TUM

Creating an Object Design Document with Javadoc

Thomas Funck

Why Object Design Documents?

- All elements (subsystems, classes, members), even private ones, have to be documented
  - – to facilitate communication among developers
  - – to make it easier to change or extend the code
  - – to support maintenance

Object Design Document (ODD)

- Detailed description of the implementation of a system
- Structure
  - Description of subsystems
  - Description of classes
  - Description of operations (methods)
  - Description of attributes (fields)
  - Class diagrams

Description of subsystems

- General description of what the subsystem does
- Description of the most important classes of the subsystem
  => How to use the subsystem from within other subsystems
- In Java usually organized in packages

Description of classes

- What is the class used for
- Which class does this class extend
- Which interfaces does it implement
- How can the class be used
  - description of its main functionality (what are the most important methods)
  - which classes use or should use this class

Description of operations

- An operation can be implemented by more than one method (method overloading)

  => With Javadoc only methods can be documented

- Description of methods
  - which functionality will be invoked
  - what types have the parameters to be of and what is their meaning
  - which result will be returned
  - which errors could occur
  - does a method override a superclass‘ method?

Description of attributes

- Description of all attributes a class uses
  - What kind of data is stored
  - What is the data used for
  - How can the data be accessed
  - Static attributes: which methods use these attr.
- Example:
  - Field: **`Hashtable students;`**
  - Comment: **`Stores a string of each student‘s name of the TUM. The Hashtable maps the string of the student‘s Matrikelmummer to his name.`**

Class diagrams

- Class diagrams graphically show
  - the inheritance hierachy
  - the relationship between classes (which class is used by another class)
- Should be created for the whole system and/or subsystems

The Javadoc Tool

Javadoc

- Creates a HTML documentation out of Java source files that contains
  - a detailed description of all elements (packages, classes, attributes, methods)
  - a tree of the class hierachy
  - an index for all elements
- Extracts special formatted comments from source files

  ```
  /** A Javadoc comment */
  ```

Using Javadoc comments

- Are typed directly before the element to document

```
/** Javadoc Comment for this class */
public class MyClass {

  /** Javadoc Comment for field text */
  String text;

  /** Javadoc Comment for method setText */
  void setText(String t) {...}
}
```

Javadoc comments

- Special tags for classes
  - @author
  - @version
- Special tags for methods
  - @param
  - @return
  - @exception
- Reference to another element
  - @see
- Can contain **any** HTML code

Writing Javadoc comments

```
/**
 * Computes the square root for the
 * specified double value.
 * @param val the value to compute the
 *            square root for
 * @return the square root of
 *         <TT>val</TT>
 * @exception IllegalArgumentException if
 *            <TT>val</TT> is < 0
 * @see #sqrt(int)
 */
public double sqrt(double val) {
     //...
}
```

Writing Javadoc comments (II)

- The first sentence is used as a kind of short description
  - therefore it should describe the meaning of the element
  - is shown at index entries
- Following the hyperlink of an element will show you the whole documentation of this element

Tips for writing comments

- Usually it is more usuful to write a detailed class description than writing long comments for each method,
especially when methods and parameters have intuitive names.

Creating an ODD with Javadoc

- Description of operations (methods)
  - Type a Javadoc comment for any method
- Description of classes
  - Type Javadoc comments for any class and all of their fields
- Description of subsystems
  - Create a HTML-file named „package.html“ that describes a package. Put it into the package directory. Do these steps for any package.

Creating an ODD with Javadoc (II)

- Run Javadoc
  - Syntax
    ```
    javadoc [options] [packagenames] [sourcefiles]
    ```
  - Example
    ```
    javadoc -private -author -d doc
      -sourcepath „c:\stars3\“
      stars.ui
      stars.communication
      stars.communication.client
    ```
  - Type **`javadoc -help`** to see all possible options

Further Information

- More information about writing Javadoc comments at **`http://java.sun.com/j2se/javadoc/writingdoccomments/index.html`**

  at the Javadoc Homepage **`http://java.sun.com/j2se/javadoc/index.html`**

Demonstration