Creating Javadocs

EHSDI

eBuzima

What Are Javadocs?

- Javadoc is a tool that generates html documentation (similar to the reference pages at java.sun.com) from Javadoc comments in the code.

Javadoc Comments

- Javadoc allows you to attach descriptions to:
  - Classes
  - Constructors
  - Methods
  - Fields
  - Interfaces
- Javadoc processes comments:/** ……..*/
- The first sentence should be descriptive.
  - Use /** * This is Dr. Phil's class. */

Javadoc Simple Example

```
/** 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.
- 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***.

Javadoc Example

```
/**
 * Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute {@link URL}. The name
 * argument is a specifier that is relative to the url argument.
 * <p>
 * This method always returns immediately, whether or not the
 * image exists. When this applet attempts to draw the image on
 * the screen, the data will be loaded. The graphics primitives
 * that draw the image will incrementally paint on the screen.
 *
 * @param  url  an absolute URL giving the base location of the image
 * @param  name the location of the image, relative to the url argument
 * @return      the image at the specified URL
 * @see         Image
 */
public Image getImage(URL url, String name) {
        try {
            return getImage(new URL(url, name));
        } catch (MalformedURLException e) {
            return null;
        }
}
```

Source: http://java.sun.com/j2se/javadoc/writingdoccomments/

Style Tips

- **Use <code> for keywords and names.** Keywords and names are offset by <code>...</code> when mentioned in a description. This includes:
  - Java keywords
  - package names
  - class names
  - method names
  - interface names
  - field names
  - argument names
  - code examples

Style Tips Cont'

- Use in-line links economically
- Omit parentheses for the general form of methods and constructors.
  - The add method enables you to insert items. (preferred)
  - The add() method enables you to insert items. (avoid)
- Use phrases instead of complete sentences,in the interests of brevity.

Language

- **Use 3rd person (descriptive) not 2nd person (prescriptive).**The description is in 3rd person declarative rather than 2nd person imperative.
  - Gets the label. (preferred)
  - Get the label. (avoid)

- **Method descriptions begin with a verb phrase.** A method implements an operation, so it usually starts with a verb phrase:
  - Gets the label of this button. (preferred)
  - This method gets the label of this button. (avoid)

Source: http://java.sun.com/j2se/javadoc/writingdoccomments/

Tag Order

- **Order of Tags**
  Include tags in the following order:

  - @author (classes and interfaces only, required)
  - @version (classes and interfaces only, required.)
  - @param (methods and constructors only)
  - @return (methods only)
  - @exception (@throws added in Javadoc 1.2)
  - @see
  - @since
  - @serial (or @serialField or @serialData)
  - @deprecated

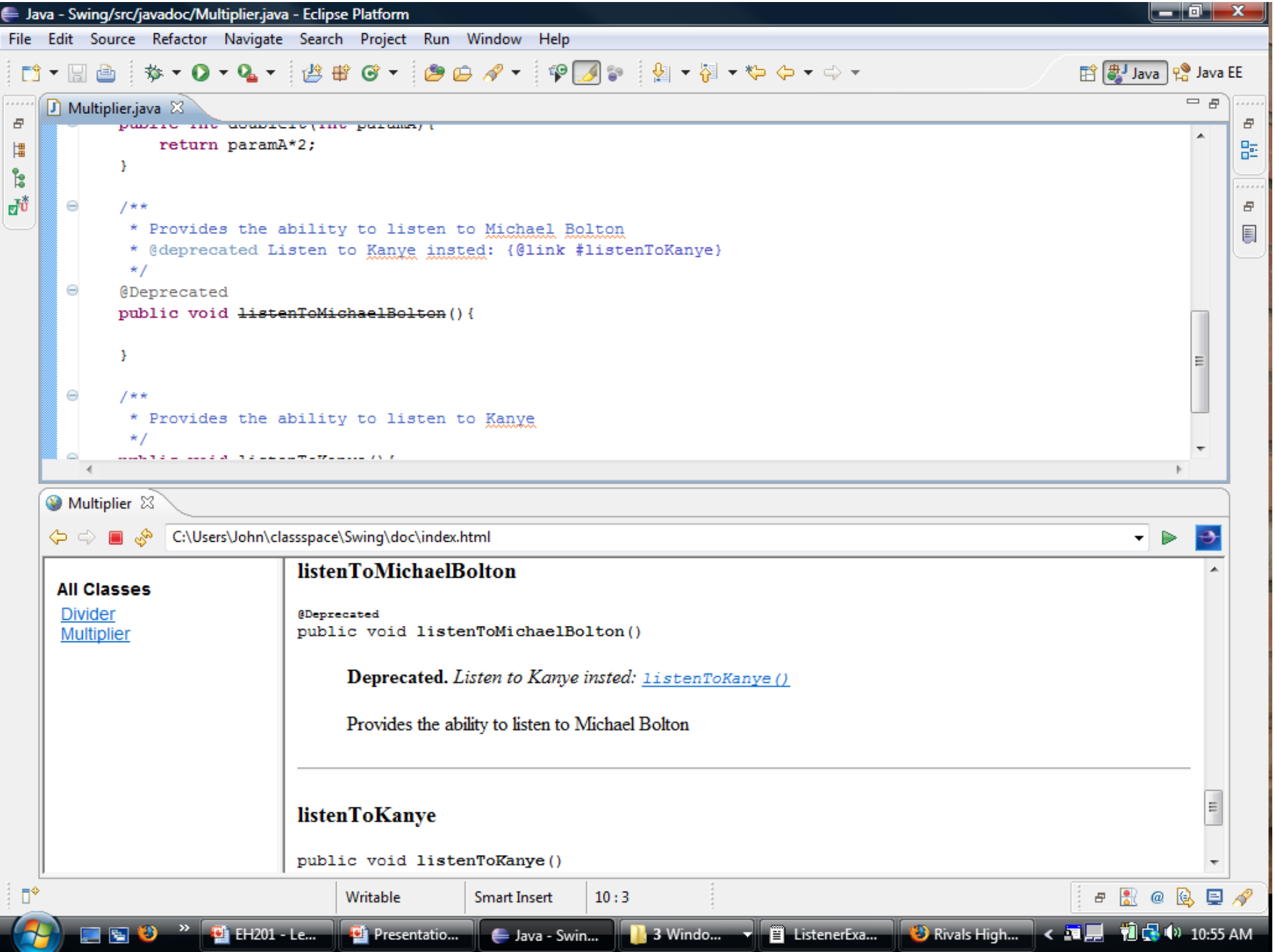
Try this! (Or something similar)

- Make two classes with javadoc using:
  - Class documentation
  - {@link …}
  - <code></code>
  - @param
  - @return
  - @version
  - @see
  - @deprecated

```
/**
 * Provides multiplying operations. Methods take parameters which are
 * multiplied, ensuing values are returned.
 * @author Johnny Dizzle
 *
 * @version 1.0
 *
 */
public class Multiplier {
    /**
     * Doubles the value of <code>paramA</code>.
     * Uses the formula <code> paramA * 2 </code> to
     * double the value of the parameter.
     *
     * For dividing operations, see the <code>Divider</code> class :
     * {@link javadoc.Divider}
     *
     * @param paramA the parameter to double
     * @return twice the value of the parameter
     * @see javadoc.Divider
     */
    public int doubleIt(int paramA){
        return paramA*2;
    }
```

Javadoc Compilation

- In Eclipse, go to the Project Menu. Select Generate Javadoc…
- Browse for the Javadoc command. This should be something like:
  - C:\Program Files\Java\jdk1.6.0_10\bin\javadoc.exe


Java - Swing/src/javadoc/Multiplier.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
Multiplier.java
return paramA*2;
}
/**
* Provides the ability to listen to Michael Bolton
* @deprecated Listen to Kanye insted: {@link #listenToKanye}
*/
@Deprecated
public void listenToMichaelBolton(){
}
/**
* Provides the ability to listen to Kanye
*/
Multiplier
C:\Users\John\classspace\Swing\doc\index.html
All Classes
Divider
Multiplier
listenToMichaelBolton
@Deprecated
public void listenToMichaelBolton()
Deprecated. Listen to Kanye insted: listenToKanye()
Provides the ability to listen to Michael Bolton
listenToKanye
public void listenToKanye()
Writable
Smart Insert
10 : 3
EH201 - Le...
Presentatio...
Java - Swin...
3 Windo...
ListenerExa...
Rivals High...
10:55 AM

Resources

- http://www.eclipse-blog.org/eclipse-ide/generating-javadoc-in-eclipse-ide.html
- http://java.sun.com/j2se/javadoc/writingdoccomments/
- http://www.mcs.csueastbay.edu/~billard/se/cs3340/ex7/javadoctutorial.html