

- Writing comments for use by javadoc
- Commenting your code is always a good idea
 - Makes it easier to update in the future, when you no longer remember what, how, and why you did stuff
 - Makes future maintenance and enhancements easier
 - Helps the other folks who support your code understand how and why you did something
 - Assuming the code you are writing is related to your job, then writing the comments is also part of your job

- Now is a good time to get into the practice of commenting your code
 - I know, documentation is rarely the highlight of your day
- I tend to do the following
 - If I'm writing code for someone else, and they may need support in the future or they are going to maintain it, I will put comments in the code and if it is complex enough, I will write a document explaining formulas and how the code works
 - If I'm writing code for myself to either solve a problem or prototype an algorithm, and I don't think I'm going to need the code again, then I won't write comments
 - If I'm prototyping an algorithm, eventually there will be some documentation written, either a software specification or software requirements that will eventually be implemented by someone
 - At least when I had a job outside of teaching

- The java documentation has lots of information about javadoc comments
 - <https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html#principles>
- The comments are written in HTML
- We will go through an example of a version of the BankAccount class
- First note that javadoc throws an error if there are no public or protected classes to document, so I updated the class definition from “**class** BankAccount” to “**public class** BankAccount”
- Private fields don't appear to be documented (in the generated documentation)
- Public and protected fields are documented (in the generated documentation)
- I also had to make the constructor public for it to show up in the documentation

- Put comments immediately before

- Classes
- Methods
- Fields

- Comments are contained in

```
/**
```

```
*
```

```
*/
```

- Some of the tags

- @author, @version, @param, @return @deprecated, @throws, @exception
- I didn't notice the content of @author or @version show up anywhere

- Here's the beginning of the BankAccount class, with the javadoc comments

```
1 /**
2  * The BankAccount class is used to implement a simple bank account.
3  * @author dave
4  * @version 0.99
5  */
6
7 public class BankAccount
8 {
9     /**
10      * Field representing the account owners last name.
11      */
12     protected String ownerLastName;
13
14     /**
15      * Field representing the account owners first name.
16      */
17     public String ownerFirstName;
18
19     /**
20      * Field representing the account number.
21      */
22     private String accountNumber;
23     private double checkingBalance;
24     private double savingsBalance;
25
26     /**
27      * The class constructor. The checking and savings balance are initialized to zero.
28      * @author dave
29      * @version 0.99
30      * @param ownerLastName the account owner's last name
31      * @param ownerFirstName the account owner's first name
32      * @param accountNumber the account number
33      */
34     public BankAccount(String ownerLastName, String ownerFirstName, String accountNumber)
35     {
36         this.ownerLastName = ownerLastName;
37         this.ownerFirstName = ownerFirstName;
38         this.accountNumber = accountNumber;
39         this.checkingBalance = 0.0;
40         this.savingsBalance = 0.0;
41     }
```

Use /**

*

*/ to document fields and methods

























- Here's some more

```
43- /**
44-  * This method is used to withdraw funds from the checking balance of the account.
45-  * @param amount the amount to withdraw
46-  * @return boolean value specifying whether the withdrawal was or was not successful
47-  */
48- public boolean withdrawFromChecking(double amount)
49- {
50-     return false;
51- }
52-
53- /**
54-  * This method is used to withdraw funds from the savings balance of the account.
55-  * @param amount the amount to withdraw
56-  * @return boolean value specifying whether the withdrawal was or was not successful
57-  */
58- public boolean withdrawFromSavings(double amount)
59- {
60-     return false;
61- }
62-
63- /**
64-  * This method is used to transfer funds from the savings balance to the checking balance of the account.
65-  * @param amount the amount to transfer
66-  * @return boolean value specifying whether the transfer was or was not successful
67-  */
68- public boolean transferFromSavingsToChecking(double amount)
69- {
70-     return false;
71- }
72-
73- /**
74-  * This method is used to transfer funds from the checking balance to the savings balance of the account.
75-  * @param amount the amount to transfer
76-  * @return boolean value specifying whether the transfer was or was not successful
77-  */
78- public boolean transferFromCheckingToSavings(double amount)
79- {
80-     return false;
81- }
```

Use @param to document parameters

Use @return to document return values

- Here's what the folder looked like after executing “javadoc BankAccount.java”, where the folder originally only contained BankAccount.java

<div> <div>< ></div> <div>javadoc</div> <div> <div>⋮ ⬆</div> <div>⋮ ⬆</div> </div> </div> <div>Back/Forward</div> <div>View</div> <div>Group</div>			
Name		^	Date Modified
 allclasses-index.html			Today at 2:06 PM
 allpackages-index.html			Today at 2:06 PM
 BankAccount.html			Today at 2:06 PM
 BankAccount.java			Today at 11:11 AM
 constant-values.html			Today at 2:06 PM
 deprecated-list.html			Today at 2:06 PM
 element-list			Today at 2:06 PM
 help-doc.html			Today at 2:06 PM
 index-all.html			Today at 2:06 PM
 index.html			Today at 2:06 PM
 jquery-ui.overrides.css			Today at 2:06 PM
 member-search-index.js			Today at 2:06 PM
 module-search-index.js			Today at 2:06 PM
 overview-tree.html			Today at 2:06 PM
 package-search-index.js			Today at 2:06 PM
 package-summary.html			Today at 2:06 PM
 package-tree.html			Today at 2:06 PM
>  resources			Today at 10:35 AM
>  script-dir			Today at 10:35 AM
 script.js			Today at 2:06 PM
 search.js			Today at 2:06 PM
 stylesheet.css			Today at 2:06 PM
 tag-search-index.js			Today at 2:06 PM
 type-search-index.js			Today at 2:06 PM

- Here's BankAccount.html

Class BankAccount

java.lang.Object
BankAccount

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

The BankAccount class is used to implement a simple bank account.

Field Summary

Fields

Modifier and Type	Field	Description
java.lang.String	ownerFirstName	Field representing the account owners first name.
protected java.lang.String	ownerLastName	Field representing the account owners last name.

Constructor Summary

Constructors

Constructor	Description
BankAccount (java.lang.String ownerLastName, java.lang.String ownerFirstName, java.lang.String accountNumber)	The class constructor.

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method	Description
void	depositToChecking (double amount)	This method is used to deposit funds to the checking balance of the account.
void	depositToSavings (double amount)	This method is used to deposit funds to the savings balance of the account.
java.lang.String	getAccountNumber ()	This method is used to return the account number.
double	getCheckingBalance ()	This method is used to return the account's checking balance.
java.lang.String	getOwnerFirstName ()	This method is used to return the account owner's first name.
java.lang.String	getOwnerLastName ()	This method is used to return the account owner's last name.
double	getSavingsBalance ()	This method is used to return the account's savings balance.
java.lang.String	toString ()	This method is used to return a string representation of the account.
boolean	transferFromCheckingToSavings (double amount)	This method is used to transfer funds from the checking balance to the savings balance of the account.
boolean	transferFromSavingsToChecking (double amount)	This method is used to transfer funds from the savings balance to the checking balance of the account.
boolean	withdrawFromChecking (double amount)	This method is used to withdraw funds from the checking balance of the account.
boolean	withdrawFromSavings (double amount)	This method is used to withdraw funds from the savings balance of the account.

Methods inherited from class java.lang.Object

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

Field Details

ownerLastName

protected java.lang.String ownerLastName

Field representing the account owners last name.