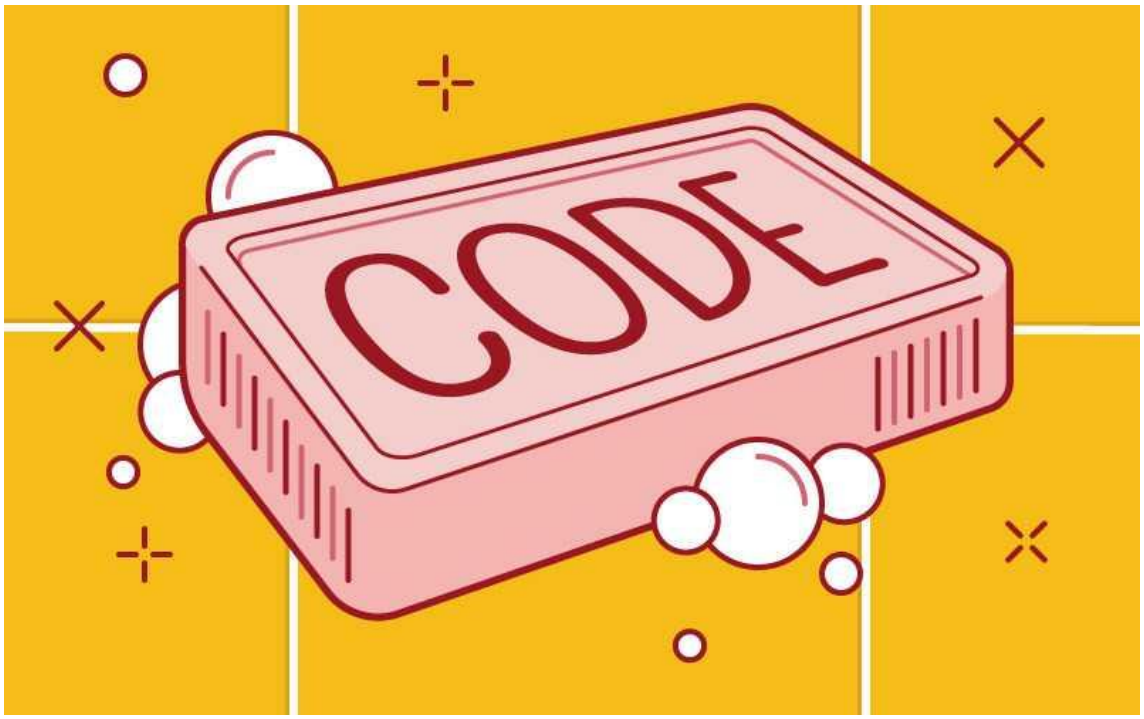
Clean Code



Anthony Castillo https://github.com/ajcastillo1/Clean-Code-Bloques-4-5-6-y-7.git

Bloque 4: Objetos y estructuras de datos

El siguiente ejemplo muestra dos clases: Figura y Triangulo.

```java
public class Figura {

    private String nombre;
    private String color;

    public Figura(String nombre) {
        this.nombre = nombre;
    }

    public Figura(String nombre, String color) {
        this.nombre = nombre;
        this.color = color;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
}
```

```java
public class Triangulo extends Figura implements OperacionesFigura{

    private double base;
    private double altura;

    public Triangulo(String nombre, String color, double base, double altura) {
        super(nombre, color);
        this.base = base;
        this.altura = altura;
    }

    @Override

    public double calcularArea() {
        return (base * altura) / 2;
    }

    public double getBase() {
        return base;
    }

    public void setBase(double base) {
        this.base = base;
    }

    public double getAltura() {
        return altura;
    }

    public void setAltura(double altura) {
        this.altura = altura;
    }

    @Override
    public Figura figura() {
        return new Figura(getNombre(), getColor());
```

Las dos clases son objetos, ya que están diseñadas de tal manera que ocultan sus datos con abstracciones y muestran funciones que operan con los datos ocultos.

Por ejemplo, en la clase “Triangulo” hereda los atributos de la clase figura y tiene los propios suyos, los cuales son privados. Estos datos se muestran a través de las funciones como puede ser “calcularArea()”.

Ley de Demeter

Triangulo:

CalcularArea(): Utiliza sus propias propiedades, no interactúa con otros objetos.

Figura(): Cumple la ley de demeter ya que se crea un nuevo objeto de tipo figura.

Figura: No tiene métodos que acceda a los atributos de otras clases.

Bloque 5: Manejo de errores

```java
public class CleanCode2Anthony {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String nombre;
        String color;
        double base;
        double altura;

        //Pedir datos y manejo de errorestr
        try {
        System.out.println("Introduce el nombre de la figura triangular");
        nombre = scan.nextLine();
        System.out.println("Introduce el color de la figura triangular");
        color = scan.nextLine();
        System.out.println("Introduce la base de la figura triangular");
        base = scan.nextDouble();
        System.out.println("Introduce la altura de la figura triangular");
        altura = scan.nextDouble();
        OperacionesFigura triangulo1 = new Triangulo(nombre, color, base, altura);
        OperacionesFigura triangulo2 = new Triangulo("Triangulo", "rojo", 10.0, 2.0);

        List<OperacionesFigura> listaTriangulos = new ArrayList<>();
        listaTriangulos.add(triangulo1);
        listaTriangulos.add(triangulo2);

        System.out.println("--------------------------------");
        calcularArea(listaTriangulos);


        }catch (InputMismatchException ex){
            System.out.println("El valor introducido no es correcto");
        }
```

En el siguiente ejemplo, he utilizado un try-catch para que el programa no colapse a la hora de introducir algún dato erróneo. Este programa en concreto recibe los datos de la instancia triangulo1 y cada dato que introduce el usuario, este lo va guardando en una variable. La función principal del try-catch es prevenir el colapso del programa, en este caso el programa podría colapsar por un valor erróneo en las variables “base” y “altura”. En caso de que colapse saldrá el mensaje “El valor introducido no es correcto” y no un mensaje de error.

Bloque 6: Test Unitarios

```java
@Test
    public void testCalcularAreaTriangulo() {
        Triangulo triangulo = new Triangulo("Triángulo", "Rojo", 5.0, 8.0);
        assertEquals(20.0, triangulo.calcularArea());
    }

    @Test
    public void testGettersSettersFigura() {
        Figura figura = new Figura("Cuadrado", "Azul");

        assertEquals("Cuadrado", figura.getNombre());
        assertEquals("Azul", figura.getColor());

        figura.setNombre("Rectángulo");
        figura.setColor("Verde");

        assertEquals("Rectángulo", figura.getNombre());
        assertEquals("Verde", figura.getColor());
    }

    @Test
    public void testGettersSettersTriangulo() {
        Triangulo triangulo = new Triangulo("Triángulo", "Rojo", 5.0, 8.0);

        assertEquals("Triángulo", triangulo.getNombre());
        assertEquals("Rojo", triangulo.getColor());
        assertEquals(5.0, triangulo.getBase());
        assertEquals(8.0, triangulo.getAltura());

        triangulo.setBase(6.0);
        triangulo.setAltura(10.0);

        assertEquals(6.0, triangulo.getBase());
        assertEquals(10.0, triangulo.getAltura());
```

He creado varios test unitarios para comprobar la funcionabilidad del sistema. En este caso he creado 3:

- TestCalcularAreaTriangulo: Este test instancia el objeto triangulo y pone a prueba el método calcularArea(), el cual debe dar el resultado que el assert espera, en este caso 20.
- TestGettersSettersFigura: Este test instancia el objeto figura con los datos Cuadrado y Azul, lo que pongo aprueba aquí es el correcto funcionamiento de los getters y los setters. Esperando por parte de los getters Cuadrado y azul, y luego haciendo uso de los setter para modificar los datos a Rectángulo y Verde. Por ultimo se hace un assertEquals con los getters para comprobar que todo se ha realizado correctamente.
- TestGettersSettersTriangulo: Prueba similar a Figura pero añadiendo los atributos de la clase triangulo.

Bloque 7: Clases

Las clases Figura y triangulo:

- Organización de clases: Se han creado varias variables privadas y encapsuladas de tal manera que ningún objeto de otras clases puedan acceder a sus datos. Para ello están las funciones publicas. (calcularArea(), getNombre, etc…)
- Clases pequeñas: Las clases están definidas para tener un fin, figura representa una figura geométrica, mientras que triangulo es un triangulo el cual hereda de figura.
- Principio de responsabilidad única:

En general las distintas clases siguen un esquema el cual esta adaptado para conseguir posibles mejoras en un futuro.

Bloque IV: Objetos y estructura de datos