

Nivel 2

Encapsulamiento

Es uno de los principios fundamentales de la Programación Orientada a Objetos que nos permite restringir el acceso a métodos y variables de las clases, evitando así que se puedan modificar directamente.[1]

Cuando encapsulamos una clase definimos que su nivel de acceso (*ver anexo 1*) sea privado por lo que la única manera en la que podemos obtener su valor o cambiarlo es a través de métodos, es aquí donde aparecen dos métodos fundamentales en la programación llamados:

- **Getters:** Obtienen y retornan el valor de los atributos.
- **Setters:** Modifican el valor almacenado en los atributos.

Estos métodos se aplican por cada uno de los atributos que tenga la clase y es necesario que sean públicos para poder invocarlos desde el *main*.

Es necesario aplicar el principio de encapsulamiento si queremos que nuestro proyecto sea más mantenible, estable y seguro.[3]

Ejemplo: [3]

Clase Gato:

Definimos una clase gato que solo cuenta con dos atributos(nombre y edad) junto con su constructor y los Getters y Setters correspondientes a cada uno de los atributos de la clase.

```

public class Gato {

    private String nombre;
    private int edad;

    public Gato(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    //-----Getters y Setters-----
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
}

```

Clase Main:

```

public class Main {

    public static void main(String[] args) {

        Gato michi = new Gato("Anacleto", 1);
        michi.setEdad(2);
        System.out.println(michi.getEdad());
    }
}

```

Cambiamos la edad del gato con el método set, ahora tiene 2 años

Se instancia un objeto de tipo gato con nombre Anacleto y edad de 1 año

Con el método get obtenemos la edad actual del gato, que es este caso es 2

Método toString

Ya que vimos el tema de encapsulamiento podemos hablar de otro método importante en la programación, el toString, este tiene el objetivo de retornar toda la información del objeto , en otras palabras, los valores almacenados en sus atributos.

Siguiendo con el ejemplo anterior de la clase Gato tenemos lo siguiente:

Clase Gato

Por defecto un objeto tiene predefinido el método toString, por tanto debemos sobrescribir el método para que nos devuelva los atributos específicos de la clase.

```
public class Gato {  
  
    private String nombre;  
    private int edad;  
  
    public Gato(String nombre, int edad) {  
        this.nombre = nombre;  
        this.edad = edad;  
    }  
    //-----Getters y Setters-----  
    public String getNombre() {  
        return nombre;  
    }  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
  
    public int getEdad() {  
        return edad;  
    }  
    public void setEdad(int edad) {  
        this.edad = edad;  
    }  
    //----toString----  
    @Override  
    public String toString() {  
        return "Gato \nNombre: "+this.getNombre()+" \nEdad: "+this.getEdad();  
    }  
}
```

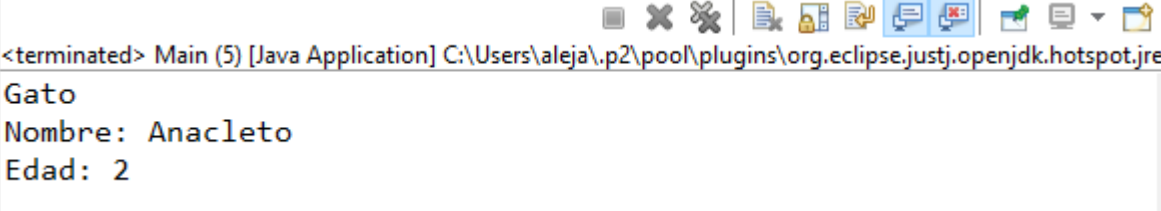
Para sobrescribir un método se usa el @Override

Clase Main

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Gato michi = new Gato("Anacleto", 1);  
  
        michi.setEdad(2);  
        System.out.println(michi.toString());  
    }  
}
```

Llamado del método
toString

Resultado en consola



The screenshot shows a Java IDE console window with a toolbar at the top containing icons for running, debugging, and other actions. The console output is as follows:

```
<terminated> Main (5) [Java Application] C:\Users\aleja\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre  
Gato  
Nombre: Anacleto  
Edad: 2
```

Anexos

Anexo 1. Niveles de acceso en Java[2]

MODIFICADOR	CLASE	PACKAGE	SUBCLASE	TODOS
public	Sí	Sí	Sí	Sí
protected	Sí	Sí	Sí	No
No especificado	Sí	Sí	No	No
private	Sí	No	No	No

Referencias

1. Image.pdf. (2021). Image.pdf. Google Docs.
https://drive.google.com/file/d/1ChoJPc0Yhpq_QZhLSwjntqhum46WJ07Z/view
2. Rodríguez, A. (2021). *public, private y protected Java.Tipos de modificadores de acceso. Visibilidad en clases, subclases. (CU00693B)*. Aprenderaprogramar.com.
https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=665:public-private-y-protected-javatipos-de-modificadores-de-acceso-visibilidad-en-clases-subclases-cu00693b&catid=68&Itemid=188
3. ThemeGrill. (2018, March 23). Java Básico Encapsulamiento - Blog de Juanla. Blog de Juanla.
<https://juanjavierrg.com/java-basico-encapsulamiento/>