INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS

MELARA ABARCA REYNA ELIA

INVESTIGACIÓN SOBRE MÉTRICAS OO

2CV13

EQUIPO 8

INTEGRANTES:

DÍAZ VITAL HECTOR

FLORES VALLADARES LUIS ENRIQUE

JUAREZ VARGAS LEOBARDO DANIEL

MARTINEZ LUNA CARLOS DANIEL

PEREZ LOPEZ LEONARDO

FECHA:

28 DE SEPTIEMBRE DE 2021

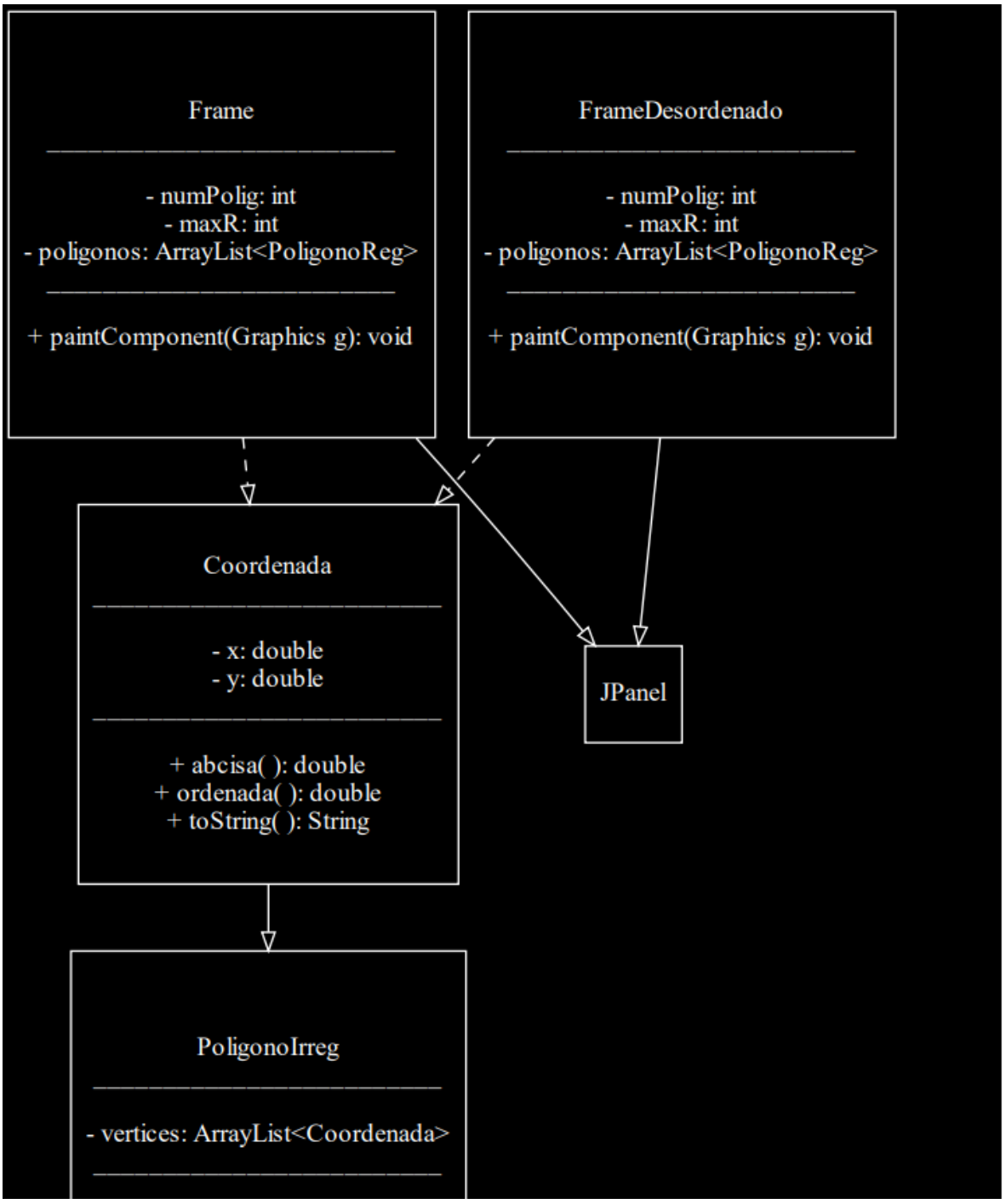
Ejercicio 1.

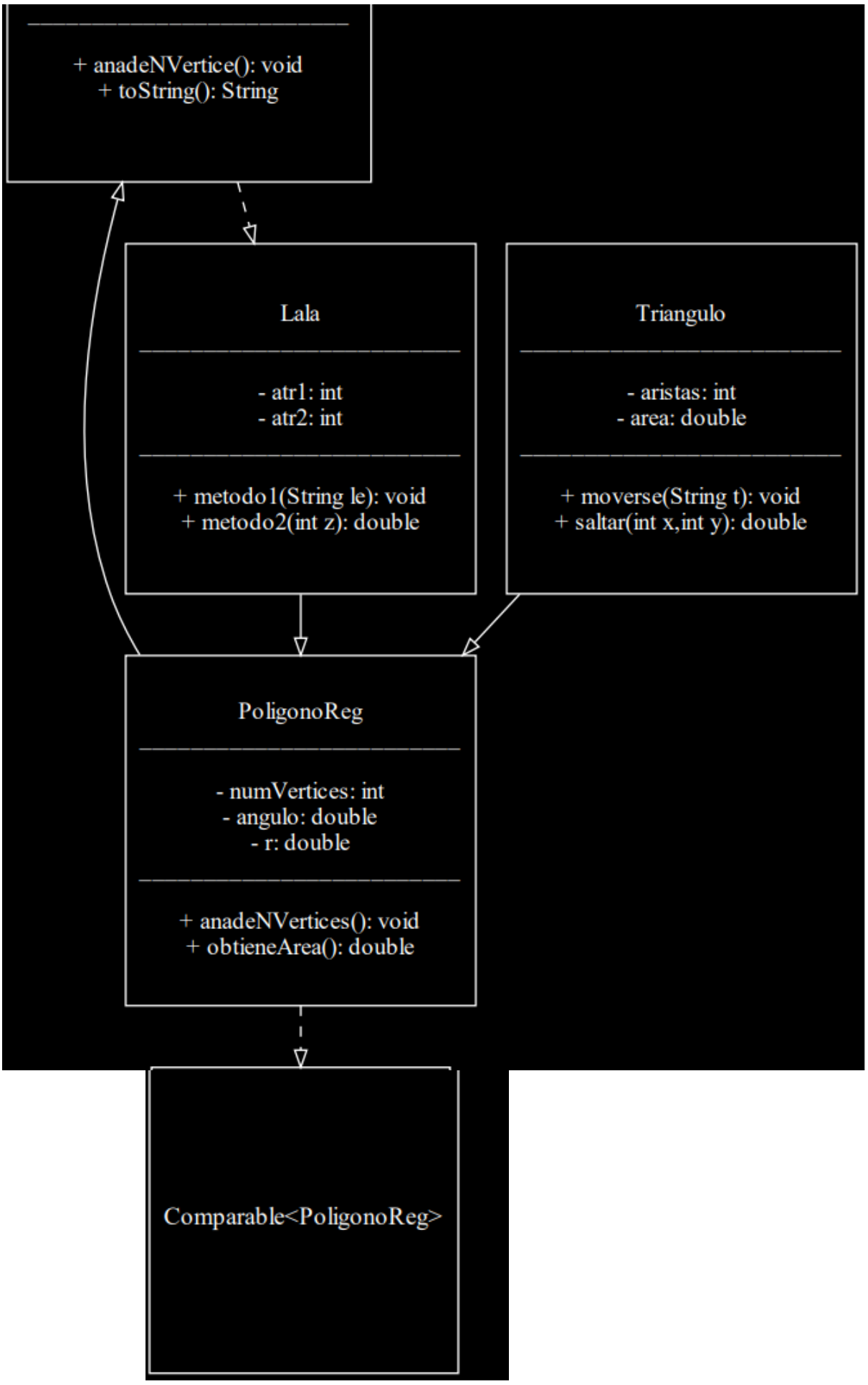
1. Complete la siguiente tabla con una posible solución que representa la medición del software al problema planteado (los problemas listados suelen ser comunes en el desarrollo de sistemas de software). Se presenta a modo de ejemplo el problema de la primera fila de la tabla.

Problema	Medir ayuda a:
Requisitos incorrectos	Describir los requisitos en términos expresados de manera que puedan ser verificables, medibles, no ambiguos.
Toma de decisiones	Determinar todas las opciones con las que se cuentan, buscar cual es la mejor para cumplir el objetivo que se busca.
Falta de control	Dialogar con el equipo de trabajo para determinar un nuevo rumbo así como las tomas de decisiones, para continuar con la continuidad de un proyecto y no afectar de manera riesgosa al mismo.
Exceso de gastos	Búsqueda de alternativas que no sigan generando más gastos y determinar qué gastos podemos reducir o incluso prescindir de ellos y continuar con solo gastos esenciales.
Costos de mantenimiento	Analizar todos los mantenimientos y determinar un costo para cada uno de manera periódica para no requerir cambio o pérdida de maquinarias.

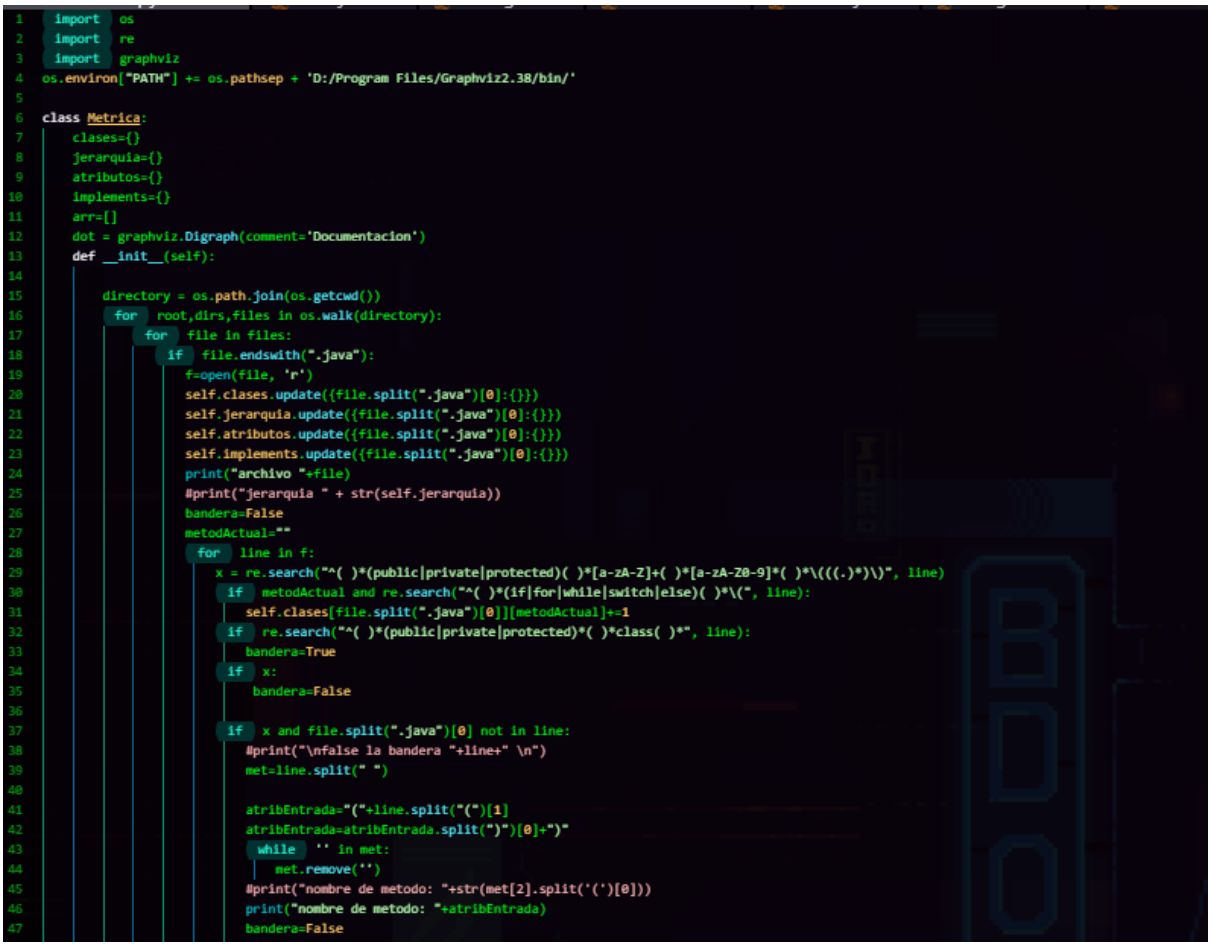
Ejercicio 2.

Diagrama de clases que genera el documentador automático programado en python




+ anadeNVertice(): void
+ toString(): String
Lala
- atr1: int
- atr2: int
+ metodo1(String le): void
+ metodo2(int z): double
Triangulo
- aristas: int
- area: double
+ moverse(String t): void
+ saltar(int x,int y): double
PoligonoReg
- numVertices: int
- angulo: double
- r: double
+ anadeNVertices(): void
+ obtieneArea(): double
Comparable<PoligonoReg>

Lo mas completo seria utilizar un compilador formado por vario autómatas finitos no deterministas pero como una aproximador se realizó un analizador sintáctico y léxico con funciones regEx



```
import os
import re
import graphviz
os.environ["PATH"] += os.pathsep + 'D:/Program Files/Graphviz2.38/bin/'

class Metrica:
    clases={}
    jerarquia={}
    atributos={}
    implements={}
    arr=[]
    dot = graphviz.Digraph(comment='Documentacion')
    def __init__(self):

        directory = os.path.join(os.getcwd())
        for root,dirs,files in os.walk(directory):
            for file in files:
                if file.endswith(".java"):
                    f=open(file, 'r')
                    self.clases.update({file.split(".java")[0]:{}})
                    self.jerarquia.update({file.split(".java")[0]:{}})
                    self.atributos.update({file.split(".java")[0]:{}})
                    self.implements.update({file.split(".java")[0]:{}})
                    print("archivo "+file)
                    #print("jerarquia " + str(self.jerarquia))
                    bandera=False
                    metodActual=""
                    for line in f:
                        x = re.search("^( )*(public|private|protected)( )*[a-zA-Z]+( )*[a-zA-Z0-9]*( )*\(((.)*)\)", line)
                        if metodActual and re.search("^( )*(if|for|while|switch|else)( )*\(", line):
                            self.clases[file.split(".java")[0]][metodActual]+=1
                        if re.search("^( )*(public|private|protected)*( )*class( )*", line):
                            bandera=True
                        if x:
                            bandera=False

                        if x and file.split(".java")[0] not in line:
                            #print("\nfalse la bandera "+line+" \n")
                            met=line.split(" ")

                            atribEntrada="("+line.split("(")[1]
                            atribEntrada=atribEntrada.split(")")[0]+")"
                            while '' in met:
                                met.remove('')
                            #print("nombre de metodo: "+str(met[2].split('(')[0]))
                            print("nombre de metodo: "+atribEntrada)
                            bandera=False
```


metodActual=str(met[2].split('(')[0])+atribEntrada+": "+met[1]
self.clases[file.split(".java")[0]].update({str(met[2].split('(')[0])+atribEntrada+": "+met[1]:0})
if re.search("^( )*(public|private|protected)*( )*class( )*", line) and "implements" in line:
print(file.split(".java")[0]+" implements "+str(line.split("implements ")[1].split("{")[0]))
self.implements.update({file.split(".java")[0]:line.split("implements ")[1].split("{")[0]})
if re.search("^( )*(public|private|protected)*( )*class( )*", line) and "extends" in line:
padre=line.split("extends ")[1].split(" ")[0].split("{")[0]
print("clase "+padre+" padre de "+ file.split(".java")[0])
self.jerarquia[file.split(".java")[0]].update({padre:{}})
if bandera==True and re.search(";$", line):
atributo=line.split(" ")
self.atributos[file.split(".java")[0]].update({atributo[len(atributo)-1].split(';')[0]+": "+atributo[len(atributo)-2]:{}})
f.close()
print("dict " + str(self.clases))
print("jerarquia " + str(self.jerarquia))
print("atributos " + str(self.atributos))
print("implements " + str(self.implements))
self.dot.attr('graph', bgcolor='black');
self.dot.attr('node', shape="square",fontcolor="white", color="white")
for i in self.clases.keys():
metodos=""
at=""
for j in self.clases[i].keys():
metodos+="+ "+j+"\n"
for k in self.atributos[i].keys():
at+="- "+k+"\n"
self.dot.node(i, i+"\n_____\n\n"+at+"_____\n\n"+metodos,shape="square",fontcolor="white", color="white")
self.dot.attr('edge', arrowhead='onormal',shape="square",fontcolor="white", color="white");
for i in self.jerarquia.keys():
self.dot.attr('edge', style='solid' ,bgcolor="white:white");
if self.jerarquia[i]:
for j in self.jerarquia[i]:
self.dot.edge(i,j)
self.dot.attr('edge', style='dashed' ,bgcolor="white:white");
if self.implements[i]:
self.dot.edge(i,self.implements[i])

```
100         self.dot.render('test-output/round-table.gv', view=True)
101
102     def creaReporte(self):
103         #Crear reporte metricas
104         for i in self.clases.keys():
105             f = open("Documentacion clase "+i+".txt","w")
106             wmc=0
107             noc=0
108             aux=i
109
110             for z in self.clases[i]:
111                 #print(int(self.clases[i][z]))
112                 wmc+=int(self.clases[i][z])
113             self.arr=[]
114             self.ramaMasLarga(i,0)
115             print(self.arr)
116             f.write("WMC: "+str(wmc)+"\n")
117             f.write("DIT: "+str(self.max(self.arr,len(self.arr)))+"\n")
118             f.write("NOC: "+str(noc)+"\n")
119             wmc=0
120             for j in self.clases[i].keys():
121                 f.write("     Metodo  "+j.split("(")[0]+": \n          WMC: "+str(self.clases[i][j])+"\n")
122             f.write("\n")
123             f.close()
124
125     def ramaMasLarga(self,claseHijo,cont):
126         for i in self.jerarquia.keys():
127             #print(i+" diferente de "+claseHijo)
128             if i!=claseHijo and i and claseHijo in self.jerarquia[i].keys():
129                 #print(i+" padre de "+claseHijo)
130                 for p in self.jerarquia[i].keys():
131                     print(p+" padre de "+i)
132                     self.ramaMasLarga(i,cont+1)
133
134
135             else :
136                 #print("final rama "+str(cont))
137                 self.arr.append(cont)
138
139     def max(self,arr,n):
140         max = arr[0]
141
142         for i in range(1, n):
143             if arr[i] > max:
144                 max = arr[i]
145         return max
```

```
146
147 if __name__ == "__main__":
148
149     met=Metrica()
150     met.creaReporte()
151
```

Ejemplos de clases con sus respectivos reportes, se pueden corroborar los datos con el diagrama de clases generado en cuanto a los extends e implements

Clase Frame

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.ArrayList;
import java.lang.Math;
import java.util.Arrays;
import java.util.Random;

class Frame extends JPanel implements Coordenada{
  int numPolig;
  int maxR;
  ArrayList<PoligonoReg> poligonos;
    public Frame(int numPolig,int maxR,ArrayList<PoligonoReg> poligonos){
      this.numPolig=numPolig;
      this.maxR=maxR;
      this.poligonos= poligonos;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        this.setPreferredSize( new Dimension(800,2*(int)((this.numPolig*this.maxR)/800+10)*maxR));


        g.setColor(Color.blue);
        Polygon poligono;

        int centrox=0;
        int centroy=30;
        int ant=0;
        int cont=0;
        int cont2=0;
        int xpoints[];
        int ypoints[];
        int npoints;
        int red;
        int green;
        int blue;
        Color randomColour;
        Random randomGenerator = new Random();

        for (PoligonoReg figura:this.poligonos){
            poligono= new Polygon();
            xpoints= new int[figura.numVertices];
```

```
51                  ypoints =   new  int[figura.numVertices];;
52
53                  cont=0;
54
55                  centrox+=ant+(int)figura.r+10;
56                   if  (centrox>1100-(int)figura.r){
57                    centroy+=ant+(int)figura.r;
58                    centrox=10+(int)figura.r;
59                  }
60                   for  (Coordenada coord:figura.vertices){
61
62                    poligono.addPoint(centrox+(int)coord.abcisa(), centroy+(int)coord.ordenada());
63                    xpoints[cont]=centrox+(int)coord.abcisa();
64                    ypoints[cont]=centroy+(int)coord.ordenada();
65                    cont+=1;
66                  }
67
68
69                  ant=(int)figura.r+10;
70
71
72                  red = randomGenerator.nextInt(256);
73                  green = randomGenerator.nextInt(256);
74                  blue = randomGenerator.nextInt(256);
75
76                  randomColour =  new  Color(red,green,blue);
77                  g.setColor(randomColour);
78
79                  g.fillPolygon(xpoints, ypoints, figura.numVertices);
80                  cont2+=1;
81
82                   if (cont2==this.numPolig){
83
84                     break ;
85                  }
86              }
87
88
89
90          }
91
92
93      }
```

```
WMC: 4
DIT: 0
NOC: 0
     Metodo  paintComponent:
          WMC: 4
```

Clase FrameDesordenado

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.ArrayList;
import java.lang.Math;
import java.util.Arrays;
import java.util.Random;

class FrameDesordenado extends JPanel implements Coordenada{
  int numPolig;
  int maxR;
  ArrayList<PoligonoReg> poligonos;
    public FrameDesordenado(int numPolig,int maxR,ArrayList<PoligonoReg> poligonos){
      this.numPolig=numPolig;
      this.maxR=maxR;
      this.poligonos= poligonos;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        this.setPreferredSize( new Dimension(1100,750));


        g.setColor(Color.blue);
        Polygon poligono;

        int centrox=0;
        int centroy=0;

        int red;
        int green;
        int blue;
        Color randomColour;
        Random randomGenerator = new Random();
        centrox =400;
        centroy =400;

        for (PoligonoReg figura:this.poligonos){
            poligono= new Polygon();
            centrox = new Random().nextInt((1000 - 50) + 1) + 50;
            centroy = new Random().nextInt((600 - 50) + 1) + 50;


            for (Coordenada coord:figura.vertices){

                poligono.addPoint(centrox + 10+(int)coord.abcisa(), centroy+(int)coord.ordenada());

            }

            red = randomGenerator.nextInt(156);
            green = randomGenerator.nextInt(156);
            blue = randomGenerator.nextInt(156);

            randomColour = new Color(red,green,blue);
            g.setColor(randomColour);
            g.drawPolygon(poligono);

        }


    }
}

```

```
WMC: 2
DIT: 0
NOC: 0
     Metodo  paintComponent:
          WMC: 2
```

Clase PoligonoIrreg

```java
import java.util.ArrayList;
import java.lang.Math;


public class PoligonoIrreg implements Lala{

  ArrayList<Coordenada> vertices;

  public PoligonoIrreg(){
    this.vertices = new ArrayList<Coordenada>();
  }

  public void anadeNVertice(){
    double x,y;
    int minRandom=-1000;
    int maxRandom=1000;

    x = (Math.random()*(maxRandom-minRandom+1)+minRandom);
    y = (Math.random()*(maxRandom-minRandom+1)+minRandom);

    vertices.add( new Coordenada(x,y));
    }

  public String toString() {
    this.vertices.forEach((n) -> System.out.println(n));
     return " ";
  }

}

```

```
WMC: 0
DIT: 2
NOC: 0
     Metodo  anadeNVertice:
          WMC: 0
     Metodo  toString:
          WMC: 0
```

Clase PoligonoReg

```
import java.lang.Math;
import java.util.Comparator;

public class PoligonoReg extends PoligonoIrreg implements Comparable<PoligonoReg>{

  int numVertices;
  double angulo;
  double r;

  public PoligonoReg(int numVertices,double r){
      this.numVertices=numVertices;
      this.angulo=Double.valueOf(360/numVertices/2);
      this.r=r;
  }


  public void anadeNVertices(){
    double x=0;
    double y=0;
      for   (int j=0;j<this.numVertices;j++){

      x= r * Math.cos(2 * Math.PI * j / this.numVertices);
      y= r * Math.sin(2 * Math.PI * j / this.numVertices);

      super.anadeVertice( x,y);

    }


  }

  public double obtieneArea(){
    double area=0;

    area=this.numVertices*Math.pow(this.r,2)/2  * Math.sin(Math.toRadians(360/this.numVertices));
     return  area;
  }


  @Override
  public int compareTo(PoligonoReg obj) {
     return  (int)(this.obtieneArea()-obj.obtieneArea());


  }

}
```

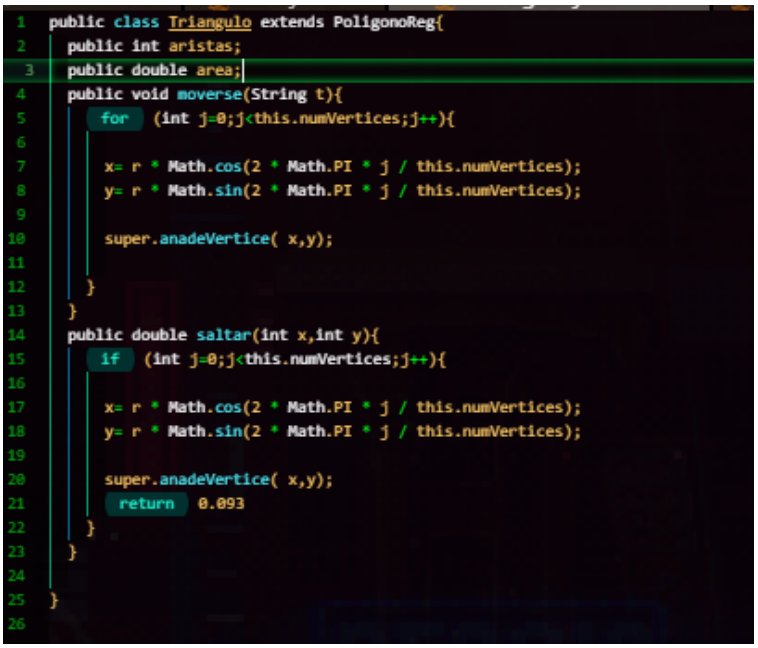
```
WMC: 1
DIT: 1
NOC: 0
      Metodo  anadeNVertices:
            WMC: 1
      Metodo  obtieneArea:
            WMC: 0
```

Clase Lala

```
public class Lala extends PoligonoReg{
  public int atr1;
  public int atr2;
  public void metodo1(String le){
    for (int j=0;j<this.numVertices;j++){

      x= r * Math.cos(2 * Math.PI * j / this.numVertices);
      y= r * Math.sin(2 * Math.PI * j / this.numVertices);

      super.anadeVertice( x,y);

    }
  }
  public double metodo2(int z){
    for (int j=0;j<this.numVertices;j++){

      x= r * Math.cos(2 * Math.PI * j / this.numVertices);
      y= r * Math.sin(2 * Math.PI * j / this.numVertices);

      super.anadeVertice( x,y);
      return 0.093
    }
  }
}
```

```
WMC: 2
DIT: 0
NOC: 0
      Metodo  metodo1:
            WMC: 1
      Metodo  metodo2:
            WMC: 1
```

Clase Triangulo



```
public class Triangulo extends PoligonoReg{
  public int aristas;
  public double area;
  public void moverse(String t){
    for (int j=0;j<this.numVertices;j++){

      x= r * Math.cos(2 * Math.PI * j / this.numVertices);
      y= r * Math.sin(2 * Math.PI * j / this.numVertices);

      super.anadeVertice( x,y);

    }
  }
  public double saltar(int x,int y){
    if (int j=0;j<this.numVertices;j++){

      x= r * Math.cos(2 * Math.PI * j / this.numVertices);
      y= r * Math.sin(2 * Math.PI * j / this.numVertices);

      super.anadeVertice( x,y);
      return 0.093
    }
  }

}
```

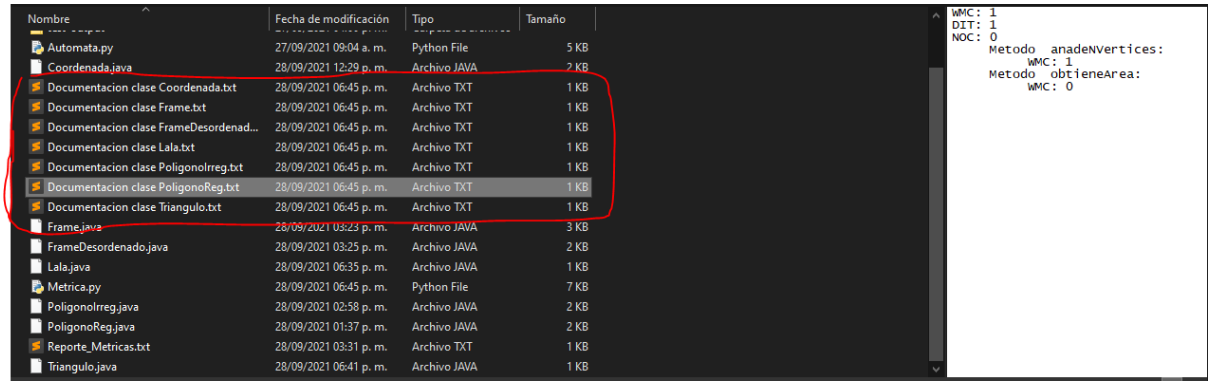
```
WMC: 2
DIT: 0
NOC: 0
      Metodo  moverse:
            WMC: 1
      Metodo  saltar:
            WMC: 1
```

Clase Coordenada

```
public class Coordenada extends PoligonoIrreg{

    private double x;
    private double y;

    public Coordenada(double x, double y) {

        this.x = x;

        this.y = y;

    }

    //Metodo getter de x

    public double abcisa( ) {
        return x;
     }



    //Metodo getter de y

    public double ordenada( ) {
        return y;
    }



    //Sobreescritura del método de la superclase objeto para imprimir con System.out.println( )

    @Override

    public String toString( ) {

        return "[" + x + "," + y + "]";

    }

}
```

```
WMC: 0
DIT: 0
NOC: 0
      Metodo  abcisa:
            WMC: 0
      Metodo  ordenada:
            WMC: 0
      Metodo  toString:
            WMC: 0
```

Archivos generados con sus respectivos nombres con su correcta documentación:



Preguntas

- ¿Por qué considera que puede ser útil estimar y medir el software?

  Es útil ya que nos permite seguir y controlar el desarrollo de los proyectos, así podemos evaluar si llevamos un buen avance o decisiones.

  Si no pudiéramos medir sería muy difícil el probar o utilizar diferentes técnicas y métodos.

- ¿Qué relación guarda con la calidad del producto y del proceso de desarrollo de un sistema de SW?

  Si revisamos periódicamente el avance del proyecto podremos identificar si hemos hecho cambios buenos o he mejorado el producto, en caso de que se encuentre alguna deficiencia o posibilidad de mejorar se realizará lo más pronto posible, siendo así, que al final entregaremos un producto de calidad.

- ¿Qué métricas utilizará para estimar su producto de software?

  Utilizaremos las 3 (WMC,DIT,NOC) ya que cada una aporta información muy valiosa para evaluar cada aspecto y perspectiva de complejidad, por ejemplo la métrica WMC nos ayudará en saber que tan compleja es una clase es decir algorítmicamente un aproximado de su peso o complejidad ya que se evalúan cuantas estructuras condicionales y ciclos tiene cada método de esa clase, la métrica DIT nos ayuda a conocer el alcance máximo que tuvo una clase padre y sus ramificaciones y por último la métrica NOC es perfecta para saber el ancho de nuestro follaje en el árbol de clases padre e hijo ya que señala cuántos hijos directos tiene cada clase