Object Oriented Programming

Documentation

Dr. Seán Russell
sean.russell@ucd.ie

School of Computer Science,
University College Dublin

September XX, 2019

Learning outcomes

After this lecture and the related practical students should...

- Understand the concept of documentation in Java
- be able to correctly annotate classes, methods and variables
- be able to generate Javadoc APIs for your classes

Documentation

- Documentation is not for explaining how code works
- It is for explaining what code does
- This is useful for allowing others to use your code without having to read the code
- Documentation in Java is done using javadoc

Table of Contents

1. Javadoc
   - Tags
   - Example
2. Compiling Javadoc

Javadoc

- Javadoc are special comments that can be used to build documentation for your classes

- A javadoc comment is a multi-line comment that starts with `/**` instead of `/*`

Javadoc Example

```
1 /** This is a javadoc comment that will be
    shown in the documentation */
2 i++;
3 /* This is a longer comment
4    The computer will ignore all of this
5    when compiling this code */
```

Table of Contents

1. Javadoc
   - Tags
   - Example
2. Compiling Javadoc

Javadoc Tags

- Javadoc uses tags to specify special pieces of information
- This could be the author of the class, an explanation of a parameter or some other useful information
- Mostly these are used to help others understand how to use your code

Tags

- `@author [name]` - who wrote this
- `@version [num]` - what version is this
- `@param [name] [explanation]` - describes one of the parameters of the method/constructor
- `@return [explanation]` - describe what is returned from the method
- `@exception [exeception] [explanation]` - describe the situation that causes a particular exception

Table of Contents

1. Javadoc
   - Tags
   - Example
2. Compiling Javadoc

What to Document

- Documentation should be completed for anything public
- This includes
  - A description of what the class/interface is used for
  - A description of any public instance variables
  - A description of any public methods
  - A description of any public constructors

Description of Class/Interface

- This should explain the main purpose of the class
- E.g. What is it used for
- It should also explain any other important information

Interface Example

```
1 /**
2  * The StringCalculator interfaces represents an object
3  * that can perform calculator like operations but
4  * instead of for numbers, it works for integer values
5  * represented as strings. This also allows for numbers
6  * to be presented in decimal or hexadecimal.
7  * @author Sean Russell
8  */
9 public interface StringCalculator {
```

Description of Instance Variable

- Should explain what the instance variable is used for
- Should only be rarely included as public instance variables should be rare

Instance Variable Example

```
1  /**
2   * A converter object that can be used to change
3   * values from strings to numbers and numbers to
4   * strings
5   */
6  public final Converter conv;
```

Description of Methods

- Should explain what the method does
- Should explain each of the parameters
- Should explain the return value

Method Example

```
1  /**
2   * A method to add two integer values as strings
3   * @param a The first number
4   * @param b The second number
5   * @param base The base that the numbers are
    represented in
6   * @return A string containing the value of the
    numbers added together
7   */
8   public String add(String a, String b, int base);
```

Method Example

```
1   /**
2    * Converts a single integer value that is
     represented as a string into a long
3    * @param number The number to be converted
4    * @param base The base the number is represented in
5    * @return a long containing the value
6    * @exception InvalidNumberForBaseException The
     string contains a character that is not acceptable
     for this base
7    */
8   public long convertToNum(String number, int base) {
```

Description of Constructors

- The parameters here are generally all that needs to be explained
- Additional explanation may be required to explain calculations that are used to create the object based on the parameters

Constructor Example

```
1    /**
2     *
3     * @param c The converter to be used
4     */
5    public StringCalc(Converter c) {
```

Table of Contents

1 Javadoc

2 Compiling Javadoc

Creating Documentation

- This documentation is not very useful hidden in the code
- We want to separate it so others do not need to read the code
- Java has a compiler to make nice documentation out of these comments

Javadoc Compiler

- The javadoc compiler is called javadoc
- This can be used on the command line
- E.g. `javadoc file.java`
- But is easier to do in eclipse

Compiling in Eclipse

- Right-Click on your project and chose **Export**
- Under **Java** choose **Javadoc** and select **Next**
- The location should automatically default to a folder named doc in your project
- Click **Finish**

Generated Javadoc

- To view the generated Javadoc open the file index.html in the doc folder
- This can be viewed in eclipse or within a web browser
- It should look just like the API documentation

Package StringCalc

Interface Summary

Interface	Description
StringCalculator	The StringCalculator interfaces represents an object that can perform calculator like operations but instead of for numbers, it works for integer values represented as strings.

Class Summary

Class	Description
Converter	
StringCalc	An implementation of the StringCalculator interface using longs to represent numbers and perform calculations.
StringCalcTest	

Exception Summary

Exception	Description
InvalidNumberForBaseException	

Method Detail

add

```
public java.lang.String add(java.lang.String a,
                            java.lang.String b,
                            int base)
```

Description copied from interface: `StringCalculator`

A method to add two integer values as strings

Specified by:

`add` in interface `StringCalculator`

Parameters:

`a` - The first number

`b` - The second number

`base` - The base that the numbers are represented in

Returns:

A string containing the value of the numbers added together