

## **Taller Refactoring**

### **Integrantes:**

- Rogwi Alexis Cajas Correa
- Jeffrey Gabriel Prado Reyes
- Walter Alfredo Santacruz Astudillo

### **Materia:**

DISEÑO DE SOFTWARE

### **Paralelo:**

3

<b>Feature Envy</b>	<b>3</b>
<b>Speculative Generality</b>	<b>4</b>
<b>Temporary Field</b>	<b>5</b>
<b>Lazy Class</b>	<b>6</b>
<b>Duplicate Code</b>	<b>7</b>
<b>Dead Code</b>	<b>8</b>
<b>Comments</b>	<b>9</b>

## Feature Envy

La clase estudiante calcula las notas de la materia por cada paralelo y luego en la nota total se encarga de acceder a las notas de la clase materia. En extra dichos métodos tampoco han sido usados.

```
//Calcula y devuelve la nota inicial contando examen, deberes, lecciones y talleres. El teorico y el practico
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
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;
}

//Calcula y devuelve la nota inicial contando examen, deberes, lecciones y talleres. Esta nota es solo el
public double CalcularNotaTotal(Paralelo p){
    double notaTotal=0;
    for(Paralelo par: paralelos){
        if(p.equals(par)){
            notaTotal=(p.getMateria().notaInicial+p.getMateria().notaFinal)/2;
        }
    }
    return notaTotal;
}
```

**Solución:** Dichos métodos son específicos de la materia más no del estudiante y peor aún de los paralelos por lo que mediante Move Method los llevamos en específico a la clase correspondiente.

```
//Calcula y devuelve la nota inicial contando examen, deberes, lecciones y talleres. El teorico y el practico
private double CalcularNotaInicial(double nexamen, double ndeberes, double nlecciones, double ntalleres){
    double notaTeorico=(nexamen+ndeberes+nlecciones)*0.80;
    double notaPractico=(ntalleres)*0.20;
    return notaTeorico+notaPractico;
}

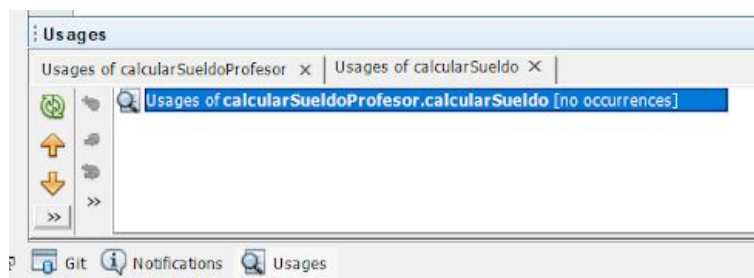
//Calcula y devuelve la nota final contando examen, deberes, lecciones y talleres. El teorico y el practico
private double CalcularNotaFinal(double nexamen, double ndeberes, double nlecciones, double ntalleres){
    double notaTeorico=(nexamen+ndeberes+nlecciones)*0.80;
    double notaPractico=(ntalleres)*0.20;
    return notaTeorico+notaPractico;
}

//Calcula y devuelve la nota inicial contando examen, deberes, lecciones y talleres. Esta nota es solo el
private double CalcularNotaTotal(double notaI, double notaF){
    return notaI+notaF/2;
}
```

## Speculative Generality

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

La clase calcularSueldoProfesor no es usada en ninguna parte como se observa en la siguiente captura:



**Solución:** Aplicar inline class para mover dicha operación a la clase encargada de almacenar la información adicional del profesor

```
public class InformacionAdicionalProfesor {  
    public int añosdeTrabajo;  
    public String facultad;  
    public double BonoFijo;  
  
    public double calcularSueldo(){  
        return this.añosdeTrabajo*600 + this.BonoFijo;  
    }  
}
```

## Temporary Field

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

En este método la variable temporal sueldo es innecesaria, pues la operación que almacena es simple. Conservarla solo ocupa líneas innecesarias y no aporta a entender mejor el código.

**Solución:** Usar la estrategia de refactorización “Inline temp” para mover la operación dentro del return del método.

```
public double calcularSueldo() {  
    return añosdeTrabajo*600 + BonoFijo;  
}
```

## Lazy Class

```
public class InformacionAdicionalProfesor {  
    public int añosdeTrabajo;  
    public String facultad;  
    public double BonoFijo;  
  
    public double calcularSueldo(Profesor prof){  
        return this.añosdeTrabajo*600 + this.BonoFijo;  
    }  
}
```

Esta clase únicamente tiene los atributos extras de la clase profesor y un único método. Mantenerla es una pérdida de recursos innecesaria.

**Solución:** usar la estrategia de refactorización “Inline Class” para mover los atributos y el método a la Clase Profesor y remover la clase vaga.

```
public int añosdeTrabajo;  
public String facultad;  
public double BonoFijo;  
  
public Profesor(String codigo, String nombre, String apellido, String facultad,  
    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(){  
    return añosdeTrabajo*600 + BonoFijo;  
}
```

## Duplicate Code

```
public class Estudiante{
    //Informacion del estudiante
    public String matricula;
    public String nombre;
    public String apellido;
    public String facultad;
    public int edad;
    public String direccion;
    public String telefono;
    public ArrayList<Paralelo> paralelos;

    //Getter y setter de Matricula
}

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;
}
```

Las clases Estudiante y Profesor tienen muchos campos duplicados al igual que sus respectivos getter y setter, Estas clases deberían estar relacionadas jerárquicamente para reutilizar código y hacer más fácil la extensión de código.

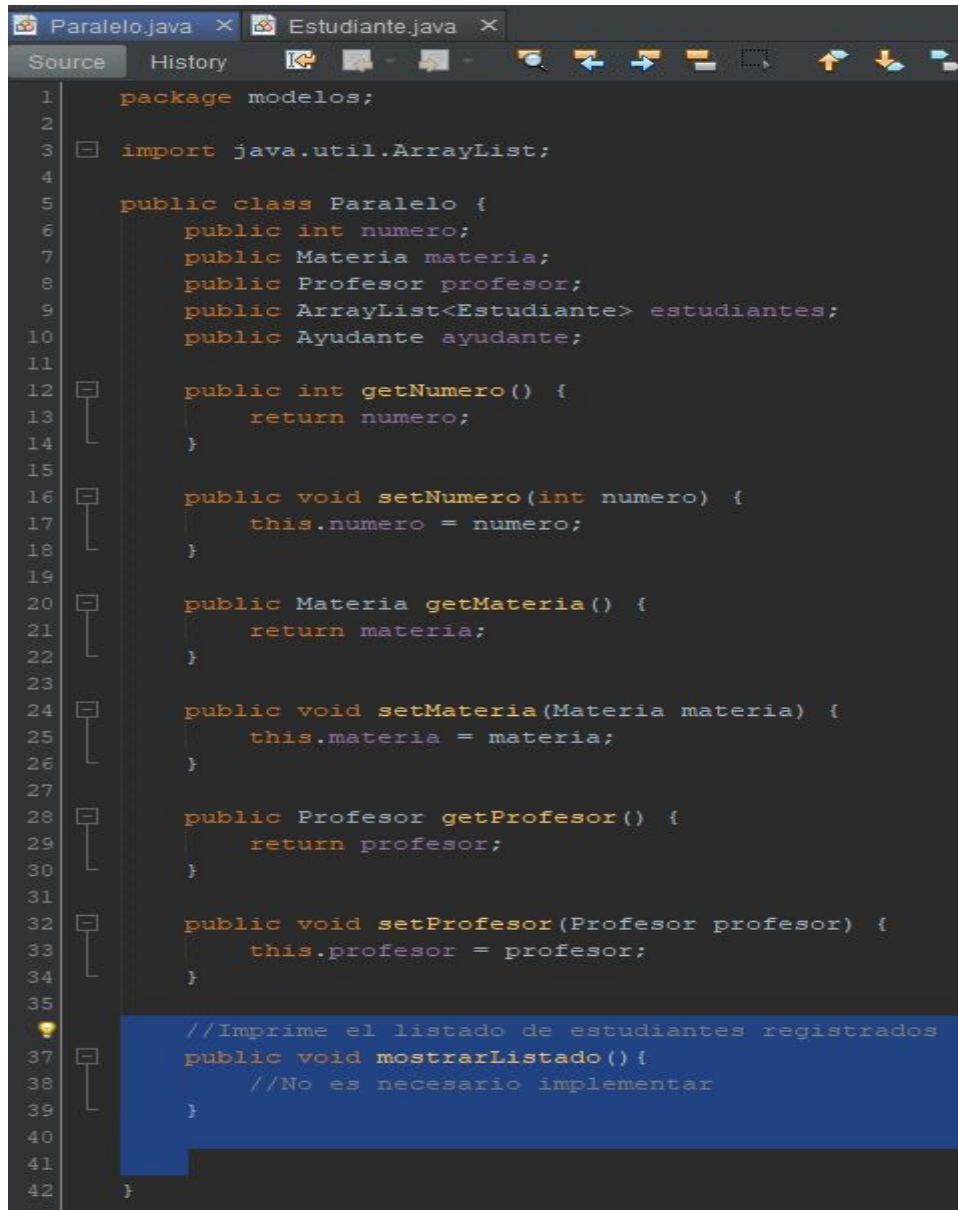
**Solución:** Usar “Extract superClass”, crear la clase padre “Persona” con los tributos y métodos repetidos.

```
public class Persona {
    public String nombre;
    public String apellido;
    public String facultad;
    public int edad;
    public String direccion;
    public String telefono;
    public ArrayList<Paralelo> paralelos;
}

public class Estudiante extends Persona{
    public String matricula;
}

public class Profesor extends Persona{
    public String codigo;
    public int añosdeTrabajo;
    public double BonoFijo;
}
```

## Dead Code

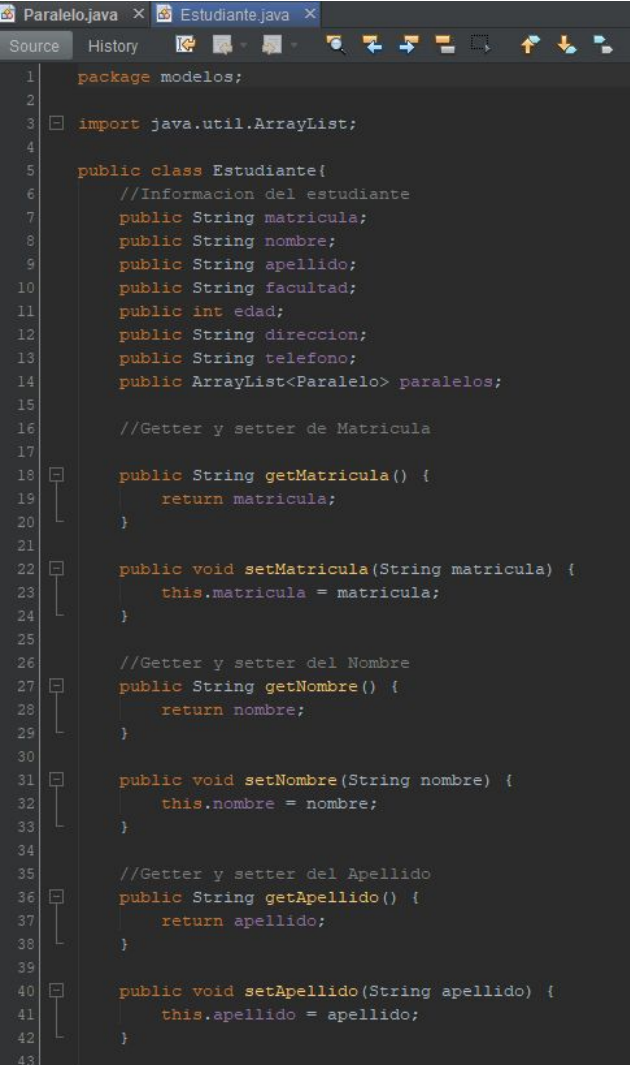


```
1 package modelos;
2
3 import java.util.ArrayList;
4
5 public class Paralelo {
6     public int numero;
7     public Materia materia;
8     public Profesor profesor;
9     public ArrayList<Estudiante> estudiantes;
10    public Ayudante ayudante;
11
12    public int getNumero() {
13        return numero;
14    }
15
16    public void setNumero(int numero) {
17        this.numero = numero;
18    }
19
20    public Materia getMateria() {
21        return materia;
22    }
23
24    public void setMateria(Materia materia) {
25        this.materia = materia;
26    }
27
28    public Profesor getProfesor() {
29        return profesor;
30    }
31
32    public void setProfesor(Profesor profesor) {
33        this.profesor = profesor;
34    }
35
36    //Imprime el listado de estudiantes registrados
37    public void mostrarListado(){
38        //No es necesario implementar
39    }
40
41
42 }
```

Está utilizando un método vacío el cual no tiene ningún sentido y no afectará al funcionamiento del programa. Por lo cual lo mejor es eliminarlo.



# Comments



```
1 package modelos;
2
3 import java.util.ArrayList;
4
5 public class Estudiante{
6     //Informacion del estudiante
7     public String matricula;
8     public String nombre;
9     public String apellido;
10    public String facultad;
11    public int edad;
12    public String direccion;
13    public String telefono;
14    public ArrayList<Paralelo> paralelos;
15
16    //Getter y setter de Matricula
17
18    public String getMatricula() {
19        return matricula;
20    }
21
22    public void setMatricula(String matricula) {
23        this.matricula = matricula;
24    }
25
26    //Getter y setter del Nombre
27    public String getNombre() {
28        return nombre;
29    }
30
31    public void setNombre(String nombre) {
32        this.nombre = nombre;
33    }
34
35    //Getter y setter del Apellido
36    public String getApellido() {
37        return apellido;
38    }
39
40    public void setApellido(String apellido) {
41        this.apellido = apellido;
42    }
43}
```

```

//Getter y setter del telefono

public String getTelefono() {
    return telefono;
}

public void setTelefono(String telefono) {
    this.telefono = telefono;
}

//Calcula y devuelve la nota inicial contando examen, deberes, lecciones y talleres
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
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;
}

//Calcula y devuelve la nota inicial contando examen, deberes, lecciones y talleres
public double CalcularNotaTotal(Paralelo p) {
    double notaTotal=0;
    for(Paralelo par:paralelos){
        if(p.equals(par)){
            notaTotal=(p.getMateria().notaInicial+p.getMateria().notaFinal);
        }
    }
    return notaTotal;
}

```

Hay muchos comentarios innecesarios como que indica que esos son getters y setters en la clase estudiante en todos los atributos y también a los otros métodos les da una explicación de que hace cuando el método en si, solo con el nombre ya se puede intuir que es lo que hace. En este caso no existe una técnica específica ya que los comentarios no irrumpen tanto en el código ya que de por sí es entendible así que lo único que se aplicaría sería borrar los comentarios.