Lombok is a Java library that helps to reduce boilerplate code. It achieves this by generating code at compile time using annotations. This can greatly simplify your Java code, especially in terms of getter, setter, and constructor methods.

Uses of Lombok

1. **Reduces Boilerplate Code**: Lombok can generate methods like getters, setters, equals, hashCode, toString, and constructors automatically.
2. **Improves Readability**: Your classes become cleaner and more readable by removing boilerplate code.
3. **Consistency**: Ensures a consistent style for common methods.
4. **Efficiency**: Saves time by reducing manual coding of repetitive tasks.

How to Install Lombok in an IDE

Step-by-Step Instructions for IntelliJ IDEA

1. **Add Lombok to Your Project**
   - **Using Maven**: Add the Lombok dependency to your `pom.xml` file.
     ```xml
     <dependency>
         <groupId>org.projectlombok</groupId>
         <artifactId>lombok</artifactId>
         <version>1.18.26</version> <!-- Check for the latest version -->
         <scope>provided</scope>
     </dependency>
     ```
   - **Using Gradle**: Add the Lombok dependency to your `build.gradle` file.
     ```groovy
     dependencies {
         compileOnly 'org.projectlombok:lombok:1.18.26' // Check for the latest version
         annotationProcessor 'org.projectlombok:lombok:1.18.26'
     }
     ```

2. **Install Lombok Plugin in IntelliJ IDEA**
   - Go to `File` > `Settings` (or `Preferences` on macOS).
   - Navigate to `Plugins` and search for "Lombok".
   - Install the Lombok plugin and restart IntelliJ IDEA.

3. **Enable Annotation Processing**
   - Go to `File` > `Settings` (or `Preferences` on macOS).
   - Navigate to `Build, Execution, Deployment` > `Compiler` > `Annotation Processors`.
   - Check "Enable annotation processing".

Step-by-Step Instructions for Eclipse

1. **Add Lombok to Your Project**
   - **Using Maven**: Add the Lombok dependency to your `pom.xml` file (as shown above).
   - **Using Gradle**: Add the Lombok dependency to your `build.gradle` file (as shown above).

2. **Install Lombok Plugin in Eclipse**
   - Download the Lombok jar from the [Lombok website](https://projectlombok.org/download).
   - Run the jar file using the command: `java -jar lombok.jar`.
   - The installer will prompt you to specify the location of your Eclipse installation. Follow the instructions to complete the installation.
   - Restart Eclipse.

3. **Enable Annotation Processing**
   - Go to `Window` > `Preferences`.

- Navigate to `Java` > `Compiler` > `Annotation Processing`.
- Check "Enable annotation processing".

How to Use Lombok Dependency

Here are some common annotations provided by Lombok and how to use them:

1. **@Getter and @Setter**: Generates getter and setter methods for fields.
   ```java
   import lombok.Getter;
   import lombok.Setter;

   public class User {
       @Getter @Setter
       private String name;

       @Getter @Setter
       private int age;
   }
   ```

2. **@ToString**: Generates a `toString` method.
   ```java
   import lombok.ToString;

   @ToString
   public class User {
       private String name;
       private int age;
   }
   ```

3. **@EqualsAndHashCode**: Generates `equals` and `hashCode` methods.
   ```java
   import lombok.EqualsAndHashCode;

   @EqualsAndHashCode
   public class User {
       private String name;
       private int age;
   }
   ```

4. **@NoArgsConstructor, @AllArgsConstructor, @RequiredArgsConstructor**: Generates constructors.
   ```java
   import lombok.NoArgsConstructor;
   import lombok.AllArgsConstructor;
   import lombok.RequiredArgsConstructor;

   @NoArgsConstructor
   @AllArgsConstructor
   @RequiredArgsConstructor
   public class User {
       private String name;
       private int age;
   }
   ```

5. **@Data**: Combines `@Getter`, `@Setter`, `@ToString`, `@EqualsAndHashCode`, and `@RequiredArgsConstructor`.
   ```java
   import lombok.Data;

   @Data
   public class User {
       private String name;
       private int age;
   }
   ```

These annotations can be added to your classes to automatically generate the necessary boilerplate code, significantly simplifying your Java development process.