Java

Javadoc

FSR Informatik

October 11, 2016

Overview

Javadoc creates a HTML documentation from your code.

Structured comments with annotations are used as source.

Eclipse

To create the Javadoc for your current project:
Project › Generate Javadoc. . .

Make sure you had installed Javadoc. If not the Finish button will stay grey.

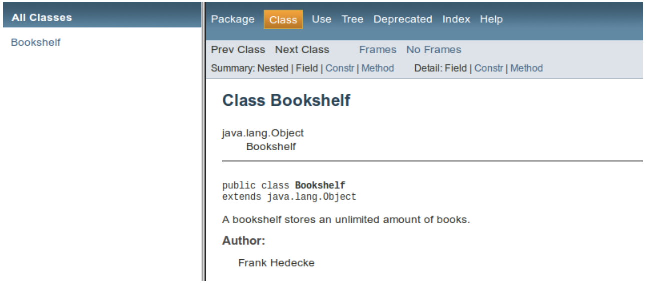
Class

An example class without methods:

```
import java.util.List;

/**
 * A bookshelf stores an unlimited amount of books.
 * @author Jane Doe
 *
 */
public class Bookshelf {

    private List<Book> books;

}

```

Class

You see the description of the class from the previous slide.



Class Bookshelf

java.lang.Object
    Bookshelf

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

A bookshelf stores an unlimited amount of books.

Author:

Frank Hedecke

Method

Add a small description of your method.

```
/**
 * A Constructor for Bookshelf.
 */
public Bookshelf() {
    books = new LinkedList<Book>();
}

```

Method with Parameter

Use **@param** to describe every parameter

```
/**
 * Puts a book into the bookshelf.
 * @param book that will be added in the bookshelf
 */
public void addBook(Book book) {
    this.books.add(book);
}

```

Method with Return Value

Use **@return** to describe the return value.

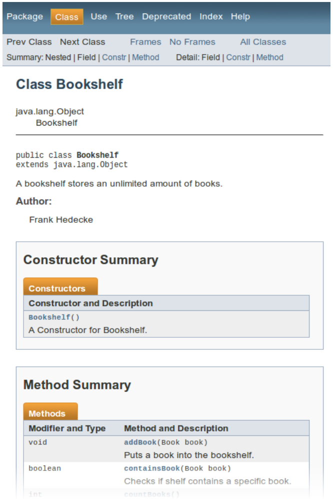
```
/**
 * Returns the number of stored books.
 * @return number of stored books
 */
public int countBooks() {
    return books.size();
}

```

Method with Parameter and Return Value

For boolean: Describe in which case a method returns true.

```
    /**
     * Checks if shelf contains a specific book.
     * @param book whose occurrence is checked
     * @return true if the shelf contains the given book
     */
    public boolean containsBook(Book book) {
        return books.contains(book);
    }

```

Javadoc shows a summary of all methods and constructors. A detailed view is below the summary.



Package Class Use Tree Deprecated Index Help
Prev Class Next Class Frames No Frames All Classes
Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

Class Bookshelf

java.lang.Object
    Bookshelf

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

A bookshelf stores an unlimited amount of books.

Author:

Frank Hedecke

Constructor Summary

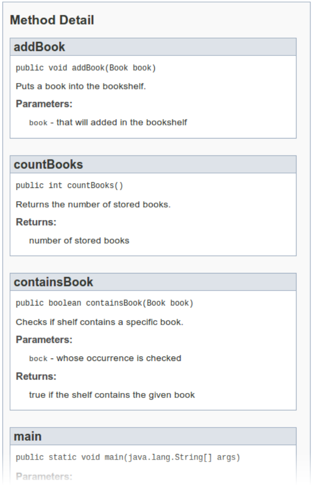
Constructors

Constructor and Description
`Bookshelf()` A Constructor for Bookshelf.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`addBook(Book book)` Puts a book into the bookshelf.
`boolean`	`containsBook(Book book)` Checks if shelf contains a specific book.
`int`	`countBooks()`



Method Detail

addBook

```
public void addBook(Book book)
```

Puts a book into the bookshelf.

Parameters:

`book` - that will added in the bookshelf

countBooks

```
public int countBooks()
```

Returns the number of stored books.

Returns:

number of stored books

containsBook

```
public boolean containsBook(Book book)
```

Checks if shelf contains a specific book.

Parameters:

`book` - whose occurrence is checked

Returns:

true if the shelf contains the given book

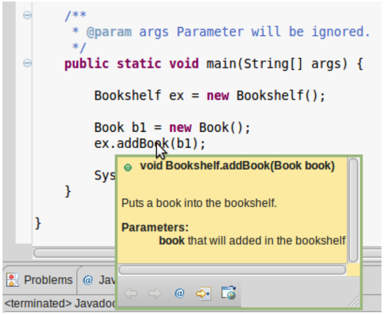
main

```
public static void main(java.lang.String[] args)
```

Parameters:

Eclipse Tooltips

Hovering a method opens a tooltip displaying information like in the Javadoc.



```
    /**
     * @param args Parameter will be ignored.
     */
    public static void main(String[] args) {

        Bookshelf ex = new Bookshelf();

        Book b1 = new Book();
        ex.addBook(b1);

        Sys
    }

}
```

void Bookshelf.addBook(Book book)

Puts a book into the bookshelf.

Parameters:
book that will added in the bookshelf

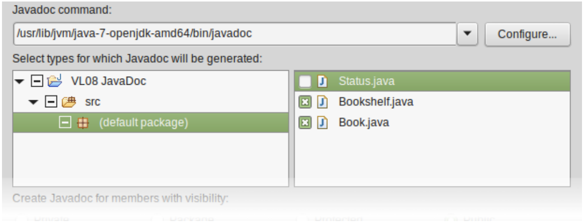
Example - A second Class

Another class.

```
/**
 * A book.
 * @author John Doe
 *
 */
public class Book {

}

```

More classes

After pressing Project 〉 Generate Javadoc. . . you can choose which classes shall be included.


Javadoc command:

/usr/lib/jvm/java-7-openjdk-amd64/bin/javadoc

Select types for which Javadoc will be generated:

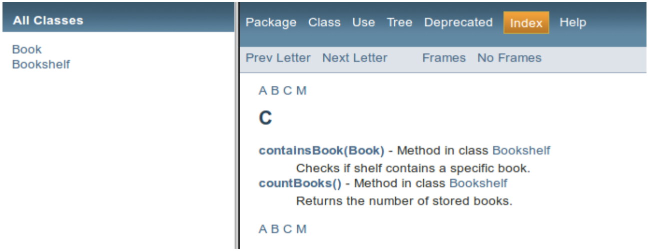
VL08 JavaDoc	☐ Status.java
src	☒ Bookshelf.java
(default package)	☒ Book.java

Create Javadoc for members with visibility:

The left view shows a tree with all packages. After selecting a package you can select your classes.

More classes

Now you see both selected classes in the Javadoc.


All Classes
Package Class Use Tree Deprecated Index Help
Prev Letter Next Letter Frames No Frames
A B C M

Book
Bookshelf

C

containsBook(Book) - Method in class Bookshelf
 Checks if shelf contains a specific book.
countBooks() - Method in class Bookshelf
 Returns the number of stored books.

The view *Index* shows a list of all methods, constructors, . . .

Constructors with this

Remember: **this** is a reference to the current object.

this() calls the constructor. So you can write multiple constructors with less code.

```
public class Example {

    public String value;

    Example(String value) {
        this.value = value;
    }

    Example() {
        this("standard");
    }
}

```

Extended use of super

In the previous lectures we used **super()** to call the constructor from the superclass. We can use **super** to call any other method from the superclass.

```java
import java.util.*;

public class StringSet extends HashSet<String> {

    @Override
    public boolean add(String elem) {
        // add a "?" to every String added to this set
        return super.add(elem + "?");
    }
}
```

Return for Void Methods

Void methods can have a return statement, too. The empty return statement stops the execution of the method early.

```java
public void foo (boolean condition, int x) {

    if (condition) {
        return;
    }

    for (int i = 1; i <= x; i++) {
        System.out.println(i);
    }
}

```