

Annotations:-

- * Note of explanation or comment added to a text or diagram.
- * In java, we need to add extra information to code, (Data about data or Meta Data)
- * Organized data about data, embedded within the code.
- * Decorative data
- * Most probably you have seen one, on top of class, function, variables.
- * Something with @, one top of function, variable, class.

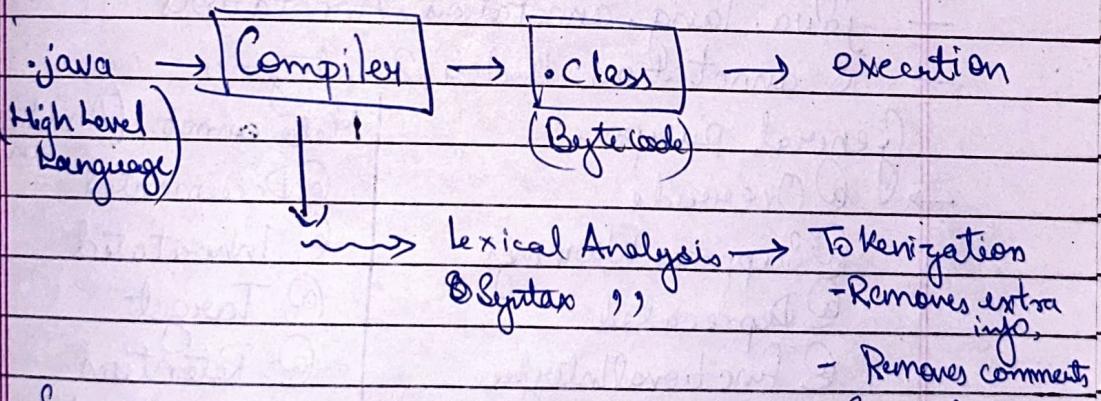
e.g. `@Override`

```
public void run();}
```

`@FunctionalInterface`

```
public class Test { }
```

We already have comments for meta data
Why we need Annotations??



So,

Comments won't be present after compilation.

" " are for human not for JVM or compiler.
Let's say we want to inform compiler about something.

→ Annotations → introduced from Java 1.5.

Earlier we used to have XML, still in Spring, Hibernate, Struts.

- These XML files you can provide externally and fetch runtime.
- ↓
XML file.
code → Runtime → O/p

XML vs Annotations

- Need to learn explicitly.
- Data outside of code, need to maintain, Location, Parsing etc.

Web.xml

@WebService ("getorder")
Public class Getorder {
3,

Types of Annotations

1 Standard Annotation / Predefined / Built in

- Defined & provided by Java.
- java.lang.annotation.Annotation
- 8 annotations under 2 categories:

General purpose

→ @Override

→ @SuppressWarnings

→ @Deprecated

→ @FunctionalInterface

Under java.lang package

Note annot. (Annotations)

→ @Documented

→ @Inherited

→ @Target

→ @Retention

Under java.lang Annotation

2 Custom Annotations / User defined Annotations

- We can create & add custom functionality.
- Created using predefined annotations.

eg. `@Documented`

`@Retention(RetentionPolicy.RUNTIME)`

`@interface TestAnnotation {`

`String name();`

`String date();`

3.

- * Anno.s can be used with Class, Method or Variable.

Based on no. of Annotation members :-

① Marker Annotations

- No values, No Members

- eg. `@Override`

② Single Valued Anno. -

- One member or Value

- eg. `@SupressWarnings(value = "unchecked")`
or .. " " ("unchecked")

③ Multi valued - - -

- Multi values

- eg. `@WebServlet(name = "abc.servlet", urlPattern = "abc.html")`

Meta Annotations .

(Anno, about Anno.)

- All annos in java need to have two annos. `@target` & `@Retention`.

① Target

- Where you want to use it
(class, variable/method)
- `@Target(SingleValueAnno)`
public @interface Target
{ ElementType[] value(); }

② Retention

- Continue to hold
- Till what level,
we want to hold
Annotation.

ElementType is an Enum in java.lang.annotations package.
8 possible values.

- ~~Anno~~ ANNOTATION TYPE
CONSTRUCTOR

FIELD

LOCAL VARIABLE

METHOD

PACKAGE

PARAMETER

TYPE (for class,
interface or
enum)

- ElementType is an Array,
Multiple Values.

General Phases:-

① Source code → Compiler → ② Runtime
java .class

③ @Retention
(Single, Threaded)

Public @interface Retention
{}

RetentionPolicy value();
{}

RetentionPolicy is an Enum
in java.lang.Annotation package

Not an array take
single Value

SOURCE Compiler will remove
annotations while
compiling source
code w/o I have

CLASS Keep in .class
(Default) but remove wile
runtime

RUNTIME Retain in all
the phases

③ ④ Inherited (Marker Annotation)

- To make out custom Annotation Inheritable
- By default in java annos are not inheritable.
- e.g. `public @interface SecurityClearance { String level; }`

`@SecurityClearance(value="Level A")`

`public class Employee { }`

Manager
public class Manager extends Employee
{ }

3

- This annotation only affects if custom Annotation is for CLASS

④ @Documented (Marker annotation)

- * Particular annotation is to be documented by Javadoc or similar tools.

General Purpose.

(Use for each in javadoc).

① @Override (Marker Anno).

- * To generate errors.

② @Deprecated (Marker annos. till 8)

- * In java 9 2 field ↳ Since - which version it is deprecated
for removal - will be removed
in upcoming version
or not.
- * To generate warnings.

③ @FunctionalInterface (Marker Anno)

- * To declare interface of functional type
- * Compiler will generate warnings if interface doesn't fit into definition of functional interface.
- * Only one abstract method.
- * Used with Lambda funcn.

④ @SupressWarnings (Marker Single Valued Anno)

- * To suppress warnings.

- * eg. `@SupressWarnings ("unchecked") or ("deprecation")`
public class Calculator {