**GUIDE TO LOMBOK LIBRARY IN JAVA**

**Author:** *By VAMOS Team*

**Date:** January 12, 2024

**I. Introduction**

- Project Lombok is an open-source Java library that helps us reduce the repeated code in many parts of an application with little alteration. This library automatically generates code of some common methods: **getter, setter, toString, equals, hashCode, and constructor.**

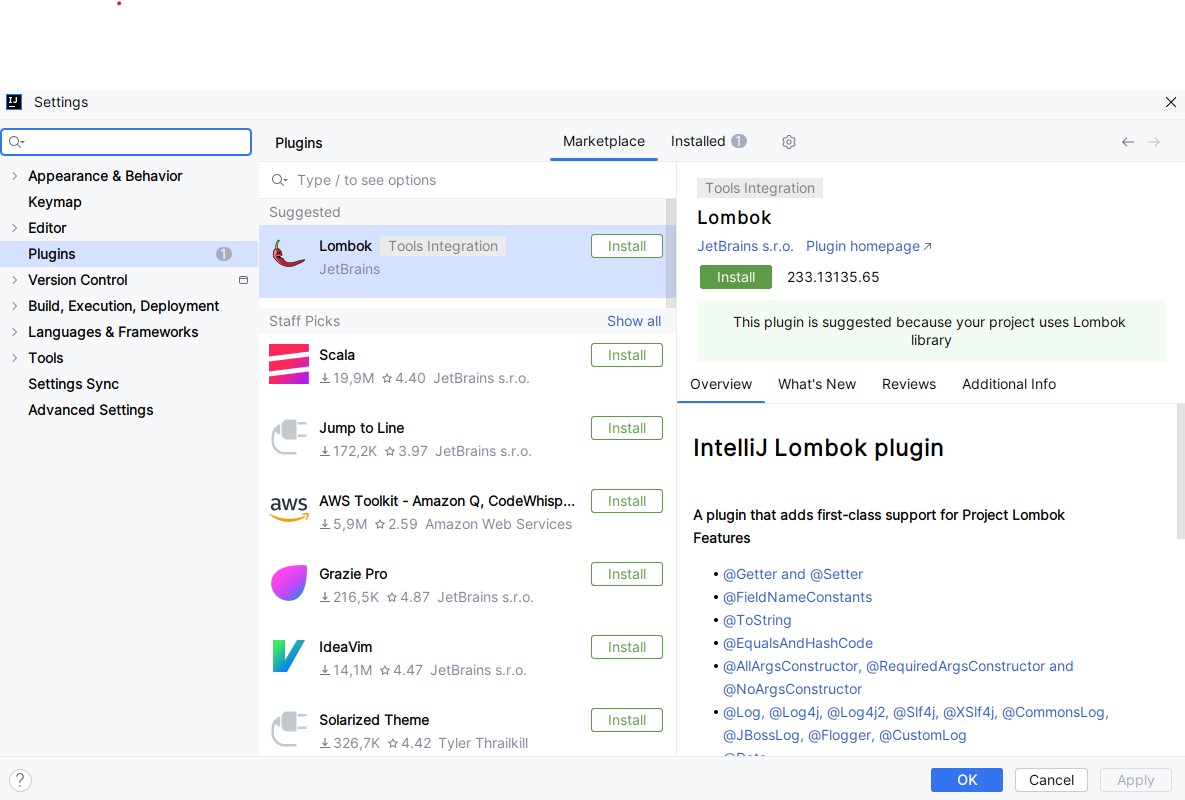
**II. Installation**

**1. Step 1**

- Open IntelliJ project, press “**Ctrl + Alt + S**” to open **Settings** windows then select “**Plugins**”.

- In the “**Marketplace**”, type “**Lombok**” to find the plugin.

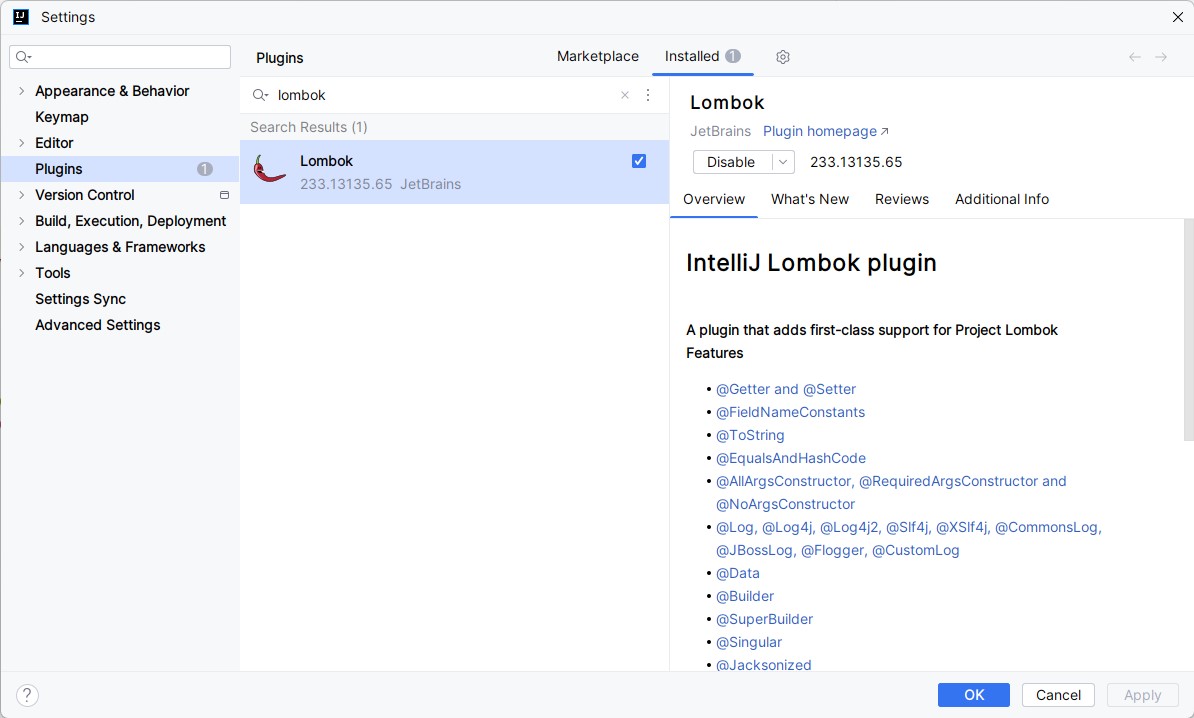
- Then press “**Install**”



**2. Step 2**

- Restart the IDE and reopen " Plugins” after installation.

- Next to the “**Marketplace**”, click “**Installed**” and search “**Lombok**”



**3. Step3**

- Open IntelliJ and add the following dependency into the ***pom.xml***file of the project:

<**dependency**>

<**groupId**>org.projectlombok</**groupId**>

<**artifactId**>lombok</**artifactId**>

<**version**>1.18.30</**version**>

<**scope**>provided</**scope**>

</**dependency**>

**Note:**

- If the project that uses Lombok library has a problem, you can restart IntelliJ, *Update Maven Project* or choose menu *Project -> Clean…* and select that project to fix.

**IV. Basic Lombok annotations**

**1. @Getter and @Setter**

- The *@Getter* and *@Setter* annotations generate a getter and setter for a field, respectively, or all. The getters generated correctly follow the convention for boolean properties, resulting in an *isFoo* getter method name instead of *getFoo* for any boolean field *foo*.

- Both the *@Getter* and *@Setter* annotations take an optional parameter to specify the access level for the generated method.

- Lombok annotated code:

@Getter

@Setter

**public class** Course {

**private** String **title**;

**private int number**;

}

- Equivalent Java source code:

**public class** Course {

**private** String **title**;

**private int number**;

**public** String getTitle() {

**return title**;

}

**public void** setTitle(String title) {

**this**.**title** = title;

}

**public int** getNumber() {

**return number**;

}

**public void** setNumber(**int** number) {

**this**.**number** = number;

}

}

**2. @EqualsAndHashCode**

* This class-level annotation will cause Lombok to generate both *equals* and *hashCode* methods, as the two are tied together intrinsically by the *hashCode* contract. By default, any field in the class that is not static or transient will be considered by both methods. Much like *@ToString*, the *exclude* parameter is provided to prevent the field from being included in the generated logic. - Also like *@ToString*, there is a *caliper* parameter for this annotation.
* Lombok annotated code:

@EqualsAndHashCode(exclude = {**"title"**})

**public class** Course {

**private** String **title**;

**private** String **university**;

}

- Equivalent Java source code:

@Override

**public boolean** equals(Object o) {

**if** (**this** == o) **return true**;

**if** (o == **null** || getClass() != o.getClass()) **return false**;

Course course = (Course) o;

**return** Objects.*equals*(**university**, course.**university**);

}

@Override

**public int** hashCode() {

**return** Objects.*hash*(**university**);

}

**3. @NoArgsConstructor, @RequiredArgsConstructor, @AllArgsConstructor**

- *@NoArgsConstructor* will generate a constructor with no parameters.

- *@RequiredArgsConstructor* generates a constructor with 1 parameter for each field that requires special handling.

- *@AllArgsConstructor* generates a constructor with 1 parameter for each field in your class. - Lombok annotated code:

@AllArgsConstructor

@NoArgsConstructor

**public class** Course {

**private** String **title**;

**private int number**;

}

- Equivalent Java source code:

**public class Course {**

**private String title;**

**private String university;**

**public Course() {}**

**public Course(String title, String university) {**

**this.title = title;**

**this.university = university;**

**}**

**}**

**5. @Builder:**

* *@Builder* annotation generates a builder pattern for your Java classes, simplifying the process of creating instances and providing a fluent API for setting values
* Lombok annotated code:

Course **course** = Course.*builder*()

.title(**"Agile Course 1"**)

.university(**"HCMUTE"**)

.build();

**6. @Data**

* The @Data annotation combines the functionality of *@ToString*, *@EqualsAndHashCode*, *@Getter*, *@Setter* and *@NoArgsConstructor*. Essentially, using *@Data* on a class is the same as annotating the class with a default *@ToString*, *@EqualsAndHashCode,* and *@NoArgsConstructor* as well as annotating each field with both *@Getter* and *@Setter*.
* This provides everything a Plain Old Java Object (POJO) needs.

**V. References**

For more information and features, please go to:

- Introduction and User guide: [***http://jnb.ociweb.com/jnb/jnbJan2010.html***](http://jnb.ociweb.com/jnb/jnbJan2010.html)

- Lombok API Documentation:[***https://projectlombok.org/api/index.html***](https://projectlombok.org/api/index.html)

- *http://awhitford.github.io/lombok.maven/lombok-maven-plugin/usage.html*

5