**JAVADOC COMMENTS**

/\*\*

\* description of class/method/field/etc.

Whatever you put “/\*\*” and ends with “\*/” is going to be javadoc comment and directly copied to the javadoc documentation.

\*

\* @tag attributes

\* @tag attributes

\* ...

\* @tag attributes

\*/

• Javadoc comments: Special comment syntax for describing detailed specifications of Java classes and methods.

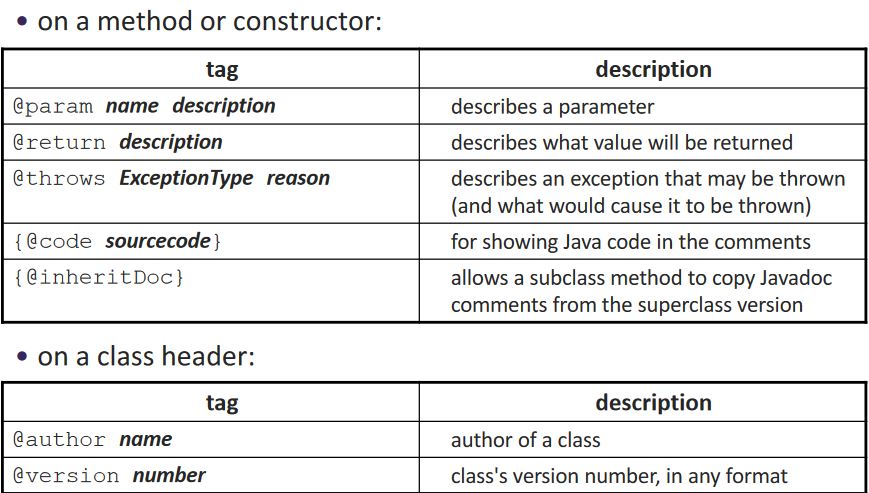
🡪 Put on all class headers, public methods, constructors, public fields, ...

• Main benefit: Tools can turn Javadoc comments into HTML spec pages.

🡪 Eclipse and other editors have useful built-in Javadoc support.

• Main drawback: Comments can become bulky and harder to read.

**JAVADOC TAGS**



**CHECK BankAccount.java**

**JAVADOC OUTPUT AS HTML**

• Java includes tools to convert Javadoc comments into web pages

🡪 from Terminal: javadoc -d doc/ \*.java OR javadoc myClass.java

🡪 Eclipse has this built in: Project → Generate Javadoc...

You can put your HTML tags in your code.

javadoc HTML example

From java.util.List interface source code:

/\*\*

\* Returns the element at the specified position

\* in this list.

\* <p>This method is <em>not</em> guaranteed to run

\* in constant time. In some implementations it may

\* run in time proportional to the element position.

\*

\* @param index index of element to return; must be

\* non-negative and less than size of this list

\* @return the element at the specified position

\* @throws IndexOutOfBoundsException if the index is

\* out of range

\* ({@code index < 0 || index >= this.size()})

\*/

public E get(int index);

Notice that HTML tags may be embedded inside the comments.

javadoc enums, constants

Each class constant or enumeration value can be commented:

/\*\*

\* An instrument section of a symphony orchestra.

\* @author John Williams

\*/

public enum OrchestraSection {

/\*\* Woodwinds, such as flute, clarinet, and oboe. \*/

WOODWIND,

/\*\* Brass instruments, such as trumpet. \*/

BRASS,

/\*\* Percussion instruments, such as cymbals. \*/

PERCUSSION,

/\*\* Stringed instruments, such as violin and cello. \*/

STRING;

}

What goes in @param/return

Don’t repeat yourself or write vacuous comments:

/\*\* Takes an index and element and adds the element there.

\* @param index index to use

\* @param element element to add

\*/

public boolean add(int index, E element) { ...

Better:

/\*\* Inserts the specified element at the specified

\* position in this list. Shifts the element currently at

\* that position (if any) and any subsequent elements to

\* the right (adds one to their indices). Returns whether

\* the add was successful.

\* @param index index at which the element is to be inserted

\* @param element element to be inserted at the given index

\* @return true if added successfully; false if not

\* @throws IndexOutOfBoundsException if index out of range

\* ({@code index < 0 || index > size()})

\*/

public boolean add(int index, E element) { ...

**Your javadoc is your spec:**

• Whenever you write a class to be used by clients, you should write full Javadoc comments for all of its public behavior.

🡪 This constitutes your specification to all clients for your class.

🡪 You can post the generated HTML files publicly for clients to view.

🡪 Common distribution of a library of classes:

• binaries (.class files, often packaged into an archive)

• specification (Javadoc .html files, or a public URL to view them)

🡪 Eclipse uses Javadoc for auto-completion.

• Effective Java Tip:

Write Javadoc comments for all exposed API elements. (anything that is non-private)

**javadoc and private**

Private internal methods do not need Javadoc comments:

/\*\* ... a Javadoc comment ... \*/

public void remove(int index) { ... }

// Helper does the real work of removing

// the item at the given index.

private void removeHelper(int index) {

for (int i = index; i < size - 1; i++) {

elementData[i] = elementData[i + 1];

}

elementData[size - 1] = 0;

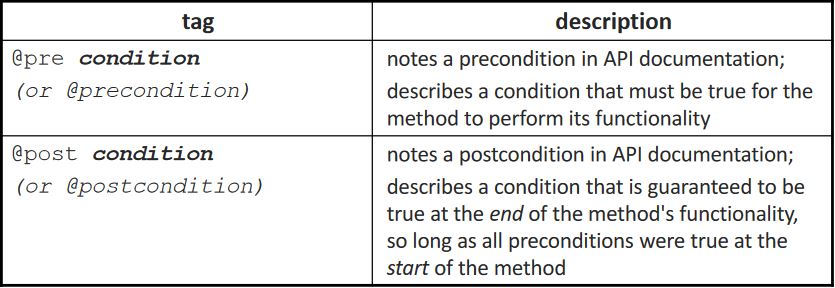
size--;

}

Private members don’t appear in the generate HTML pages.

**Custom javadoc tags**

Javadoc doesn’t have tags for pre/post, but you can add them:



By default, these tags won’t show up in the generated HTML. But …

**Applying Custom javadoc Tags**

from Terminal:

javadoc -d doc/

-tag pre:cm:"Precondition:"

-tag post:cm:"Postcondition:" \*.java

The generated JAVA API web pages will now be able to display pre and post tags properly!