COP 3330, Spring 2013

The Javadoc tool

Instructor : Arup Ghosh

04-12-13

School of Electrical Engineering and Computer Science
University of Central Florida

Javadoc

- There is a program included in the standard Java installation that will generate documentation for your Java code.
  - `javadoc.exe` in the bin directory, where `javac.exe` and `java.exe` are also located.

- It processes source code, extracts Javadoc comments and builds a series of HTML documents containing all the formatted documentation.

- Quite usefully, they are in the same standard format that you've seen for the classes in the Java standard library.

Comments

- So far, you've seen the two types of comments Java borrowed from C.
  - // Line comments
  - /* Block comments */
- Java has a third type of comment, which we sometimes call the Javadoc comment.
  - /** Javadoc comment */
- The text of the comment can be automatically converted into documentation for elements of the program.
- The position of the comment decides which element the documentation in it refers to.

Writing Javadoc comments

- You can document classes, methods, and even fields.

- Place the Javadoc comment **immediately before** the element you wish to document.

```
/**
 * This is a silly method with useless
 * documentation.
 */
public String sillyMethod()  { … }
```

Javadoc comment structure

- The general structure of a comment is:

```
/**
 * [Description]
 * [Tags]
 */
```

- The description is just HTML, so you can use HTML formatting markup if you wish (But it is not required).

- Tags are a Javadoc-specific form of describing certain special features of the document.

Javadoc tags

- Tags describe important things that the documentation will draw attention to, usually by formatting them in some special way.

- For example:
  - Describing method parameters
    - `@param parameter_name description`
  - Describing the return value of a method
    - `@return description`
  - Which exceptions does it throw
    - `@throws exceptionType description`

Others

- **@author** *description* (For classes only)
- **@version** *versionNumber* (For classes only)
- **@see** package.class#member *label*
  - Used to refer to methods or fields in the documentation of another class.
- @**since** *versionNumber*
- @**deprecated** *description*
  - Used to indicate that this method may be removed in future versions.

Example

```
/**
 * This method divides the first parameter by
 * the second and returns the truncated result.
 *
 * @param num The numerator of the division
 * @param denom The denominator of the division
 * @return The result of the division
 * @throws ArithmeticException if denom is zero
 */
public static int divide(int num, int denom)
throws ArithmeticException
{...}
```

Description

```
/**
 * This method divides the first parameter by
 * the second and returns the truncated result.
 *
 * @param num The numerator of the division
 * @param denom The denominator of the division
 * @return The result of the division
 * @throws ArithmeticException if denom is zero
 */
public static int divide(int num, int denom)
throws ArithmeticException
{...}
```

Parameter description

```
/**
 * This method divides the first parameter by
 * the second and returns the truncated result.
 *
 * @param num The numerator of the division
 * @param denom The denominator of the division
 * @return The result of the division
 * @throws ArithmeticException if denom is zero
 */
public static int divide(int num, int denom)
throws ArithmeticException
{...}
```

Return value

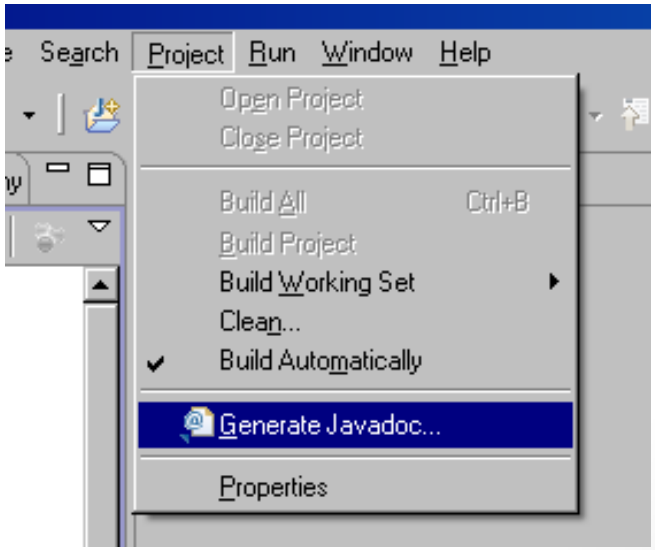
```
/**
 * This method divides the first parameter by
 * the second and returns the truncated result.
 *
 * @param num The numerator of the division
 * @param denom The denominator of the division
 * @return The result of the division
 * @throws ArithmeticException if denom is zero
 */
public static int divide(int num, int denom)
throws ArithmeticException
{...}
```

Thrown Exceptions

```
/**
 * This method divides the first parameter by
 * the second and returns the truncated result.
 *
 * @param num The numerator of the division
 * @param denom The denominator of the division
 * @return The result of the division
 * @throws ArithmeticException if denom is zero
 */
public static int divide(int num, int denom)
throws ArithmeticException
{...}
```

Generating Javadoc

- Eclipse will help generate and format Javadoc comments quite neatly.
  - After writing your comments, you can use `Project > Generate Javadoc` to build the documentation.
  - By default, this places documentation in a `doc` folder in the project directory.



Command line

- As with javac and java, you must be set up for command line compilation.
- Simply navigate to the project directory, and type ‘javadoc’ followed by the name of the .java file or package you want to build documentation for.

- There are lots of command-line options to control the output. Type ‘javadoc’ followed by nothing to get a full list of them.

- Reference:
  - http://docs.oracle.com/javase/1.5.0/docs/tooldocs/windows/javadoc.html