Compile Time Annotations in Java

@Override ensure at compile time that method annotated with override must be overriding super class method.I want a compile time annotation, which ensures that any method annotated with @MyAnnotation, must be returning Boolean type only.

## Define Annotations:- Annotations are metadata or data about data. An annotation indicates that the declared element should be processed in some special way by a compiler, development tool, deployment tool, or during run time.

Now let us consider case one by one from its usage at compile time, development time, deployment time, or during run time. So that is why the retention policy of annotation can be of 3 kind as per its usage.

1. Development time--> Retention is set to source only, such annotation are use full at source level only and removed by compiler while generating binary class file.

2. Compile time or Deployment time--> Retention is set be to class only. Such annotation are used by compiler or deployment tools for taking some specific action e.g. producing compile time error in case of some rule or guideline violation.

3. Run time(Execution time)--> Retention is set run-time, such annotation are retained in class file are used for deducing some meta information at run time and may effect run time processing. These annotations are not ignored by VM.

So, let us consider case 2 only.

How to create compile time annotations or How to create annotation very similar @Override which ensure that any method annotated with it must have Boolean return type?

The first thing which come to my mind is, Is it possible to create such annotation? My first thought was, If java providers can make such @Override annotation, then I should. But after checking the source code and other stuff I was not able to find, how compiler got to know that when any method annotated with @Override and not actually overriding then it should generate error. Nothing is mentioned in the Override annotation source code.

Are you still thinking that such compile time annotations are possible? I think you might have started thinking that, it is not possible because I have created enough negative stories around it.

But I know you guyz are true Bollywood fence, until and unless its not happiness its not the end. "Picture abhi baki hai mere dost"(the movie is still not ended my friend!!!).

So, as usual answer is yes, such annotations can be created which can be used for compile time validation, as @override makes sure you are overriding.

What are the steps that need to be performed, to create such annotations?

Create the annotation.

Create the annotation processor class for your custom annotation, if want to take some actions at compile time. Here goes the basic example.

@SupportedAnnotationTypes(value = {"com.radhe.test.annotations.Hello"})

public class YourAnnotationProcessor extends AbstractProcessor {

@Override

public boolean process(Set aAnnotations, RoundEnvironment aRoundEnv) {

//your annotation processor code goes here...

//return something based on your annotation processing logic.

return false;

}

}

}

Now, pass this annotation processor with your compilation command. e.g javac -processor com.radhe.test.annotations.YourAnnotationProcessor

There are several ways to include this processor in eclipse and other java IDE's to include this annotation processor.

### Hoping this will help you and me. Compile Time Annotations in Java

Hello,

While thinking about solution of the one of the problem, I found that I need an annotation, which do some validation at compile time. i.e. @Override ensure at compile time that method annotated with override must be overriding super class method.

I want a compile time annotation, which ensures that any method annotated with @MyAnnotation, must be returning Boolean type only.

I searched here and there, went through multiple web pages, tutorials, articles and white papers.

Finally I didn't invented anything but point here is that it is easy to find multiple blogs, examples and tutorials regarding same and same thing. But, the simple page, which shows how to create compile time annotations, is not available. So, I decided write it.

All you know about annotation, to refresh for yourself, I am putting it from some other resource for ready reference.

Let's start it from definition.

**Define Annotations?**

Annotations are metadata or data about data. An annotation indicates that the declared element should be processed in some special way by a compiler, development tool, deployment tool, or during run time.

Now let us consider case one by one from its usage at compile time, development time, deployment time, or during run time. So that is why the retention policy of annotation can be of 3 kind as per its usage.

1.     **Development time-->** Retention is set to source only, such annotation are use full at source level only and removed by compiler while generating binary class file.

2.     **Compile time or Deployment time-->** Retention is set be to class only. Such annotation are used by compiler or deployment tools for taking some specific action e.g. producing compile time error in case of some rule or guideline violation.

3.     **Run time(Execution time)-->** Retention is set run-time, such annotation are retained in class file are used for deducing some meta information at run time and may effect run time processing. These annotations are not ignored by VM.

Now I am coming to the main motto of this blog. i.e. coming up with annotation that can be used at compile time to have some more compilation validation.

So, let us consider case 2 only.

**How to create compile time annotations or How to create annotation very similar @Override which ensure that any method annotated with it must have Boolean return type?**

The first thing which come to my mind is, Is it possible to create such annotation? My first thought was, If java providers can make such @Override annotation, then I should. But after checking the source code and other stuff I was not able to find, how compiler got to know that when any method annotated with @Override and not actually overriding then it should generate error. Nothing is mentioned in the Override annotation source code.

Are you still thinking that such compile time annotations are possible? I think you might have started thinking that, it is not possible because I have created enough negative stories around it.

But I know you guyz are true Bollywood fence, until and unless its not happiness its not the end. **"Picture abhi baki hai mere dost"**(the movie is still not ended my friend!!!).

So, as usual answer is yes, such annotations can be created which can be used for compile time validation, as @override makes sure you are overriding.

**What are the steps that need to be performed, to create such annotations?**

* Create the annotation.
* Create the annotation processor class for your custom annotation, if want to take some actions at compile time. Here goes the basic example.

@SupportedAnnotationTypes(value = {"com.radhe.test.annotations.Hello"})

public class YourAnnotationProcessor extends AbstractProcessor {

@Override

     public boolean process(Set aAnnotations, RoundEnvironment aRoundEnv) {

  //your annotation processor code goes here...

  //return something based on your annotation processing logic.

return false;

     }

 }

}

* Now, pass this annotation processor with your compilation command. e.g javac -processor com.radhe.test.annotations.YourAnnotationProcessor
* There are several ways to include this processor in eclipse and other java IDE's to include this annotation processor.

Hoping this will help you and me.