Contents

[**Java Annotations** 2](#_Toc33649413)

[**What are Java Annotations** 2](#_Toc33649414)

[**Purpose of Annotations** 2](#_Toc33649415)

[**Are Java annotations same as Java comments ?** 2](#_Toc33649416)

[**Where can annotations be used ?** 2](#_Toc33649417)

[**Java Annotations Format** 2](#_Toc33649418)

[**Built-In Java Annotations** 2](#_Toc33649419)

[**Annotations that can be applied to other annotations** 3](#_Toc33649420)

# **Java Annotations**

## **What are Java Annotations**

Java annotations are used to provide meta data for your Java code. Being meta data, Java annotations do not directly affect the execution of your code, although some types of annotations can actually be used for that purpose.

## **Purpose of Annotations**

Information for the compiler - to detect errors or suppress warnings

compile-time or deployment/build-time processing -

Runtime processing –

Java annotations can be used at build time while building Software project. The build process includes generating source code, compiling source code,generating XML files(eg deployment desrciptors), packaging the compiled code and files int JAR file etc. Building the software is typically done by an automatic build tool like Apache Ant or Apache Maven. Build tools can actually scan Java code for annotations and generate source code or other files based on annotations

## **Are Java annotations same as Java comments ?**

Annotations are used to give detailed information to the compiler whereas Comments are for the convenience of the programmer so that he know how the code is structured.

## **Where can annotations be used ?**

You can place Java annotations above classes, interfaces, methods, method parameters, fields and local variables.

## **Java Annotations Format**

@ sign indicates to compiler that what follows is annotation

@Entity(value=”yes”,tableName=”vehicle”,primaryKey=”id”)

Annotations can also Repeat

@Author(name = "Jane Doe")

@Author(name = "John Smith")

class MyClass { ... }

## **Built-In Java Annotations**

**@Deprecated** - The @Deprecated annotation is used to mark a class, method or field as deprecated, meaning it should no longer be used. If your code uses deprecated classes, methods or fields, the compiler will give you a warning.

**@Override** - The @Override Java annotation is used above methods that override methods in a superclass. If the method does not match a method in the superclass, the compiler will give you an error. The @Override annotation is not necessary in order to override a method in a superclass. It is a good idea to use it still, though. In case someone changed the name of the overridden method in the superclass, your subclass method would no longer override it. Without the @Override annotation you would not find out. With the @Override annotation the compiler would tell you that the method in the subclass is not overriding any method in the superclass.

**@SuppressWarnings** - The @SuppressWarnings annotation makes the compiler suppress warnings for a given method. For instance, if a method calls a deprecated method, or makes an insecure type cast, the compiler may generate a warning. You can suppress these warnings by annotating the method containing the code with the @SuppressWarnings annotation.The @SuppressWarnings annotation makes the compiler suppress warnings for a given method. For instance, if a method calls a deprecated method, or makes an insecure type cast, the compiler may generate a warning. You can suppress these warnings by annotating the method containing the code with the @SuppressWarnings annotation.

## **Annotations that can be applied to other annotations**

@Retention(RetentionPolicy.RUNTIME)

This is what signals to the Java compiler and JVM that the annotation should be available via reflection at runtime.

RetentionPolicy.CLASS means that the annotation is stored in the .class file, but not available at runtime. This is the default retention policy, if you do not specify any retention policy at all.

RetentionPolicy.SOURCE means that the annotation is only available in the source code, and not in the .class files and not a runtime.

@Target - You can specify which Java elements your custom annotation can be used to annotate. You do so by annotating your annotation definition with the @Target annotation.

@Target({ElementType.METHOD})

ElementType.ANNOTATION\_TYPE

ElementType.CONSTRUCTOR

ElementType.FIELD

ElementType.LOCAL\_VARIABLE

ElementType.METHOD

ElementType.PACKAGE

ElementType.PARAMETER

ElementType.TYPE

@Documented

The @Documented annotation is used to signal to the JavaDoc tool that your custom annotation should be visible in the JavaDoc for classes using your custom annotation.