Assignment 1.1: Program Developement and Debugging

Due: 8am Friday, September 12th, 2008

Directions

The problems for this assignment are described below. You will submit this assignment by emailing it to me as a Zip file. The file will be named **Assignment-1.1-LastName.zip**, where [LastName] is obviously your last name. All of your submitted assignments will be named in this fashion. The Zip file will contain a folder of the same name (so, Assignment-1.1-LastName), which will contain a folder for each of the subsequent problems. Each of these subfolders will have the assignment number and name of its particular problem (for example, **Assignment-1.1-isPrime** or **Assignment-1.1-dataTypes**). Each of these problem subfolders will contain all of the necessary files for that particular problem. It's very important that you follow all of these naming conventions exactly as specified, or you could lose credit.

Problems

1) isPrime

You will develop an algorithm for determining whether a certain number, N, is prime. I suggest first just thinking of a few numbers in your head (preferably above 50) and determine whether each is prime. How did you figure it out in your head (or on paper)? Isolating *that* process is how you develop an algorithm. Once you have a general idea of the process, try to write it down as steps. These steps should be very simple, involving only simple math. If you need to (you will) you can say something like "If the result of step 4 is blah blah, go back to step 2, otherwise, go to step 5." A fifth grader who knows what a prime number is should be able to follow your directions. Test your steps on a few more numbers, perhaps some larger ones that you think might be prime but aren't sure about. Once you feel pretty good about your algorithm (that's what these steps are!), create a text file, **algorithm.txt**, using a simple text editor (Notepad or Wordpad in Windows) and record the numbered steps you came up with.

Remember, these steps need to be very, very specific. Saying "test whether or not N is divisible by all the numbers smaller than it" is not enough. The idea behind this problem is that you must always *think* about how you're going to approach a problem before you actually start coding it.

2) debugCode

You will take a piece of code (**MovingDisk.java**. which you must download from our class website) with some errors in it and fix those errors. "But," you say, "how

can I fix these errors if I don't really know what I'm doing (it *is* only the first assignment). Nevertheless, you have an indispensable friend: Eclipse's error highlighting and correction suggestion. When you open the file in Eclipse (you will need to create a new project called **Assignment-1.1-debugCode** and add **MovingDisk.java** as a class to its **src** folder), it should show the file with all sorts of syntax highlighting and have some jagged red underlines. Those red underlines represent the *syntax errors* that you must fix. If you hover your mouse over the error underline, a little pop-up box should show up with the error message and some possible solutions.

3) dataTypes

You will create a simple program (project called **Assignment-1.1-dataTypes** and main class named **DataTypes.java**)that explores the assignment and ranges of a few common primitive data types. Your program must do the following for each of the data types listed below.

Data Type	Assign & Display	Display maximum and minimum
`int`	`int` value	`Integer` values
`double`	`double` value	`Double` values
`char`	`char` value	`Char` values
`boolean`	`boolean` value	
`String`	`String` value	
(symbolic) constant	`final` value	

Your variable names should adhere to acceptable style (see Litvin & Litvin 6.2), including your constant (all caps), which should also represent a constant value, like your birthday or something that doesn't change.