**SHOPPING BASKET ASSESSMENT**

**Useful Hints (3)**

**Documenting the Classes**

In order to document the classes already written, you can add Javadoc comments, and auto-generate the documentation through eclipse.

Add javadoc comments to your classes.

**JavaDoc Review**

When you implement classes and methods, you should get into the habit of thoroughly *commenting* their behaviours. In Java there is a very useful standard form for **documentation comments**. If you use this form in your classes, a program called javadoc can automatically generate a neat set of HTML pages that describe them.

A documentation comment is placed before the class or method declaration that is being documented. It starts with a /\*\*, a special comment delimiter used by the javadoc utility. Then you describe the method's *purpose*. Then, for each method parameter, you supply a line that starts with @param, followed by the parameter name and a short explanation. Finally, you supply a line that starts with @return, describing the return value. You omit the @param tag for methods that have no parameters, and you omit the @return tag for methods whose return type is void.

**NOTE**

Use documentation comments to describe the classes and public methods of your programs.

The javadoc utility copies the *first* sentence of each comment to a summary table in the HTML documentation. Therefore, it is best to write that first sentence with some care. It should start with an uppercase letter and end with a period. It does not have to be a grammatically complete sentence, but it should be meaningful when it is pulled out of the comment and displayed in a summary.

Here are two typical examples.

/\*\*

Withdraws money from the bank account.

@param amount the amount to withdraw

\*/

public void withdraw(double amount)

{

**implementation**--**filled in later**

}

/\*\*

Gets the current balance of the bank account.

@return the current balance

\*/

public double getBalance()

{

**implementation**—**filled in later**

}

The comments you have just seen explain individual *methods*. Supply a brief comment for each *class*, explaining its purpose. The comment syntax for class comments is very simple: Just place the documentation comment above the class.

/\*\*

A bank account has a balance that can be changed by

deposits and withdrawals.

\*/

public class BankAccount

{

...

}

Your first reaction may well be "Whoa! Am I supposed to write all this stuff?" These comments do seem pretty repetitive. But you should take the time to write them, even if it feels silly.

It is always a good idea to write the method comment *first*, before writing the code in the method body. This is an excellent test to see that you firmly understand what you need to program. If you can't explain what a class or method does, you aren't ready to implement it.

What about very simple methods? You can easily spend more time pondering whether a comment is too trivial to write than it takes to write it. In practical programming, very simple methods are rare. It is harmless to have a trivial method over-commented, whereas a complicated method without any comment can cause real grief to future maintenance programmers. According to the standard Java documentation style, *every*class, *every* method, *every*parameter, and *every* return value should have a comment.

**NOTE**

Provide documentation comments for every class, every method, every parameter, and every return value.

The javadoc utility formats your comments into a neat set of documents that you can view in a web browser. It makes good use of the seemingly repetitive phrases. The first sentence of the comment is used for a *summary table* of all methods of your class (see [**Figure 3**](javascript:moveTo('a_method_summary_generated_by_javadoc');)). The@param and @return comments are neatly formatted in the detail description of each method (see [**Figure 4**](javascript:moveTo('method_detail_generated_by_javadoc');)). If you omit any of the comments, then javadoc generates documents that look strangely empty.

This documentation format should look familiar. The programmers who implement the Java library use javadoc themselves. They too document every class, every method, every parameter, and every return value, and then use javadoc to extract the documentation in HTML format.

