Javadoc

Angel Gruev
Dreamix ltd.

Outline

- Motivation
- Writing Javadoc comments
- Running the javadoc tool

Motivation

- Why document programs?
  - To make it easy to understand, e.g., for reuse and maintenance
- What to document?
  - Semantics
  - Internal working

Motivation (Cont.)

- Why Javadoc?
  - To combine source code with documentation and other reference materials
  - To make it easier to keep the documentation and code in sync
  - To generate API specifications (or interface specifications) from source code

Approach

- Javadoc comments
  - Attach special comments, called *documentation comment* (or *doc comment*) to classes, fields, and methods.

    /** … */
- Javadoc tool
  - Use a tool, called *javadoc*, to automatically generate HTML pages from source code.
  - MIF Doclet - for Generating FrameMaker and PDF Documents

Javadoc Example

```
/** An abstract class representing various kinds of
shapes. */
public abstract class Shape {
  /** The x-coordinate of this shape. */
  private int x;
  // …

  /** Returns the x-coordinate of this shape. */
  public int getX() { … }

  /** Set the x-coordinate of this shape to the
    * argument
    * <code>x</code>.
    */
  public void setX(int x) { … }

  // …
}
```

Javadoc Tags

- Javadoc Tags
  - Special keyword recognized by javadoc tool.
  - Will be specially formatted
- Common Tags
  - @author — Author of the feature
  - @version — Current version number
  - @since — Since when
  - @param — Meaning of parameter
  - @return — Meaning of return value
  - @throws — Meaning of exception
  - @see — Link to other features

Example

```
/** An abstract class representing various
 * kinds of windows.
 *
 *  @author Bill Gates
 *  @version 1.0 (01/22/88)
 *  @since version 3.11
 */
public abstract class Windows {
   // …
}
```

Specifying Parameters and Return Value

- **Syntax**
  - **@param *name description***
  - **@return *description***
  - **@throws *exception description***

- **Example**

```
/** Returns the definition of a given word in this
     dictionary.
 *
 *   @param word a word whose definition is being looked
     up.
 *   @return the definition of the word; null if no definition
     is
 *       found.
 *   @throws NullPointerException if the word is null.
 */
public String lookup(String word) { /* ... */ }
```

Linking to Other Features

- Syntax
  - @see *featureName*

  where *featureName* is class, field, or method.
- Example

  @see Dictionary

  @see #elems

  @see #lookup(String)

  @see SpanishDictionary#lookup(String)

  @see cs3331.Score#lookup(String)

Linking to Other Features (Cont.)

- In-line link
  - Used to refer features in a sentence.
  - Syntax: {@link *featureName*}

    where *featureName* is class, field, or method.
- Example

```
/** Returns the definition of a given word in this
     dictionary. This
 *   method is overridden here from the class {@link
     Dictionary}
 *   to implementSpanish-specific, efficient lookup
     algorithm.
 *
 *   @see Dictionary#lookup(String)
 *   ….
 */
public String lookup(String word) { /* … */ }
```

Principles

- **The doc comment for a method should describe succinctly the contract between the method and its client.**
  - the contract should say *what the method does rather than how it does its job*
  - method's *preconditions and postcoditions*

Specifications

- The "precondition": constraints that hold before the method is called (if not, all bets are off)
  - **requires** – spells out any obligations on client
- The "postcondition": constraints that hold after the method is called (if the precondition held)
  - **modifies** – lists objects that may be affected by method; any object not listed is guaranteed to be untouched
  - **throws** – lists possible exceptions
  - **effects** – gives guarantees on the final state of modified objects
  - **returns** – describes return value

Example

```
public static int test(List lst, Object oldelt,
                       Object newelt)
```

requires lst, oldelt and newelt are non null. oldelt occurs in lst
modifies lst
effects change the first occurrence of oldelt in lst to newelt & makes no other changes to lst
returns an i such that ith element of lst was oldelt

Example (2)

```
public static List<Integer> listAdd(
                            List<Integer> lst1,
                            List<Integer> lst2
                            )
```

requires lst1 and lst2 are not null. lst1 and lst2 are the same size
modifies none
effects none
returns a list of same size where the ith element is the sum of the ith elements of lst1 and lst2

Example (javadoc)

```
/**
 * Returns the element at the specified position in this list.
 *
 * @param index
 *             index of element to return; must be nonnegative
 *              and less than the size of this list.
 * @return the element at the specified position in this list.
 * @throws IndexOutOfBoundsException
 *              if the index is out of range (<tt>index &lt; 0
 *              || index &gt;= this.size()</tt>).
 */
Object get(int index)
```

Bad Example

```
/**
   * Sets the tool tip text.
   *
   * @param text  the text of the tool tip
   */
   public void setToolTipText(String text) {
```

Preferred

```
/**
 * Registers the text to display in a tool tip. The
 * text displays when the cursor lingers over the
 * component.
 *
 * @param text
 *            the string to display. If the text is
 * null, the tool tip is turned off for this component.
 */
 public void setToolTipText(String text) {
```

HTML nature

- Javadoc uses HTML metacharacters and tags
- Commonly used tags and metachars:
  - <p>, <code>, <tt>
  - HTML metachars must be escaped
  - < > &

Running javadoc Tool

- Similar to running javac, e.g.,
  - javadoc A.java
  - javadoc A.java B.java
  - javadoc *.java
  - javadoc –d javadocs *.java

Running javadoc (Cont.)

- Other options, refer to JDK tool docs, e.g.,
  - http://java.sun.com/j2se/1.4.1/docs/tooldocs/tools.html or
  - http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html

Other Resources

- Many on-line tips, guidelines, and other documents, e.g.,
  - "How to Write Doc Comments for the Javadoc™ Tool" available at http://java.sun.com/j2se/javadoc/writingdoccuments/index.html

DREAMIX
Software Development

Questions?