Relaciones de Dependencia en Java

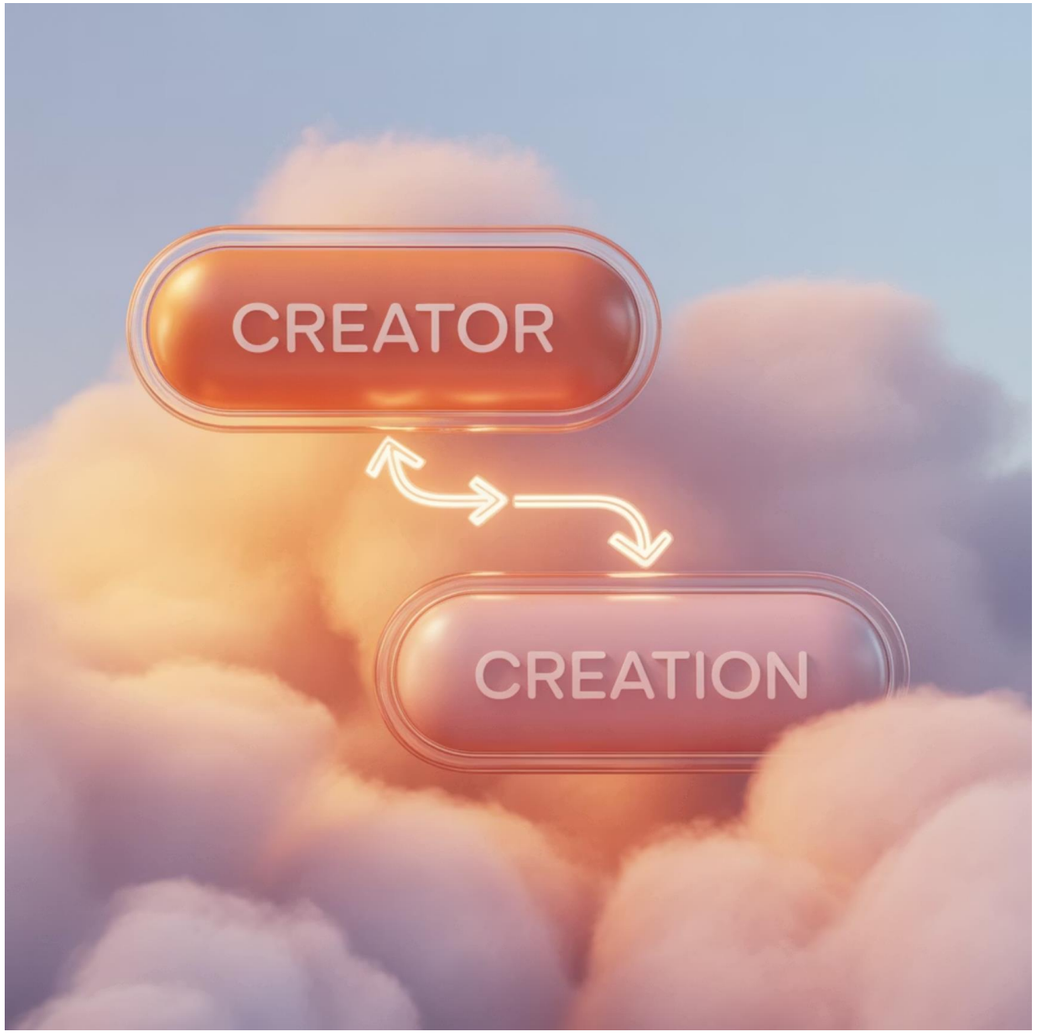
Una guía completa para programadores que buscan comprender las relaciones de dependencia en la Programación Orientada a Objetos con Java. Exploraremos los diferentes tipos de dependencias, cuándo utilizarlas y las mejores prácticas para implementarlas correctamente.

¿Qué es una Relación de Dependencia?

Una dependencia indica que una clase utiliza temporalmente a otra clase, creando un acoplamiento débil y circunstancial entre ambas. A diferencia de otras relaciones más estrechas como la herencia o la composición, una dependencia representa una necesidad momentánea.

Situaciones típicas de dependencia:

- Invocar un método de otra clase
- Crear un objeto temporalmente
- Recibir un objeto como parámetro



En UML, una dependencia se representa mediante una línea punteada con una flecha que apunta desde la clase dependiente hacia la clase de la que depende.

Tipos de Dependencia

Dependencia de Uso

Una clase utiliza otra momentáneamente por parámetro o dentro de un método, sin mantener una referencia permanente a ella.

- Menor acoplamiento
- Mayor flexibilidad
- Facilita las pruebas unitarias

Dependencia de Creación

Una clase crea una instancia de otra dentro de un método, asumiendo la responsabilidad de su ciclo de vida.

- Mayor acoplamiento
- Control sobre la instanciación
- Menor flexibilidad

Ambos tipos de dependencia son temporales, pero difieren significativamente en el nivel de acoplamiento que generan entre las clases.

Dependencia de Uso – Concepto

En una dependencia de uso, una clase utiliza temporalmente a otra sin conservar una referencia permanente a ella. Esta es la forma más débil de acoplamiento entre clases.

Ocurre típicamente en:

- Parámetros de métodos
- Variables locales dentro de métodos
- Llamadas a métodos estáticos

```
public void procesar(Documento doc) {
         // Usa el documento sin almacenarlo
         doc.imprimir();
}
```

Dependencia de Uso - Ejemplo

Clase Impresora

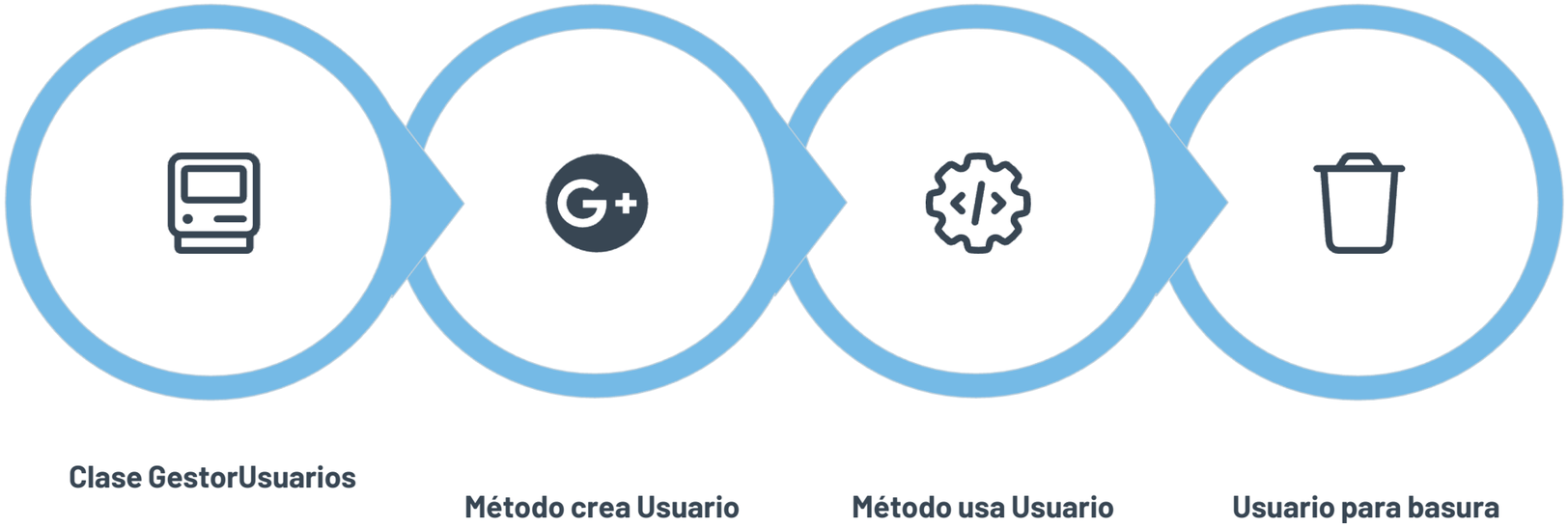
```
public class Impresora {
   public void imprimir(Documento doc) {
 // Dependencia de uso: utiliza doc
 // sin mantener una referencia
System.out.println(doc.getContenido());
          }
}
```

Clase Documento

```
public class Documento {
    private String contenido;
public Documento(String c) {
         contenido = c;
     }
public String getContenido() {
         return contenido;
     }
}
```

La clase Impresora depende temporalmente de Documento solo durante la ejecución del método imprimir().

Dependencia de Creación - Concepto



En una dependencia de creación, una clase asume la responsabilidad de crear instancias de otra clase internamente, sin recibirlas como parámetros.

Características principales:

- La clase dependiente conoce detalles de construcción de la otra clase
- Genera un acoplamiento más fuerte que la dependencia de uso
- Controla el ciclo de vida del objeto creado

```
// Ejemplo de dependencia de creaciónUsuario
nuevo = new Usuario("Juan");
```

Dependencia de Creación - Ejemplo

Clase GestorUsuarios

```
public class GestorUsuarios {
public void crearYMostrar(String nombre) {
        // Dependencia de creación:
// GestorUsuarios crea un Usuario
Usuario user = new Usuario(nombre);
// Utiliza el objeto creado
System.out.println(user.getNombre());
    }
}
```

Clase Usuario

```
public class Usuario {
    private String nombre;
        public Usuario(String n) {
         nombre = n;
     }
        public String getNombre() {
         return nombre;
     }
}
```

GestorUsuarios depende de Usuario porque crea instancias directamente, conociendo detalles de su constructor.

Comparación General

Dependencia de Uso

- Relación temporal durante la ejecución de un método
- Objeto recibido como parámetro
- Mayor independencia entre clases

Dependencia de Creación

- Relación temporal, pero con conocimiento de instanciación
- Objeto creado internamente con new
- Mayor acoplamiento entre clases
- Dificulta el cambio de implementación
- Complica las pruebas unitarias

¿Cuándo usar cada una?

Dependencia de Uso

Ideal cuando:

- Solo necesitas el objeto momentáneamente
- Necesitas aplicar inyección de dependencias
- Necesitas flexibilidad para pruebas unitarias
- Buscas bajo acoplamiento entre componentes

Dependencia de Creación

Apropiada cuando:

- Tu clase es responsable de la creación de objetos
- Implementas patrones de diseño como Factory
- Necesitas controlar el ciclo de vida del objeto
- El objeto creado es una implementación específica

La elección entre ambos tipos de dependencia afectará significativamente la mantenibilidad y flexibilidad de tu código.

Buenas Prácticas

Prioriza Dependencias de Uso

Siempre que sea posible, prefiere recibir objetos como parámetros en lugar de crearlos internamente.

Aplica Inyección de Dependencias

Utiliza frameworks como Spring para gestionar la creación e inyección de objetos dependientes.

Interfaces sobre Implementaciones

Depende de interfaces o clases abstractas en lugar de implementaciones concretas.