

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

---

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

**DISEÑO DE SOFTWARE**

**TALLER DE REFACTORING - I TÉRMINO 2020**

**PARALELO: 102**

**GRUPO: 10**

**INTEGRANTES:**

**WILLIAM VENEGAS**

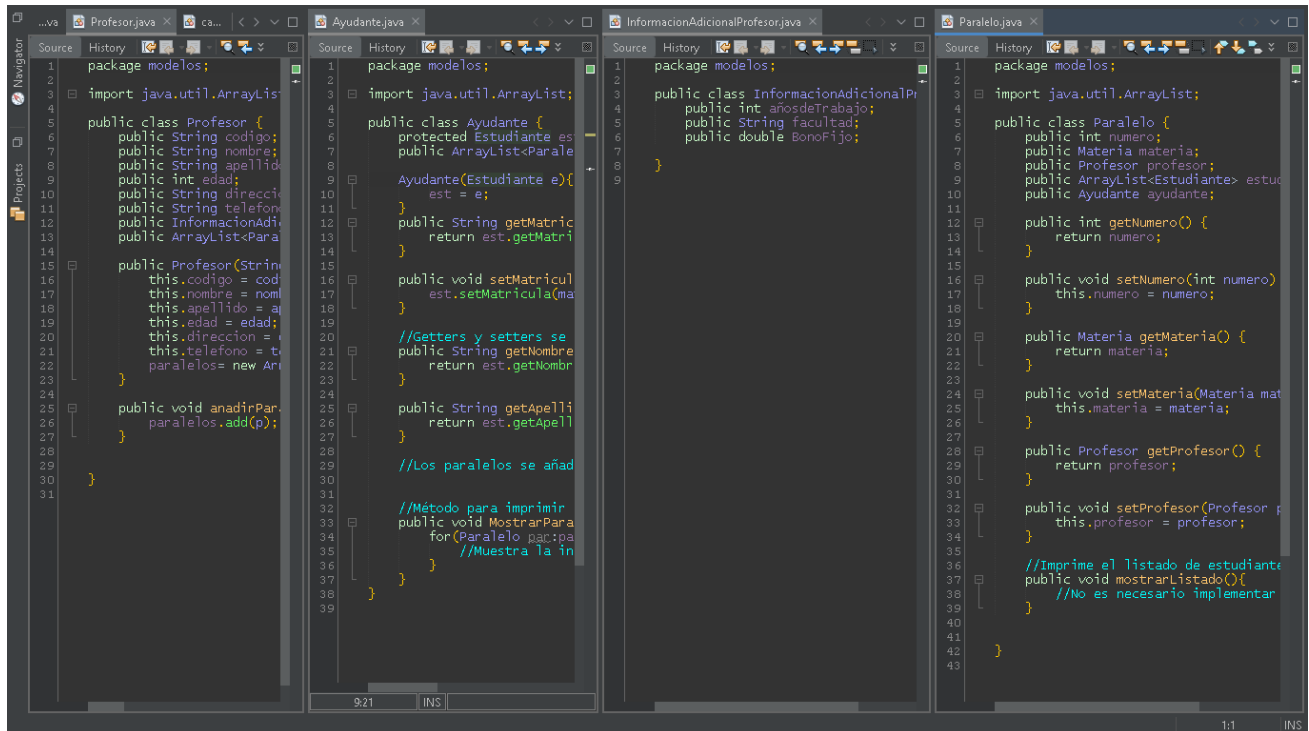
**GUSTAVO CHONILLO**

## Contenido

INAPPROPRIATE INTIMACY.....	3
DATA CLUMPS.....	4
REFUSED BEQUEST.....	5
DEAD CODE.....	6
DUPLICATE CODE .....	7
DATA CLASS .....	8

## INAPPROPRIATE INTIMACY

Las clases Estudiante, Profesor, InformacionAdicionalProfesor, Ayudante, Paralelo y Materia, tienen declarados sus atributos como públicos, es decir que cualquier otra clase puede acceder a sus atributos, en este caso tenemos el **Code Smell Inappropriate Intimacy**, ¿Cómo lo podemos solucionar?, debemos refactorizar y para eso aplicaremos **Encapsulate Field**, es decir cambiaremos el modificador de acceso de los atributos a privado y agregaremos los **getters** y **setters** para cada atributo.



```
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 InformacionAdi...
    public ArrayList<Para...

    public Profesor(String...
        this.codigo = cod...
        this.nombre = nom...
        this.apellido = a...
        this.edad = edad;
        this.direccion = ...
        this.telefono = t...
        paralelos = new Ar...
    }

    public void anadirPar...
        paralelos.add(p);
    }
}

package modelos;
import java.util.ArrayList;

public class Ayudante {
    protected Estudiante es...
    public ArrayList<Para...

    Ayudante(Estudiante e){
        est = e;
    }

    public String getMatri...
        return est.getMatri...
    }

    public void setMatricul...
        est.setMatricula(ma...
    }

    //Getters y setters se...
    public String getNombre...
        return est.getNombr...
    }

    public String getApelli...
        return est.getApelli...
    }

    //Los paralelos se añad...

    //Método para imprimir...
    public void MostrarPara...
        for (Paralelo pac:pa...
        //Muestra la in...
    }
}

package modelos;

public class InformacionAdicionalPr...
    public int añosdeTrabajo;
    public String facultad;
    public double BonoFijo;
}

package modelos;
import java.util.ArrayList;

public class Paralelo {
    public int numero;
    public Materia materia;
    public Profesor profesor;
    public ArrayList<Estudiante> estud...
    public Ayudante ayudante;

    public int getNumero() {
        return numero;
    }

    public void setNumero(int numero) {
        this.numero = numero;
    }

    public Materia getMateria() {
        return materia;
    }

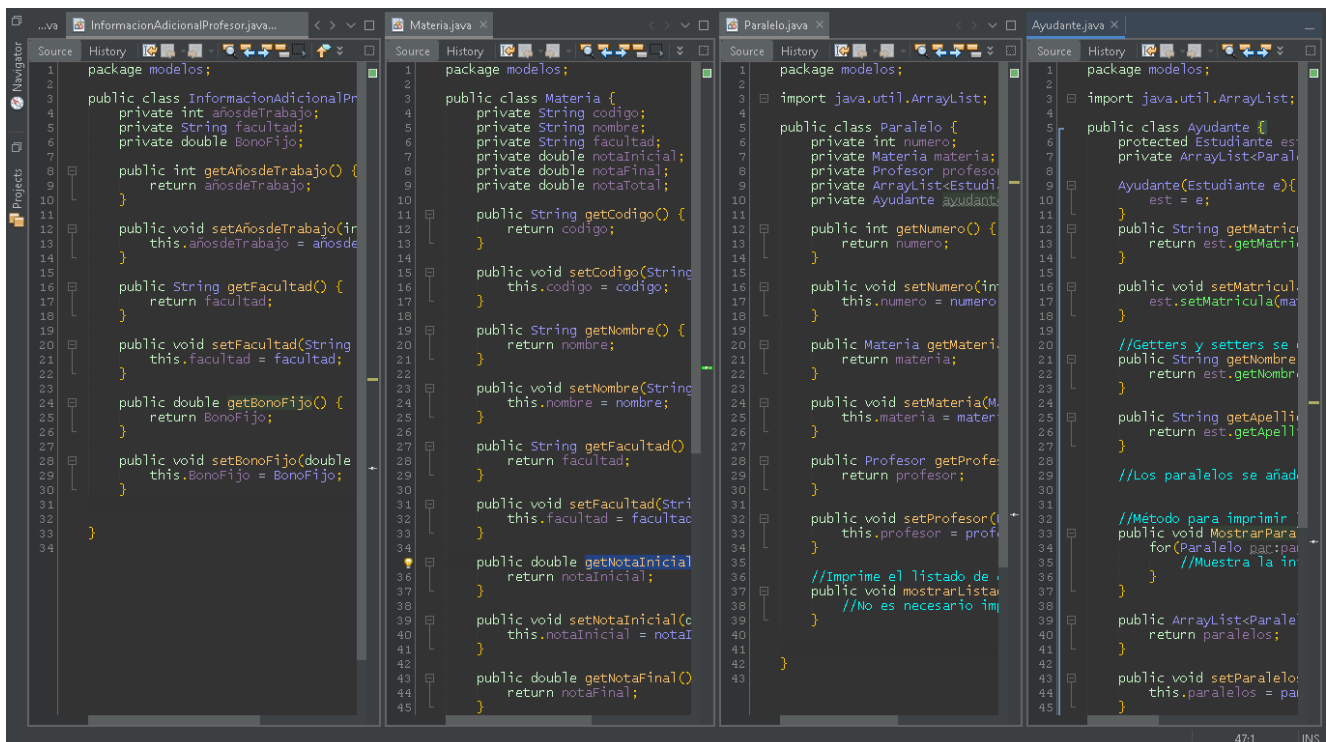
    public void setMateria(Materia mat...
        this.materia = materia;
    }

    public Profesor getProfesor() {
        return profesor;
    }

    public void setProfesor(Profesor p...
        this.profesor = profesor;
    }

    //Imprime el listado de estudiante...
    public void mostrarListado() {
        //No es necesario implementar...
    }
}
```

Figura 1: Code Smell Inappropriate Intimacy



```
package modelos;

public class InformacionAdicionalPr...
    private int añosdeTrabajo;
    private String facultad;
    private double BonoFijo;

    public int getAñosdeTrabajo() {
        return añosdeTrabajo;
    }

    public void setAñosdeTrabajo(int...
        this.añosdeTrabajo = añosde...
    }

    public String getFacultad() {
        return facultad;
    }

    public void setFacultad(String...
        this.facultad = facultad;
    }

    public double getBonoFijo() {
        return BonoFijo;
    }

    public void setBonoFijo(double...
        this.BonoFijo = BonoFijo;
    }
}

package modelos;

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

    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String...
        this.codigo = codigo;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String...
        this.nombre = nombre;
    }

    public String getFacultad() {
        return facultad;
    }

    public void setFacultad(Stri...
        this.facultad = facultad;
    }

    public double getNotaInicial...
        return notaInicial;
    }

    public void setNotaInicial(c...
        this.notaInicial = notai...
    }

    public double getNotaFinal() {
        return notaFinal;
    }
}

package modelos;
import java.util.ArrayList;

public class Paralelo {
    private int numero;
    private Materia materia;
    private Profesor profesor;
    private ArrayList<Estudi...
    private Ayudante ayudante;

    public int getNumero() {
        return numero;
    }

    public void setNumero(int...
        this.numero = numero;
    }

    public Materia getMateri...
        return materia;
    }

    public void setMateria(M...
        this.materia = mater...
    }

    public Profesor getProfe...
        return profesor;
    }

    public void setProfesor(...
        this.profesor = profi...
    }

    //Imprime el listado de...
    public void mostrarLista...
        //No es necesario im...
    }
}

package modelos;
import java.util.ArrayList;

public class Ayudante {
    protected Estudiante es...
    private ArrayList<Para...

    Ayudante(Estudiante e){
        est = e;
    }

    public String getMatri...
        return est.getMatri...
    }

    public void setMatricul...
        est.setMatricula(ma...
    }

    //Getters y setters se...
    public String getNombre...
        return est.getNombr...
    }

    public String getApelli...
        return est.getApelli...
    }

    //Los paralelos se añad...

    //Método para imprimir...
    public void MostrarPara...
        for (Paralelo pac:pa...
        //Muestra la in...
    }

    public ArrayList<Para...
        return paralelos;
    }

    public void setParalelo...
        this.paralelos = pai...
    }
}
```

Figura 2: Refactoring Encapsulate Field

## DATA CLUMPS

En la clase Profesor y Estudiante ambos tienen el campo Telefono y Dirección, es decir tenemos grupos idénticos de variables en clases diferentes es decir que está presente el **Code Smell Data Clumps**, para solucionar esto refactorizamos el código aplicando **Extract Class**. De esta manera, se crea una nueva clase y colocamos los campos y métodos responsables de la funcionalidad relevante en ella.

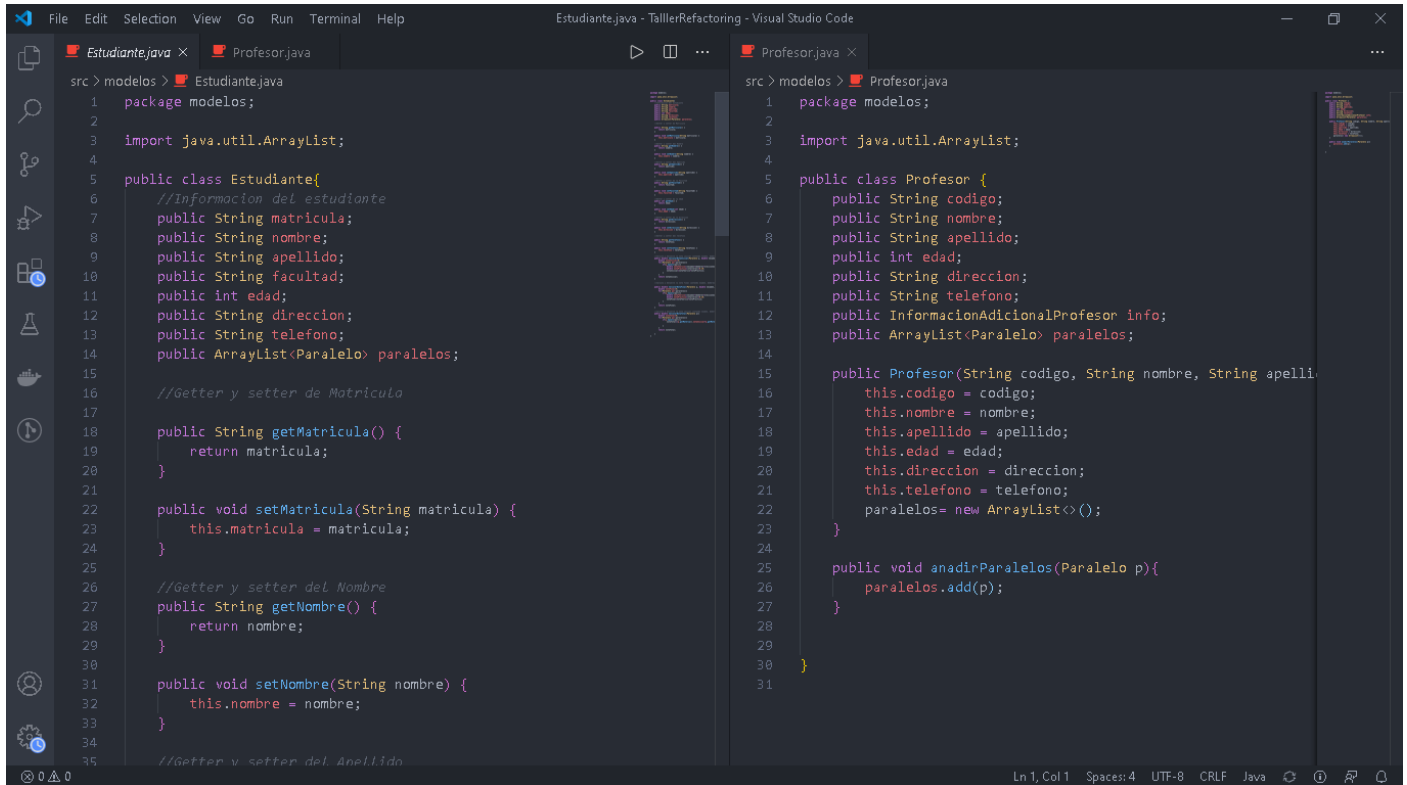


Figura 3: Code Smell Data Clumps

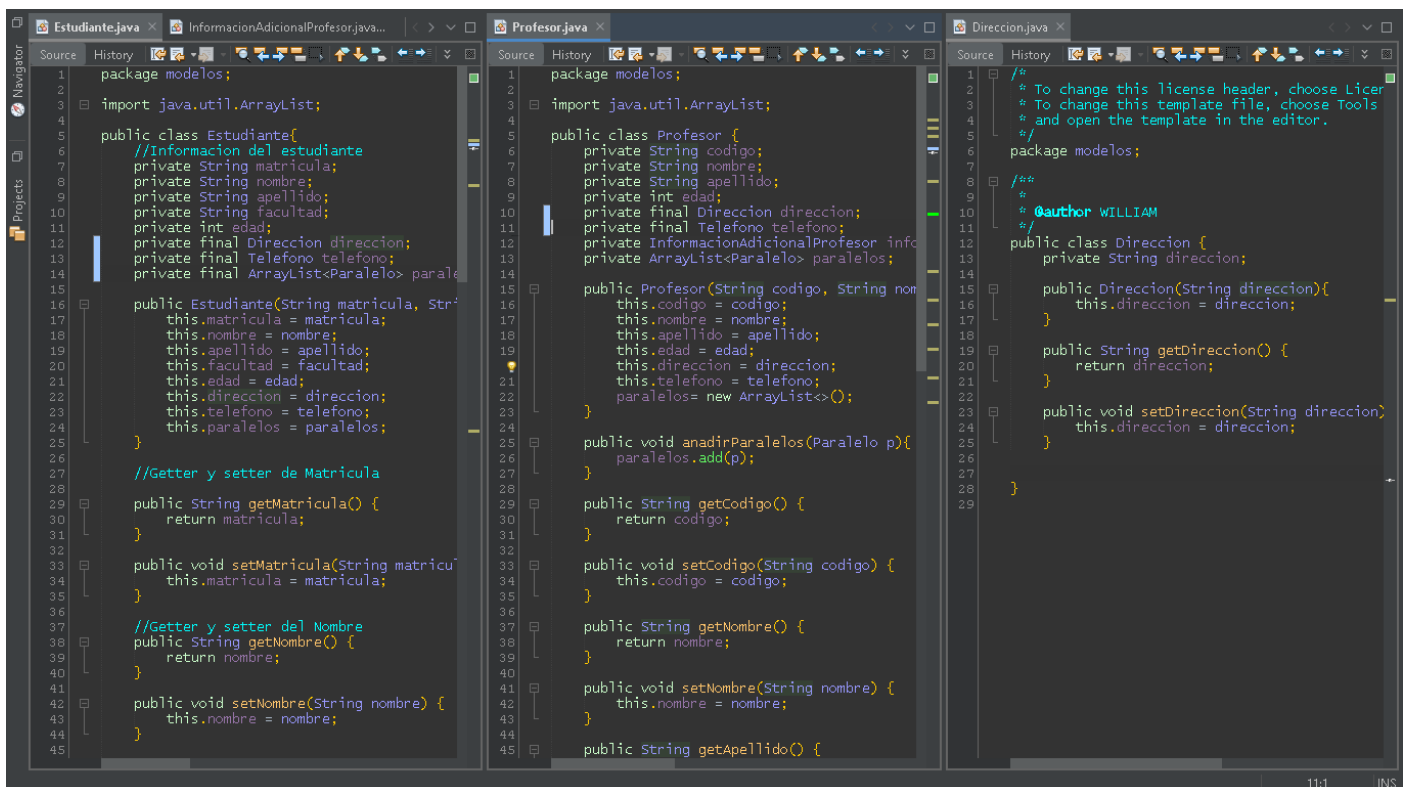
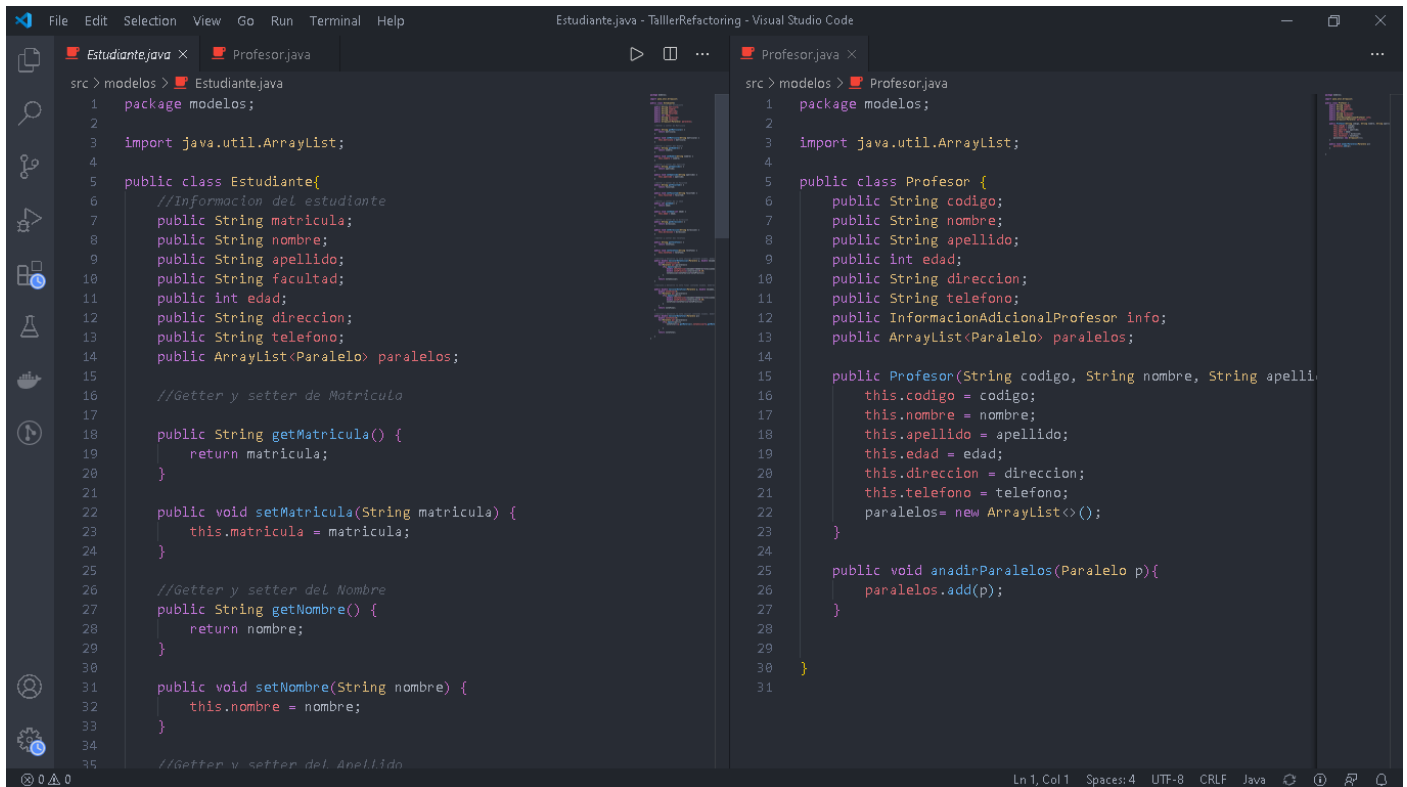


Figura 4: Refactoring Extract Class

## REFUSED BEQUEST

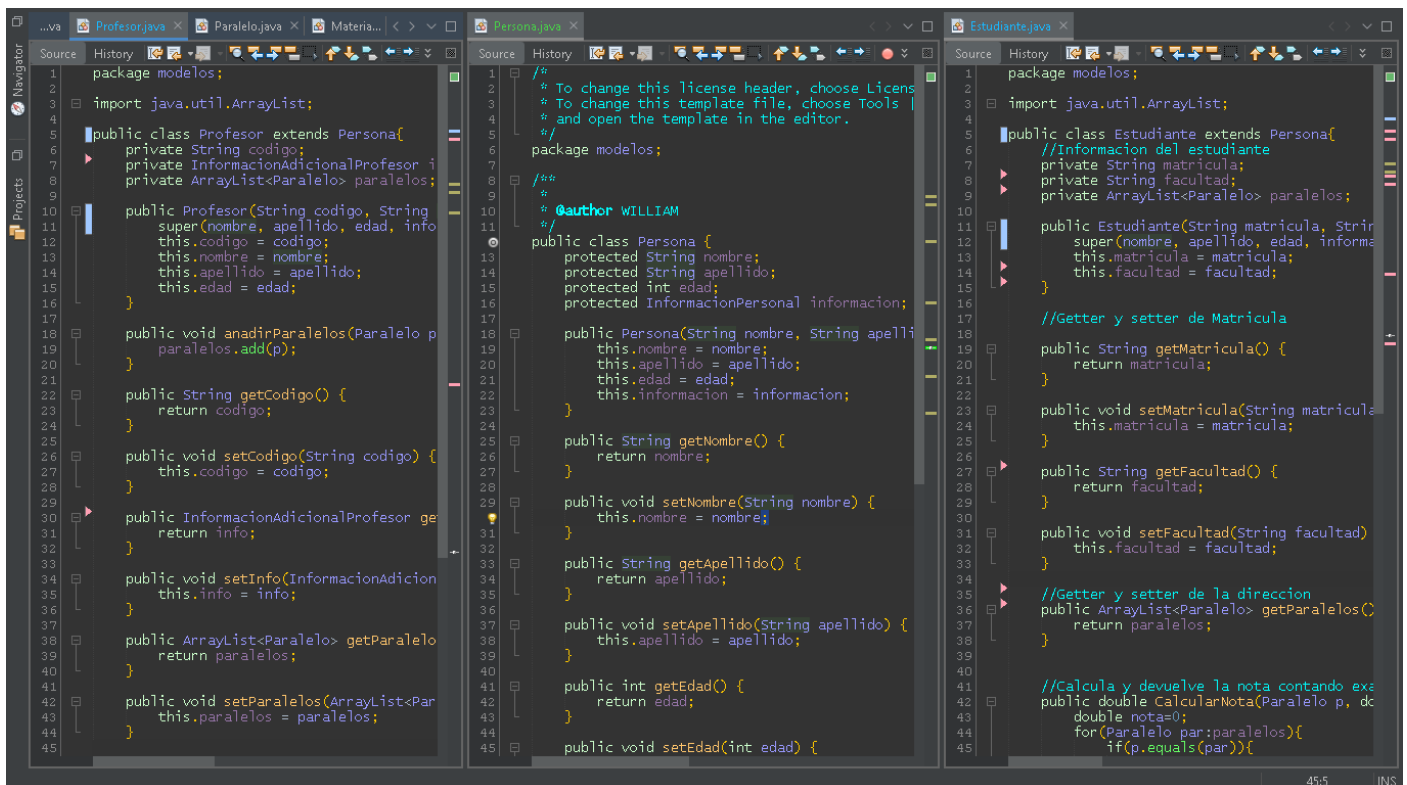
La clase Profesor y Estudiante comparten campos similares y campos similares, es decir que podemos crear una super clase para las clases Profesor y Estudiante. En este caso está presente el **Code Smeel Refused Bequest**, ya que ambas pueden ser hijas de una misma clase padre, refactorizamos con **Extract Superclass**.



```
src > modelos > Estudiante.java
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
```

```
src > modelos > Profesor.java
1 package modelos;
2
3 import java.util.ArrayList;
4
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 apelli
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    }
24
25    public void anadirParalelos(Paralelo p){
26        paralelos.add(p);
27    }
28
29
30 }
31
```

Figura 5: Code Smell Refused Bequest



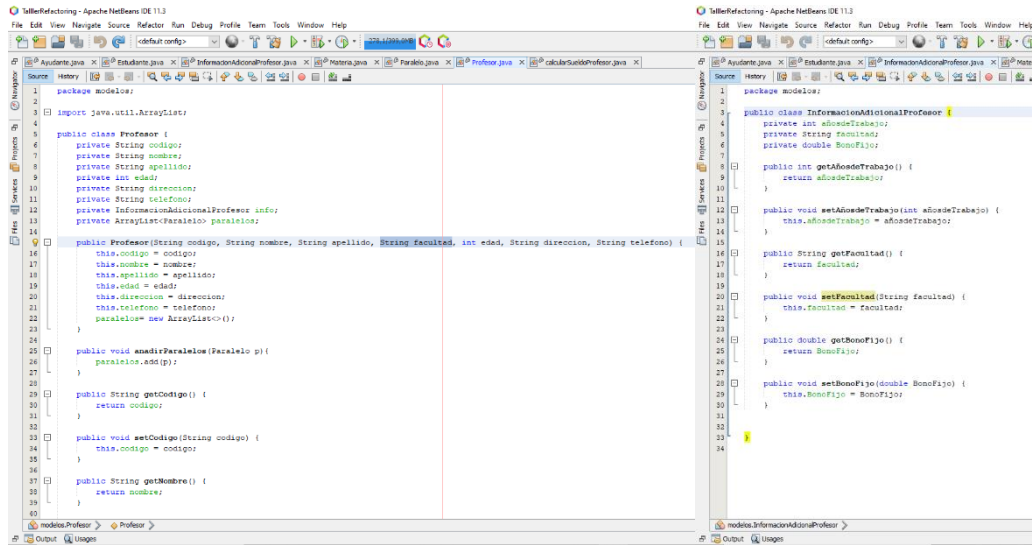
```
...va Profesor.java | Paralelo.java | Materia... | Persona.java | Estudiante.java
Source History
package modelos;
import java.util.ArrayList;
public class Profesor extends Persona{
    private String codigo;
    private InformacionAdicionalProfesor i
    private ArrayList<Paralelo> paralelos;
    public Profesor(String codigo, String
        super(nombre, apellido, edad, info
        this.codigo = codigo;
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
    }
    public void anadirParalelos(Paralelo p
        paralelos.add(p);
    }
    public String getCodigo() {
        return codigo;
    }
    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }
    public InformacionAdicionalProfesor ge
        return info;
    }
    public void setInfo(InformacionAdicion
        this.info = info;
    }
    public ArrayList<Paralelo> getParalelo
        return paralelos;
    }
    public void setParalelos(ArrayList<Par
        this.paralelos = paralelos;
    }
}

Source History
/* To change this license header, choose Licens
/* To change this template file, choose Tools |
/* and open the template in the editor.
package modelos;
/**
 * @author WILLIAM
 */
public class Persona {
    protected String nombre;
    protected String apellido;
    protected int edad;
    protected InformacionPersonal informacion;
    public Persona(String nombre, String apelli
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
        this.informacion = informacion;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellido() {
        return apellido;
    }
    public void setApellido(String apellido) {
        this.apellido = apellido;
    }
    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {

Source History
package modelos;
import java.util.ArrayList;
public class Estudiante extends Persona{
    //Informacion del estudiante
    private String matricula;
    private String facultad;
    private ArrayList<Paralelo> paralelos;
    public Estudiante(String matricula, Strir
        super(nombre, apellido, edad, inform
        this.matricula = matricula;
        this.facultad = facultad;
    }
    //Getter y setter de Matricula
    public String getMatricula() {
        return matricula;
    }
    public void setMatricula(String matricula
        this.matricula = matricula;
    }
    public String getFacultad() {
        return facultad;
    }
    public void setFacultad(String facultad)
        this.facultad = facultad;
    }
    //Getter y setter de la direccion
    public ArrayList<Paralelo> getParalelos()
        return paralelos;
    }
    //Calcula y devuelve la nota contando exa
    public double CalcularNota(Paralelo p, dc
        double nota=0;
        for(Paralelo par:paralelos){
            if(p.equals(par)){
```

Figura 6: Refactoring - Extract SuperClass

## DEAD CODE



La clase `Profesor` hay un parámetro en el constructor llamado `facultad` que no se está usando, y en la clase `InformacionAdicionalProfesor` hay un atributo con el mismo nombre que no se usa en ningún lado del código. La solución a este problema es ***Remove Parameter***.

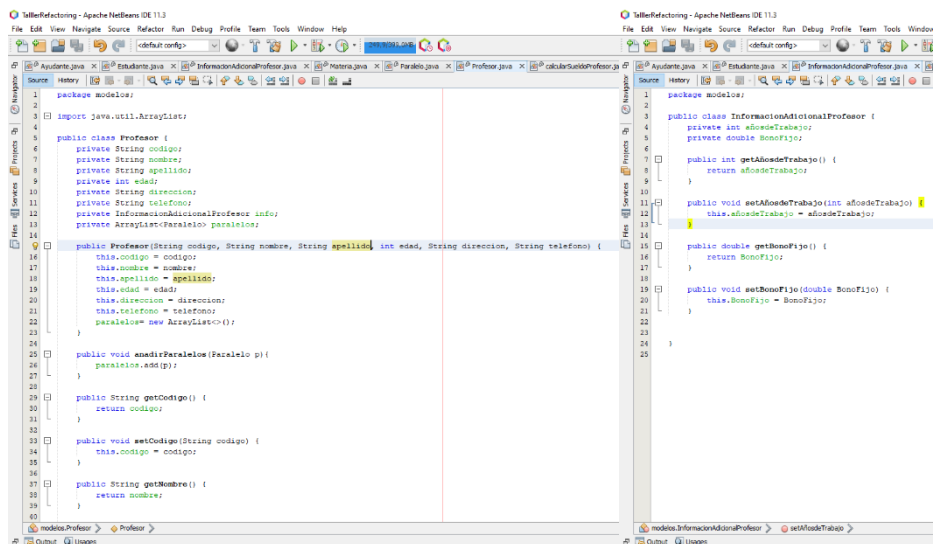
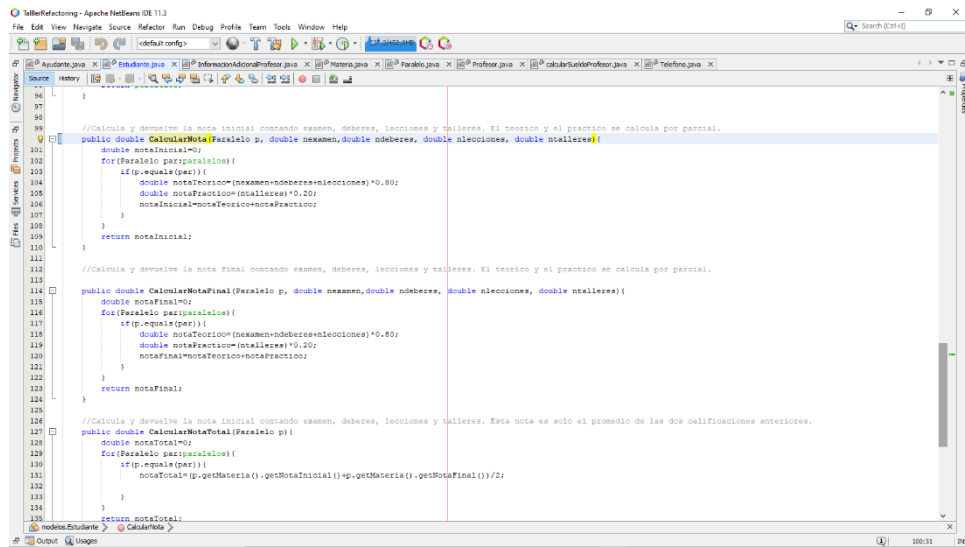


Figura 7: Code Smell Dead Code

Figura 8: Refactoring Remove Parameter

## DUPLICATE CODE



En la clase estudiante el código en los métodos `calcularNotaInicial` y `calcularNotaFinal` es el casi el mismo por lo que reae en el code smell **Duplicate Code**, esto puede solucionarse utilizando **Parametrize Method**. Se eliminan los 2 métodos identicos y se los hace uno solo.

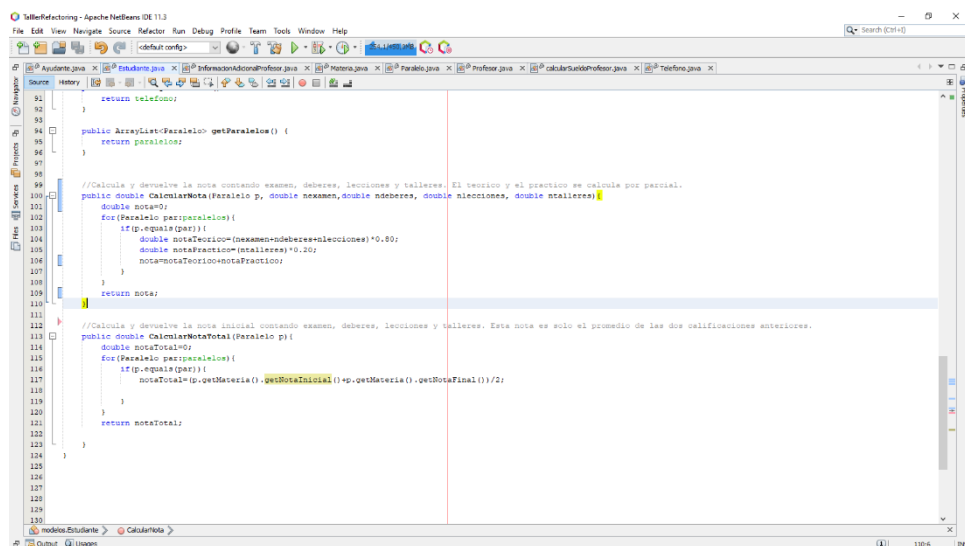


Figura 9: Code Smell Duplicate Code

Figura 10: Refactoring Parametrize Method

# DATA CLASS

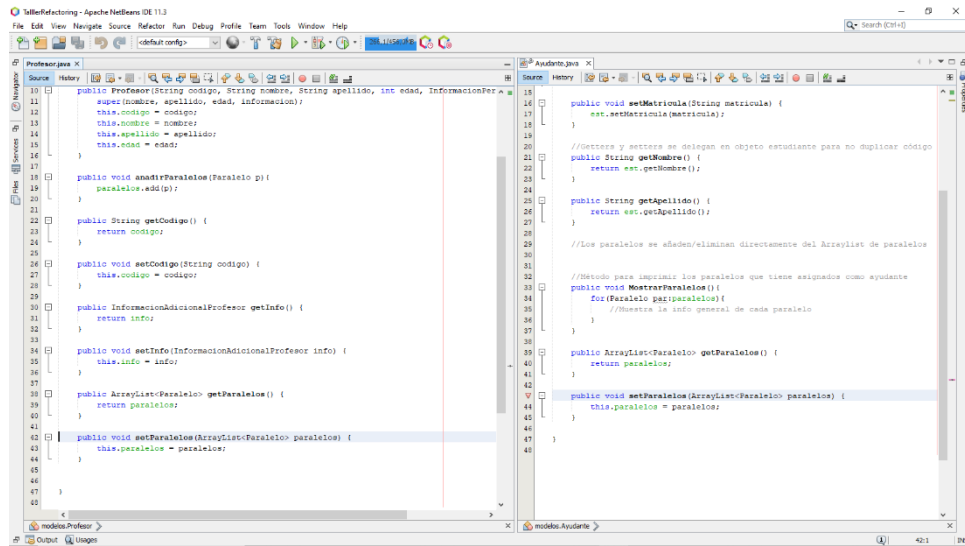




Figura 12: Refactor Encapsulate Collection

Figura 13: Refactor Move Method