# HW0: Setting up Dr. Java

DUE DATE: Sunday May 18th, 2008 7pm EST

## Do everything on the on list provided here: http://coweb.cc.gatech.edu/cs1316/11

(Remember the username and password for the coweb is *attach* and *carmen* respectively)

## Introduction to the interactions pane

One of the appeals of Dr. Java is the *interactions pane*, which can be used to run simple Java commands without setting up a whole class. The interactions pane works like an independent main method. You will learn more about the main method later. The interactions pane is the bottom pane in Dr. Java's interface.

Type the following line in the interactions pane:

```
System.out.println("Hello World.");
```

and press enter. `System.out.println(...)` is used to output to the terminal and the interactions pane in the case of Dr. Java and is very useful for simple debugging.

Now try creating two integers:

```
int a = 6;
int b = 5;
```

You can add the two integers together:

```
int sum = a + b;
```

You can also multiply them:

```
int product = a *b;
```

subtract them:

```
int difference = a - b;
```

divide them:

```
int quotient = a / b;
```

find the remainder after division:

```
int remainder = a % b;
```

and many other things as well. The interactions pane will allow you to do all of the previous lines without giving the variable a particular type (such as `int`). However, you must give types when you later have to write an actual class. You will also be required to include types on all examinations.

Now maybe it would be best to print out each result. Try the following lines:

```
System.out.println(a);
System.out.println(b);
System.out.println(sum);
System.out.println(product);
System.out.println(difference);
System.out.println(quotient);
```

It is kind of hard to discern one number from the other since we just printed the numbers (as `String`s) without distinguishing them in any way. This notion will be especially important when you have several printouts within a program that will print out in different orders or even not at all. String concatenation is where you combine two or more `String`s together. To concatenate two `String`s in Java you use the "`+`" operator. Thus if we had the

following:

```
String s1 = "Georgia";
String s2 = "Tech";
String space = " ";
```

We can create the `String` "`Georgia Tech`" by concatenating `s1`, `s2` and `space` together.

```
System.out.println(s1 + space + s2);
```

Notice that the following lines will yield the same result as `System.out.println(s1 + space + s2);`

```
String gatech = s1 + space + s2;
System.out.println(gatech);
```

Now do some more `String` manipulations:

```
 String myName = "Dawn";
System.out.println("Hi, my name is " + myName);
System.out.println("I have two numbers " + a + " and " + b);
System.out.println("The sum is " + sum);
System.out.println("The product is " + product);
System.out.println("The difference is " + difference);
System.out.println("The quotient is " + quotient);
```

## Setting up the media path

Open up `FileChooser.java` and `Picture.java` in the java-sources folder we provided to you. Compile those two files using the compile button found in the upper right hand area (Ignore any warnings) and then close them (Closing them will help us make sure that you have set up your Dr. Java correctly, because having them open can sometimes will cause an improperly set up Dr. Java to function correctly).

Let us go ahead and set your media path (Keep in mind that `C:/CS1316/mediasources/` is the media path and may be different on your computer. Please adjust as needed).

```
FileChooser.setMediaPath("C:/CS1316/mediasources/");
```

To make sure it worked, try loading a picture:

```
Picture p = new Picture(FileChooser.getMediaPath("swan.jpg"));
p.show();
```

Setting up your media path means that the program will store the path to your media sources folder inside of a file (`SimplePictureProperties.txt`) and retrieve it each time you need to use it. So instead of hardcoding `C:/CS1316/mediasources/swan.jpg` each time, you use `FileChooser.getMediaPath("swan.jpg")` which retrieves the media path, `C:/CS1316/mediasources/`, and concatenates it with `swan.jpg` to create `C:/CS1316/mediasources/swan.jpg`. This is necessary for your assignments, because media paths will differ from computer to computer (namely between yours and the grading TA's). However it is important that you never include a `setMediaPath` command with any of your assignments, because doing so will mess up the grading TA's existing media path.

## Creating your first class

In Dr. Java, make a new file and save it as `HW0.java`. Now inside `HW0.java` you will define your first class `HW0` and your first main method (Note: the name of the class and the file that contains the class must match):

```
public class HW0 {
   public static void main(String [] args){
            //This is my first comment
            System.out.println("I have just written my first class!");
   }
}
```

You will have no problem understanding the code above a little later on in the course. For now go ahead and save the code and compile it using the compile button.

Now run the code using the interactions pane:

```
java HW0
```

Or you simply hit the run button if there is one on your Dr. Java interface.

You should get an output of `I have just written my first class!`

All the lines we had you write in the interactions pane previously can also be written in a main method like the one provided above. Go ahead and put all the work you did in the interactions pane into the new main method above except the `setMediaPath` command!

Now try to write some comments. Above public class HW0 write your name, t-square log in name (same as your gatech email) and your recitation section. You can write it in one of two ways:

```
//Name: Dawn Finney
//T-square log in: gth683e
//Recitation section: A2
```

Or

```
/*Name: Dawn Finney
T-square log in: gth683e
Recitation section: A2*/
```

`/**/` are for multi-line comments while `//` are for single-line.

Make sure your file compiles and run HW0 again. You should get the same (or really similar) output to the interactions pane as when you entered each line manually by hand. It is very important to always make sure to compile and run your homework before turning it in. TAs will not even consider grading your homework if it does not compile.

## The Syntax versus Semantics

Sometimes it is easier to think of a compiler like the spelling and grammar check you would find in a word processor such as Microsoft Word. Having correct syntax means that your code is arranged in the correct form and structure for a particular language. The compiler will try to make sure that the commands that you are trying to send to the computer are syntactically correct before converting it in actual form that the machine understands. This is why when you compile an error-free `HW0` you will notice an additional file appear, `HW0.class`. The `.class` file contains the code that the machine will understand, but is not readable by humans at all. Thus, when you submit your homework we only want the `.java` files, because they are technically the only ones that are readable by humans. While `a .class` is technically functional, if a student submits the `.class` file it becomes impossible to tell whether the student did his/her own work, because the file is not readable by humans. With Dr. Java you will notice an additional file with the `.java~` extension. This `.java~` file extension is sometimes termed “the tilde file” or “the java tilde file,” where tilde is the name of the “~” symbol. We also do not accept `.java~` file, because they are auto-save files that are generated by Dr. Java. The tilde files tend to be incomplete versions of the .java file and usually do not reflect the most recent changes. It is important to not rely on these tilde files.

So let us say that your code has gotten past the compiler meaning that it is completely syntactically correct, but does it mean it will run? Let us go back to the word processor. Your English paper passes the spelling and grammar check, but does it make sense when you read through it? Consider the following sentence:

Steve, the red dog, visits clouds when his feet are shiny and when the potato drives.

Now it is safe to say that the above sentence is syntactically correct but what is it trying to say? Imagine having a supervisor that spoke like this when he or she is trying to give you a new task. What a headache! Your code, like the sentence, may be completely syntactically correct but not make any sense to the machine. With the supervisor, you may try to remedy the situation by asking questions such as *What did you mean by potato?* and *Who is Steve?*. The machine is not as flexible as you are; it does not know how to ask questions. The machine is going to take whatever commands you give it quite literally and try to work with them, but if some sort of problem occurs, the problem will probably affect other areas of the program as well.

What do we do if get an exception? Well it is good to know that an exception is basically an error that occurs when the machine is executing your commands and thus is also sometimes known as a runtime exception. You try to read and figure why the exceptions are occurring. The exceptions typically have a type such as `NullPointerException` or `IndexOutOfBoundsException` and a line number associated with them. You should look at the line where the error is occurring and at each part of the line picking out some good suspects. You will get better at this as you move through the course. Do not be disappointed if TAs are much faster at picking out possible places with problems than you are, but instead strive to be at the level where TA intervention is not necessary.

## Answer the questions

Answer the following questions in a text file and submit it with the rest of your homework. DO NOT SUBMIT ANY OTHER FILE FORMAT.

1. What is an appeal of using the Dr. Java interactions pane?

2. How would you find the remainder of 10/4 in Java?

3. What is the result of 3/4 in Java? Why do you think this is? (Hint: Try more numbers and look for a pattern)

4. What is String concatentation? What operator do you use in Java?

5. How do `setMediaPath` and `getMediaPath` work?

6. Why is it important to NEVER include a `setMediaPath` command in your homework assignment?

7. What are two ways to "run" your code (based on the HW0 description)?

8. What is difference between `/**/` and `//` ?

9. What is the result when `HW0Tester` is run?

10. Of the file extension, `.java`, `.java~` and `.class`, which is the ONLY one that TAs will accept? Why is that the only one? The answer is not simply just because we the TAs said that is the only one we want. (Hint: For each one of the file extensions think about what type of data and in what format the file actually contains.)

## What to turn in

- HW0.java.
- HW0 questions.txt

## How to turn it in

- Turn in via TSquare.