Annotation provides data about a program that is not part of the program itself. This have no direct effect on the operation of the code that annotate. Annotations have number of uses, among them:

* **Information for compiler:** Annotations can be used by the compiler to detect errors or suppress warnings.
* **Compiling time and deployment time:** Software tools can process annotations information to generate code, XML files and so forth.
* **Runtime Processing:** Some annotation are available to be examined at runtime.

**#.** Annotations can be applied to programs declaration of classes, fields, methods and other program elements.

**#. Annotation used by the compiler:** There are three annotation types that are predefined by the language specification, itself.

* **@Deprecated:** The @Deprecated annotation indicates that the marked element is deprecated and should no longer be used. The compiler generates a warning whenever a program uses a method, class, or field with @Deprecated annotations.
* **@Override:** The Annotation informs the compiler that the element is meant to override an element declared in a **SuperClass.** If a method with this annotation does not not override its SuperClass’s method, then compiler will generate an error at compile time.
* **@SuppressWarnings:** This annotation tells the compiler to suppress (ignore, Keep down) specific warning that it would otherwise generate.

**Meta Annotations**

Meta annotation which are actually known as the **annotations of annotations**, contain four types:

**@target, @Retention, @Documented, and @Inherited**

**No-1. @Target (optional):** The target annotations indicates the targeted elements of a class in which the annotation type will be applicable. It contains the following enumerated types as it value.

* **@Target (Element.TYPE):** can be applied to any element of class.
* **@Target (ElementType.FIELD):** can be applied to a Filed level annotation.
* **@Target (ElementType.METHOD):** can be applied to a method level annotation.
* **@Target (ElementType.PARAMETER):** can be applied to the parameter of a method.
* **@Target (ElementType.CONSTRUCTOR):** can be applied to a constructor.
* **@Target (ElementType.LOCAL\_VARIABLE):** can be applied to a local variable.
* **@Target (ElementType.ANNOTATION\_TYPE):** indicates that the declared type itself is an annotation type.

**Declaring of Target Example:**

@Target **(**ElementType**.**METHOD**)0**

public @interface TestTarget**{**

/\* It is a user-defined Filed of this annotation. Don't confuse with parentheses. \*/

public String doTarget**();**

**}**

**No-2. @Retention:** The retention annotation indicates where and how long annotation with this type are to be retained (continue). There are three values.

* **RetentionPolicy.SOURCE:** Annotation with this type will be retained (continue) only at the source level and will be ignored by the compiler.
* **RetentionPolicy.CLASS:** Annotation with this type will be retained by the compiler at compile time, but will be ignored by the JVM.
* **RetentionPolicy.RUNTIME:** Annotation with this type will be retained by the VM so they can be read only at runtime.

**Declaration Example:**

@Target **(**ElementType**.**METHOD**)**

**@Retention (RetentionPolicy.RUNTIME)**

public @interface TestTarget**{**

/\* It is a user-defined Filed of this annotation. Don't confuse with parentheses. \*/

public String doTarget**();**

**}**

**No-3. @Documented:** This annotation indicates that an annotation with this type should be documented by the Javadoc tool. By default annotations are not included in Javadoc but if **@Documented is used,** it then will processed by **Javadoc –like** tools and the annotation type information will also be included in the generated document.

@Target **(**ElementType**.**METHOD**)**

@Retention (RetentionPolicy.RUNTIME)

**@Documented**

public @interface TestTarget**{**

/\* It is a user-defined Filed of this annotation. Don't confuse with parentheses. \*/

public String doTarget**();**

**}**

**No-4. @Inherited:** This is a bit of complex annotation type. It indicating that the annotated class with this type is automatically inherited more specifically, if we define an annotation with **@Inherited** tag then annotate class with our annotation, and finally extend the class in a subclass, all properties of the parent class will be inherited into its subclass.

@Target **(**ElementType**.**METHOD**)**

@Retention (RetentionPolicy.RUNTIME)

@Documented

**@Inherited**

public @interface TestTarget**{**

/\* It is a user-defined Filed of this annotation. Don't confuse with parentheses. \*/

public String doTarget**() default “Do What?”;**

**}**