Javadoc Tutorial

Introduction

- Javadoc is a tool that generates html documentation (similar to the reference pages at java.sun.com) from Javadoc comments in the code. In this tutorial we will go over how to write basic Javadoc comments and how to use features of Javadoc to generate html documentation.

Javadoc Comments

- Javadoc recognizes special comments /** .... */ which are highlighted blue by default in Eclipse (regular comments // and /* ... */ are highlighted green).
- Javadoc allows you to attach descriptions to classes, constructors, fields, interfaces and methods in the generated html documentation by placing Javadoc comments directly before their declaration statements.
- Here's an example using Javadoc comments to describe a class, a field and a constructor:

```
/** Class Description of MyClass */
public class MyClass
{
    /** Field Description of myIntField */
    public int myIntField;

    /** Constructor Description of MyClass() */
    public MyClass()
    {
            // Do something ...
    }
}
```

Javadoc Tags

- ***Tags*** are keywords recognized by Javadoc which define the type of information that follows.
- Tags come after the description (separated by a new line).
- Here are some common pre-defined tags:
    - ***@author [author name]*** - identifies author(s) of a class or interface.
    - ***@version [version]*** - version info of a class or interface.
    - ***@param [argument name] [argument description]*** - describes an argument of method or constructor.
    - ***@return [description of return]*** - describes data returned by method (unnecessary for constructors and void methods).

- ***@exception [exception thrown] [exception description]*** - describes exception thrown by method.
- ***@throws [exception thrown] [exception description]*** - same as ***@exception***.

Here's an example with tags:

```
/** Description of JavaDoc sample MyClass
*
* @author John Doe
* @author Jane Doe
* @version 6.0z Build 9000 Jan 3, 2018.
*/
public class MyClass
{
    /** Description of myIntField */
    public int myIntField;

    /** Description of the constructor MyClass()
    *
    * @throws MyException Description of myException
    */
    public MyClass() throws myException
    {
            // some statements...
    }

    /** Description of myMethod(int a, String b)
    *
    * @param a Description of a
    * @param b Description of b
    * @return Description of c
    */
    public Object myMethod(int a, String b)
    {
            Object c;
            // some statements...
            return c;
    }
}
```

Javadoc Compilation

- To generate the html documentation, run Javadoc followed by the list of source files, which the documentation is to be generated for, in the command prompt (i.e. ***Javadoc [files]***).
- Javadoc also provides additional options which can be entered as switches following the Javadoc command (i.e. ***Javadoc [options] [files]***).
- Here are some basic Javadoc options:
    - ***-author*** - generated documentation will include a author section
    - ***-classpath [path]*** - specifies path to search for referenced .class files.
    - ***-classpathlist [path];[path];...;[path]*** - specifies a list locations (separated by ";") to search for referenced .class files.
    - ***-d [path]*** - specifies where generated documentation will be saved.
    - ***-private*** - generated documentation will include private fields and methods (only public and protected ones are included by default).
    - ***-sourcepath [path]*** - specifies path to search for .java files to generate documentation form.
    - ***-sourcepathlist [path];[path];...;[path]*** - specifies a list locations (separated by ";") to search for .java files to generate documentation form.
    - ***-version*** - generated documentation will include a version section
- Some examples
    - Basic example that generates and saves documentation to the current directory (c:\MyWork) from A.java and B.java in current directory and all .java files in c:\OtherWork\.
        - *c:\MyWork>* ***Javadoc A.java B.java c:\OtherWork*.java***
    - More complex example with the generated documentation showing version information and private members from all .java files in c:\MySource\ and c:\YourSource\ which references files in c:\MyLib and saves it to c:\MyDoc.
        - *c:\>* ***Javadoc -version -private -d c:\MyDoc -sourcepathlist c:\MySource;c:\YourSource\ -classpath c:\MyLib***

Javadoc In Eclipse

- Eclipse can generate Javadoc comments for classes and methods.
    1. Place the cursor in the text of class of method declaration.
    2. Right Click->Source->Add Javadoc Comment.
    3. Javadoc comments with appropriate tags are generated, but you still have to write the descriptions.
- Eclipse can also compile Javadocs for projects/packages/classes in the workspace.
    - Set location of Javadoc command and export your project/package/class as a Javadoc:

1. ***File->Export***.
2. Select ***Javadoc*** and enter the path of ***Javadoc.exe***, i.e. ***[Path of Java SDK]\bin\javadoc.exe*** (e.g. ***c:\j2sdk1.5.0\bin\javadoc.exe***).
3. Also choose your export destination and click ***Next***.
4. In the ***Generate Javadoc Windo***w, select the project/package(s)/class(es) you want to compile Javadocs for, select the visibility, and enter the path of the destination folder.
5. Click ***Finish***.

Javadoc References

- Javadoc is a powerful tool with many more features, for more information check out the following links:
  - http://www.oracle.com/technetwork/java/javase/documentation/javadoc-137458.html
  - http://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html
  - https://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#exampleresult