**206. Java Documentation – Javadoc**

1. Java Documentation is given on oracle documentation. From Java Documentation <https://docs.oracle.com/en/java/>.

2. To prepare same documentation as official website for our java classes we use a Java Doc tool.

3. Following are the tags provided by Java Doc.

**Class**

a. @author

b. @version

c. @since

d. @see : References.

**Method.**

e. @param

f. @return

g. @throws

h. @exception

i. @deprecated

j. @code

**Others**

k. @link

l. @value

m. @serial

4. For documentation /\*\* \*\*/ is used.

5. It is a good practice to document the code properly.

6. It’s like we are writing the comments.

7. Its so useful.

**207. Built-in Annotations.**

1. Annotations are used to send the metadata to class or interface or so on.

2. Information related to class like data, author and etc.

3. Built-in Annotations types:

a. Applied to Code.

b. Applied to Other Annotations.

4. Applied Annotations:

a. @Override

b. @Depracated

c. @FunctionalInterface

d. @SuppressWarnings

e. @SafeVarargs

5. Annotations help’s us to detect the errors in code.

6. It is good practice to write @Override annotation while overriding the method.

7. The @Deprecated annotation is used to label the method as deprecated. The ide will show deprecated methods in strike off form. We will get warning when we compile the program, if ever used the deprecated method.

8. The @SuppressWarnings annotation takes attribute. We can suppress the warning. For example:

@SuppressWarnings(“deprication”)

We will not get the warnings regarding deprication.

9. @SafeVarargs give us the warning about that variable argument methods must be safe.

10. If an interface is having the single method then it is called as functional interface. The @FunctionalInterface annotation gives warning regarding the functional interface.

**208. User-Defined Annotation.**

1. Defining an user-defined annotation is similar to defining the interface. Just start the interface with the @. It will automatically inherit from annotation class.

2. The package name is java.lang.annotation.Annotation;

3. Annotations can be used at methods, class level.

4. Annotations can be written for the parameters also.

5. Annotations should be used at suitable places when it contains any metadata.

6. Elements of annotation are similar to abstract methods of an interface.

For example @interface MyAnno

{

//meta data

}

7. It is mandatory for a programmer to provide values for all elements.

8. Annotations give data about the data(program).

9. Annotations helps to maintain the important details about program.

10. Elements of annotation cannot have body. They are just similar to abstract methods. But they will have return type.

11. We can set default values of elements of annotation.

For example:

String name default “Shreyash”;

**209. Built-in Annotations #2.**

1. Other built-in annotations.(These annotations affects other annotations.)

a. Retention.

b. Documented.

c. Target.

d. Inherited.

2. @Retention(RetentionPolicy…)

a. There are three types of annotations.

b. The policy of retention tells us about availability of annotation

3. By using reflection you can get the metadata.

4. The @Documented annotation use to get the annotation in java document also. It doesn’t have elements.

5. The @Target annotation is used to decide the target of the annotation. It takes type value argument.

For example:

@Target(value={ElementType.LOCAL\_VARIABLE)

6. The @Inherited if the annotation is used upon of the class then the annotation will also applicable to its child classes.

7. Annotation should be used only once. But using @Repeatable annotation we can use the annotation multiple times.

For Example: @Repeatable(MyAnno.class)