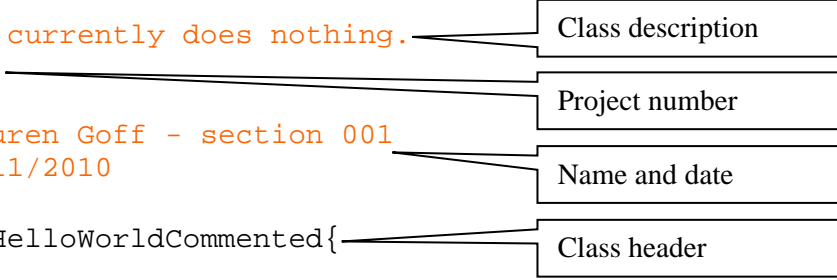


## Class documentation (Chapter 1):

Every class in your program should have a Javadoc tag that specifies the program's purpose, the project number, the author, and the date. Tags that start with the @ symbol represent special information that is read differently when documentation is generated. **IMPORTANT:** make sure that your Javadoc tags start with two asterisks (\*\*) and that your class Javadoc tag is placed directly before your class declaration (after any import statements).

Example:

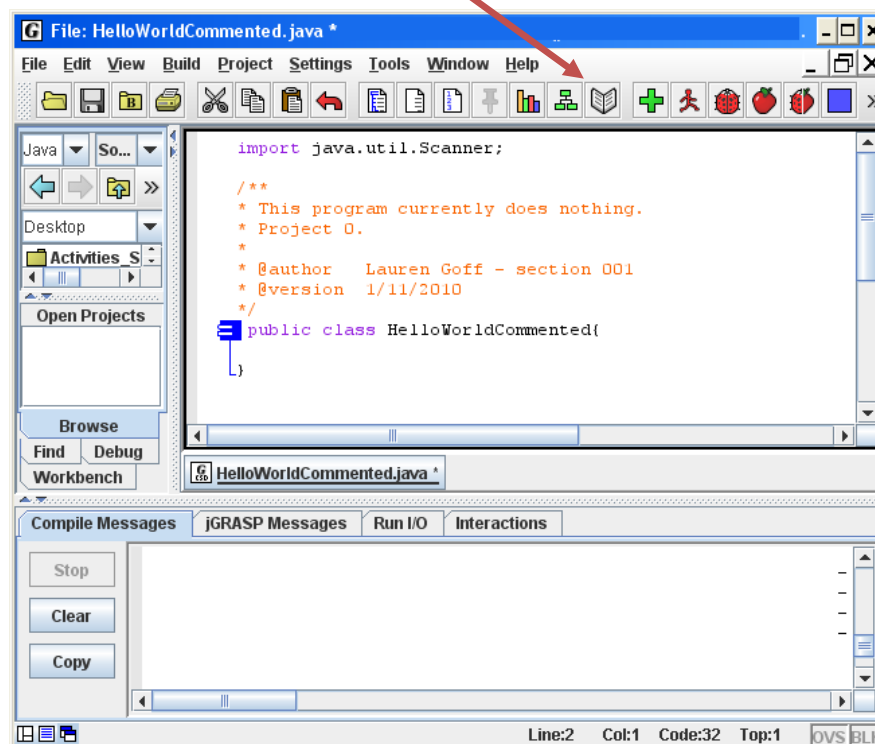
```
/**
 * This program currently does nothing.
 * Project 0.
 *
 * @author Lauren Goff - section 001
 * @version 1/11/2010
 */
public class HelloWorldCommented{
}
```



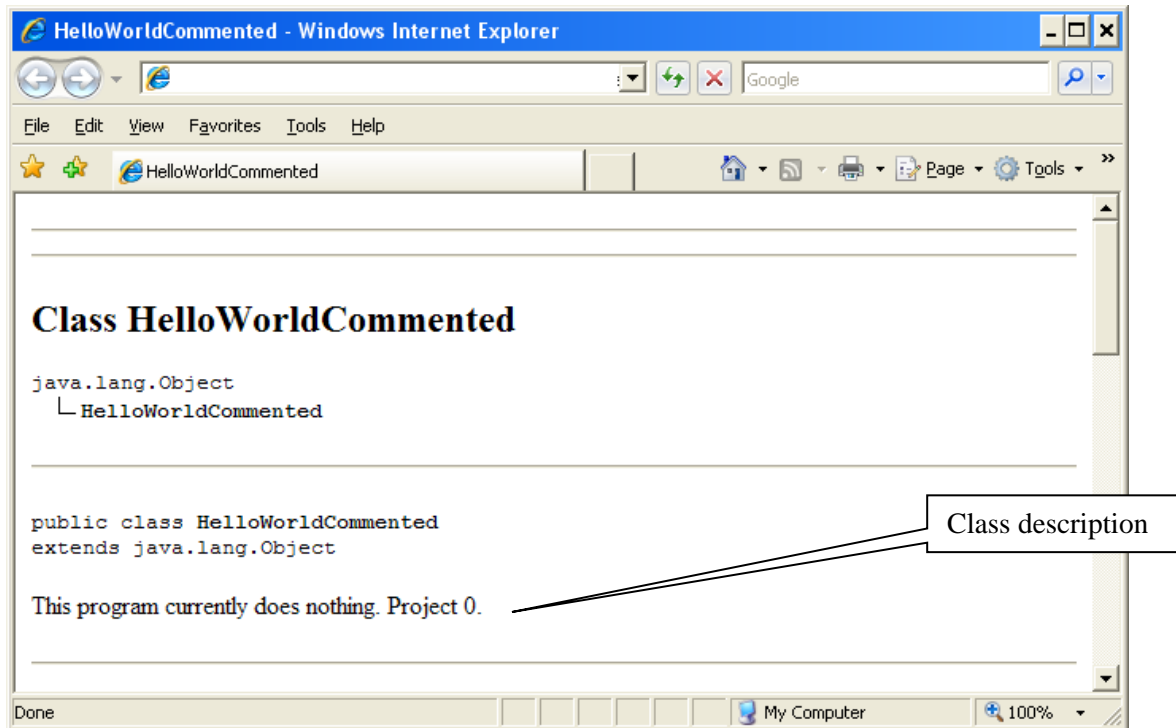
## Checking your documentation:

In order to make sure that your documentation is properly formatted, press the Generate

Documentation () button in jGRASP:



You should now see documentation for your project. Make sure that your class description shows up under the name of your class (note that your name and the date should not show up when documentation is generated):



## Main method documentation (Chapter 1):

Your main method should also have a Javadoc tag. Note that this time your Javadoc tag includes an @param tag; you can just copy the tag from below for your program, as method parameters and command-line arguments will be discussed at a later date.

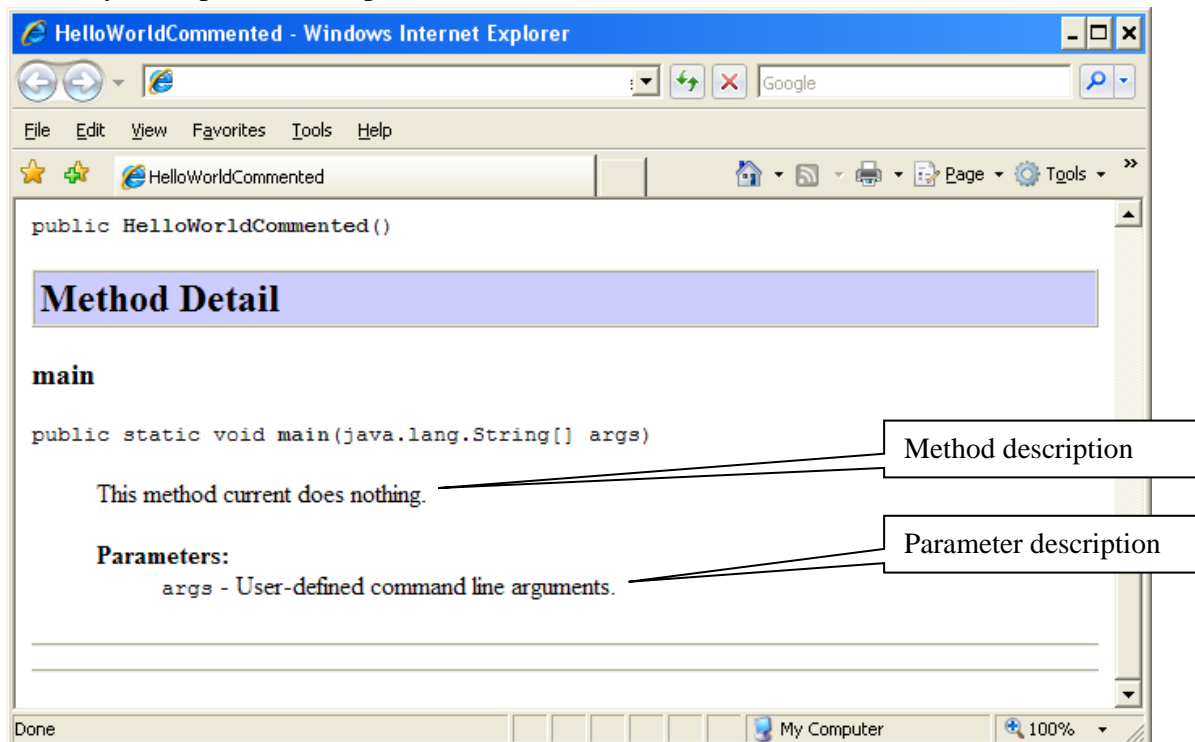
Example:

```
/**
 * This method current does nothing.
 *
 * @param args Command line arguments - not used.
 */
public static void main(String[] args){

}
```

## Checking your documentation:

Generate the documentation for your class using the same method as page 1. Scroll down to find the documentation for your main method. It should contain your method description as well as your @param description as below.



**Generating documentation outside of jGRASP (Chapter 1):**

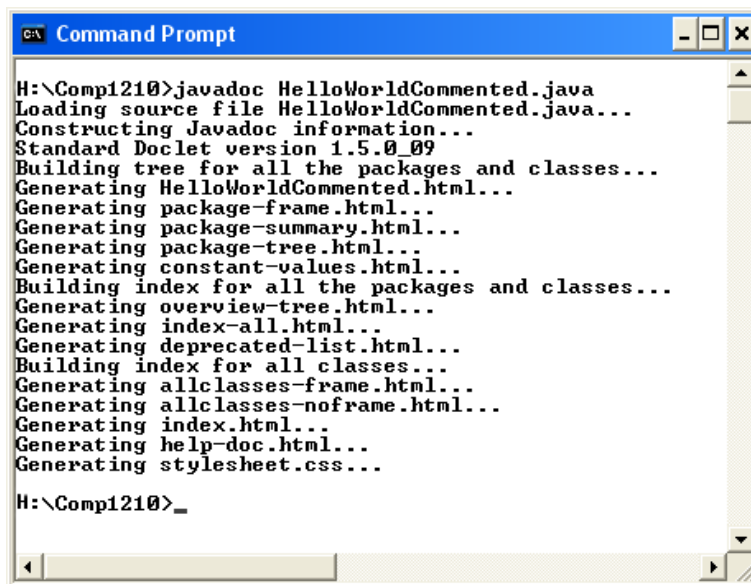
Open a command prompt by going to Start -> All Programs -> Accessories -> Command Prompt. Navigate to the directory in which your source code is located using the following commands:

Command	Function
<b>dir</b>	Displays all of the files/folders in the current directory
<b>cd</b> <i>foldername</i>	Navigates to a folder called <i>foldername</i>
<b>cd..</b>	Goes back one folder
<b>cls</b>	Clears the command prompt screen
<b>H:</b>	Moves to the H: drive (or any other drive specified)

Go to the command prompt and use the javadoc program to generate the documentation for your program. Type the following:

```
> javadoc ProgramName.java
```

Where ProgramName is the name of your Java source code file. Example:



```
Command Prompt
H:\Comp1210>javadoc HelloWorldCommented.java
Loading source file HelloWorldCommented.java...
Constructing Javadoc information...
Standard Doclet version 1.5.0_09
Building tree for all the packages and classes...
Generating HelloWorldCommented.html...
Generating package-frame.html...
Generating package-summary.html...
Generating package-tree.html...
Generating constant-values.html...
Building index for all the packages and classes...
Generating overview-tree.html...
Generating index-all.html...
Generating deprecated-list.html...
Building index for all classes...
Generating allclasses-frame.html...
Generating allclasses-noframe.html...
Generating index.html...
Generating help-doc.html...
Generating stylesheet.css...
H:\Comp1210>_
```

Go to Start -> My Computer and then go to the file where your program is located. Find the file that is named *ProgramName.html* (where ProgramName is the name of your program). Open the html file. This is where your documentation is located. Check your documentation to make sure that all of the information has been stored properly as in page 2-3 of this file.



You will also need to specify what each method returns. If the method does not return a value (void) as in the methods above, you do not need an @return tag. Take a look at the following method which has a return type but no parameters:

```
/**
 * Returns a String representation of the object,
 * including customer name, interest rate, balance,
 * and accumulated travel miles.
 *
 * @return String representation of credit card object.
 */
public String toString(){
    String output = "Name: " + customer + "\n" +
        "Interest rate: " + interest + "%\n" +
        "Balance: $" + balance + "\n" +
        "Travel Miles: " + travelMiles + "\n";
    return output;
}
```

Note that the method description includes exactly what is included in the output of toString, and a description of the return after the @return tag (you should never have a variable name in your @return tag).

@return The String representation of credit card object.

↑  
Description only

If a method has both a return and multiple parameters, then you will need to have an @param tag for each parameter and an @return tag for the return. Example:

```
/**
 * Adds interest to a card by multiplying the balance by the
 * interest rate and adding that amount to the old balance.
 * Returns the balance of the card after the interest has been
 * added.
 *
 * @param interestRate The interest rate in decimal format.
 * @return The current balance after adding the interest.
 */
public balance chargeInterest(double interestRate){
    balance += balance*interest;
    return balance;
}
```

**Constructor documentation (Chapter 4):**

You should document your constructor exactly as you would a method. Two things to take into consideration when documenting a constructor are as follows:

- You will never have an @return statement for a constructor, as constructors do not return a value.
- You need to describe everything that happens when an object is set up. This is especially important when values are initialized in the constructor as with no parameters.

See the example below. Note that it describes how the balance and travel miles are initialized even though there are no inputs given for those values.

```
/**
 * Creates a new credit card object with a given customer name
 * and interest rate. The balance of the account and the number
 * of travel miles are initialized to 0.
 *
 * @param customerName The name of the customer.
 * @param interestRate The interest rate on the credit card.
 */
public CreditCard(String customerName, double interestRate){
    customer = customerName;
    interest = interestRate;
    balance = 0;
    travelMiles = 0;
}
```