

CS1322 Homework 3

(Due: Tuesday, Sep.20, 2005 @ 6:00 pm)

General – Use of comments

Up to now, we have not required any code comments just for simplicity. Comments are very important, however, for software maintenance and understanding of your code. There is an old adage in development that says, “Comment your code so that someone who has no idea what the code does can understand it, because in six months that someone will be you!”

In this assignment we will begin requiring you to comment your code and we will gradually build this level up over the term. For HW3 we will begin to “javadoc” our code. If you have ever looked at the API at Sun’s website, you have seen javadoced classes before. For example, if you look at the API for the String class, you will see a class description, instance variables, method descriptions, and other important information.

To generate pages like those API pages, the Java SDK comes with a program called javadoc. The way javadoc works is it reads a .java file and looks for comments in this format:

```
/**
 * stuff
 * more stuff
 */
```

It then translates what is inside those special comments into HTML to create pages similar to the ones you have seen in the API. Using javadoc from the command line is very similar to compiling your code, so you would type something like “javadoc *.java” to create documentation for all the classes in your current directory. If you are using JGrasp, you can invoke javadoc by using **Project->Generate Documentation** from the pull-down menu. That will create .html files for all of your .java files in your current project.

For HW3 we will create only one type of javadoc comments: **The class header**. The class header is important because it states what the class does and a general description of how to use it. You can also include special information like who authored the code and what version it is.

Here’s an example:

```
/**
 * This is HW3, problem 1.
 * This class rids the Earth of bad guys.
 * @author James Bond gtg007m
 */
```

As you can see, the “@author” tells javadoc that the rest of that line is the author of that file. There are many “@” tags in javadoc that we will tell you more about in later homeworks.

Although not required, you should also place good comments into your code when the algorithm you are using is not obvious. For example, for the statement:

```
int perimeter = a + b + c;
```

a good comment might be:

```
// calculate perimeter by summing the lengths of the sides
```

and a bad comment would be:

```
// add a b and c together
```

A frequent syntax problem developers have is mismatched braces. One commenting technique that can help you is to comment the close brace with a short description of what it is enclosing. For example:

```
    } // end switch(x)
  } // end main method
} // end class def Demo
```

We'll be looking at more javadoc and commenting issues in future HWs. For now, if you have more questions, check out the javadoc info page under the Java Resources page on the class website, or talk to your TA.

Assignment 3.1: Jcalculator

In this part of the homework, you will create a class, called Jcalculator, that implements a simple prefix-notation calculator using Java's arithmetic operators and the Math class. The class has a main method that reads input from the user and performs the requested mathematical operations. To read input from the user, you will use a Scanner object, as you did in previous assignments. You will use conditional statements to determine which part of the code to execute based on the input.

The calculator provides 7 operations: +, -, /, *, pow, abs, and sqrt. When run, the calculator asks the user to enter the operation he/she wants to perform. At that point, the user can enter one of the following commands:

```
+ PARAM1 PARAM2
- PARAM1 PARAM2
/ PARAM1 PARAM2
* PARAM1 PARAM2
pow PARAM1 PARAM2
abs PARAM1
sqrt PARAM1
```

PARAM1 and **PARAM2** are real numbers. After the user enters the command, the program performs the requested operation, prints the result, and terminates. In your solution, assume that the user provides the right number of parameters for the operations. In other words, assume that operators will be followed by the right number (one or two) and type (real numbers) of operands. Your code should report an error if the user requests an unknown operation (e.g., "foo"). To clarify, here are some examples of use of the program:

```
> java Jcalculator
Please input a command: + 32 46.5
Result: 78.5
> java Jcalculator
Please input a command: bar
Error: unrecognized command
```

```
> java Jalculator
Please input a command: k 2.2 6.1
Error: unrecognized command
> java Jalculator
Please input a command: sqrt 68.9
Result: 8.300602387778854
> java Jalculator
Please input a command: pow 2.2 6.1
Result: 122.68129538201431
```

Remember that you can use a Scanner object to input different types of values (consult the Java API docs, at <http://java.sun.com/j2se/1.5.0/docs/api/>, for details about Scanner's methods).

Assignment 3.2: Creating a business card

Write a java applet that displays a business card of your own design. The card should include both graphics and text. At a minimum, your text should include your name. The graphics should include at least 3 or 4 different shapes from the Graphics class. You should also use a least 3 different colors on your card. Make the business card (applet) 400 pixels wide by 200 pixels tall, an aspect ratio similar to a real card. Make sure to include an HTML file that displays your applet in a web page. We will put some of the "coolest" cards off our class web page (with the student's permission of course).

To view your applet, you can run it as an applet from inside JGrasp, but you should also embed it in a web page for WWW viewing. You can do this in a web page editor or even a simple text editor. Since you may not know how to code html, the only line of code needed is given below:

```
<applet code="BusinessCardApplet.class" height=200 width=400></applet>
```

(assuming that your applet file is called BusinessCardApplet.java)

Turn-in Procedures

After you have finished the above assignments, turn them via [Webwork](#) You will be submitting three files:

- Jalculator.java
- BusinessCardApplet.java
- The html file you created for viewing your applet