# Procesos de la ingeniería del Software

Análisis estático del código.

Curso 2019/2020

Pablo Almansa Torres

10 de marzo de 2020

# I N D I C E

# 1 Introducción

En este documento, se va a dar una explicación de los puntos a reforzar de “Proyecto1” y del motivo de esto.

# 2 Revisión del proyecto “SourceMonitor”

En este apartado se hará un análisis basándonos solamente en los datos que obtenemos en el “SourceMonitor”

En general, podemos ver que hay escasez en la cantidad de comentarios. Además, podemos ver que para las clases que se tratan, hay bastante complejidad y profundidad en “Poligono”. Lo que la convierte en la clase principal.

También podemos ver que todos los .java tienen dentro una sola clase. Lo que es bueno considerando que estamos en Java.

# 3 Análisis del proyecto “Proyecto1”

En este apartado se va a hacer un análisis más extensivo de las clases de este proyecto.

## 3.1 Línea

```java
package poligono;

//Linea creada a partir de la pendiente y un punto.
public class Linea {
    private int n, x;

    public Linea (int m, Punto punto){
        x = m;
        n = -m*punto.getPosX() + punto.getPosY();
    }

    public String toString() { return "y = "+x+"x + ("+n+")"; }

    public String implicita() { return x+"x - y ("+n+") = 0"; }

    public String puntoCorte(Linea otra){
        double coordenadaX, coordenadaY;
        if (this.x == otra.x) {
            return "paralelas o coincidentes";
        }
        coordenadaX = (this.n-otra.n)/(this.x-otra.x)*(-1);
        coordenadaY = this.x*coordenadaX+this.n;
        return "("+coordenadaX+", "+coordenadaY+")";
    }

}
```

```java
package poligono;

///Clase Linea.
///Creada apartir de una pentiente (m) y de un Punto (punto)
public class Linea {
    //n: DESCRIPCION
    //x: DESCRIPCION
    private int n, x;

    public Linea (int m, Punto punto){
        x = m;
        n = -m*punto.getPosX() + punto.getPosY();
    }

    //Calcula el punto de corte de esta Linea con "otra"
    public String puntoCorte(Linea otra){
        if (this.x == otra.x) return "paralelas o coincidentes";
        double coordenadaX, coordenadaY;

        coordenadaX = (this.n-otra.n)/(this.x-otra.x)*(-1);
        coordenadaY = this.x*coordenadaX+this.n;
        return "("+coordenadaX+", "+coordenadaY+")";
    }

    public String toString(){
        return "y = "+x+"x + ("+n+")";
    }
}
```

El código así queda mucho más claro, se sabe para que vale la clase, cómo se crea, para que valen los métodos y además, he eliminado “implícita()”. Ya que parece un toString() y encima no se usaba.

## 3.2 Polígono

En esta clase, para empezar, no tiene el “enconde”

```java
package poligono;

import java.util.ArrayList;

public class Poligono {

    ArrayList<Punto> poligono;

    public Poligono(ArrayList<Punto> poligon) {
        poligono = poligon;
    }

    public boolean concavo(){
        Vector vec1, vec2;

        //Comprobamos el signo del primer producto vectorial
        int i = 0;
        //Creamos vectores a partir de los primeros puntos
        vec1 = new Vector(poligono.get(i), poligono.get(i+1));
        vec2 = new Vector(poligono.get(i+1), poligono.get(i+2));
        double prodVectorial = vec1.productoVectorial(vec2);
        boolean positivo = prodVectorial >= 0;

        //Calculamos el signo de los demás productos vectoriales
        for (i = 1; i < poligono.size()-2; i++) {
            vec1 = new Vector(poligono.get(i), poligono.get(i+1));
            vec2 = new Vector(poligono.get(i+1), poligono.get(i+2));
            prodVectorial = vec1.productoVectorial(vec2);
            //Si tienen distinto signo es cóncavo
            if (positivo != (prodVectorial >= 0)) {
                return true;
            }
        }
        //Calculamos el producto vectorial del último y el primer vector
        vec1 = new Vector(poligono.get(i+1), poligono.get(0));
        prodVectorial = vec2.productoVectorial(vec1);
        if (positivo != (prodVectorial >= 0)) {
            return true;
        }
        return false;
    }
```

```java
package poligono;

import java.util.ArrayList;
//Clase Poligono.
//Se construye con una lista de puntos.
public class Poligono {

    private ArrayList<Punto> poligono;

    public Poligono(ArrayList<Punto> puntos) {
        poligono = puntos;
    }

    public boolean concavo(){
        Vector vec1, vec2;
        int i = 0;
        double prodVectorial;
        boolean signoPositivo;
        //Creamos vectores a partir de los primeros puntos y sacamos el signo
        vec1 = new Vector(poligono.get(i), poligono.get(i+1));
        vec2 = new Vector(poligono.get(i+1), poligono.get(i+2));
        prodVectorial = vec1.productoVectorial(vec2);
        signoPositivo = prodVectorial >= 0;

        //Calculamos el signo de los demás productos vectoriales
        for (i = 1; i < poligono.size()-2; i++) {
            vec1 = new Vector(poligono.get(i), poligono.get(i+1));
            vec2 = new Vector(poligono.get(i+1), poligono.get(i+2));
            prodVectorial = vec1.productoVectorial(vec2);
            //Es concavo si alguno tiene distinto signo
            if (signoPositivo != (prodVectorial >= 0)) {
                return true;
            }
        }
        //Calculamos el producto vectorial del último y el primer vector
        vec1 = new Vector(poligono.get(i+1), poligono.get(0));
        prodVectorial = vec2.productoVectorial(vec1);
        //Si es distinto, es concavo
        if (signoPositivo != (prodVectorial >= 0)) {
            return true;
        }
        return false;
    }
```

Si se mejoran un poco los comentarios y, además, se agrupan las declaraciones de variables, el código queda mucho más claro.

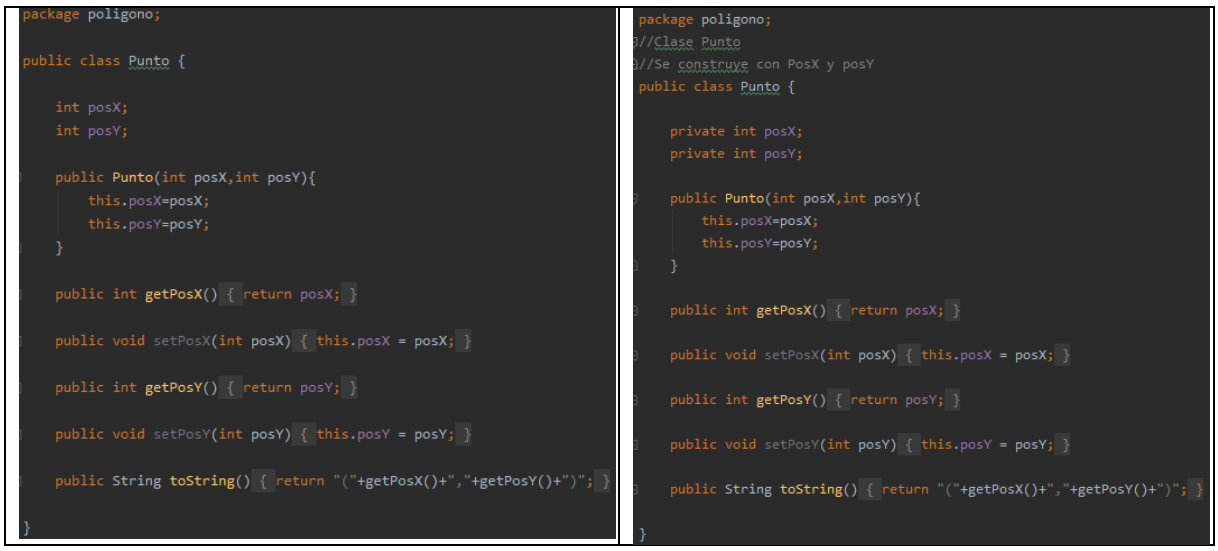
```java
public Punto centroide(){
    double X=0, Y=0, A=0;
    int mod = poligono.size();
    for (int k = 0; k < poligono.size(); k++) {
        A+=poligono.get(k).getPosX()*poligono.get((k+1)%mod).getPosY()-poligono.get((k+1)%mod)
                .getPosX()*poligono.get(k).getPosY();
    }
    A=A/2;

    for (int k = 0; k < poligono.size(); k++) {
        X+=(poligono.get(k).getPosX()+poligono.get((k+1)%mod).getPosX())*(poligono.get(k)
                .getPosX()*poligono.get((k+1)%mod).getPosY()-poligono.get((k+1)%mod).getPosX()
                *poligono.get(k).getPosY());
    }
    X=X/(6*A);

    for (int k = 0; k < poligono.size(); k++) {
        Y+=(poligono.get(k).getPosY()+poligono.get((k+1)%mod).getPosY())*(poligono.get(k)
                .getPosX()*poligono.get((k+1)%mod).getPosY()-poligono.get((k+1)%mod).getPosX()
                *poligono.get(k).getPosY());
    }
    Y=Y/(6*A);
    return new Punto((int) X, (int)Y);
}

public String toString(){
    return poligono.toString();
}
```

```java
//Calcula el centroide del Poligono
public Punto centroide(){
    double X=0, Y=0, A=0;
    int mod = poligono.size();
    for (int k = 0; k < poligono.size(); k++) {
        A+=poligono.get(k).getPosX()*poligono.get((k+1)%mod).getPosY()-poligono.get((k+1)%mod)
                .getPosX()*poligono.get(k).getPosY();
        X+=(poligono.get(k).getPosX()+poligono.get((k+1)%mod).getPosX())*(poligono.get(k)
                .getPosX()*poligono.get((k+1)%mod).getPosY()-poligono.get((k+1)%mod).getPosX()
                *poligono.get(k).getPosY());
        Y+=(poligono.get(k).getPosY()+poligono.get((k+1)%mod).getPosY())*(poligono.get(k)
                .getPosX()*poligono.get((k+1)%mod).getPosY()-poligono.get((k+1)%mod).getPosX()
                *poligono.get(k).getPosY());
    }
    A=A/2;
    X=X/(6*A);
    Y=Y/(6*A);
    return new Punto((int) X, (int)Y);
}

public String toString(){
    return poligono.toString();
}
```

Aquí lo que he hecho es juntar todas las cuentas en un solo bucle for, no es necesario que haya más de uno cuando los límites iniciales y finales son exactamente los mismos. Además de añadir un pequeño comentario

### 3.3 Punto

En esta clase realmente hay poco que modificar. Los modificadores de acceso a private.



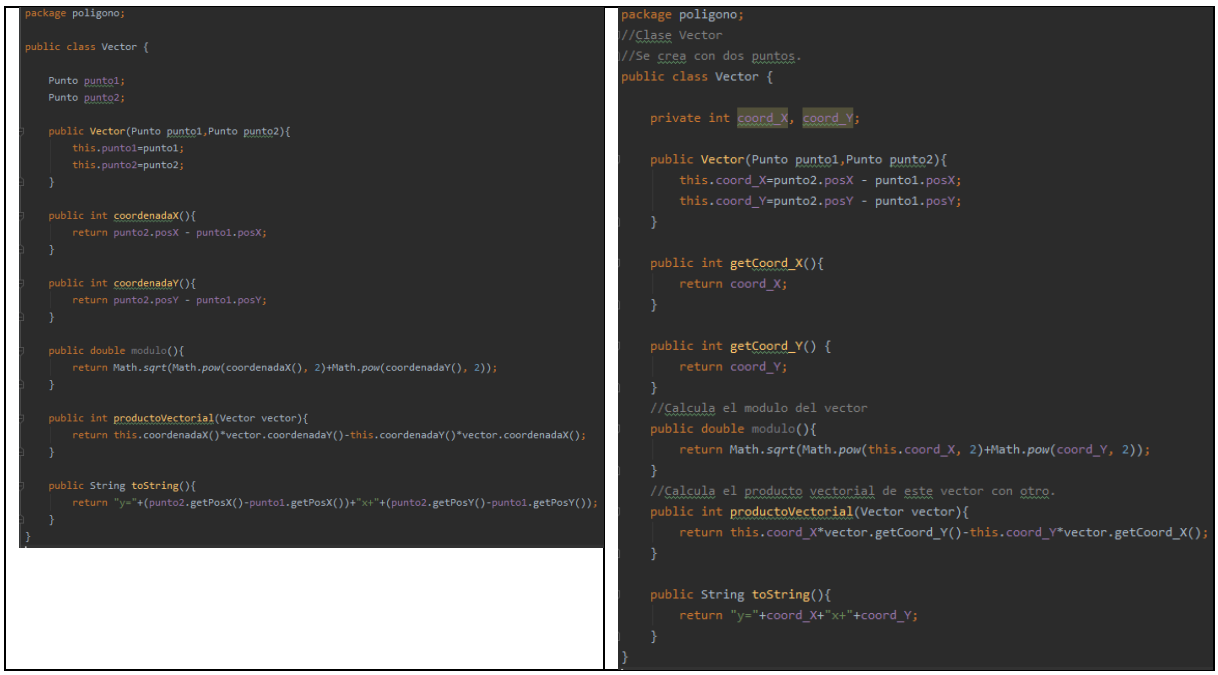
```java
package poligono;

public class Punto {

    int posX;
    int posY;

    public Punto(int posX,int posY){
        this.posX=posX;
        this.posY=posY;
    }

    public int getPosX() { return posX; }

    public void setPosX(int posX) { this.posX = posX; }

    public int getPosY() { return posY; }

    public void setPosY(int posY) { this.posY = posY; }

    public String toString() { return "("+getPosX()+","+getPosY()+")"; }

}
```

```java
package poligono;
//Clase Punto
//Se construye con PosX y posY
public class Punto {

    private int posX;
    private int posY;

    public Punto(int posX,int posY){
        this.posX=posX;
        this.posY=posY;
    }

    public int getPosX() { return posX; }

    public void setPosX(int posX) { this.posX = posX; }

    public int getPosY() { return posY; }

    public void setPosY(int posY) { this.posY = posY; }

    public String toString() { return "("+getPosX()+","+getPosY()+")"; }

}
```

### 3.4 UsoPoligono

Esta clase no tiene mucho que comentar. No está en UTF8 y es un Main. El nombre podría ser mejor. Como por ejemplo “App” o “Program”.

### 3.5 Vector



```java
package poligono;

public class Vector {

    Punto punto1;
    Punto punto2;

    public Vector(Punto punto1,Punto punto2){
        this.punto1=punto1;
        this.punto2=punto2;
    }

    public int coordenadaX(){
        return punto2.posX - punto1.posX;
    }

    public int coordenadaY(){
        return punto2.posY - punto1.posY;
    }

    public double modulo(){
        return Math.sqrt(Math.pow(coordenadaX(), 2)+Math.pow(coordenadaY(), 2));
    }

    public int productoVectorial(Vector vector){
        return this.coordenadaX()*vector.coordenadaY()-this.coordenadaY()*vector.coordenadaX();
    }

    public String toString(){
        return "y="+(punto2.getPosX()-punto1.getPosX())+"x+"+(punto2.getPosY()-punto1.getPosY());
    }
}
```

```java
package poligono;
//Clase Vector
//Se crea con dos puntos.
public class Vector {

    private int coord_X, coord_Y;

    public Vector(Punto punto1,Punto punto2){
        this.coord_X=punto2.posX - punto1.posX;
        this.coord_Y=punto2.posY - punto1.posY;
    }

    public int getCoord_X(){
        return coord_X;
    }

    public int getCoord_Y() {
        return coord_Y;
    }
    //Calcula el modulo del vector
    public double modulo(){
        return Math.sqrt(Math.pow(this.coord_X, 2)+Math.pow(coord_Y, 2));
    }
    //Calcula el producto vectorial de este vector con otro.
    public int productoVectorial(Vector vector){
        return this.coord_X*vector.getCoord_Y()-this.coord_Y*vector.getCoord_X();
    }

    public String toString(){
        return "y="+coord_X+"x+"+coord_Y;
    }
}
```

Esta clase está mejor con una reforma general, no tiene sentido guardar los dos puntos cuando lo que hace falta es la resta de estos. Además, hacen falta modificadores de accesibilidad y algunos comentarios.