

Commenting and javadoc

Reference:

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

javadoc Comments

- Must precede a class, interface, field, constructor or method declaration.
 - put anywhere else will be ignored by the javadoc tool... or perhaps confuse it...
- start with `/**` (not `/*`)
- Sun suggests each subsequent line starts with `'*` (but this is not required).

Example

```
/**  
 * adds two integers using  
 * the plus (+) operator.  
 */  
public int add(int x, int y) {  
    ...  
}
```

Description and *tags*

- Each comment has two parts
 - description (text description of the class, method, constructor or field).
 - block tags that tell javadoc that what follows describes a parameter, return value, author, version, etc.
 - block tags start with @

Some tags

@param: describe one parameter to a method or constructor.

@return: describe the return value of a method.

@version: user defined version name/number

@author: take a wild guess.

More tags

@exception, @throws: describe/list exceptions that can be generated by a method.

@see: list where additional information can be found (link to another page, etc.)

There are more. You can also create custom tags (you need to tell javadoc how to handle them...).

HTML

- The text in javadoc comments can include HTML markup.
 - if you don't know HTML, don't worry – you don't need to learn it
- Keep in mind that the result of running the javadoc tool is a bunch of HTML documents...

The First Sentence

- The first sentence should be a concise summary.
 - javadoc treats the first sentence as special.
- The first sentence ends with a period!

Class comment guidelines

- Each class (or interface) should have a comment that describes what the class does (represents).
- include `@author` tag (and possibly `@version`).
- Don't describe methods (they each have their own comments!).

Field comments

- Each *public* field should have a comment that describes what the field is for.
- Sometimes a list of valid values is relevant (for example, see the API docs for a class and look at the Field Summary).
 - javadoc will automatically generate info about the data type.

Method/Constructor comments

- Description of what the method does.
- One `@param` tag for each parameter
 - describing the parameter, see examples...
- `@return` tag describing the return value.
- `@exception` or `@throws` tag for each type of exception thrown (if not obvious).

Class Example

```
/**
 * Class Grader defines an automated
 * grading object that can grade homework
 * assignments.
 * @author Dave H.
 * @version 1/1/2006-1.043-a176
 * @see CheaterCatcher
 */
public class Grader {
    ...
}
```

Field Examples

```
/**
 * cohort holds the name of the cheating
 * partner (if any is found).
 */
public String cohort;

/**
 * similarity ranges between 0 and 1, with
 * 1 meaning identical
 */
public double similarity;
```

Method Example

```
/**
 * extraCredit records any additional
 * homework points for a student.
 * <b>This method should never be
 *   called.</b>
 *
 * @param stu The student.
 * @param points Additional points.
 * @return total points for the student.
 */
public int extraCredit(Student stu, int
    points) { ...
```

Generating HTML

- The program javadoc will read source code and generate HTML files.

```
javadoc *.java
```

Homework Requirements

- You must include javadoc comments
 - for each class, interface, method, constructor and public field.
 - Methods require @param and @return tags.
- You don't need to submit javadoc generated HTML files.
 - we will do this.