Prifysgol Abertawe
Swansea University

CS-230 Software Engineering

L11: Javadoc

Dr. Liam O’Reilly

Semester 1 – 2020

What is Javadoc?

- Javadoc is a documentation generator [. . .] [originally] from Sun Microsystems for generating documentation in HTML format from Java source code.

- The “doc comments” format used by Javadoc is the de facto industry standard for documenting Java classes.

  [Wikipedia]

  - There are others like Doxygen (www.doxygen.org).

- Official Website(?): http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html

Motivation

- If we write a separate document that documents the code then it will be produced during the initial development.
- Developer who edit the code might not update the documentation.
  - Due to time pressure or cost.
- Over time the documentation and code drift apart until the documentation is totally out-of-date.
- Solution (?): Keep the documentation next to the code it documents.
  - Hopefully developers will quickly update the documentation as they edit the code.
  - It seems to help.

Motivation (2)

- Javadoc is used to document the usage of your classes.
  - By reading the Javadoc of someone else's class you should understand how to use it.

- Javadoc is not intended to document the intricacies of how the code functions. You use normal comments for that.

- **You should not include in any Javadoc (of public entities) any implementation details of the class.**

Structure of a Java File with Javadoc

Javadoc is placed directly above Java code to document the Java code that directly follows it.

```
/**
 * Javadoc for class.
 */
public class SomeClass{
     ...

     /**
      * Javadoc for constructor.
      */
     public SomeClass(...) { ... }

     /**
      * Javadoc for method.
      */
     public void method1(...) { ... }

     /**
      * Javadoc for method2.
      */
     public void method2(...) { ... }
}
```

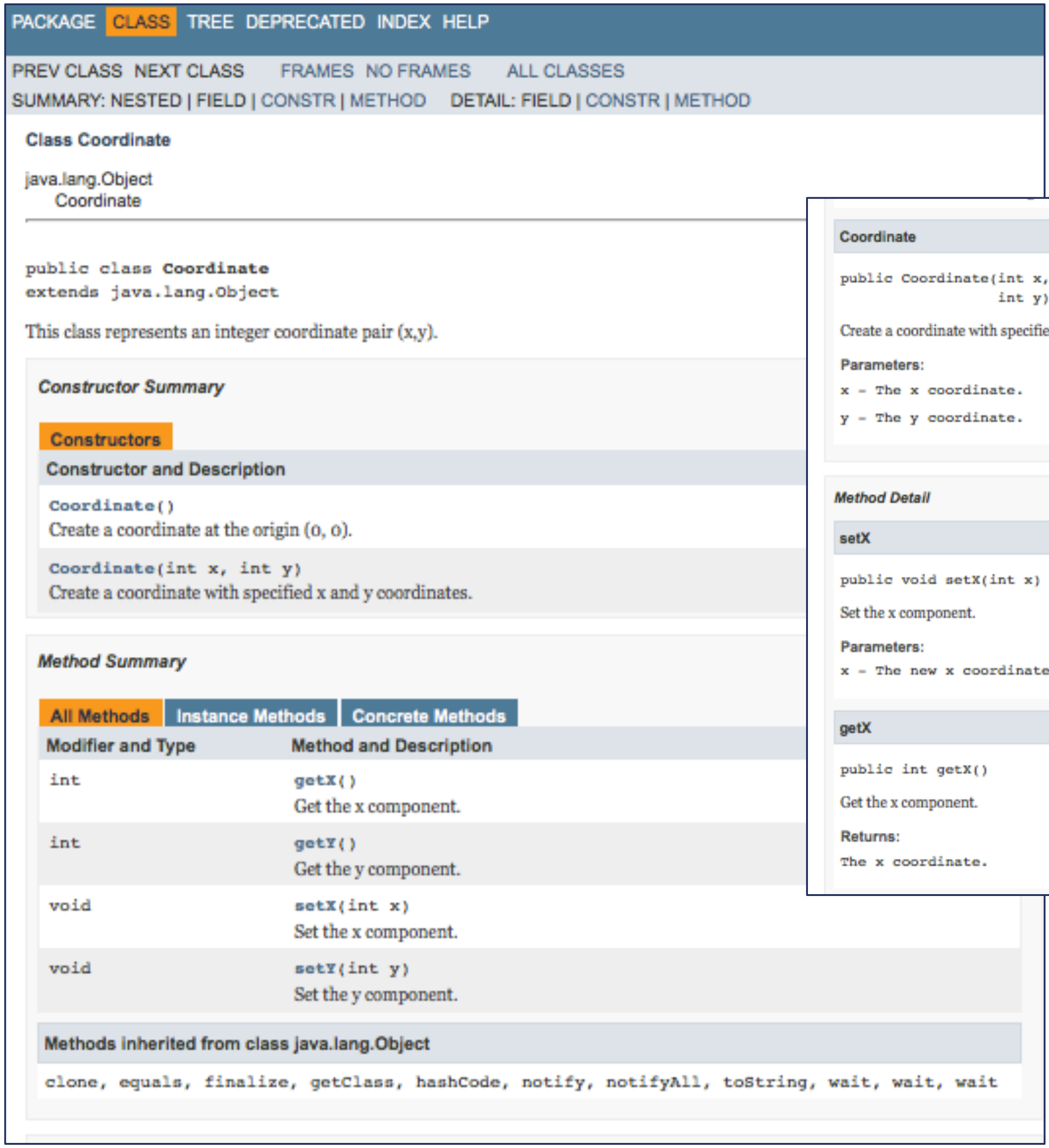
Javadoc uses special comment syntax:

```
/**
…
*/
```

Javadoc Example

```
/**
 * This class represents an integer coordinate pair (x, y).
 * @author Liam O'Reilly
 * @version 1.6
 */
public class Coordinate {
      ...

      /**
       * Create a coordinate at the origin (0, 0).
       */
      public Coordinate() { ... }

      /**
       * Create a coordinate with a specified x-coordinate and y-coordinate.
       * @param x The x coordinate.
       * @param y The y coordinate.
       */
      public Coordinate(int x, int y) { ... }

      /**
       * Set the x component.
       * @param x The new x coordinate.
       */
      public void setX(int x) { ... }

      /**
       * Get the x component.
       * @return The x coordinate.
       */
      public int getX() { ... }
}
```

Javadoc Example (2)


PACKAGE CLASS TREE DEPRECATED INDEX HELP
PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Coordinate

java.lang.Object
Coordinate

```
public class Coordinate
extends java.lang.Object
```

This class represents an integer coordinate pair (x,y).

Constructor Summary

Constructors

Constructor and Description
`Coordinate()` Create a coordinate at the origin (0, 0).
`Coordinate(int x, int y)` Create a coordinate with specified x and y coordinates.

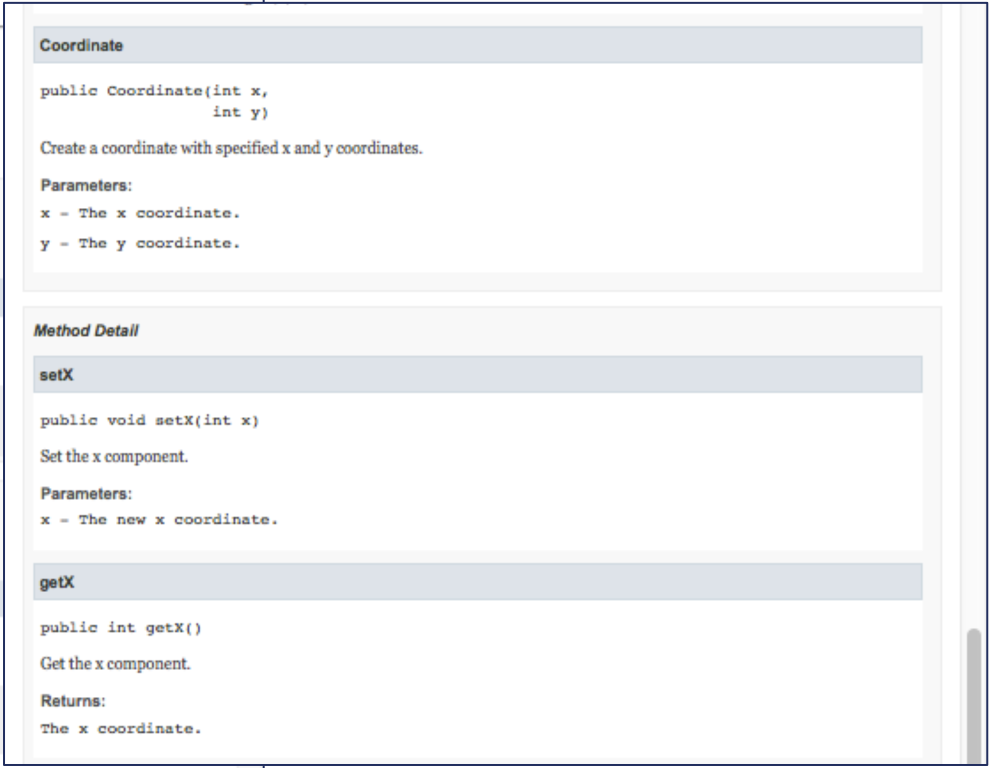
Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`int`	`getX()` Get the x component.
`int`	`getY()` Get the y component.
`void`	`setX(int x)` Set the x component.
`void`	`setY(int y)` Set the y component.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`


Coordinate

```
public Coordinate(int x,
                  int y)
```

Create a coordinate with specified x and y coordinates.

Parameters:

`x - The x coordinate.`

`y - The y coordinate.`

Method Detail

setX

```
public void setX(int x)
```

Set the x component.

Parameters:

`x - The new x coordinate.`

getX

```
public int getX()
```

Get the x component.

Returns:

`The x coordinate.`

7

Running Javadoc:

Javadoc can be run from the command line:

- javadoc -d <directory-name> -version –author *.java

It can also be run from Eclipse (or other IDEs):

- Eclipse commands
- <ALT SHIFT> + J
    - Generate Javadoc comments template in code
- <Project><generate Javadoc>
    - To create the javadoc web site.

Viewing the generated Javadoc:

- Open index.html in your favourite web browser.

The -d option allows you to specify a directory where javadoc will place the generated files.

Using this will stop javadoc “polluting” your source code directory with loads of html, css, etc.

Available Tags

Tags allow you "attach" documentation to various pieces of java code.

- `@author, {@code}, {@docRoot}, @deprecated,`
- `@exception, {@inheritDoc}, {@link},`
- `{@linkplain}, {@literal}, @param,`
- `@return, @see, @serial, @serialData,`
- `@serialField, @since, @throws,`
- `{@value}, @version`

- Javadoc can be written in HTML and can use HTML tags.

Public vs Private

- Normally Javadoc is used to document only the public aspects of a class.
  - When you run Javadoc it builds a webpage which only shows the public entities.
  - This is useful for giving to the public or other developers that should use your classes.

- Javadoc can be called with the option `-private`.
  - The generated documentation will now include all classes, constructor and methods.
  - This is useful for in-house (i.e., not public) documentation (e.g., for the developer).

Javadoc Class Comments

Javadoc Class Comments

Each class should contain a Javadoc comment above it.

- The first sentence should describe the class concisely. This is required.
- The subsequent sentences should provide more detail. These are optional.
- You must use full sentences and full stops with class comments.

Javadoc Class Comments - Tags

The following tags can be used in Javadoc Class Comments:

`@author name-text`

- Identifies the author of a class or interface with the specified name-text.
- Only shown in generated javadoc when the `-author` option is used on the javadoc command.
- Should only be used in the class javadoc comment.

`@version version-text`

- Specifies the version of a class or interface.
- Only shown in generated javadoc when the `-version` option is used on the javadoc command.
- Should only be used in the class javadoc comment.

Javadoc Class Comments - Example

```
/**
 * This class represents an integer coordinate pair (x, y).
 * @author Liam O'Reilly
 * @version 1.6
 */
public class Coordinate {
    ...
}
```

Simple description of the class, along with the author and version.

Javadoc Constructor & Method Comments

Javadoc Constructor & Method Comments

Each public method should contain a Javadoc comment above it.

- The first sentence should describe what the constructor / method does as concisely as possible. This is required.
- The subsequent sentences should provide more detail. These are optional.
- Can also comment private methods – depends on the coding convention in use.

Javadoc Constructor & Method Comments - Tags

The following tags can be used in Javadoc Class Comments:

`@param parameter-name description`

- Provides a description of a parameter of a method or constructor; with the specified parameter-name followed by the specified description.
- These are required.

`@return description`

- Describes data returned by a method. This text should describe the permissible range of values.
- This is unnecessary for constructors and void methods, but required otherwise.

Note: Can use sentences (start with a capital letter and end with full stop) or write phrases (but do not capitalise nor use full stops).

Javadoc Constructor & Method Comments - Example

```
/**
 * Create a coordinate with a specified x-coordinate and y-coordinate.
 * @param x The x coordinate.
 * @param y The y coordinate.
 */
public Coordinate(int x, int y) { ... }

/**
 * Set the x component.
 * @param x The new x coordinate.
 */
public void setX(int x) { ... }

/**
 * Get the x component.
 * @return The x coordinate.
 */
public int getX() { ... }
```

Everything someone needs to know in order to use the constructors and methods.

Other Tags

The following tags can also be used with constructors and methods:

- `@throws exception-name description`
  - Describe situations that cause exceptions to be thrown.
- `{@code text}`
  - Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.
  - This enables you to use regular angle brackets (< and >) instead of the HTML entities (&lt; and &gt;)

Javadoc Example: Stacks

Stacks

What is it?

How to Manipulate Stacks

- Create an empty stack
- Put something on top – "push"
- Look what is on top – "top"
- Take the top item away – "pop"

- Example Application:
  - Main implementation strategy in compilers for method calls.
  - Many other examples (used heavily in Computer Science).

Example: Bad Javadoc

```
/**
 * Puts data into stack array and increments
 * topOfStack integer by 1.
 * @param value The value to add to the array
 * holding the stack data.
 */
public void push(int value) {...}
```

- This Javadoc discusses implementation details such as the array and topOfStack – bad!
- We should be able to change the underlying data structure (say from an array to an ArrayList) and the user (programmer) using the Stack should not notice. We should not have to change the documentation – it still does the same thing – i.e., implements a working stack.

Example: Good Javadoc

```
/**
 * Adds a new value to the (top of the) stack.
 * @param value The value to add to the stack.
 */
public void push(int value) {...}
```

- This Javadoc describes what happens when the method is executed. It does not describe how it happens.
- Encapsulation: Hide the implementation detail. If we change how the stack works, we will not have to change the public interface nor the documentation.

Summary

- Javadoc is the defacto documentation tool for Java.
- Its easy and useful.
- Describe how to use class, constructor and methods.
- Do not expose implementation details.