



# 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:

```
main.py > Triangle > perimetro
1  from abc import ABC, abstractmethod
2  import math
3
4  class Figure(ABC):
5      def __init__(self, color):
6          self.color = color
7
8      def getColor(self):
9          return self.color
10
11     @abstractmethod
12     def perimetro(self):
13         pass
14
15     @abstractmethod
16     def area(self):
17         pass
18
19     class Triangle(Figure):
20         def __init__(self, color, a, b, c):
21             super().__init__(color)
22             self.a = a
23             self.b = b
24             self.c = c
25
26         def perimetro(self):
27             return self.a + self.b + self.c
28
29         def area(self):
30             s = self.perimetro() / 2
31             return math.sqrt(s * (s - self.a) * (s - self.b) * (s - self.c))
```

```

main.py > Triangle > perimetro
33 class Circle(Figure):
34     def __init__(self, color, radius):
35         super().__init__(color)
36         self.radius = radius
37
38     def perimetro(self):
39         return 2 * math.pi * self.radius
40
41     def area(self):
42         return math.pi * self.radius**2
43
44 class Rectangle(Figure):
45     def __init__(self, color, width, height):
46         super().__init__(color)
47         self.width = width
48         (method) def perimetro(self: Self@Rectangle) → Any
49
50     def perimetro(self):
51         return 2 * (self.width + self.height)
52
53     def area(self):
54         return self.width * self.height
55
56 class Hexagon(Figure):
57     def __init__(self, color, side):
58         super().__init__(color)
59         self.side = side
60
61     def perimetro(self):
62         return 6 * self.side
63

```

```

63
64     def area(self):
65         return (3 * math.sqrt(3) * self.side**2) / 2
66
67 # Demostración de polimorfismo
68 figures = [
69     Triangle("Red", 3, 4, 5),
70     Circle("Blue", 10),
71     Rectangle("Green", 4, 7),
72     Hexagon("Yellow", 6)
73 ]
74
75 for figure in figures:
76     print(f"Color: {figure.getColor()}, Perimeter: {figure.perimetro()}, Area: {figure.area()}")

```

## 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:

```
1 abstract class Figure {
2     protected String color;
3
4     public Figure(String color) {
5         this.color = color;
6     }
7
8     public String getColor() {
9         return color;
10    }
11
12    public abstract double perimetro();
13    public abstract double area();
14 }
15
16 class Triangle extends Figure {
17     private double a, b, c;
18
19     public Triangle(String color, double a, double b, double c) {
20         super(color);
21         this.a = a;
22         this.b = b;
23         this.c = c;
24     }
25
26     @Override
27     public double perimetro() {
28         return a + b + c;
29     }
30
31     @Override
32     public double area() {
33         double s = perimetro() / 2;
34         return Math.sqrt(s * (s - a) * (s - b) * (s - c));
35     }
36 }
37
38 class Circle extends Figure {
39     private double radius;
40
41     public Circle(String color, double radius) {
42         super(color);
43         this.radius = radius;
44     }
45
46     @Override
47     public double perimetro() {
48         return 2 * Math.PI * radius;
49     }
50
51     @Override
52     public double area() {
53         return Math.PI * radius * radius;
54     }
55 }
56
```

```

56
57 class Rectangle extends Figure {
58     private double width, height;
59
60     public Rectangle(String color, double width, double height) {
61         super(color);
62         this.width = width;
63         this.height = height;
64     }
65
66     @Override
67     public double perimetro() {
68         return 2 * (width + height);
69     }
70
71     @Override
72     public double area() {
73         return width * height;
74     }
75 }
76
77 class Hexagon extends Figure {
78     private double side;
79
80     public Hexagon(String color, double side) {
81         super(color);
82         this.side = side;
83     }
84
85     @Override
86     public double perimetro() {
87         return 6 * side;
88     }
89
90     @Override
91     public double area() {
92         return (3 * Math.sqrt(3) * side * side) / 2;
93     }
94 }
95
96 public class Main {
97     public static void main(String[] args) {
98         Figure[] figures = {
99             new Triangle(color:"Red", a:3, b:4, c:5),
100             new Circle(color:"Blue", radius:10),
101             new Rectangle(color:"Green", width:4, height:7),
102             new Hexagon(color:"Yellow", side:6)
103         };
104
105         for (Figure figure : figures) {
106             System.out.println("Color: " + figure.getColor()
107                 + ", Perimeter: " + figure.perimetro() +
108                 ", Area: " + figure.area());
109         }
110     }
111 }

```

## 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.