Java Annotations

1.Java annotations are metadata (data about data) for our program source code.

2.They provide additional information about the program to the compiler but are not part of the program itself. These annotations do not affect the execution of the compiled program.

3.Annotations start with @. Its syntax is:



4.Built-In Java Annotations

**Built-In Java Annotations used in Java code**

@Override

@SuppressWarnings

@Deprecated

**Built-In Java Annotations used in other annotations**

@Target

@Retention

@Inherited

@Documented

**6.@Override**

@Override annotation assures that the subclass method is overriding the parent class method. If it is not so, compile time error occurs.

class Animal{

void eatSomething(){System.out.println("eating something");}

}

class Dog extends Animal{

@Override

void eatsomething(){System.out.println("eating foods");}//should be eatSomething

}

class TestAnnotation1{

public static void main(String args[]){

Animal a=new Dog();

a.eatSomething();

}}

**7.@SuppressWarnings**

@SuppressWarnings annotation: is used to suppress warnings issued by the compiler.

**@Deprecated**

@Deprecated annoation marks that this method is deprecated so compiler prints warning. It informs user that it may be removed in the future versions. So, it is better not to use such methods.

**8.Java Custom Annotations**

Java Custom annotations or Java User-defined annotations are easy to create and use. The *@interface* element is used to declare an annotation. For example:

**@interface** MyAnnotation{}

Here, MyAnnotation is the custom annotation name.

**9.Points to remember for java custom annotation signature**

There are few points that should be remembered by the programmer.

a.Method should not have any throws clauses

b.Method should return one of the following: primitive data types, String, Class, enum or array of these data types.

c.Method should not have any parameter.

d.We should attach @ just before interface keyword to define annotation.

e.It may assign a default value to the method.

**10.Types of Annotation**

There are three types of annotations.

1.Marker Annotation

2.Single-Value Annotation

3.Multi-Value Annotation

**1.Marker Annotation**

An annotation that has no method, is called marker annotation. For example:

@interface MyAnnotation{}

The @Override and @Deprecated are marker annotations.

**2) Single-Value Annotation**

An annotation that has one method, is called single-value annotation. For example:

**@interface** MyAnnotation{

**int** value();

}

**We can provide the default value also. For example:**

**@interface** MyAnnotation{

**int** value() **default** 0;

}

**How to apply Single-Value Annotation**

Let's see the code to apply the single value annotation.

@MyAnnotation(value=10)

The value can be anything.

**3) Multi-Value Annotation**

An annotation that has more than one method, is called Multi-Value annotation. For example:

**@interface** MyAnnotation{

**int** value1();

String value2();

String value3();

}

}

We can provide the default value also. For example:

**@interface** MyAnnotation{

**int** value1() **default** 1;

String value2() **default** "";

String value3() **default** "xyz";

}

**How to apply Multi-Value Annotation**

Let's see the code to apply the multi-value annotation.

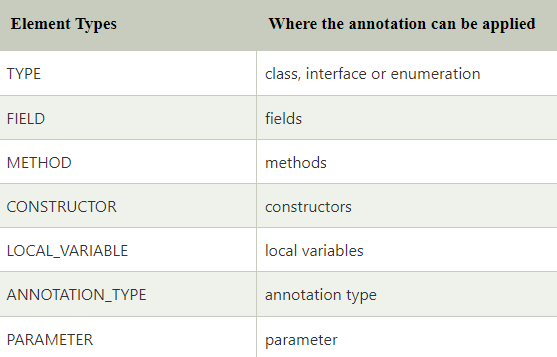
@MyAnnotation(value1=10,value2="Arun Kumar",value3="Ghaziabad")

## **11.Built-in Annotations used in custom annotations in java**

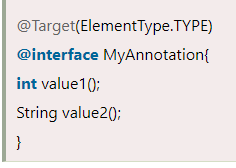
* @Target
* @Retention
* @Inherited
* @Documented

## **@Target**

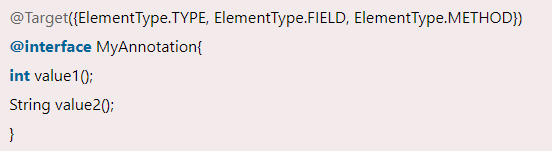
**@Target** tag is used to specify at which type, the annotation is used.



## **Example to specify annoation for a class**

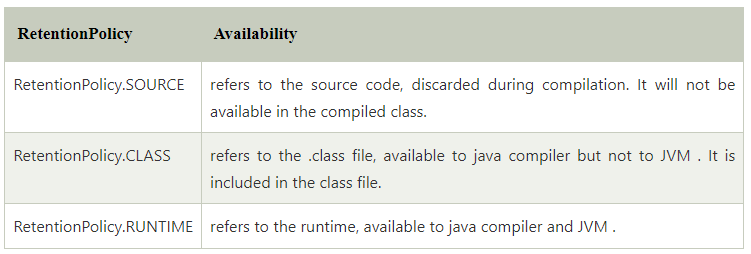


## **Example to specify annotation for a class, methods or fields**

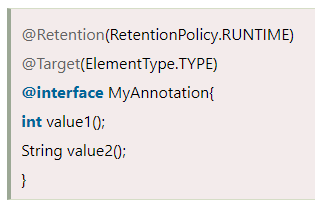


## **@Retention**

**@Retention** annotation is used to specify to what level annotation will be available.

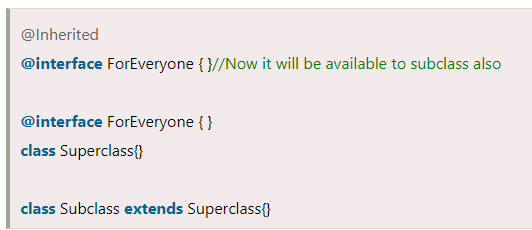


## **Example to specify the RetentionPolicy**



## **@Inherited**

By default, annotations are not inherited to subclasses. The @Inherited annotation marks the annotation to be inherited to subclasses.



## **@Documented**

The @Documented Marks the annotation for inclusion in the documentation.