Herencia y Polimorfismo en Java

Pedro Gordillo

March 2, 2025

Índice

Herencia

¿Qué es la Herencia?

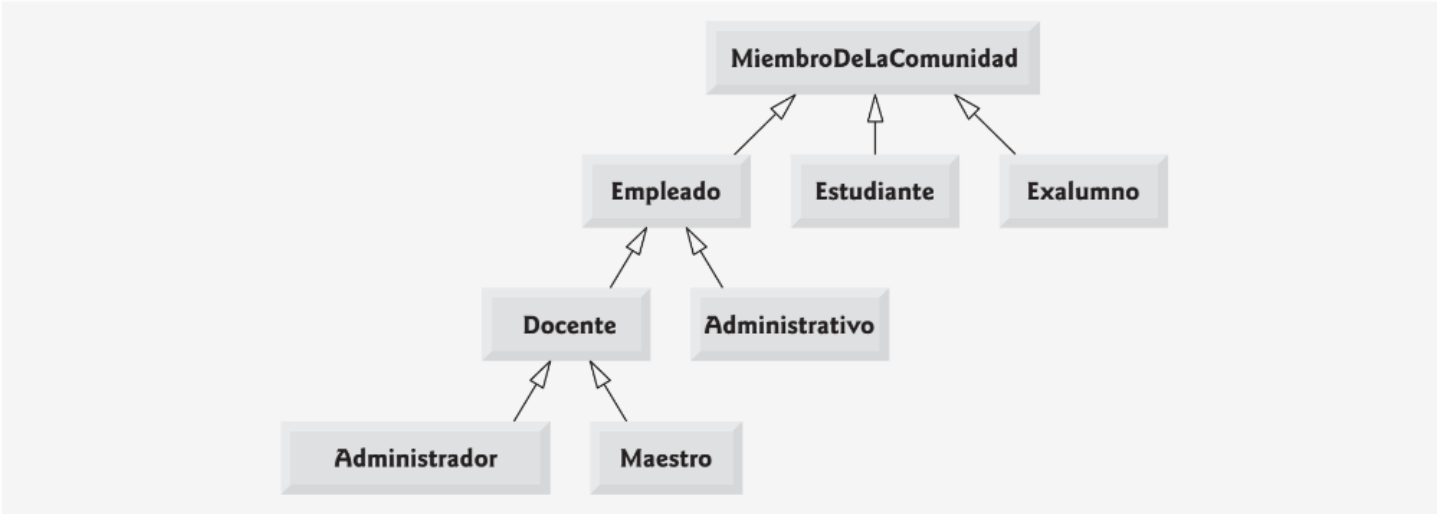
La herencia es un mecanismo fundamental en la programación orientada a objetos (POO). Permite que una clase derive de otra, heredando sus propiedades y métodos.

Herencia

La clase existente se denomina superclase. La clase derivada de la superclase se denomina la subclase. También se conoce a la superclase como clase padre y una subclase se conoce como clase hija, clase extendida o clase derivada.

Herencia

En este ejemplo, vemos cómo se representa la herencia



Sintaxis de Herencia

```
class Superclass {
    // atributos y métodos
}

class Subclass extends Superclass {
    // atributos y métodos adicionales
}
```

Uso de `super` en Java

En Java, el keyword `super` se utiliza para hacer referencia a la clase **padre** (superclase) desde una clase **hija** (subclase). Se usa principalmente para dos propósitos:

1. **Llamar al constructor de la clase padre**.
2. **Acceder a los métodos y variables de la clase padre**.

1. Llamar al constructor de la clase padre:
Cuando se crea una clase hija, se usa `super()` para invocar el constructor de la clase padre. Si la clase padre tiene un constructor sin parámetros, `super()` es opcional, ya que se llama automáticamente si no se especifica otro constructor.

2. Acceder a métodos o variables de la clase padre:
Si una clase hija quiere acceder a un método o variable que ha sido sobrescrito en la clase hija, se puede usar `super` para llamar al método o variable de la clase padre.

Uso de super en Java

A continuación se muestra un ejemplo de cómo se utiliza super en Java:

```java
// Clase Padre (Superclase)
class Animal {
    // Atributo de la clase padre
    String nombre;

    // Constructor de la clase padre
    public Animal(String nombre) {
        this.nombre = nombre;
    }

    // Método de la clase padre
    public void hacerSonido() {
        System.out.println("Haciendo sonido...");
    }
}
```

Uso de super en Java

```java
// Clase Hija (Subclase)
class Perro extends Animal {

    // Constructor de la clase hija
    public Perro(String nombre) {
        // Llamada al constructor de la clase padre (Animal)
        super(nombre);
    }

    // Método de la clase hija
    public void hacerSonido() {
        // Llamada al método de la clase padre
        // Llama al método 'hacerSonido()' de la clase Animal
        super.hacerSonido();
        System.out.println("Guau!");
    }

    // Método para mostrar el nombre del perro
    public void mostrarNombre() {
     // Accede al atributo 'nombre' de la clase Animal
        System.out.println("El nombre del perro es: " + super.nombre);
    }
}
```

Ejemplo de ejecución

```java
public class Main {
    public static void main(String[] args) {
        // Crear un objeto de la clase Perro
        Perro perro = new Perro("Firulais");

        // Llamar al método de la clase hija
        //(que sobrescribe al de la clase padre)
        // Llama a hacerSonido() de Perro, pero primero invoca a la clase padre
        perro.hacerSonido();

        // Llamar al método que muestra el nombre del perro
        perro.mostrarNombre();
    }
}
```

Modificadores de acceso

El lenguaje Java proporciona diversos modificadores que se pueden utilizar para modificar aspectos del proceso de herencia.
public: se puede acceder desde el exterior de la definición de la clase. A una clase pública se puede acceder fuera del paquete en el que está declarada.
private: se puede acceder sólo dentro de la definición de la clase en que aparece.
protected: sólo se puede acceder dentro de la definición de la clase en la que aparece, dentro de otras clases del mismo paquete o dentro de la definición de subclases (Herencia).

Polimorfismo

¿Qué es el Polimorfismo?

El polimorfismo permite que objetos de diferentes clases se traten de manera uniforme. Es un concepto clave que nos permite usar un solo nombre de método para diferentes tipos de objetos.
En Java, el polimorfismo se logra principalmente de dos maneras:

- Polimorfismo de sobrecarga
- Polimorfismo de sobrescritura

Polimorfismo de sobrecarga

El polimorfismo de sobrecarga se da cuando el mismo nombre de método se puede utilizar con parámetros diferentes y se permite que áparentemente el mismo método sea declarado un cierto número de veces dentro de la misma clase.

Firma

En Java, es posible aplicar sobrecarga de métodos (method overloading) debido a la firma del método, que está compuesta por:

1. **Nombre del método: El nombre con el que se invoca el método.**
2. **Número de parámetros: La cantidad de parámetros que el método recibe.**
3. **Tipo de parámetros: El tipo de datos de los parámetros que el método recibe.**
4. **Orden de los parámetros: En caso de que los tipos sean iguales, el orden de los parámetros también puede ser diferente.**

La sobrecarga de métodos no está relacionada con el tipo de valor de retorno.

```java
public class Sobrecarga {
    public int add(int a, int b) {
        return a + b;
    }

    public double add(double a, double b) {
        return a + b;
    }

    public static void main(String[] args) {
        Sobrecarga obj = new Sobrecarga();
        System.out.println("Suma de integers: " + obj.add(5, 10));
        System.out.println("Suma de doubles: " + obj.add(5.5, 10.5));
    }
}
```

Polimorfismo de Sobrescritura (Override)

El polimorfismo de sobrescritura se logra cuando una subclase proporciona una implementación específica para un método que ya está definido en su clase base.

```
class Animal {
    public void sonido() {
        System.out.println("El animal hace un sonido");
    }
}

class Perro extends Animal {
    @Override
    public void sonido() {
        System.out.println("Ladrido");
    }
}
```

Uso del Polimorfismo

Muestra cómo un objeto de tipo `Animal` puede referirse a un objeto de tipo `Perro` y usar el método sobrescrito.

```
public class Main {
    public static void main(String[] args) {
        Animal mascota = new Perro();
        mascota.sonido(); // Salida "Ladrido"
    }
}
```

Aunque `mascota` es de tipo `Animal`, se ejecuta el método sobrescrito de `Perro`.