# Java Annotations

Asked 16 years, 4 months ago Modified 2 years, 8 months ago

Viewed 38k times

125

What is the purpose of annotations in Java? I have this fuzzy idea of them as somewhere in between a comment and actual code. Do they affect the program at run time?

What are their typical usages?

Are they unique to Java? Is there a C++ equivalent?

java annotations glossary

Share

Improve this question

Follow

edited Jan 31, 2013 at 17:36

Igor Popov
10.1k ● 7 ● 56 ● 70

asked Aug 23, 2008 at 13:23

Lehane
48.6k ● 14 ● 55 ● 54

11 You can do some really wicked things with annotations. For example, if you know Spring Framework and their @Transactional annotation which starts a transaction before method invocation and rollbacks on exception/error and commits on successfull transaction. It totally supports the SOC-principle (seperation of concern). Very clean ! Java rules ! :-) – user1830397 Nov 16, 2012 at 17:37

1 That looks like 3 questions to me. The second in itself is too broad. – Raedwald Jul 22, 2013 at 9:50

All annotations by syntax looks like interfaces. So, how compiler or JVM knows its purpose? Suppose if we take @ Override -> performs check on whether method overriden properly or not @ Transactional, as explained above, it performs transaction related stuff. For all these annotations, where can we found definition/logic? – Paramesh Korrakuti Mar 20, 2015 at 11:09

There is no such thing as an annotation, in the sense that there is no common feature of all annotations that could be used to put them in an abstract conceptual group, other than that they all start with an @ symbol. So, you need to learn every kind of annotation (@Override, @interface, user defined annotations) separately, and don't expect understanding one kind of annotation to tell you anything about the others. – Sean Apr 13, 2022 at 23:17

## 9 Answers

Sorted by: Highest score (default)

89

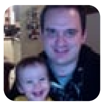
Annotations are primarily used by code that is inspecting other code. They are often used for modifying (i.e. decorating or wrapping) existing classes at run-time to change their behavior. Frameworks such as JUnit and

Hibernate use annotations to minimize the amount of code you need to write yourself to use the frameworks.

Oracle has a good explanation of the concept and its meaning in Java on their site.

Share Improve this answer Follow

edited Mar 22, 2013 at 17:56



Michael Hogenson
1,381 ● 1 ● 15 ● 31

answered Aug 23, 2008 at 13:38

Anders Sandvig
21k ● 16 ● 61 ● 74

This is only true of user defined annotations, which are one specific kind of annotation; for example, @Override does something completely different. – Sean Apr 13, 2022 at 23:15

19

> Also, are they unique to Java, is there a C++ equivalent?

No, but VB and C# have attributes which are the same thing.

Their use is quite diverse. One typical Java example, `@Override` has no effect on the code but it can be used by the compiler to generate a warning (or error) if the decorated method doesn't actually override another method. Similarly, methods can be marked obsolete.

Then there's reflection. When you reflect a type of a class in your code, you can access the attributes and act according to the information found there. I don't know any examples in Java but in .NET this is used by the compiler to generate (de)serialization information for classes, determine the memory layout of structures and declare function imports from legacy libraries (among others). They also control how the IDE form designer works.

/EDIT: Attributes on classes are comparable to tag interfaces (like Serializable in Java). However, the .NET coding guidelines say not to use tag interfaces. Also, they only work on class level, not on method level.

Share Improve this answer Follow

edited Nov 26, 2013 at 5:05
Afshin Moazami
2,098 ● 5 ● 33 ● 57

answered Aug 23, 2008 at 13:31
Konrad Rudolph
545k ● 139 ● 956 ● 1.2k

---

15

Anders gives a good summary, and here's an example of a JUnit annotation

```
@Test(expected=IOException.class)
public void flatfileMissing() throws IOException {
    readFlatFile("testfiles"+separator+"flatfile_doesn
}
```

Here the `@Test` annotation is telling JUnit that the `flatfileMissing` method is a test that should be executed and that the expected result is a thrown `IOException`. Thus, when you run your tests, this method will be called and the test will pass or fail based on whether an `IOException` is thrown.

Share Improve this answer Follow

edited Nov 22, 2016 at 8:37
Lahiru Ashan
767 ● 9 ● 16

answered Aug 23, 2008 at 16:16
Eli Courtwright
193k ● 68 ● 218 ● 257

---

8

Java also has the Annotation Processing Tool (apt) where not only you create annotations, but decide also how do these annotations work on the source code.

[Here](#) is an introduction.

Share Improve this answer Follow

edited Jul 31, 2012 at 12:17
S.L. Barth is on codidact.com
8,265 ● 71 ● 53 ● 67

answered Aug 25, 2008 at 17:02
pek
18k ● 28 ● 88 ● 99

---

6

To see some cool stuff you can do with Annotations, check out [my JavaBean annotations and annotation processor](#).

They're great for generating code, adding extra validations during your build, and I've also been using them for an error message framework (not yet published -- need to clear with the bosses...).

Share Improve this answer Follow

edited Aug 9, 2012 at 10:06
forsvarir
10.8k ● 6 ● 47 ● 81

answered Sep 16, 2008 at 15:29
Scott Stanchfield
30.6k ● 9 ● 51 ● 67

---

4

The first thing a newcomer to annotations will ask about annotations is: "What is an annotation?" It turns out that there is no answer to this question, in the sense that there is *no common behavior* which is present in *all* of the various kinds of java annotations. There is, in other words, nothing that binds them together into an abstract conceptual group other than the fact that they all start with an "@" symbol.

For example, there is the @Override annotation, which tells the compiler to check that this member function overrides one in the parent class. There is the @Target annotation, which is used to specify what kinds of objects a user defined annotation (a third type of construct with nothing in common with other kinds of annotation) can be attached to. These have nothing to do with one another *except for starting with an @ symbol*.

Basically, what appears to have happened is that some committee responsible for maintaining the java language definition is gatekeeping the addition of new keywords to the java language, and therefore other developers are doing an end run around that by calling new keywords "annotations". And that's why it is hard to understand, in general what an annotation is: because there is no common feature linking all annotations that could be used to put them in a conceptual group. In other words, annotations as a concept do not exist.

Therefore I would recommend studying the behavior of every different kind of annotation individually, and do not expect understanding one kind of annotation to tell you anything about the others.

Many of the other answers to this question assume the user is asking about *user defined annotations* specifically, which are one kind of annotation that defines a set of integers or strings or other data, static to the class or method or variable they are attached to, that can be queried at compile time or run time. Sadly, there is no marker that distinguishes this kind of annotation from other kinds like @interface that do different things.

Share Improve this answer Follow

edited Apr 13, 2022 at 23:11

answered Apr 13, 2022 at 22:45

Sean

**379** ● 3 ● 5

1

By literal definition an annotation adds notes to an element. Likewise, Java annotations are tags that we insert into source code for providing more information about the code. Java annotations associate information with the annotated program element. Beside Java annotations Java programs have copious amounts of informal documentation that typically is contained within comments in the source code file. But, Java annotations are different from comments they annotate the program elements directly using annotation types to describe the form of the annotations. Java Annotations present the information in a standard and structured way so that it could be used amenably by processing tools.

Share Improve this answer Follow

edited Jan 11, 2021 at 19:34
Machavity ♦
**31.6k** ● 27 ● 95 ● 105

answered Nov 30, 2013 at 7:55
Krishan
**337** ● 4 ● 8

This is only *user defined annotations*; built in ones like @Override do nothing similar to this. – Sean Apr 13, 2022 at 23:10

0

[When do you use Java's @Override annotation and why?](When do you use Java's @Override annotation and why?)
The link refers to a question on when one should use the

override annotation(@override).. This might help understand the concept of annotation better.Check out.

Share Improve this answer Follow

edited May 23, 2017 at 12:02
Community Bot
1 ● 1

answered Apr 8, 2014 at 10:32
Koushik Balaji Venkatesan
115 ● 1 ● 8

---

0

Annotations when it comes to EJB is known as choosing Implicit middle-ware approach over an explicit middle-ware approach , when you use annotation you're customizing what you exactly need from the API for example you need to call transaction method for a bank transfer : without using annotation : the code will be

```
transfer(Account account1, Account account2, long amou
{
   // 1: Call middleware API to perform a security che
   // 2: Call middleware API to start a transaction
   // 3: Call middleware API to load rows from the dat
   // 4: Subtract the balance from one account, add to
   // 5: Call middleware API to store rows in the data
   // 6: Call middleware API to end the transaction
}
```

while using Annotation your code contains no cumbersome API calls to use the middle- ware services. The code is clean and focused on business logic

```
transfer(Account account1, Account account2, long amou
{
   // 1: Subtract the balance from one account, add to
}
```

Share Improve this answer

Follow

edited Jan 21, 2016 at 15:46

Chewpers
**2,440** ● 5 ● 23 ● 30

answered Jan 21, 2016 at 15:21

Harvester
**3** ● 2