**Java Project Lombok**

Project Lombok is a very useful tool for java projects to reduce boiler plate code.

### Problem Statement

In Java vs. other language debate, the first knock which you get from other language supporters is that Java requires lot of boiler plate code and you just cannot get over it, and you are defenceless. The same issue is also reported in multiple platforms and developer communities.

A class should have getter-setters for the instance variables, equals & hashCode method implementation, all field constructors and an implementation of toString method. This class so far has no business logic and even without it is 80+ lines of code. This is insane.

### Project Lombok

Project Lombok is a java library that automatically plugs into your editor and build tools and helps reduce the boiler plate code. Let’s see how to setup Lombok project first.

### How does Java Project Lombok work?

Lombok has various annotations which can be used within our code that is be processed during the compile time and appropriate code expansion would take place based on the annotation used.

Lombok only does the code reduction in view time, after the compiling the byte code is injected with all the boiler plate. This helps keeping our codebase small, clean and easy to read and maintain.

### Project Lombok Maven

Adding Lombok in your project is simple. Just add the below dependency in your maven project pom.xml file.

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

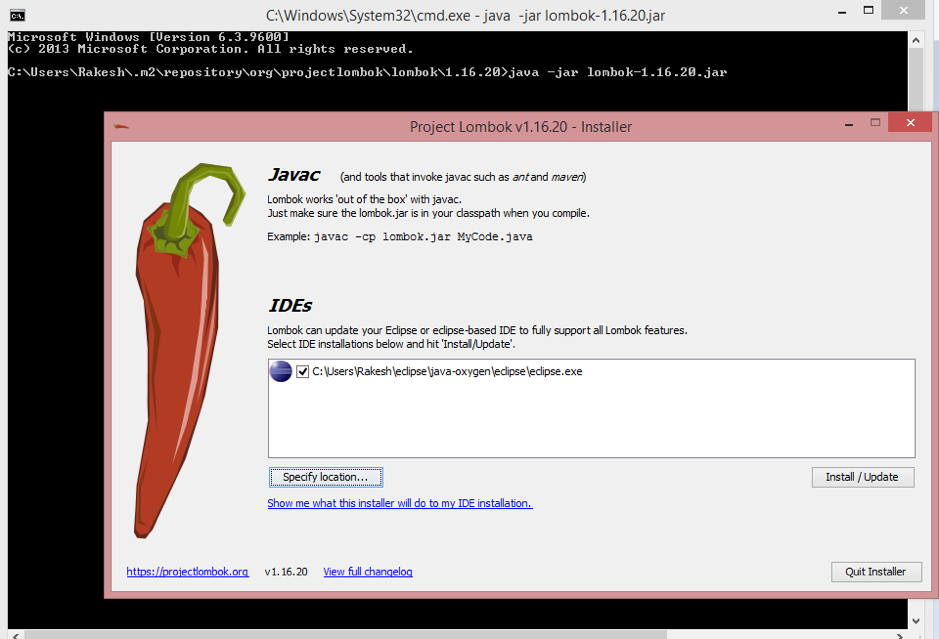
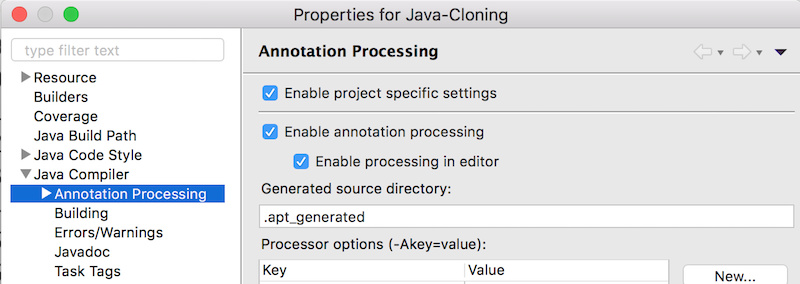
<version>1.16.20</version>

</dependency>

### Adding the Lombok Plugin in IDE (Eclipse)

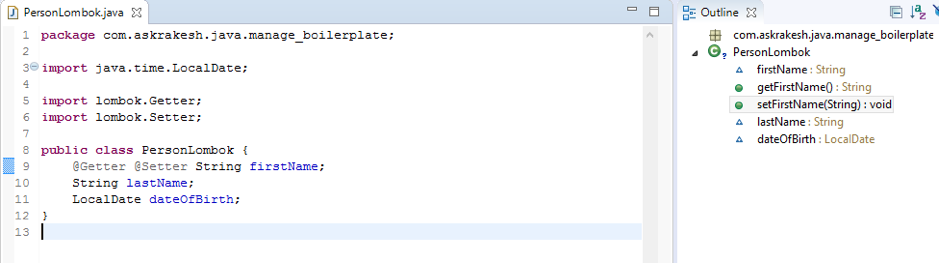
Here are the installation steps for Windows:

1. Downloaded jar from <https://projectlombok.org/download> or use the jar which is downloaded from your maven build.
2. Execute command in terminal: java -jar lombok.jar
3. This command will open window as show in the picture below, install and quit the installer and restart eclipse.

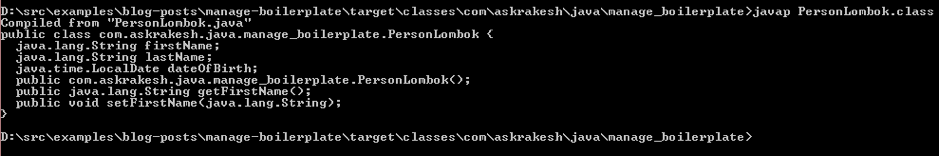
Restart Eclipse and enable “Annotation Processing” in project properties as shown in below image.  


### Lombok’s peek in Eclipse outline

After installation let’s check how we can see our reduced boiler plate code? I have recreated the same class as PersonLombok. Eclipse outline which displays getter and setter for firstName. This was done based on the Lombok’s @Getter & @Setter [annotation](https://www.journaldev.com/721/java-annotations) set for instance variable firstName.



### Lombok’s peek in Java byte code

We can check the addition of getter & setter methods for the firstName from class bytecode.  


### Project Lombok Annotations

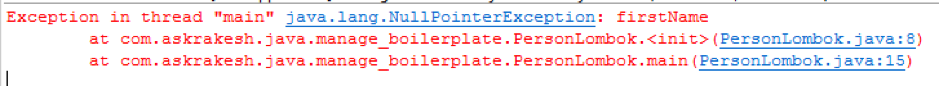
Project Lombok provides many annotation which helps to reduce the boiler plate code in various scenarios. Let’s go through a few of them.

#### Constructor Annotation

1. @AllArgsConstructor
2. public class PersonLombok {
3. @NonNull String firstName;
4. String lastName;
5. LocalDate dateOfBirth;
6. public static void main(String[] args) {
7. new PersonLombok(null, "Kumar", LocalDate.now());
8. }
9. }

Above code injects the following in the class:

* + A constructor with all arguments by @AllArgsConstructor
  + Null check during when passing an argument in the Constructor by @NonNull. @NonNull annotation can also be used when passing an argument as a parameter to a method

Here’s the result of the program run.  
  
@RequiredArgsConstructor generates a constructor with 1 parameter for each field that requires special handling. All non-initialized final fields get a parameter, as well as any fields that are marked as @NonNull that aren’t initialized where they are declared.

#### Getter/Setter Annotations

These annotations can be used either at the field or class level. If you want fine grained control then use it at field level. When used at class level all the getter/setters are created. Let’s work on the class we had created above.

@AllArgsConstructor @Getter @Setter

public class PersonLombok {

String firstName;

String lastName;

LocalDate dateOfBirth;

}

#### equals, hashCode and toString annotations

It’s recommended to override the hashCode() and equals() methods while creating a class. In Lombok we have @EqualsAndHashCode annotation which injects code for equals() & hashCode() method as they go together. Additionally a @ToString annotation provides a toString() implementation. Let’s see this:

@AllArgsConstructor @Getter @Setter

@EqualsAndHashCode

@ToString

public class PersonLombok {

String firstName;

String lastName;

LocalDate dateOfBirth;

}

We now have achieved to create Person class without any boiler plate code with help of Lombok annotations. However it gets even better we can replace all the annotations used in the above class with @Data and get the same functionality.

#### Design Pattern based Annotations

@Builder lets you automatically produce the code required to have your class be instantiable using [builder pattern](https://www.journaldev.com/1425/builder-design-pattern-in-java).

@Builder

public class Employee {

String firstName;

String lastName;

LocalDate dateOfBirth;

public static void main(String[] args) {

Employee emp = new EmployeeBuilder().firstName("Rakesh")

.lastName("Kumar")

.dateOfBirth(LocalDate.now())

.build();

}

}

@Delegate generates delegate methods that forward the call to this field on which annotation is used.

“[Favour Composition over Inheritance](https://www.journaldev.com/12086/composition-vs-inheritance)“, but this creates a lot of boiler plate code similar to [Adapter Pattern](https://www.journaldev.com/1487/adapter-design-pattern-java). Lombok’s took the clue from Groovy’s annotation by the same name while implementing this functionality. Let’s see an implementation:

@RequiredArgsConstructor

public class AdapterImpl implements Adapter {

@Delegate

private final Adaptee instance;

public static void main(String[] args) {

AdapterImpl impl = new AdapterImpl(new Adaptee());

impl.display();

}

}

interface Adapter {

public void display();

}

class Adaptee {

public void display() {

System.out.println("In Adaptee.display()");

}

}

Lombok provides functionality for fine grained control in all the annotations.

### Boiler-Plate: Are Java Architects Listening?

Yes, they are. You need to understand unlike other languages, Java has taken utmost care of upgrading the language such that they do not break any existing codebase which are in older versions of java. This itself is a huge task and cannot be undermined.

They are already modifying and building better type inference capabilities in the language which has been rolled out. One of the important features planned for Java 10 is **Local-Variable Type Inference**. Though the feature has more to do with adding dynamic typing than the boiler plate, but it is a small drop in the Ocean to manage the boiler-plate code.

### Summary

Reducing boiler plate code helps in better readability, less code also means less error. Project Lombok is heavily used today in almost all the major organizations. We provided you with the most useful features from Lombok. Hope you give it a try.

**Source Code**: You can visit my [Github link](https://github.com/arunkumarsa7/examples.git" \t "_blank) to download complete source code used in this tutorial.