion = direccion;
        this.telefono = telefono;
        paralelos= new ArrayList<>();
    }

    public void anadirParalelos(Paralelo p){
        paralelos.add(p);
    }
```

```java
package modelos;

public class InformacionAdicionalProfesor {
    public int añosdeTrabajo;
    public String facultad;
    public double BonoFijo;

    public double calcularSueldo(){
        double sueldo=0;
        sueldo= añosdeTrabajo*600 + BonoFijo;
        return sueldo;
    }

}
```

Técnica de Refactorización

Remove middle man: con esta técnica lo que buscamos en eliminar elementos intermediaron que añaden complejidad al código base, al eliminar la clase innecesaria los atributos y el único método que contenía se trasladan a la clase base Profesor.

Código refactorizado

```java
package modelos;

import java.util.ArrayList;

public class Profesor {
    public String codigo;
    public String nombre;
    public String apellido;
    public int edad;
    public String direccion;
    public String telefono;
    public ArrayList<Paralelo> paralelos;
    public int añosdeTrabajo;
    public String facultad;
    public double BonoFijo;

    public Profesor(String codigo, String nombre, String apellido, int edad, String direccion, String telefono, int año
    }

    public void anadirParalelos(Paralelo p){
        paralelos.add(p);
    }

    public double calcularSueldo(){
        double sueldo=0;
        sueldo= añosdeTrabajo*600 + BonoFijo;
        return sueldo;
    }

}
```

Long Paramether List

Explicación, Asunciones y Consecuencias

El método calculoNota definido en la clase Estudiante recibe una muy larga lista de parámetros.

Captura del código Inicial

```java
// Realiza en cálculo de las notas parciales ya sea inicial o final.
public double CalcularNota(Paralelo p, double nexamen,double ndeberes, double nlecciones, double ntalleres){
    double nota=0;
    for(Paralelo par:paralelos){
        if(p.equals(par)){
            double notaTeorico=(nexamen+ndeberes+nlecciones)*0.80;
            double notaPractico=(ntalleres)*0.20;
            nota=notaTeorico+notaPractico;
        }
    }
    return nota;
}
```

Técnica de Refactorización

Introduce parameter object: en lugar de recibir los 4 parámetros enviaremos un objeto Notas que contendrá las diferentes notas.

Código refactorizado

```java
// Realiza en cálculo de las notas parciales ya sea inicial o final.
public double CalcularNota(Paralelo p, ArrayList<double> notas){
    double nota=0;
    for(Paralelo par:paralelos){
        if(p.equals(par)){
            double notaTeorico= (notas[0]+notas[1]+notas[2])*0.80;
            double notaPractico=(notas[3])*0.20;
            nota=notaTeorico+notaPractico;
        }
    }
    return nota;
}
```