Comments, Documentation?

- Most programmers don't read documentation
  - Instead they prefer to work with the code.
- Programmers often look for sample code
  - Well-written unit tests can provide such a working specification of your code.
  - Unit tests effectively become a significant portion of your technical documentation.

Why Document Your Code?

- Just because your code compiles doesn't mean you can read it and understand it.
- Machines don't read comments or documentation; the developers who have to maintain your code do.
- Your code does *not fully* "self document."
- Because it's part of your grade.

Where To Document (at minimum)

- At the beginning of every class.
  - Tell where the class fits in relation to the system.
  - Justify its existence.
  - Explain any special considerations, (new and exciting data structures… etc).
- At the beginning of every method.
  - Tell what it does.
  - Why it is a method.
  - If there are any side effects of your method.
- Inside your code.
  - When you use a nonstandard method of problem solving. (read: hack)
  - Whenever your code could confuse you when you update it.

Javadoc

- Javadoc is a program that generates HTML documentation, using comments you embed in code.
- The comments must be in a standardized format.
- Javadoc provides an easy way to generate impressive web pages of documentation of your code.
- Javadoc is available to you in Eclipse as well as being a standalone program (written, of course, in Java).

Javadoc Comment Format

- Javadoc comments have this general form (note the two opening asterisks).

```
/**
 * Comments go here.
 * Tags optionally go here.
 */
```

- First sentence: brief summary (appears in method list)
- After that: more detailed description (also appears in the method list)
- At end: special tags for parameters, return values, author, version, exceptions

Javadoc Tags

- Special comment blocks are tagged as documentation comments.
- These comment blocks are then run through a javadoc utility.
- Special tags are then read which are turned into html fields for presentation.
- A complete list of tags is available at:
  - http://java.sun.com/j2se/javadoc/writingdoccomments/

Common Tags

- `@param`
  - Used in classes, constructors, methods and interfaces.
  - Used to designate what input parameters a method, constructor, etc., is using.

```
/**
* @param row the row you wish to get the value
*                  from
* @param col the column you wish to get the value
*                  from
*/
public int getAt(int row, int col)
```

Common Tags 2

- *`@return`*
  - Used in classes, constructors, methods and interfaces.
  - Used to designate what input parameters a method, constructor… etc, is returning.

```
/**
* @return     the value at row/col
*/
public int getAt(int row, int col)
```

Common Tags 3

- *`@throws`* (also *`@exception`*)
  - Used in classes, constructors, methods and interfaces.
  - Used to designate what type of exceptions may be thrown and why.

```
/**
* @throws    NullPointerException when row or col
*             are out of bounds
*/
public int getAt(int row, int col) throws
  NullPointerException
```

Other Useful Tags

- *`@author name`*
  - Should be used for classes and interfaces
- *`@version text`*
  - Often used in conjunction with version control systems like CVS and SVN and git
- *`@see reference`*
  - Adds a “See Also:” entry

Javadoc In Eclipse

- Eclipse automates much of the javadoc experience.
  - Typing /** will cause a javadoc comment to be started.
  - Highlighting a class, or method declaration and pressing alt-shift-j will cause Eclipse to create a comment with the appropriate tags.
  - Finally, Eclipse will interface with a javadoc utility to create the html and output it to the correct folder.