TITULO DE LABORATORIO

FIGURAS GEOMETRICAS

HANNA KATHERINE ABRIL GÓNGORA
KAROL ASLEY ORJUELA MAPE

UNIVERSIDAD MANUELA BELTRÁN

PROGRAMACIÓN ORIENTADA A OBJECTOS

DOCENTE

DIANA MARCELA TOQUICA RODRÍGUEZ

BOGOTÁ DC VIERNES 31 DE MARZO

1. **PREGUNTAS ORIENTADORAS**

1.1. ¿Cómo se diferencia el polimorfismo de la sobrecarga de métodos en Python y Java?

Polimorfismo

- En Python, el polimorfismo se logra de manera dinámica sin requerir una estructura formal de herencia o interfaces. Los objetos pueden responder a los mismos métodos incluso si no comparten una relación de herencia explícita, siempre y cuando implementen esos métodos.
- En Java, el polimorfismo se basa en la herencia y las interfaces. Los objetos pueden ser tratados polimórficamente si comparten una relación de herencia común o si implementan una interfaz común. Esto permite que se invoquen métodos de manera polimórfica a través de referencias de clase base o interfaces, y que los objetos reales (instancias de subclases) proporcionen su propia implementación de esos métodos.

Sobrecarga

- Python: Python no admite la sobrecarga de métodos en el sentido tradicional. Esto significa que no puedes definir múltiples métodos con el mismo nombre, pero diferentes firmas (número o tipos de parámetros) en una clase. Cuando defines múltiples métodos con el mismo nombre, el último método definido sobrescribe los anteriores.
- Java: Java admite la sobrecarga de métodos, lo que significa que puedes definir múltiples métodos en una clase con el mismo nombre, pero diferentes firmas (número o tipos de parámetros). Java puede determinar cuál método invocar en función de la firma del método y los argumentos proporcionados en la llamada al método.

1.2. ¿Cómo se puede aplicar el polimorfismo para mejorar la legibilidad y mantenibilidad del código?

El polimorfismo, la capacidad de un objeto de comportarse de diferentes maneras en diferentes contextos, ofrece una serie de beneficios para la legibilidad y mantenibilidad del código:

1. Mayor legibilidad:

- **Código más conciso:** Se evita la duplicación de código para funcionalidades similares en diferentes tipos de objetos.
- **Nombres más descriptivos:** Los métodos pueden tener nombres que reflejen su propósito general, sin necesidad de detalles específicos del tipo de objeto.
- **Mayor claridad:** El código se vuelve más fácil de entender al separar la lógica del comportamiento específico de cada tipo de objeto.

2. Mayor mantenibilidad:

- **Facilidad de cambio:** Se pueden modificar los detalles de implementación de un método sin afectar a otras partes del código que lo utilizan.
- **Extensibilidad:** Se pueden agregar nuevos tipos de objetos sin necesidad de modificar el código existente.
- **Facilidad de prueba:** Se pueden probar los métodos de forma independiente, sin necesidad de crear instancias de todas las clases que los implementan.

1.3. ¿Cuándo es apropiado utilizar una clase abstracta en lugar de una clase concreta?

Es apropiado utilizar una clase abstracta en lugar de una clase concreta cuando necesitas definir una interfaz común, evitar la duplicación de código, proporcionar una implementación base para métodos comunes, establecer un contrato claro para las clases derivadas o restringir la creación de instancias directas.

1.4. ¿Qué ventajas ofrece el uso de clases abstractas en el diseño de software?

- Reutilización de código: Las clases abstractas nos permiten definir una jerarquía de clases para un proyecto de software. Esto facilita el uso de herencia en la programación, lo que a su vez promueve la reutilización de métodos y variables de la superclase.
- Agrupación de comportamientos comunes: Las clases abstractas nos permiten agrupar los comportamientos comunes entre las clases de un proyecto de software. Al hacerlo, simplificamos los procesos de desarrollo y mantenimiento. Además, proporcionan una forma de encapsular métodos y variables en una unidad, mejorando la seguridad de los datos.

En resumen, las clases abstractas son una herramienta valiosa para estructurar y organizar el código, fomentando la eficiencia y la claridad en el diseño de software.

2. EXPLICACIÓN DEL CÓDIGO EN PYTHON

- Esta línea importa las funcionalidades necesarias para trabajar con métodos abstractos en Python. '*abstractmethod'*, se utiliza para definir métodos abstractos y se exporta ABC para trabajar con clases abstractas, es decir, métodos que deben ser implementados por las clases hijas, pero no por la clase abstracta en sí. Además, se importa el módulo *math* para realizar operaciones matemáticas.

```
1   from abc import ABC, abstractmethod
2   import math
```

- *class Figura():* Aquí se define la clase abstracta Figura. Esta clase tiene un constructor *(__init__)* que toma un parámetro llamado figura y lo asigna a un atributo de instancia también llamado figura. @abstractmethod: Este decorador indica que los métodos *calcular_area()* y *calcular_perimetro()* son métodos abstractos, es decir, deben ser implementados por las clases hijas de Figura pero no por Figura en sí.

  Luego, se definen varias clases que heredan de la clase abstracta Figura, como *Cuadrado, Rectangulo, Triangulo,* etc. Cada una de estas clases implementa los métodos *calcular_area()* y en algunos casos calcular_perimetro().

```
4   class Figura(ABC):
5       def __init__(self):
6           pass
7
8       @abstractmethod
9       def calcular_area():
10          pass
11
```

- *class Cuadrado(Figura):* Se implementa la clase abstracta de calcular área, utilizando la fórmula para calcular el área de un cuadrado, la cual es:

$$A = L * L = L^2$$

Donde:

- A es el área del cuadrado.
- L es la longitud de uno de los lados del cuadrado.

```
13  class Cuadrado(Figura):
14      def __init__(self, lado):
15          super().__init__()
16          self.lado = lado
17
18      def calcular_area(self):
19          return self.lado**2
20
```

- *class Rectangulo(Figura):* Se implementa la clase abstracta de calcular área, utilizando la fórmula para calcular el área de un rectángulo, la cual es:

$$A = B * H$$

Donde:

- A es el área del rectángulo.
- B es la longitud de la base del rectángulo.
- H es la altura del paralelogramo, que es la distancia perpendicular entre la base y el lado opuesto.

```
22  class Rectangulo(Figura):
23      def __init__(self, base, altura):
24          super().__init__()
25          self.base = base
26          self.altura = altura
27
28      def calcular_area(self):
29          return self.base * self.altura
```

- *class Triangulo(Figura):* Se implementa la clase abstracta de calcular área, utilizando la fórmula para calcular el área de un triángulo, la cual es:

$$A = \frac{B * H}{2}$$

Donde:

- A es el área del rectángulo.
- B es la longitud de la base del rectángulo.
- H es la altura del paralelogramo, que es la distancia perpendicular entre la base y el lado opuesto.

```
31  class Triangulo(Figura):
32      def __init__(self, base, altura):
33          super().__init__()
34          self.base = base
35          self.altura = altura
36
37      def calcular_area(self):
38          return (self.base * self.altura) / 2
39
```

- *class Rombo(Figura):* Se implementa la clase abstracta de calcular área, utilizando la fórmula para calcular el área de un rombo, la cual es:

$$A = \frac{D * d}{2}$$

Donde:

- A es el área del rombo.
- D es la longitud de la diagonal mayor del rombo.
- d es la longitud de la diagonal menor del rombo.

```
40  class Rombo(Figura):
41      def __init__(self, diagonal_l, diagonal_c):
42          super().__init__()
43          self.diagonal_l = diagonal_l
44          self.diagonal_c = diagonal_c
45
46      def calcular_area(self):
47          return (self.diagonal_l * self.diagonal_c) / 2
48
```

- *class Pentagono(Figura):* Se implementa la clase abstracta de calcular área, utilizando la fórmula para calcular el área de un pentágono, la cual es:

$$apotema = \frac{L}{2 * \tan\left(\frac{\pi}{5}\right)}$$

$$A = \frac{L * apotema}{2}$$

Donde:

- A es el área del pentágono.
- L es la longitud de uno de los lados del pentágono.
- La apotema (apotema) es la distancia perpendicular desde el centro del pentágono hasta uno de sus lados. La apotema se puede calcular usando la fórmula apotema=$\frac{L}{2*\tan\left(\frac{\pi}{5}\right)}$ donde tan $\left(\frac{\pi}{5}\right)$ es la tangente del ángulo central de 72 grados.

```
49  class Pentagono(Figura):
50      def __init__(self, lado, apotema):
51          super().__init__()
52          self.lado = lado
53          self.apotema = apotema
54
55      def calcular_area(self):
56          return (5 * self.lado * self.apotema) / 2
57
```

- *class Hexagono(Figura):* Se implementa la clase abstracta de calcular área, utilizando la fórmula para calcular el área de un hexágono, la cual es:

$$apotema = \frac{L}{2 * \tan\left(\frac{\pi}{6}\right)}$$

$$A = \frac{3 * \sqrt{3} * L^2}{2}$$

Donde:

- A es el área del hexágono.
- L es la longitud de uno de los lados del hexágono.

```
58  class Hexagono(Figura):
59      def __init__(self, lado, apotema):
60          super().__init__()
61          self.lado = lado
62          self.apotema = apotema
63
64      def calcular_area(self):
65          return (3 * math.sqrt(3) * self.apotema**2) / 2
66
```

- *class Circulo(Figura):* Se implementa la clase abstracta de calcular área, utilizando la fórmula para calcular el área de un círculo, la cual es:

$$A = \pi * r^2$$

Donde:

- A es el área del círculo.
- π es una constante aproximadamente igual a 3.14159. Es la relación entre la circunferencia de un círculo y su diámetro.
- r es el radio del círculo, es decir, la distancia desde el centro del círculo hasta cualquier punto de su circunferencia.

```
67  class Circulo(Figura):
68      def __init__(self, radio):
69          super().__init__()
70          self.radio = radio
71
72      def calcular_area(self):
73          return math.pi * self.radio**2
74
```

- *class Trapecio(Figura):* Se implementa la clase abstracta de calcular área, utilizando la fórmula para calcular el área de un trapecio, la cual es:

$$A = \frac{(B_1 + B_2) * H}{2}$$

Donde:

- A es el área del trapecio.
- B_1 es la longitud de la base larga del trapecio.

- B_2 es la longitud de la base corta del trapecio.
- H es la altura del trapecio, que es la distancia perpendicular entre las bases.

```
75  class Trapecio(Figura):
76      def __init__(self, base_larga, base_corta, altura):
77          super().__init__()
78          self.base_larga = base_larga
79          self.base_corta = base_corta
80          self.altura = altura
81
82      def calcular_area(self):
83          return ((self.base_larga + self.base_corta) / 2) * self.alt
84
```

- *class Paralelogramo(Figura):* Se implementa la clase abstracta de calcular área, utilizando la fórmula para calcular el área de un paralelogramo, la cual es:

$$A = B * H$$

Donde:

- A es el área del paralelogramo.
- B es la longitud de la base del paralelogramo.
- H es la altura del paralelogramo, que es la distancia perpendicular entre la base y el lado opuesto.

```
85  class Paralelogramo(Figura):
86      def __init__(self, base, altura):
87          super().__init__()
88          self.base = base
89          self.altura = altura
90
91      def calcular_area(self):
92          return self.base * self.altura
93
```

- *class Gestor_Figuras():* Esta clase gestiona las figuras geométricas. Tiene un constructor *(__init__)* que inicializa una lista vacía llamada figuras.

  agregar_figura(self, figura): Método para agregar una figura a la lista figuras.

  imprimir_areas(self): Método para imprimir las áreas de todas las figuras en la lista figuras, como una especie de historial de calculos.

```
 97  class Gestor_Figuras():
 98      def __init__(self):
 99          self.figuras = []
100
101      def agregar_figura(self, figura):
102          self.figuras.append(figura)
103
104      def imprimir_areas(self):
105          for figura in self.figuras:
106              area = figura.calcular_area()
107              print(f"El área de la {figura.__class__.__name__} es: {a
108
```

- Finalmente, la función *main()* es donde se crea una instancia de *Gestor_Figuras* y se ejecuta el programa principal en un bucle mientras el usuario no seleccione la opción de salir (11). Dentro del bucle, se presentan las opciones de figuras para calcular áreas y se captura la entrada del usuario para realizar los cálculos correspondientes.

```python
def main():
    gestor_figuras = Gestor_Figuras()

    while True:
        print("""

            Figuras geometricas:

            1. Cuadrado
            2. Rectangulo
            3. Triangulo
            4. Rombo
            5. Pentagono
            6. Hexagono
            7. Circulo
            8. Trapecio
            9. Paralelogramo
            10. Historial de areas calculadas
            11. Salir

            """)

        opcion=input("Ingrese el número de la figura que desea calcular o '11' para salir: ")

        match opcion:

            case '1':
                lado = float(input("Ingrese la longitud del lado del cuadrado: "))
                cuadrado = Cuadrado(lado)
                gestor_figuras.agregar_figura(cuadrado)
                area = cuadrado.calcular_area()
                print(f"El área del cuadrado es: {area}")

            case '2':
                lado_1 = float(input("Ingrese el lado 1 del rectángulo: "))
                lado_2 = float(input("Ingrese el lado 2 del rectángulo: "))
                rectangulo = Rectangulo(lado_1, lado_2)
                gestor_figuras.agregar_figura(rectangulo)
                area = rectangulo.calcular_area()
                print(f"El área del rectangulo es: {area}")

```

```
150                 case '3':
151                     base = float(input("Ingrese la base del triángulo: "))
152                     altura = float(input("Ingrese la altura del triángulo: "))
153                     triangulo = Triangulo(base, altura)
154                     gestor_figuras.agregar_figura(triangulo)
155                     area = triangulo.calcular_area()
156                     print(f"El área del triángulo es: {area}")
157 
158                 case '4':
159                     diagonal_l = float(input("Ingrese la diagonal mayor del rombo: "))
160                     diagonal_c = float(input("Ingrese la diagonal menor del rombo: "))
161                     rombo = Rombo(diagonal_l, diagonal_c)
162                     gestor_figuras.agregar_figura(rombo)
163                     area = rombo.calcular_area()
164                     print(f"El área del rombo es: {area}")
165 
166                 case '5':
167                     lado = float(input("Ingrese el lado del pentágono: "))
168                     apotema = float(input("Ingrese la apotema del pentágono: "))
169                     pentagono = Pentagono(lado, apotema)
170                     gestor_figuras.agregar_figura(pentagono)
171                     area = pentagono.calcular_area()
172                     print(f"El área del pentagono es: {area}")
173 
174                 case '6':
175                     lado = float(input("Ingrese el lado del hexágono: "))
176                     apotema = float(input("Ingrese la apotema del hexágono: "))
177                     hexagono = Hexagono(lado, apotema)
178                     gestor_figuras.agregar_figura(hexagono)
179                     area = hexagono.calcular_area()
180                     print(f"El área del hexagono es: {area}")
181 
182                 case '7':
183                     radio = float(input("Ingrese el radio del círculo: "))
184                     circulo = Circulo(radio)
185                     gestor_figuras.agregar_figura(circulo)
186                     area = circulo.calcular_area()
187                     print(f"El área del círculo es: {area}")
188 
```

```
189                 case '8':
190                     base_l = float(input("Ingrese la base larga del trapecio: "))
191                     base_c = float(input("Ingrese la base corta del trapecio: "))
192                     altura = float(input("Ingrese la altura del trapecio: "))
193                     trapecio = Trapecio(base_l,base_c,altura)
194                     gestor_figuras.agregar_figura(trapecio)
195                     area = trapecio.calcular_area()
196                     print(f"El área del trapecio es: {area}")
197 
198                 case '9':
199                     base = float(input("Ingrese la base del paralelogramo: "))
200                     altura = float(input("Ingrese la altura del paralelogramo: "))
201                     paralelogramo = Paralelogramo(base,altura)
202                     gestor_figuras.agregar_figura(paralelogramo)
203                     area = paralelogramo.calcular_area()
204                     print(f"El área del paralelogramo es: {area}")
205 
206                 case '10':
207                     print("Historial de calculos.")
208                     gestor_figuras.imprimir_areas()
209 
210                 case '11':
211                     print("Saliendo del programa...")
212                     break
213 
214                 case _:
215                     print("Opción no válida. Intente de nuevo.")
216 
217     if __name__ == '__main__':
218         main()
```

3. **EXPLICACION DEL CODIGO UML**

```
@startuml Figuras

abstract class Figura {
    + calcular_area(): float
    + calcular_perimetro(): float
}

class Cuadrado {
    - lado: float
    + calcular_area(): float
}

class Rectangulo {
    - lado_1: float
    - lado_2: float
    + calcular_area(): float
}

class Triangulo {
    - base: float
    - altura: float
    + calcular_area(): float
}

class Rombo {
    - diagonal_l: float
    - digonal_c: float
    + calcular_area(): float
}

class Pentagono {
    - lado: float
    + calcular_area(): float
}

```

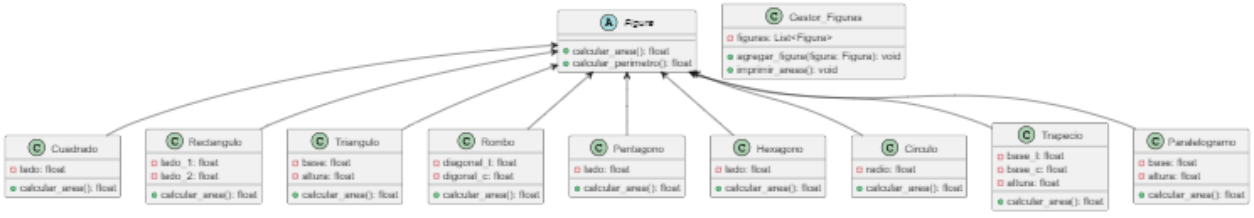
```
36  class Hexagono {
37      - lado: float
38      + calcular_area(): float
39  }
40
41  class Circulo {
42      - radio: float
43      + calcular_area(): float
44  }
45
46  class Trapecio {
47      - base_l: float
48      - base_c: float
49      - altura: float
50      + calcular_area(): float
51  }
52
53  class Paralelogramo {
54      - base: float
55      - altura: float
56      + calcular_area(): float
57  }
58
59  class Gestor_Figuras {
60      - figuras: List<Figura>
61      + agregar_figura(figura: Figura): void
62      + imprimir_areas(): void
63  }
64
65  Figura <|-- Cuadrado
66  Figura <|-- Rectangulo
67  Figura <|-- Triangulo
68  Figura <|-- Rombo
69  Figura <|-- Pentagono
70  Figura <|-- Hexagono
71  Figura <|-- Circulo
72  Figura <|-- Trapecio
73  Figura <|-- Paralelogramo
74
75
76  @enduml
77
```

- Línea 3-5: Se define una clase abstracta “Figura” con un método para calcular el perímetro, pero no se implementa (es decir, se espera que las subclases proporcionen la implementación).
- Línea 8-11: Se define una clase “Cuadrado” con un atributo para el lado y un método para calcular el área.
- Línea 13-17: Similar a “Cuadrado”, se tiene la clase “Rectángulo” con atributos para dos

lados diferentes y un método para calcular el área.

- Línea 19-23: La clase “Triángulo” tiene atributos para la base y la altura, junto con su propio método de cálculo del área.
- Línea 25-29: La clase “Rombo” tiene atributos para las dos diagonales y un método de cálculo del área.
- Línea 31-34: La clase “Pentágono” es similar a la del cuadrado, pero no muestra cómo calcula el área.
- Línea 36-39: Similar al pentágono, la clase “Hexágono” también tiene un atributo para el lado, pero no muestra cómo calcula el área.
- Línea 41-44: La clase “Círculo” tiene un atributo de radio y un método de cálculo del área sin implementar aquí.
- Línea 46-51: La clase “Trapecio” contiene cuatro flotantes representando los lados y una función sin cuerpo definido aún para calcular su área.
- Línea 53-57: Clase “Paralelogramo”, similar a las anteriores, contiene métodos sin cuerpo definido aún.
- Línea 59–63: Una nueva estructura llamada Gestor_Figuras que contiene una lista de figuras e incluye métodos como agregar_figura e imprimir_areas.
- Línea 65–76: Aquí parece ser parte de algún tipo de inicialización o asignación donde se crean objetos específicos desde las clases definidas anteriormente.

4. DIAGRAMA DEL CODIGO EN UML



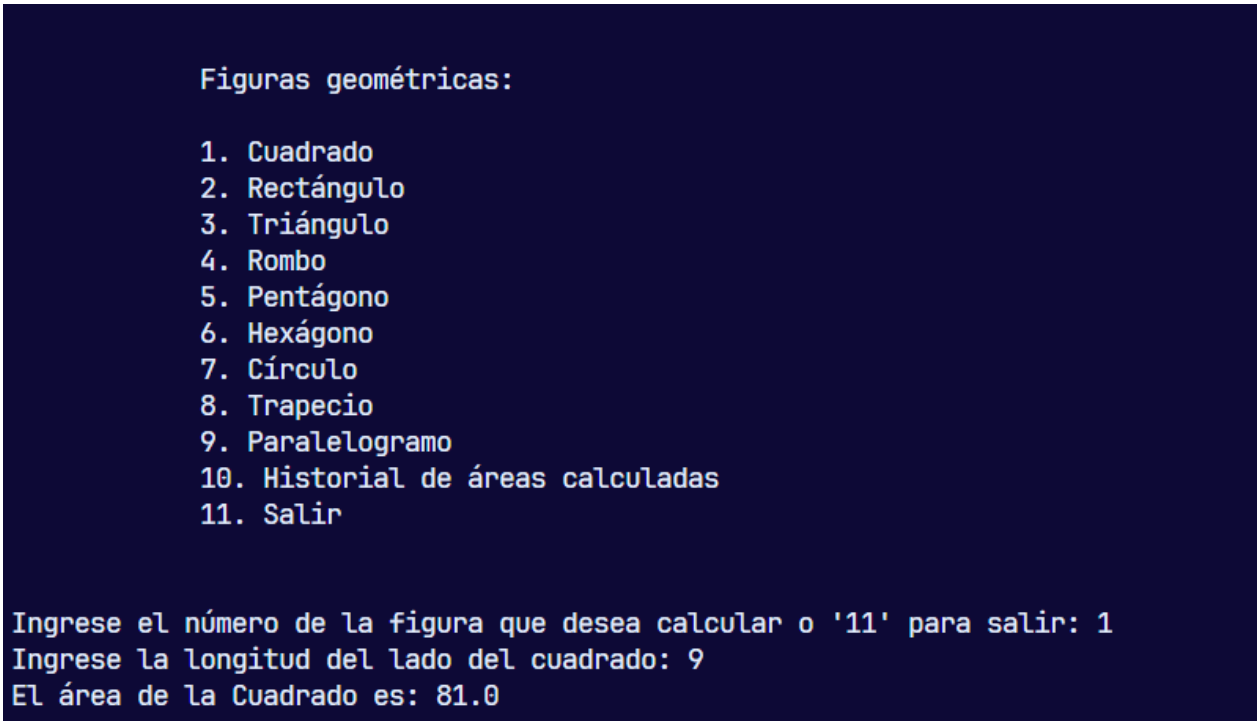
Nota. Puede que el diagrama no se vea bien por la extensión, por tal motivo se adjunta el archivo del código de este.

5. EXPLICACION Y USO

El código proporcionado es un programa en Python que permite calcular el área de diversas figuras geométricas, como cuadrados, rectángulos, triángulos, rombos, pentágonos, hexágonos, círculos, trapecios y paralelogramos. La estructura del programa se basa la programación orientada a objetos (POO), utilizando clases y métodos para representar cada figura geométrica y su respectivo cálculo de área. El programa presenta un menú interactivo mediante la estructura match, donde el usuario puede seleccionar la figura deseada y proporcionar los datos necesarios para realizar el cálculo del área. Además, el programa incluye un gestor de figuras que permite agregar nuevas figuras calculadas y mostrar un historial de las áreas calculadas. Este código es útil para realizar cálculos precisos y rápidos de áreas de figuras geométricas de manera organizada y eficiente, ofreciendo una herramienta práctica tanto para aprendizaje como para aplicaciones más avanzadas en geometría y matemáticas.

6. DEMOSTRACION DE PRUEBAS Y FUNCIONAMIENTO DEL CODIGO

6.1 Prueba 1.



En esta primera prueba se realiza el cálculo del cuadrado exitosamente.

6.2 Prueba 2.

```
            Figuras geométricas:

            1. Cuadrado
            2. Rectángulo
            3. Triángulo
            4. Rombo
            5. Pentágono
            6. Hexágono
            7. Círculo
            8. Trapecio
            9. Paralelogramo
            10. Historial de áreas calculadas
            11. Salir


Ingrese el número de la figura que desea calcular o '11' para salir: 2
Ingrese la base del rectángulo: 6
Ingrese la altura del rectángulo: 7
El área de la Rectangulo es: 42.0
```

En esta segunda prueba prueba se realiza el cálculo del rectángulo exitosamente.

6.3 Prueba 3.

```
            Figuras geométricas:

            1. Cuadrado
            2. Rectángulo
            3. Triángulo
            4. Rombo
            5. Pentágono
            6. Hexágono
            7. Círculo
            8. Trapecio
            9. Paralelogramo
            10. Historial de áreas calculadas
            11. Salir


Ingrese el número de la figura que desea calcular o '11' para salir: 3
Ingrese la base del triángulo: 8
Ingrese la altura del triángulo: 14
El área de la Triangulo es: 56.0
```

En esta tercera prueba se realiza el cálculo del triángulo exitosamente.

6.4 Prueba 4.

```
            Figuras geométricas:

            1. Cuadrado
            2. Rectángulo
            3. Triángulo
            4. Rombo
            5. Pentágono
            6. Hexágono
            7. Círculo
            8. Trapecio
            9. Paralelogramo
            10. Historial de áreas calculadas
            11. Salir


Ingrese el número de la figura que desea calcular o '11' para salir: 4
Ingrese la diagonal mayor del rombo: 59
Ingrese la diagonal menor del rombo: 78
El área de la Rombo es: 2301.0
```

En esta cuarta prueba se realiza el cálculo del rombo exitosamente.

6.5 Prueba 5.

```
            Figuras geométricas:

            1. Cuadrado
            2. Rectángulo
            3. Triángulo
            4. Rombo
            5. Pentágono
            6. Hexágono
            7. Círculo
            8. Trapecio
            9. Paralelogramo
            10. Historial de áreas calculadas
            11. Salir


Ingrese el número de la figura que desea calcular o '11' para salir: 5
Ingrese el lado del pentágono: 6
Ingrese la apotema del pentágono: 7
El área de la Pentagono es: 105.0
```

En esta quinta prueba se realiza el cálculo del pentágono exitosamente.

6.6 Prueba 6.

```
            Figuras geométricas:

            1. Cuadrado
            2. Rectángulo
            3. Triángulo
            4. Rombo
            5. Pentágono
            6. Hexágono
            7. Círculo
            8. Trapecio
            9. Paralelogramo
            10. Historial de áreas calculadas
            11. Salir


Ingrese el número de la figura que desea calcular o '11' para salir: 6
Ingrese el lado del hexágono: 8
Ingrese la apotema del hexágono: 3
El área de la Hexagono es: 23.382685902179844
```

En esta sexta prueba se realiza el cálculo del hexágono exitosamente.

6.7 Prueba 7.

```
            Figuras geométricas:

            1. Cuadrado
            2. Rectángulo
            3. Triángulo
            4. Rombo
            5. Pentágono
            6. Hexágono
            7. Círculo
            8. Trapecio
            9. Paralelogramo
            10. Historial de áreas calculadas
            11. Salir


Ingrese el número de la figura que desea calcular o '11' para salir: 7
Ingrese el radio del círculo: 7
El área de la Circulo es: 153.93804002589985
```

En esta séptima prueba se realiza el cálculo del círculo exitosamente.

6.8 Prueba 8.

```
            Figuras geométricas:

            1. Cuadrado
            2. Rectángulo
            3. Triángulo
            4. Rombo
            5. Pentágono
            6. Hexágono
            7. Círculo
            8. Trapecio
            9. Paralelogramo
            10. Historial de áreas calculadas
            11. Salir


Ingrese el número de la figura que desea calcular o '11' para salir: 8
Ingrese la base larga del trapecio: 9
Ingrese la base corta del trapecio: 5
Ingrese la altura del trapecio: 3
El área de la Trapecio es: 21.0
```

En esta octava prueba se realiza el cálculo del trapecio exitosamente.

6.9 Prueba 9.

```
            Figuras geométricas:

            1. Cuadrado
            2. Rectángulo
            3. Triángulo
            4. Rombo
            5. Pentágono
            6. Hexágono
            7. Círculo
            8. Trapecio
            9. Paralelogramo
            10. Historial de áreas calculadas
            11. Salir


Ingrese el número de la figura que desea calcular o '11' para salir: 1
Ingrese la longitud del lado del cuadrado: 95
El área de la Cuadrado es: 9025.0
```

Se realiza una última prueba del cálculo del área de un cuadrado por propósitos de muestra del

historial.

6.10 Prueba 10.

```
            Figuras geométricas:

            1. Cuadrado
            2. Rectángulo
            3. Triángulo
            4. Rombo
            5. Pentágono
            6. Hexágono
            7. Círculo
            8. Trapecio
            9. Paralelogramo
            10. Historial de áreas calculadas
            11. Salir


Ingrese el número de la figura que desea calcular o '11' para salir: 10
Historial de cálculos.
El área de la Cuadrado es: 81.0
El área de la Rectangulo es: 42.0
El área de la Triangulo es: 56.0
El área de la Rombo es: 2301.0
El área de la Pentagono es: 105.0
El área de la Hexagono es: 23.382685902179844
El área de la Circulo es: 153.93804002589985
El área de la Trapecio es: 21.0
El área de la Cuadrado es: 9025.0
```

Y finalmente en esta prueba se muestra que el programa mantiene un historial de todos los cálculos realizados incluyendo si se repiten figuras

7. DECISIÓN DE DISEÑO O CONSIDERACION IMPORTANTE

En este caso una consideración importante de diseño del programa, era que queríamos que fuera interactivo con el usuario y que sirviera como una especie de calculadora del cálculo del área de algunas figuras geométricas, por tal razón hicimos que las entradas de las medidas fueran realizadas directamente por el usuario, y que además de eso se pudieran realizar varios cálculos, no solo uno por figura, y además de eso para mantener al usuario al tanto de sus cálculos decidimos implementar una opción para ver el historial de todos sus cálculos en cuanto a los resultados de estos.

8. PILARES DE POO IMPLEMENTADOS

8.1. Abstracción:

Se utiliza en la clase abstracta Figura y en sus métodos abstractos calcular_area. La abstracción permite definir una estructura base (clase abstracta) para las figuras geométricas sin especificar cómo se implementan exactamente estas operaciones para cada figura concreta.

8.2. Herencia:

Las clases concretas como Cuadrado, Rectangulo, Triangulo, etc., heredan de la clase abstracta Figura. Esto permite reutilizar código y definir comportamientos específicos para cada tipo de figura geométrica.

8.3. Polimorfismo:

Se manifiesta en el método agregar_figura de la clase Gestor_Figuras, donde se puede agregar cualquier objeto que herede de la clase Figura, sin importar el tipo específico de figura. Además, en el método imprimir_areas, se utiliza el polimorfismo para llamar al método calcular_area de cada figura sin saber de antemano qué tipo de figura es.

9. CONCLUCIONES

Al trabajar en este código y aplicar la estructura match de Python para el menú de opciones, nuestro equipo aprendió a manejar de manera más eficiente la lógica de selección y control de flujo en programas orientados a objetos. La implementación de clases y métodos abstractos nos permitió organizar de manera más clara la jerarquía de figuras geométricas, facilitando la extensión del programa con nuevas clases de figuras en el futuro. Además, la interacción con el usuario a través de la consola nos enseñó a gestionar la entrada de datos de manera más robusta y a responder adecuadamente a las acciones del usuario, mejorando así la usabilidad y la experiencia general del programa.