Documentación de Figuras Geométricas

¿Qué es la Programación Orientada a Objetos (POO)?

Es un paradigma en el mundo de la programación que modela el diseño de objetos y clases que queremos replicar en el mundo real. En sí, disminuye los errores y hace la reutilización del código. La POO permite a los devs organizar sus códigos en unidades lógicas, lo que mejora la claridad, mantenibilidad y escalabilidad de los proyectos. Este enfoque se basa en cuatro principios fundamentales de las cuales son:

1. Abstracción
2. Encapsulamiento
3. Herencia
4. Polimorfismo

¿Qué es la Abstracción?

Permite trabajar con conceptos generales y simples sin preocuparse por los detalles internos, en términos más simples, es cuando separamos los datos de un objeto para luego generar un molde

¿Qué es el Encapsulamiento?

Es la agrupación de datos y métodos que operan sobre las clases, en sí protegen el estado interno del objeto, es decir, podemos reutilizar cuando deseamos ciertas propiedades sin comprometer lo que queremos replicar

¿Qué es la Herencia?

Permite crear nuevas clases basadas en cosas ya existentes, esto realmente promueve la reutilización y las jerarquías, el ejemplo más claro es el árbol genealógico, donde existen los padres y los hijos, cada uno tiene características en común pero también cosas únicas

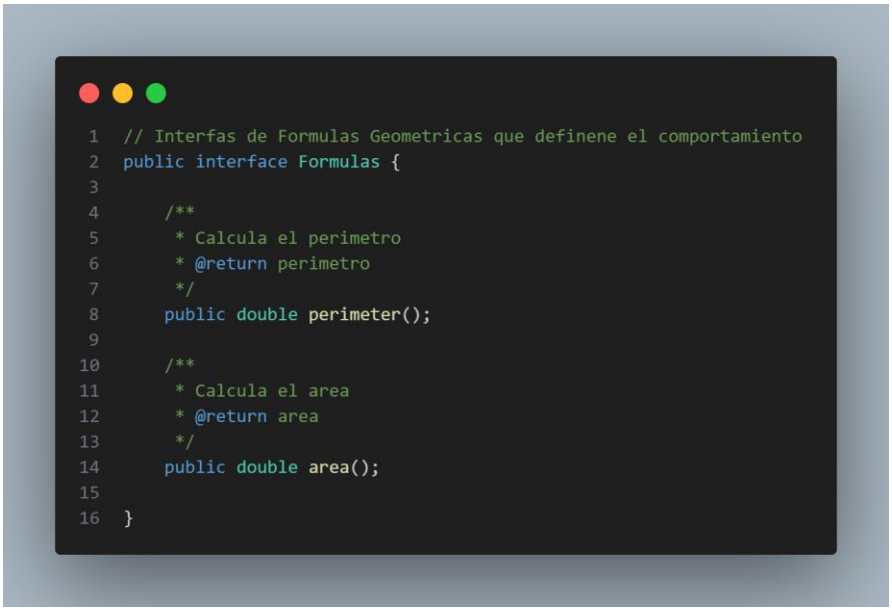
¿Qué es el Polimorfismo?

Si derivamos la palabra, poli proviene de muchos y morfismo de formas, se utiliza para crear múltiples formas ya sea un objeto, método o clase. En lo particular, es como tener variantes de cosas específicas, como lo sería de hacer funcionar algo de diversas maneras.

Con estos conceptos claros, ahora podemos tener un claro ejemplo de representación aplicando todos los pilares de la programación orientada a objetos.

Clase Formulas

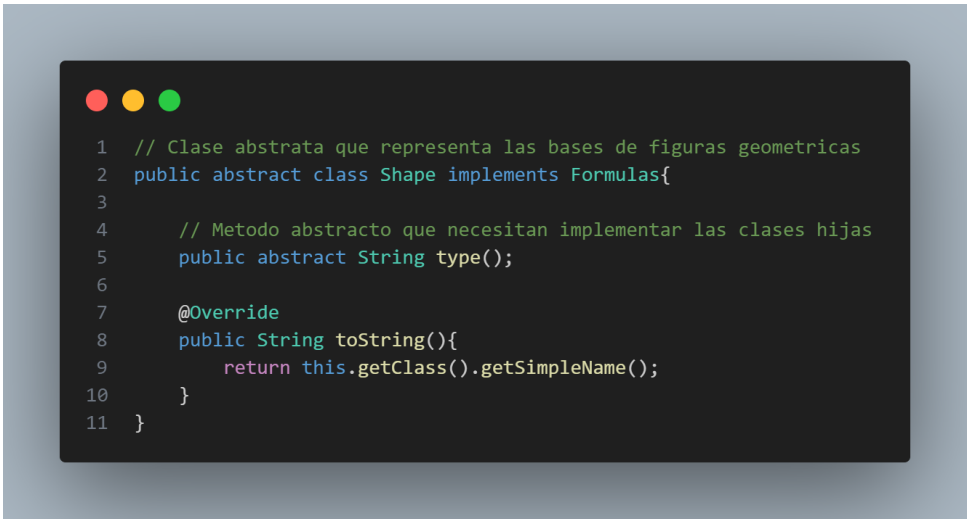
En esta clase se definen métodos para calcular el área y el perímetro, al ser una clase de tipo interfaz definirán el molde necesario paras las fórmulas de las figuras geométricas.



```java
// Interfas de Formulas Geometricas que definene el comportamiento
public interface Formulas {

    /**
     * Calcula el perimetro
     * @return perimetro
     */
    public double perimeter();

    /**
     * Calcula el area
     * @return area
     */
    public double area();

}
```

Clase Shape

En esta clase abstracta definimos los métodos comunes a todas las figuras geométricas y al implementar con la interfaz, hay una relación directa para que futuras clases puedan hacer uso de las formulas



```java
// Clase abstrata que representa las bases de figuras geometricas
public abstract class Shape implements Formulas{

    // Metodo abstracto que necesitan implementar las clases hijas
    public abstract String type();

    @Override
    public String toString(){
        return this.getClass().getSimpleName();
    }
}
```

Clase Square

En esta clase hacemos la creación de un objeto, en este caso de una figura cuadrada, usaremos la herencia de *Shape* y reutilizaremos su código de la clase. Al igual que implementaremos las formulas de la interfaz *Formulas*. Tambien usaremos la encapsulación para continuar con todos los pilares.

```java
// Clase hija que extiende a la clase Shape y al mismo tiempo implementa la interfaz
public class Square extends Shape{

    private double side;

    public Square(double side){
        this.side = side;
    }

    // Implementacion de metodos de la interfaz Formulas
    @Override
    public double perimeter() {
        return side * 4;
    }

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

    // Implementacion de metodos de la clase abstracta Shape
    @Override
    public String toString() {
        return "The shape is: " + super.toString();
    }

    @Override
    public String type() {
        return "Line";
    }
}
```

Clase Main

En esta clase observamos todo el funcionamiento de todo el código y observaremos el polimorfismo y toda la instancia para la creación de figuras

```java
// Clase main para probar el programa
public class App {
    public static void main(String[] args) throws Exception {
        // Declaracación e inicialización de variables de referencias guardas en un arreglo
        Shape[] shape = new Shape[] {new Square(5),
                                     new Triangle(4, 6, 8),
                                     new Circule(10)
        };
        // Invocación de método para inicizalizar con el arreglo
        show(shape);
    }
    /**
     * Metodo para mostrar todo la información de las figuras
     * @param s Recorre el arreglo de formas geometricas
     */
    public static void show(Shape[] s){
        // Polimorfismo
        for (Shape shape : s) {
            System.out.println(shape.toString() + ", your perimeter is: "
            + shape.perimeter() + " and your area is: " + shape.area() + "type: " + shape.type());
        }
    }
}
```