**Annotations**

Annotations are simply a way to add some extra information about a piece of code. Such information can be used by other tools to treat that piece of code differently. You will be using a lot of annotations when you develop your own application.

You should note that all annotations in Java always start with ‘@’. For example, @Override annotation starts with '@'.

In the previous segment, you understood the basics of Annotations.

## @Override Annotation

Let’s learn about a commonly used annotation- @Override annotation.

#### Q1: @Override Annotation

@Override annotation necessary for compilation?

Ans: False

**✓ Correct**

**Feedback:**

 @Override annotation is indeed not necessary for compilation. Your code will compile without it too. The reason for adding @Override annotation is so that some run-time errors become compile time errors. Compile-time errors are easier to deal with. Therefore, this is the correct answer.

**What is @Override annotation?**  
When a method is marked with the @Override annotation, the compiler searches for that particular method in the parent class. If the method is not found in the parent class, then the compiler throws an error.

**Advantages of @Override annotation**

* Fewer chances of bugs because of an incorrect method name
* Better readability

## @Deprecated Annotation

You have already seen how @Override works. Now, let’s look at another annotation called @Deprecated annotation.

**What is @Deprecated annotation?**  
@Deprecated annotation is used to mark a piece of code that should not be used to write new code. If your code includes the deprecated code, the compiler will show a warning.

**Advantages of @Deprecated annotation**

* Warns the team/programmer against the deprecated elements
* Helps in maintaining the hygiene of the code

#### Q2: @Override Annotation

State whether the following statement is true or false.

If you annote a method with the @Override annotation, then that method should be declared and defined directly in the direct superclass or implemented interface.

Ans: False

**✓ Correct**

**Feedback:**

If you annotate a method with @Override annotation, then that method may not be present in the direct superclass or implemented interface.

Consider the following example to understand better.

Suppose there are three classes, A, B and C. A is the superclass of B, and B is the superclass of C. In C, you are trying to override a method named m1(), then this method m1() should be compulsorily present in either B or A. It is not required for method m1() to be present in the direct superclass (i.e B).

Therefore, this option is correct.

#### Q3. @Deprecated Annotation

State whether the following statement is true or false.

If a class is annotated with the @Deprecated annotation, then that class cannot be extended.

Ans: False

**✓ Correct**

**Feedback:**

Recall that only the final classes cannot be extended.  Even if a class is declared deprecated with @Deprecated annotation, that class can be extended although the compiler will show a warning. This option is correct.

**Annotations**  
Annotations are simply a way to add some extra information about a piece of code. Such information can be used by other tools to treat that piece of code differently. For example:

* **@Override Annotation -**When a method is marked with the @Override annotation, the compiler searches for that particular method in the parent class. If the method is not found in the parent class, then the compiler throws an error.
* **@Deprecated Annotation -**@Deprecated annotation is used to mark a piece of code that should not be used to write new code. If your code includes the deprecated code, the compiler will show a warning.