Javadoc

What is Javadoc?

- Utility that parses code documentation from Java source files and produces a document, typically in HTML.
- Output format can be customized.

What can be documented?

- Packages
- Classes
- Fields
- Methods

General Syntax

- Start with /** and end with */ .

```
/**
 * 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;
        }
 }
```

General Syntax: useful HTML tags

- Comments can contain some HTML tags
  - Paragraph

    <p>
  - Line break

    text **
**
  - Pre-formatted text

    <pre> text **</pre>**
  - Lists

    <ul>

    **
Item 1
**

    **
item 2
**

    **
**

General Syntax: documenting classes

- Put Javadoc before the class declaration.
- Recommended tags:
  - @author: who wrote the class or interface
  - @see: other related class, package or interface
  - @version: current version of the class
  - @deprecated: indicates that the class is no longer in use

General Syntax: documenting methods

- Put Javadoc before the method declaration.
- Recommended tags:
  - @param: parameters of the method
  - @return: return value of the method
  - @throws: exceptions the method may throw
  - @see: other related class, package or interface

General Syntax: documenting methods

- What to document:
  - Expected behavior: expected or desired behavior of this operation.
  - State transitions: specify what state transitions this operation may trigger.
  - Range of valid argument values
  - Null argument values - for each reference type argument, specify the behavior when null is passed in.
  - Range of return values - specify the range of possible return values, including where the return value may be null.
  - Exception - when exception may be thrown.

General Syntax: summary

- Tags start with @ and should follow this order:
  - @author: who wrote the class or interface (classes and interfaces only, required)
  - @version: current version of the class (classes and interfaces only, required)
  - @param: parameters of the method (methods and constructors only)
  - @return: return value of the method (methods only)
  - @throws: exceptions the method may throw
  - @see: other related class, package or interface
  - @since: when the code was introduced (e.g. version of the file when you added a method)
  - @deprecated: indicates that the class is no longer in use, describe deprecated item and what alternative to use instead