Taller Code Smells

Integrantes:

Juan Demera

Luis Quezada

Contenido

sección A

comments

```java
public class Estudiante{
    //Informacion del estudiante
    public String matricula;
    public String [redacted];
    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 String getMatricula() {
        return matricula;
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    //Getter y setter del Nombre
    public String getNombre() {
        return [redacted];
    }

    public void setNombre(String nombre) {
        this.[redacted] = nombre;
    }

    //Getter y setter del Apellido
    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }
```

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

    //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;
    }

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

            }
        }
```

los comentarios en los métodos son innecesarios y pueden generar ruidos al programador

en este caso se debería aplicar “Rename Method” pero el nombre y es obvio e intuitivo, solo se procede a borrar el comentario.

Data Class

```java
package modelos;

public class Materia {
    public String codigo;
    public String nombre;
    public String facultad;
    public double notaInicial;
    public double notaFinal;
    public double notaTotal;

}
```

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

}
```

Las clases Materia e InformacionAdicionalProfesor son clases que solo contienen datos públicos, para mejorar la clase Materia se aplicó Encapsule Field para cambiar sus atributos públicos a privados y agregar getters y setters, en cambio con InformacionAdicionalProfesor se aplicó Collapse Hierarchy para agregar los atributos de InformacionAdicionalProfesor a la clase Profesor.

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

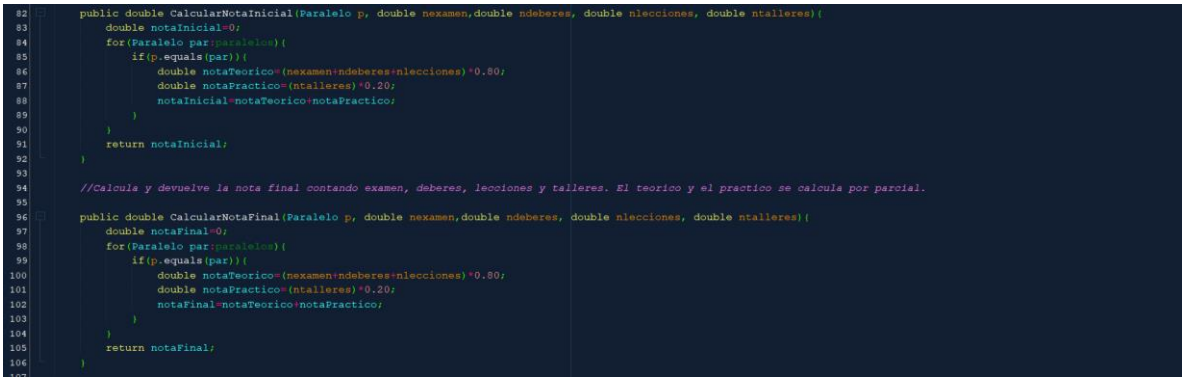
Data Clumps

```
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;
```

```
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;
```

La clase Estudiante y Profesor tiene atributos iguales, para evitar la repetición y mejorar la organización de código se puede crear una superclase Usuario con los atributos repetidos que Estudiante y Profesor hereden.

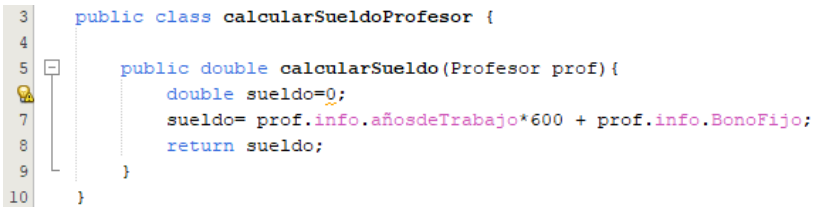
Long Parameter List



```
    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;
    }
```

La clase estudiante cuenta con los métodos calcularNotaInicial y calcularNotaFinal que tienen muchos parámetros, para solucionar este problema se usó “Introduce Parameter Object” para crear la clase notas con los atributos del método.

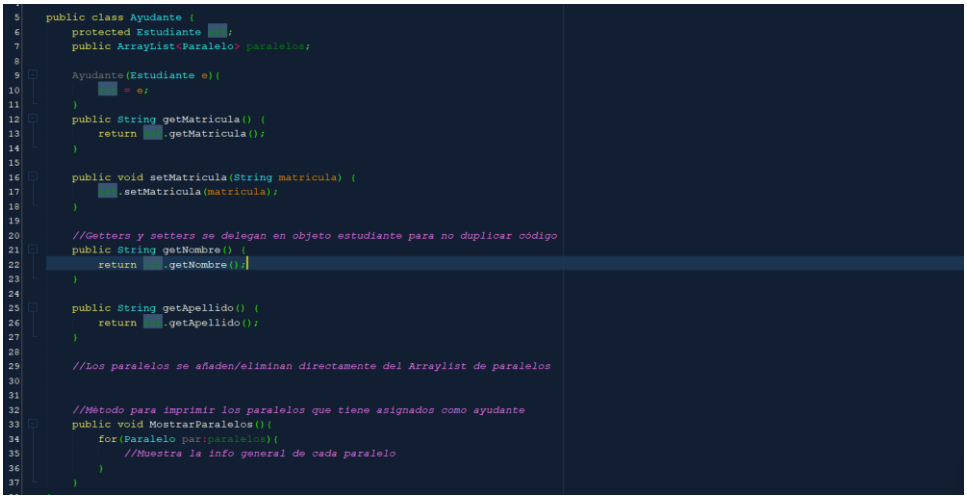
Lazy Class



```
public class calcularSueldoProfesor {

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

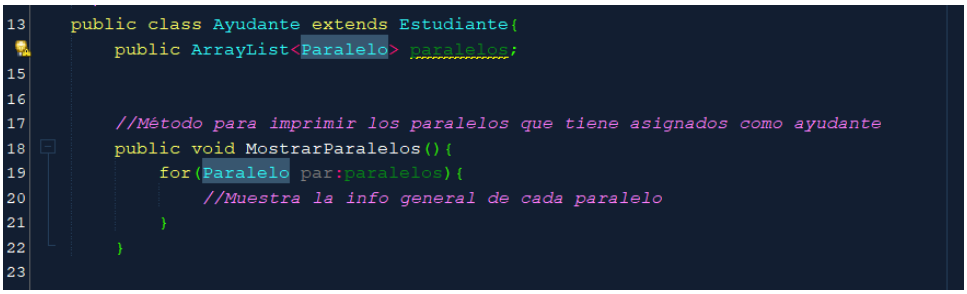
La clase calcularSueldoProfesor tiene un método que accede a los atributos de la clase Profesor, para arreglar esto se usó “Inline Class” para mover el método a la clase profesor.

Featury Envy



```
public class Ayudante {
    protected Estudiante ___;
    public ArrayList<Paralelo> paralelos;

    Ayudante(Estudiante e){
        ___ = e;
    }
    public String getMatricula() {
        return ___.getMatricula();
    }

    public void setMatricula(String matricula) {
        ___.setMatricula(matricula);
    }

    //Getters y setters se delegan en objeto estudiante para no duplicar código
    public String getNombre() {
        return ___.getNombre();
    }

    public String getApellido() {
        return ___.getApellido();
    }

    //Los paralelos se añaden/eliminan directamente del Arraylist de paralelos


    //Método para imprimir los paralelos que tiene asignados como ayudante
    public void MostrarParalelos(){
        for(Paralelo par:paralelos){
            //Muestra la info general de cada paralelo
        }
    }
```

En la clase Ayudante los métodos getters y setters acceden a los datos de otro objeto, para solucionar esto se uso “Replace Delegation with Inheritance”



```
public class Ayudante extends Estudiante{
    public ArrayList<Paralelo> paralelos;


    //Método para imprimir los paralelos que tiene asignados como ayudante
    public void MostrarParalelos(){
        for(Paralelo par:paralelos){
            //Muestra la info general de cada paralelo
        }
    }

```