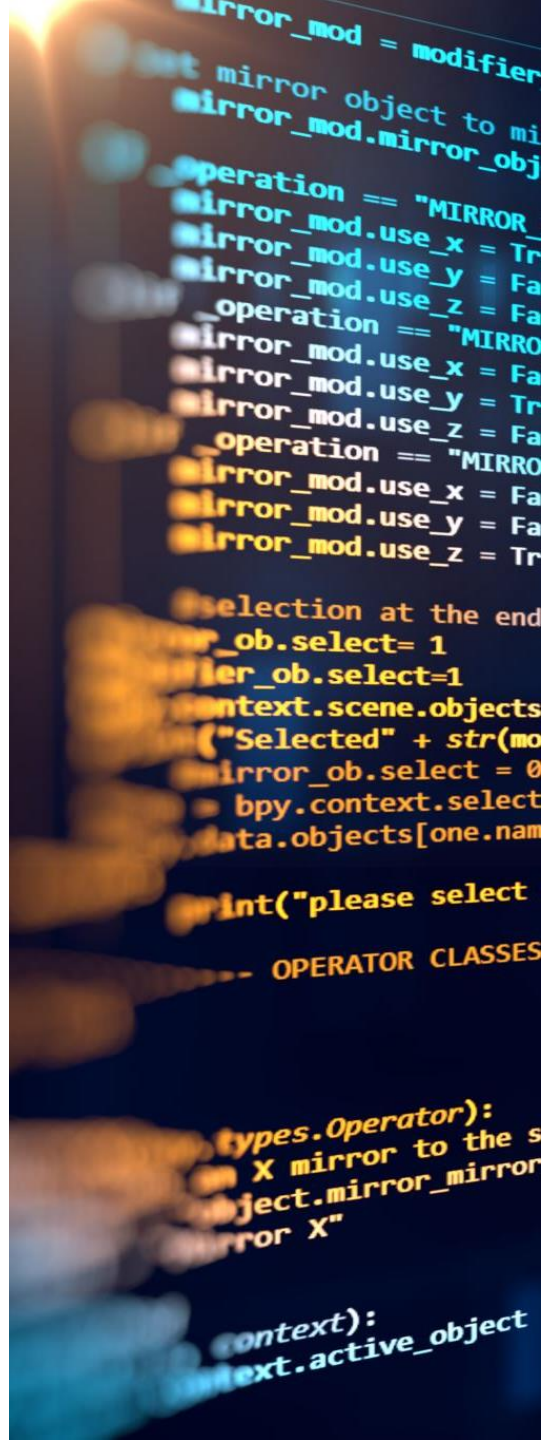
CS 1027
Fundamentals of Computer Science II

Javadoc Review

Ahmed Ibrahim



Agenda

- What is Javadoc?
- Javadoc Comments
- Javadoc Tags
- Generating Javadoc Documentation
- References

What is Javadoc?

- JavaDoc is a software tool developed by Sun Microsystems for generating API documentation format from Java source code augmented with special tags in the code's comments.
- Javadoc is an industry standard for documenting Java classes.
- How does JavaDoc work?
  - Instead of writing and maintaining separate documentation, the programmer writes formatted comments in the Java code itself.
  - Javadoc takes these comments and transforms them into documentation.

Javadoc Tools

- Many such systems exist that can be used for various programming languages: Javadoc, Doxygen, … and more.
- These systems offer a variety of output formats, catering to different user preferences and needs. You can choose from HTML, RTF, PDF, LaTeX, … and more.
- HTML has many advantages—portable, browsable, adaptable

Why Javadoc?

Advantages

- Program documentation process is coupled with the programming process.
- Automated generation of documentation: less error-prone
- Ease of generation of documentation
- Ease of update of documentation
- Short code-to-documentation cycle: all programmers can be made aware of others' developments almost in real time.
- Can generate highly browsable documentation, accessible electronically over the web (HTML)

Disadvantages

- There is a learning curve to learn how to use the tool, though it is minimal
- It requires dedication, or else the documentation will be obsolete or incomplete.

```java
/**
 * This is a simple class that simulates drawing a circle.
 * It prints out a message that represents drawing a circle.
 *
 * @author Student
 * @since 1.0.0
 */
public class SimpleCircleDrawer {

    // Method that simulates drawing a circle
    public void drawCircle() {
        System.out.println(x:"Drawing a circle...");
    }

    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        // Create an instance of SimpleCircleDrawer
        SimpleCircleDrawer drawer = new SimpleCircleDrawer();

        // Call the drawCircle method to "draw" a circle
        drawer.drawCircle();
    }
}
```

Javadoc Tags

Javadoc Comments

- A Javadoc comment begins with the **/**** marker and ends with the ***/** marker.
- All the lines in the middle start with an asterisk lined up under the first asterisk in the first line.

Javadoc Tages: on class header

tag	description
`@author` ***name***	author of a class
`@version` ***number***	class's version number, in any format
`@since` ***number***	Indicates when a class, method, or feature was introduced.

Example

```
2   /**
3    * <h1>SimpleCircleDrawer Class</h1>
4    * <p>
5    * This class simulates drawing a circle by printing a message.
6    * It is designed as a basic example for beginners learning Java.
7    * </p>
8    *
9    * <p><strong>Features:</strong></p>
10   * <ul>
11   *     <li>Defines a simple method to simulate drawing a circle.</li>
12   *     <li>Uses console output to represent the action of drawing.</li>
13   * </ul>
14   *
15   * <p>Below is an example usage:</p>
16   * <pre>
17   * SimpleCircleDrawer drawer = new SimpleCircleDrawer();
18   * drawer.drawCircle();
19   * </pre>
20   *
21   * <p>This class is part of a beginner's tutorial for learning Java classes and methods.</p>
22   *
23   * @author Student
24   * @version 1.0
25   * @since 1.0
26   */
```

Javadoc Tages: on a method or constructor

tag	description
`@param` ***name description***	describes a parameter
`@return` ***description***	describes what value will be returned
`@throws` ***ExceptionType reason***	describes an exception that may be thrown (and what would cause it to be thrown)
`{@code` ***sourcecode***`}`	for showing Java code in the comments
`{@inheritDoc}`	allows a subclass method to copy Javadoc comments from the superclass version

Example

```java
/**
 * The main method where the program starts.
 * It creates an instance of {@code SimpleCircleDrawer2}
 * and calls the {@code drawCircle()} method.
 *
 * <p>Example usage:</p>
 * <pre>
 * {@code
 * public static void main(String[] args) {
 *     SimpleCircleDrawer2 drawer = new SimpleCircleDrawer2();
 *     drawer.drawCircle();
 * }
 * }
 * </pre>
 *
 * @param args Command-line arguments (not used in this program).
 * @throws IllegalArgumentException If an invalid argument is passed.
 */
Run | Debug | Run main | Debug main
public static void main(String[] args) throws IllegalArgumentException {
    // Create an instance of SimpleCircleDrawer2
    SimpleCircleDrawer2 drawer = new SimpleCircleDrawer2();

    // Call the drawCircle method to "draw" a circle
    drawer.drawCircle();
}
```

Javadoc Comments (cont.)

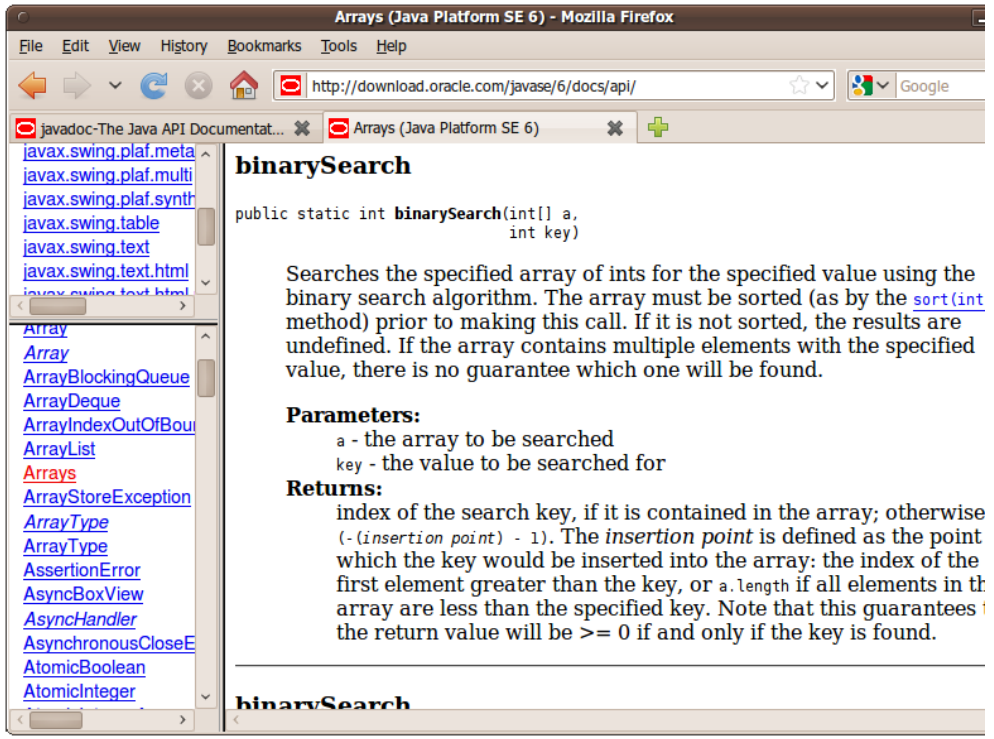
- For the Javadoc comments to be recognized as such by the **Javadoc** tool, they must appear **immediately** before the class, interface, constructor, method, or data member declarations.
- It will be ignored if you put the **Javadoc** comment for the class before the **import** statements.
- The first sentence is a "**summary sentence**". This should be a short description of the element described by the comment.

Oracle's Spec. Webpages

- Java has tools to convert Javadoc comments into web pages
  - from Terminal:

    ```
    javadoc -d doc/ *.java
    ```

  - Eclipse has this built in: Project → Generate Javadoc…
- The actual Java API spec web pages are generated from Oracle's Javadoc comments on their own source code:



HTML Tags

- Because **Javadoc** generates HTML files, any valid HTML can be embedded.
- A **Javadoc** comment may be composed of multiple lines,
- Another useful HTML marker is <code>, which we can use to include a sample code in a **Javadoc** comment.

Class-Level Javadoc

```
/**
 * <h1>SimpleCircleDrawer Class</h1>
 * <p>
 * This class simulates drawing a circle by printing a message.
 * It is designed as a basic example for beginners learning Java.
 * </p>
 *
 * <p><strong>Features:</strong></p>
 * <ul>
 *     <li>Defines a simple method to simulate drawing a circle.</li>
 *     <li>Uses console output to represent the action of drawing.</li>
 * </ul>
 *
 * <p>Below is an example usage:</p>
 * <pre>
 * SimpleCircleDrawer drawer = new SimpleCircleDrawer();
 * drawer.drawCircle();
 * </pre>
 *
 * <p>This class is part of a beginner's tutorial for learning Java classes and methods.</p>
 *
 * @author Student
 * @version 1.0
 * @since 1.0
 */
```

Example of HTML Output for the Javadoc

SimpleCircleDrawer Class

This class simulates drawing a circle by printing a message. It is designed as a basic example for beginners learning Java.

Features:

- Defines a simple method to simulate drawing a circle.
- Uses console output to represent the action of drawing.

Below is an example usage:

```
SimpleCircleDrawer drawer = new SimpleCircleDrawer();
drawer.drawCircle();
```

This class is part of a beginner's tutorial for learning Java classes and methods.

Author: Student

Version: 1.0

Since: 1.0

Constructor & Method-Level Javadoc

```java
public class SimpleCircleDrawer2 {

    /**
     * <p>
     * The constructor for the <code>SimpleCircleDrawer</code> class.
     * </p>
     * <p>
     * This constructor creates an instance of the class, but doesn't take any parameters
     * because the functionality of this class is very basic.
     * </p>
     */
    public SimpleCircleDrawer2() {
        // Constructor does nothing but is defined for clarity
    }

    /**
     * <p>
     * This method simulates drawing a circle by printing
     * the message "Drawing a circle..." to the console.
     * </p>
     * <p>Use this method when you want to simulate the action of drawing a circle.</p>
     */
    public void drawCircle() {
        System.out.println(x:"Drawing a circle...");
    }
```

References

- Sun Microsystems, “How to Write Doc Comments for the Javadoc Tool”, http://java.sun.com/j2se/javadoc/writingdoccomments/index.html
- Emil Vassev, “DMS API Documentation”, Concordia University, Montreal, Quebec, Canada, 2005
- Wikipedia. “Comparison of Documentation Generators”. http://en.wikipedia.org/wiki/Comparison_of_documentation_generators
- These slides are inspired by Mimi Opkins' CECS 274 course, as well as the course by Emil Vassev and Joey Paquet, and include some of their content.