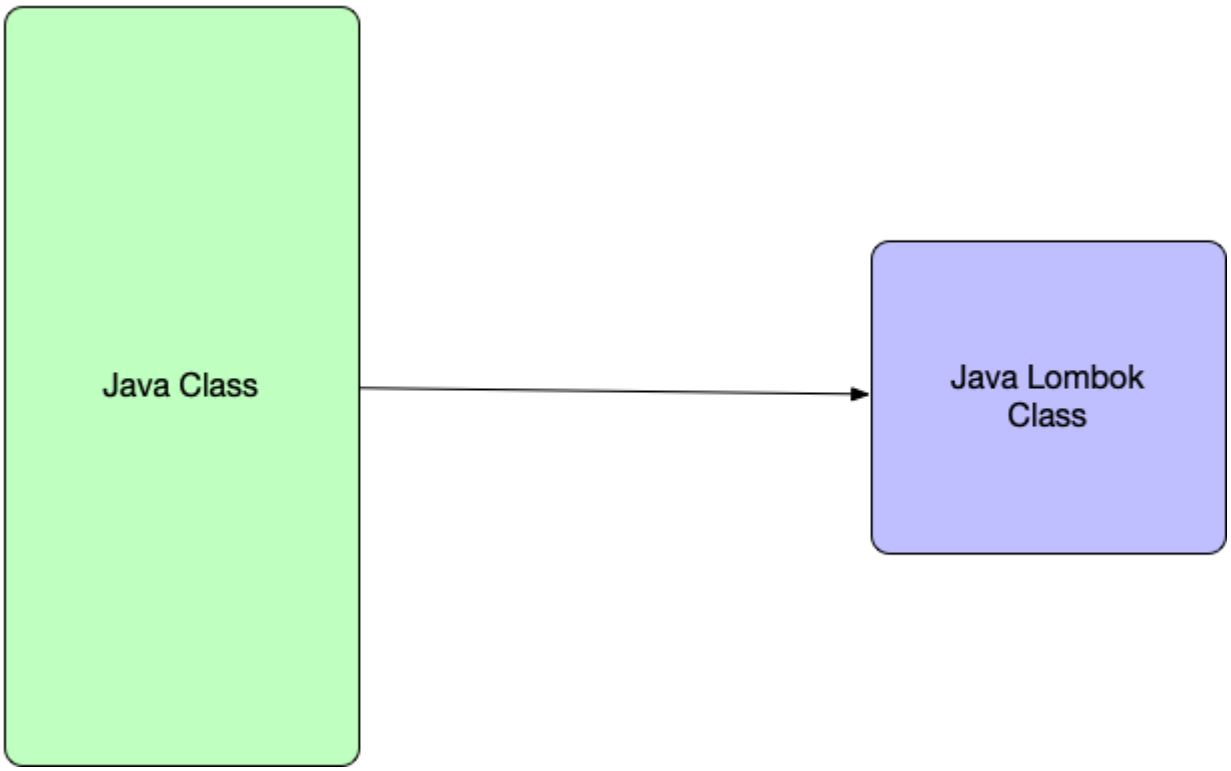
En Java, una parte importante del código que escribimos no siempre aporta lógica de negocio: constructores, getters, setters, `toString`, `equals` y `hashCode` suelen repetirse una y otra vez. Lombok aparece como una herramienta muy útil para reducir ese ruido, hacer las clases más limpias y, en contextos donde trabajamos con inteligencia artificial o generación de código, también puede ayudarnos a ahorrar tokens.



Lombok en Java: menos código, menos tokens

Lombok es una librería para Java que permite generar automáticamente código repetitivo mediante anotaciones. En lugar de escribir manualmente todos los métodos de acceso, constructores o implementaciones de `toString`, podemos usar anotaciones como `@Getter`, `@Setter`, `@NoArgsConstructor`, `@AllArgsConstructor` o `@Data`. El resultado es una clase mucho más corta, fácil de leer y mantener.

Cuando hablamos de ahorrar tokens, nos referimos especialmente a reducir la cantidad de texto necesario para representar una idea o una implementación. Esto es muy útil al

trabajar con modelos de IA, documentación técnica, prompts o generación automática de código. Una clase Java tradicional puede ocupar muchas líneas solo para declarar métodos básicos, mientras que con Lombok la misma intención se expresa en pocas líneas.

Además de ahorrar espacio, Lombok mejora la legibilidad del código cuando se usa correctamente. En lugar de distraernos con métodos repetitivos, podemos concentrarnos en lo que realmente importa: los atributos de la clase y su comportamiento específico. Eso sí, conviene conocer bien qué genera cada anotación para evitar confusiones, especialmente en proyectos grandes o con reglas estrictas de arquitectura.

Clase Persona: comparando Java con y sin Lombok

Veamos primero una clase `Persona` escrita en Java tradicional, sin usar Lombok. Esta implementación incluye atributos privados, constructor vacío, constructor con parámetros, getters, setters y el método `toString`. Aunque es código válido y claro, también es bastante extenso para una clase tan simple.

```
public class Persona {

    private String nombre;
    private String apellido;
    private int edad;

    public Persona() {
    }

    public Persona(String nombre, String apellido, int edad) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
```

```
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

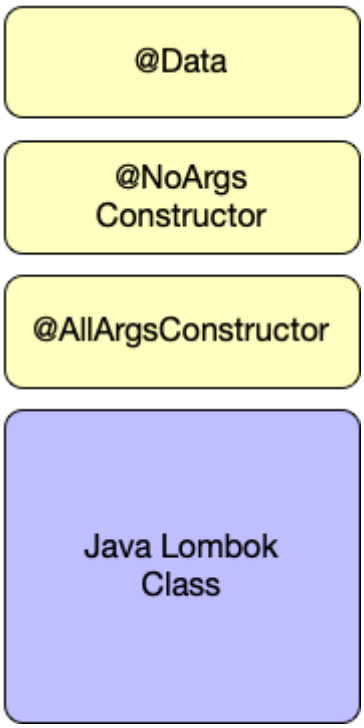
public int getEdad() {
    return edad;
}

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

@Override
public String toString() {
    return "Persona{" +
            "nombre='" + nombre + ''' +
            ", apellido='" + apellido + ''' +
```

```
                ", edad=" + edad +
                '}';
    }
}
```

Ahora veamos la misma clase usando Lombok. Con la anotación `@Data`, Lombok genera automáticamente los getters, setters, `toString`, `equals` y `hashCode`. Además, con `@NoArgsConstructor` y `@AllArgsConstructor`, también generamos el constructor vacío y el constructor con todos los atributos.




```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
```

```
@AllArgsConstructor
public class Persona {

    private String nombre;
    private String apellido;
    private int edad;
}
```

La diferencia es evidente: pasamos de una clase larga y repetitiva a una versión mucho más compacta. En términos de tokens, la implementación con Lombok necesita mucho menos texto para expresar la misma estructura. Esto no solo ahorra espacio, sino que también hace que el código sea más cómodo de leer, compartir y generar automáticamente.

Lombok es una herramienta muy práctica para reducir código repetitivo en Java y hacer que nuestras clases sean más expresivas con menos líneas. En el ejemplo de la clase `Persona`, se ve claramente cómo una implementación tradicional puede simplificarse usando unas pocas anotaciones. Si se usa con criterio, Lombok no solo ayuda a escribir menos código, sino también a ahorrar tokens y mantener proyectos más limpios.

Recordemos que la anotación @Data esta muy orientada a DTOS y hoy por hoy los Java Records suelen ser mejor opción.

- Java Lombok , clases y productividad
- Introducción a JSON Web Token y la seguridad
- Creando un JWT token con Node.js y Express
- Java @Data y preprocesado de JavaBeans
- Java toString overriding y Eclipse