Annotations

- Definition
- Standard Annotations
- Types of Annotations
- Syntax

Definition

An annotation is a form of syntactic metadata that can be added to Java source code.

Annotations is not part of the program itself.

Annotations have no direct effect on the operation of the code they annotate.

What can be annotated:

- classes,
- methods,
- variables,
- parameters,
- packages,
- class instance creation,
- type cast,
- implements clause,
- thrown exception declaration.

Annotations from the library

Annotations applied to Java code

@Override	Checks that the method is an override. Causes a compile error if the method is not found in one of the parent classes or implemented interfaces.
@Deprecated	Marks the method as obsolete. Causes a compile warning if the method is used.
@SuppressWarnings	Instructs the compiler to suppress the compile time warnings specified in the annotation parameters.
@FunctionalInterface	Specifies that the type declaration is intended to be a functional interface, since Java 8.
@SafeVarargs	Suppress warnings for all callers of a method or constructor with a generics varargs parameter, since Java 7.

Annotations applied to other annotations (also known as "Meta Annotations")

@Retention	Specifies how the marked annotation is stored—Whether in code only, compiled into the class, or available at runtime through reflection.
@Documented	Marks another annotation for inclusion in the documentation.
@Target	Marks another annotation to restrict what kind of Java elements the annotation may be applied to.
@Inherited	Marks another annotation to be inherited to subclasses of annotated class (by default annotations are not inherited to subclasses).
@Repeatable	Specifies that the annotation can be applied more than once to the same declaration, since Java 8.

Types of annotations

```
public enum RetentionPolicy {
    /**
     * Annotations are to be discarded by the compiler.
     */
    SOURCE,
    /**
     * Annotations are to be recorded in the class file by the compiler
     * but need not be retained by the VM at run time.  This is the default
     * behavior.
     */
    CLASS,
    /**
     * Annotations are to be recorded in the class file by the compiler and
     * retained by the VM at run time, so they may be read reflectively.
     *
     * @see java.lang.reflect.AnnotatedElement
     */
    RUNTIME
}
```

```
public enum ElementType {
    /** Class, interface (including annotation type), or enum declaration */
    TYPE,

    /** Field declaration (includes enum constants) */
    FIELD,

    /** Method declaration */
    METHOD,

    /** Formal parameter declaration */
    PARAMETER,

    /** Constructor declaration */
    CONSTRUCTOR,

    /** Local variable declaration */
    LOCAL_VARIABLE,

    /** Annotation type declaration */
    ANNOTATION_TYPE,

    /** Package declaration */
    PACKAGE,

    /** Type parameter declaration */
    TYPE_PARAMETER,

    /** Use of a type */
    TYPE_USE
}
```

Syntax

```
import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
/**
 * https://en.wikipedia.org/wiki/Java_annotation
 */
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE,ElementType.METHOD,
        ElementType.CONSTRUCTOR,ElementType.ANNOTATION_TYPE,
        ElementType.PACKAGE,ElementType.FIELD,ElementType.LOCAL_VARIABLE})
@Inherited
public @interface Unfinished {
    public enum Priority { LOW, MEDIUM, HIGH }
    String value();
    String[] changedBy() default "";
    String[] lastChangedBy() default "";
    Priority priority() default Priority.MEDIUM;
    String createdBy() default "James Gosling";
    String lastChanged() default "2011-07-08";
}
```