Instituto Politécnico Nacional

Escuela Superior de Cómputo

Unidad de académica: Paradigmas de programación

Actividad: “Creación y Uso de la Clase Lectura Números para Lectura de Datos Numéricos”

Equipo:

• Elizalde Hernández Alan

• Reyes Ruíz Yoselyn Estefany

• Solares Velasco Arturo Misael

• Solís Lugo Mayra

• Toral Hernández Leonardo Javier

Grupo: 3CV1

Profesor: García Floriano Andrés

Fecha:

17 de Mayo 2024

Reporte de Práctica: Análisis y Comparación

Introducción

Ambos archivos implementan un sistema de clases para figuras geométricas, incluyendo un ejemplo de polimorfismo. Este reporte analiza la implementación y funcionalidad de ambos archivos, destacando similitudes, diferencias y mejores prácticas.

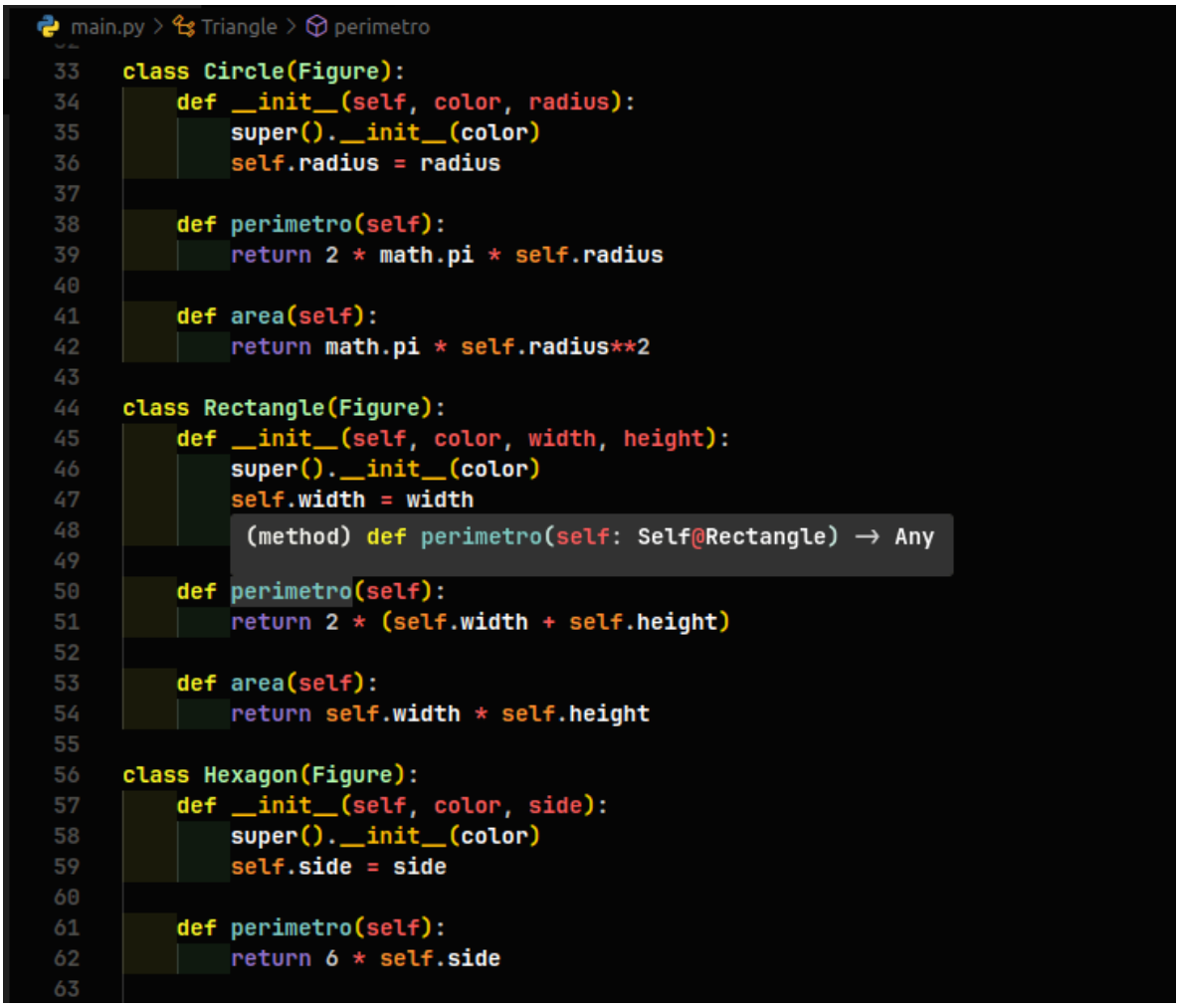
Contenido del archivo `main.py`

El archivo `main.py` define una jerarquía de clases para representar figuras geométricas. A continuación, se detallan las clases y sus funcionalidades:

1. **Clase `Figure`**:
   - Clase abstracta que define los métodos abstractos `perimetro` y `area`.
   - Tiene un atributo `color` y un método `getColor`.
2. **Clase `Triangle`**:
   - Hereda de `Figure`.
   - Define los atributos `a`, `b` y `c` para los lados del triángulo.
   - Implementa los métodos `perimetro` y `area` usando la fórmula de Herón.
3. **Clase `Circle`**:
   - Hereda de `Figure`.
   - Define el atributo `radius`.
   - Implementa los métodos `perimetro` y `area` basándose en el radio del círculo.
4. **Clase `Rectangle`**:
   - Hereda de `Figure`.
   - Define los atributos `width` y `height`.
   - Implementa los métodos `perimetro` y `area`.
5. **Clase `Hexagon`**:
   - Hereda de `Figure`.
   - Define el atributo `side`.
   - Implementa los métodos `perimetro` y `area`.
6. **Demostración de polimorfismo**:
   - Se crean instancias de cada figura y se almacenan en una lista.

- Se itera sobre la lista para imprimir el color, perímetro y área de cada figura.

7. Código:



Contenido del archivo `Main.java`

El archivo `Main.java` también define una jerarquía de clases para figuras geométricas. A continuación, se detallan las clases y sus funcionalidades:

1. **Clase `Figure`**:

    - Clase abstracta que define los métodos abstractos `perimetro` y `area`.
    - Tiene un atributo `color` y un método `getColor`.
2. **Clase `Triangle`**:
    - Hereda de `Figure`.
    - Define los atributos `a`, `b` y `c` para los lados del triángulo.
    - Implementa los métodos `perimetro` y `area` usando la fórmula de Herón.
3. **Clase `Circle`**:
    - Hereda de `Figure`.
    - Define el atributo `radius`.
    - Implementa los métodos `perimetro` y `area`.
4. **Clase `Rectangle`**:
    - Hereda de `Figure`.
    - Define los atributos `width` y `height`.
    - Implementa los métodos `perimetro` y `area`.
5. **Clase `Hexagon`**:
    - Hereda de `Figure`.
    - Define el atributo `side`.
    - Implementa los métodos `perimetro` y `area`.
6. **Clase `Main`**:
    - Método `main` que crea instancias de cada figura y las almacena en un arreglo.
    - Se itera sobre el arreglo para imprimir el color, perímetro y área de cada figura.

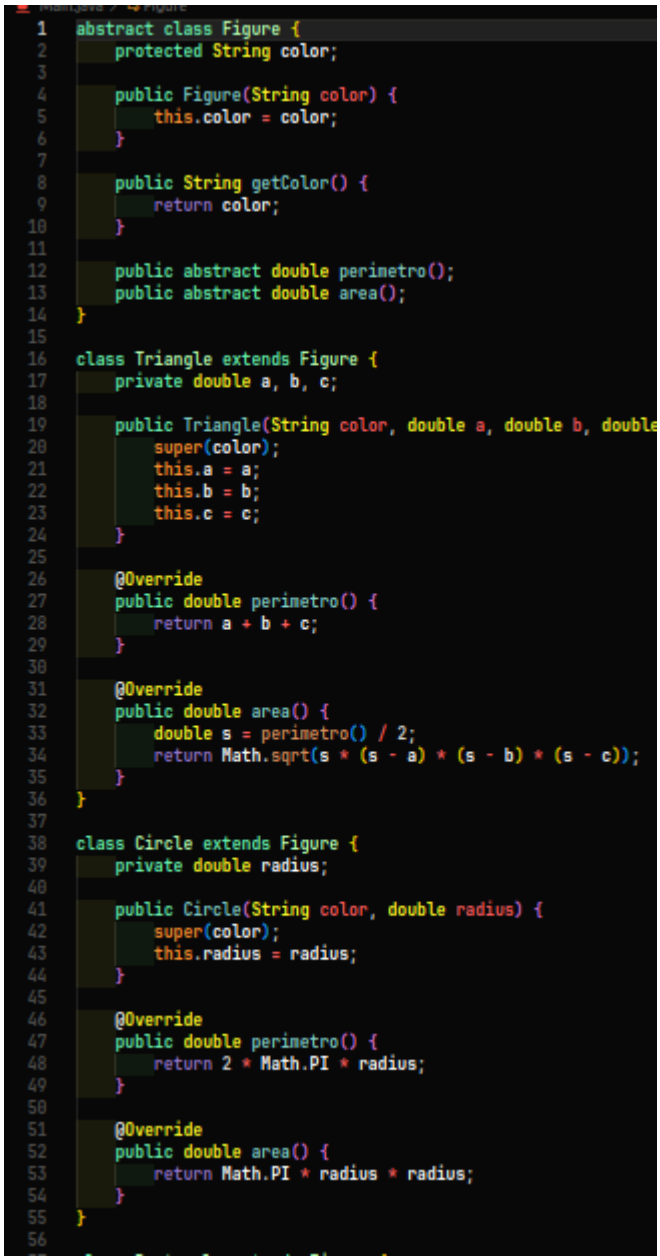
Comparación y Análisis

1. **Diseño de Clases**:
    - Ambas implementaciones siguen un diseño similar, utilizando clases abstractas para definir la interfaz común (`Figure`) y clases concretas para las diferentes figuras geométricas (`Triangle`, `Circle`, `Rectangle`, `Hexagon`).
2. **Polimorfismo**:
    - Ambas implementaciones demuestran polimorfismo creando instancias de diferentes figuras y almacenándolas en una colección

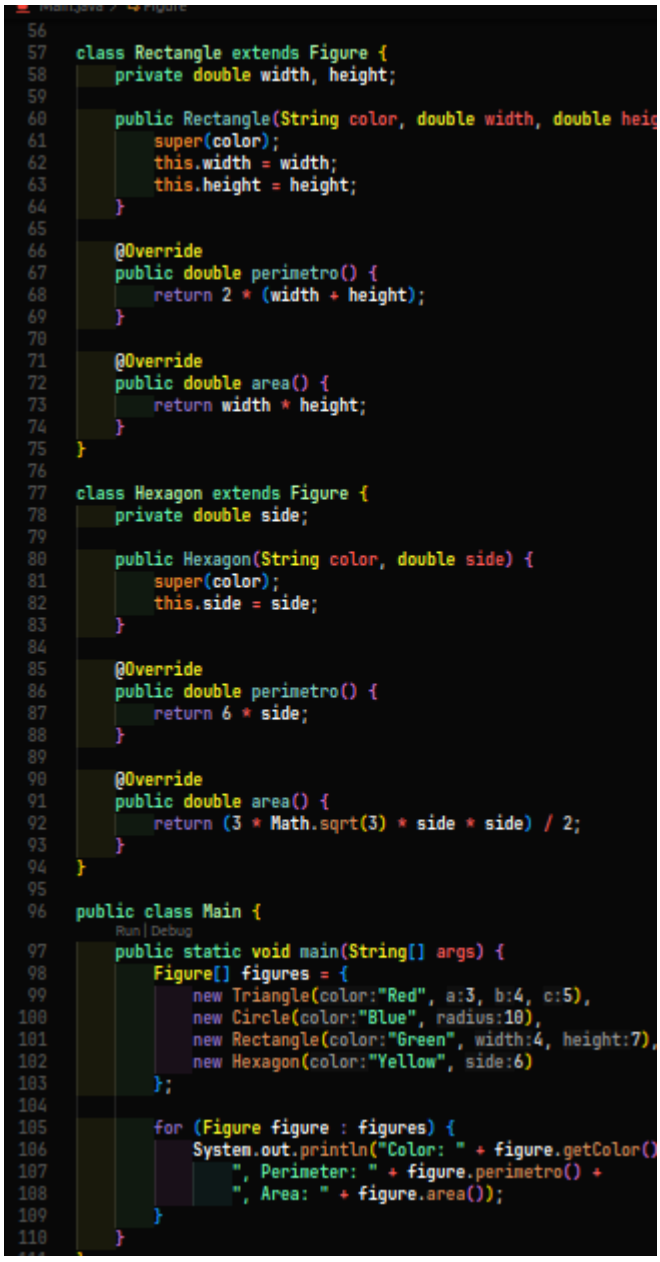
(lista en Python, arreglo en Java), luego iterando sobre la colección para llamar a los métodos comunes.

3. **Sintaxis y Estilo**:
    - La implementación en Python es más concisa debido a la naturaleza del lenguaje, mientras que Java requiere más código boilerplate (por ejemplo, la declaración explícita de tipos y el método `main`).
4. **Cálculo de Perímetro y Área**:
    - Ambos archivos implementan correctamente las fórmulas para calcular el perímetro y el área de cada figura.

Código:



```java
abstract class Figure {
    protected String color;

    public Figure(String color) {
        this.color = color;
    }

    public String getColor() {
        return color;
    }

    public abstract double perimetro();
    public abstract double area();
}

class Triangle extends Figure {
    private double a, b, c;

    public Triangle(String color, double a, double b, double
        super(color);
        this.a = a;
        this.b = b;
        this.c = c;
    }

    @Override
    public double perimetro() {
        return a + b + c;
    }

    @Override
    public double area() {
        double s = perimetro() / 2;
        return Math.sqrt(s * (s - a) * (s - b) * (s - c));
    }
}

class Circle extends Figure {
    private double radius;

    public Circle(String color, double radius) {
        super(color);
        this.radius = radius;
    }

    @Override
    public double perimetro() {
        return 2 * Math.PI * radius;
    }

    @Override
    public double area() {
        return Math.PI * radius * radius;
    }
}
```



```java
class Rectangle extends Figure {
    private double width, height;

    public Rectangle(String color, double width, double heig
        super(color);
        this.width = width;
        this.height = height;
    }

    @Override
    public double perimetro() {
        return 2 * (width + height);
    }

    @Override
    public double area() {
        return width * height;
    }
}

class Hexagon extends Figure {
    private double side;

    public Hexagon(String color, double side) {
        super(color);
        this.side = side;
    }

    @Override
    public double perimetro() {
        return 6 * side;
    }

    @Override
    public double area() {
        return (3 * Math.sqrt(3) * side * side) / 2;
    }
}

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Figure[] figures = {
            new Triangle(color:"Red", a:3, b:4, c:5),
            new Circle(color:"Blue", radius:10),
            new Rectangle(color:"Green", width:4, height:7),
            new Hexagon(color:"Yellow", side:6)
        };

        for (Figure figure : figures) {
            System.out.println("Color: " + figure.getColor()
                ", Perimeter: " + figure.perimetro() +
                ", Area: " + figure.area());
        }
    }
```

Conclusión

Ambas implementaciones son efectivas y demuestran un buen uso de la programación orientada a objetos y polimorfismo. La elección entre Python y Java puede depender del contexto del proyecto, requisitos de rendimiento, y preferencias del equipo de desarrollo.