Material descargable

JAVA Programación Orientada a Objetos

egg

Programación Orientada a Objetos

Repasemos: la programación orientada a objetos (POO) es un paradigma de programación que se basa en el uso de "objetos" y sus interacciones para diseñar aplicaciones y programas de software. Los objetos son instancias de "clases", las cuales pueden incluir variables de instancia, métodos (funciones) y otros datos necesarios para su funcionamiento.

Encapsulamiento y Ocultamiento

Encapsulación y ocultamiento de información son dos conceptos fundamentales de la programación orientada a objetos que ayudan a mantener la integridad de los datos y a hacer el código más manejable.

Cuando encapsulamos, estamos agrupando los datos (atributos) y las operaciones (métodos) que actúan sobre esos datos en una sola unidad, la clase. La ocultación de información ocurre cuando limitamos el acceso a los detalles internos de esa clase, ocultando sus atributos y permitiendo la interacción con ellos únicamente a través de los métodos que hemos definido. Esto se realiza a menudo utilizando modificadores de acceso, como `private`, y proporcionando métodos públicos, como los métodos *getters* y *setters*, para acceder y modificar los datos.

Beneficios de Encapsulación y Ocultamiento:

- **Control de Acceso:** Al utilizar métodos getter y setter en lugar de permitir el acceso directo a los atributos, tienes un control completo sobre cómo y cuándo se accede o modifica la información de un objeto. Por ejemplo, puedes realizar una validación en un método setter para asegurarte de que nadie pueda establecer un valor inválido para un atributo.

- **Flexibilidad y mantenimiento:** Al ocultar los detalles internos de cómo se implementa una clase, puedes cambiar la implementación en cualquier momento sin afectar a las otras partes del código que utilizan esa clase. Si otras partes del código usan directamente los atributos de la clase en lugar de usar los métodos getter y setter, cualquier cambio en esos atributos requeriría cambiar también todas esas partes del código.
- **Seguridad de los datos**: Los atributos de un objeto pueden ser sensibles o críticos para el estado coherente del objeto. Permitir el acceso directo a estos atributos puede poner en riesgo la seguridad e integridad de los datos, ya que pueden ser alterados de maneras no deseadas o inesperadas.

Modificadores de Acceso:

Los modificadores de acceso en Java determinan la visibilidad de las clases, los métodos y los atributos. Hay cuatro niveles de acceso:

- **public**: La clase, el método o el atributo es accesible desde cualquier lugar.
- **protected**: La clase, el método o el atributo es accesible dentro del mismo paquete y también por subclases de cualquier paquete.
- **default**: La clase, el método o el atributo es solo accesible dentro del mismo paquete. No es necesario declararlo, si no se especifica un modificador de acceso es el comportamiento por defecto.
- **private**: La clase, el método o el atributo es accesible solamente dentro de la clase.

Métodos Getters y Setters

Los métodos *getters* y *setters* son funciones públicas que se utilizan para acceder y modificar los atributos privados de una clase. Este enfoque garantiza que los datos se manipulen de manera controlada y segura.

- **Getters:** Son métodos que permiten recuperar el valor de un atributo privado. Siempre tienen un tipo de retorno que coincide con el tipo del atributo que devuelven. Su nombre suele comenzar con la palabra "get" seguida del nombre del atributo con la primera letra en mayúscula.
- **Setters:** Son métodos que permiten modificar el valor de un atributo privado. Siempre tienen un tipo `void` porque no devuelven ningún valor. Su nombre suele comenzar con la palabra "set" seguida del nombre del atributo con la primera letra en mayúscula. Los *setters* también pueden incluir validaciones para asegurar que los datos asignados sean correctos.

Ejemplo de Getters y Setters en Java:

```
public class Persona {
    private String nombre;
    private int edad;

    // Getter para el atributo nombre
    public String getNombre() {
        return nombre;
    }

    // Setter para el atributo nombre
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    // Getter para el atributo edad
    public int getEdad() {
        return edad;
    }

    // Setter para el atributo edad con validación
    public void setEdad(int edad) {
        if (edad > 0) {
            this.edad = edad;
        } else {
                System.out.println("Error: La edad debe ser un
valor positivo.");
        }
    }
}
```

En este ejemplo:

- Los atributos `nombre` y `edad` están declarados como privados.
- Los métodos `getNombre()` y `getEdad()` devuelven los valores de los atributos correspondientes, por lo que tienen un tipo de retorno (`String` e `int`, respectivamente).
- Los métodos `setNombre()` y `setEdad()` modifican los valores de los atributos y son del tipo `void` porque no devuelven ningún valor.

A continuación, se muestra un ejemplo que demuestra el uso de modificadores de acceso en atributos y métodos en Java:

```java
// Clase principal
public class EjemploModificadoresAcceso {

    public static void main(String[] args) {
        // Crear un objeto de la clase Ejemplo
        Ejemplo objeto = new Ejemplo();

        // Usar los métodos para interactuar con el objeto
        objeto.setNumero(10); // Modifcar dato
        System.out.println("El número es: " + objeto.getNumero()); // Obtener dato

        // Intentar asignar un valor inválido
        objeto.setNumero(-5); // Esto generará un mensaje de error
    }
}

public class Ejemplo {
    // Atributo privado
    private int numero;

    // Setter para el atributo con validación
    public void setNumero(int n) {
        if (n > 0) {
            this.numero = n;
        } else {
            System.out.println("Error: El número debe ser positivo.");
        }
    }

    // Getter para el atributo
    public int getNumero() {
        return numero;
    }
}
```

Análisis del Ejemplo

- Se define una clase llamada `Ejemplo`.
- El atributo `numero` se declara como privado utilizando el modificador `private`.
- Los métodos `setNumero()` y `getNumero()` son públicos y se utilizan para establecer y obtener el valor del atributo `numero`, respectivamente.
- Se asegura que la validación del valor asignado al atributo `numero` sea manejada dentro del método *setter*, eliminando la necesidad de validaciones adicionales en el programa principal.

- La clase principal interactúa únicamente con los métodos *getters* y *setters*, lo que garantiza que la lógica de validación permanezca encapsulada dentro de la clase.

Este enfoque hace que el código sea más claro, modular y fácil de mantener.

Explora la documentación oficial y fortalece tu conocimiento

Te invitamos a profundizar en el encapsulamiento y ocultamiento de información explorando la documentación oficial de Java. Aquí tienes dos recursos clave:

- **Modificadores de acceso**: Aprende a controlar la visibilidad de atributos y métodos.
- **Clases y objetos:** Descubre cómo estructurar datos y comportamientos en Java.

La documentación oficial es confiable, precisa y esencial para afianzar tu conocimiento. ¡Anímate a consultarla y lleva tus habilidades al siguiente nivel!