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.

Ahora profundicemos en otras de sus características:

Encapsulamiento y Ocultación

Encapsulación y ocultación 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 tal caso. 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 Ocultación:

- **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.

Aquí tienes un ejemplo simple que demuestra cómo se utilizan los 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();
        // Acceder a los atributos y métodos de la clase Ejemplo
        objeto.mostrarMensaje();
        objeto.setNumero(10);
        int numero = objeto.getNumero();
        System.out.println("El número es: " + numero);
    }
}
```

```java
// Clase Ejemplo
class Ejemplo {
    // Atributo privado
    private int numero;

    // Método público que establece el valor del atributo privado
    public void setNumero(int n) {
        // Validar que el número sea positivo antes de asignarlo al atributo
        if (n > 0) {
            numero = n;
        } else {
            System.out.println("Error: El número debe ser positivo.");
        }
    }

    // Método público que devuelve el valor del atributo privado
    public int getNumero() {
        return numero;
    }

    // Método público que muestra un mensaje
    public void mostrarMensaje() {
        System.out.println("¡Hola desde la clase Ejemplo!");
    }
}
```

En este ejemplo:

- Se define una clase llamada Ejemplo.
- El atributo numero se declara como privado usando el modificador private.
- Los métodos setNumero() y getNumero() son públicos y se utilizan para establecer y obtener el valor del atributo numero, respectivamente.
- El método mostrarMensaje() es público y simplemente muestra un mensaje en la consola.
- En la clase EjemploModificadoresAcceso, se crea un objeto de la clase Ejemplo y se utilizan los métodos públicos para acceder al atributo y mostrar mensajes.

El uso de los modificadores de acceso asegura que el atributo numero solo pueda ser modificado a través del método setNumero() y que su valor solo pueda ser obtenido a través del método getNumero(). Esto ayuda a encapsular los datos y garantiza que se mantenga la coherencia de los objetos.