CS 1331 Homework 1

Due Thursday August 30th, 2012 8:00 PM

Introduction

Welcome to CS1331, Introduction to Object Oriented Programming! This assignment will introduce you to our collaboration policy, help you install the software for this class (you might already have it), write, compile, and run a program, and even break the program with errors. First is the collaboration policy.

Homeworks will usually be released late Thursday night or Friday, and will always be due the following Thursday at **8pm.** There will be absolutely no exceptions in regards to late homeworks past the grace period (which extends to 2am on T-Square). If I get an email from you at 2:01 saying you can't submit on T-Square, I cannot help you. In situations of absolute T-Square meltdown, make sure you email your code as an attachment **before the end of the grace period** to either the Head TA or your grading TA.

1.1: Academic Integrity and Collaboration Policy

You are expected to uphold this course's Academic Integrity and Collaboration Policy. As outlined in your syllabus, the policy is:

We expect academic honor and integrity from students. You are to be aware of and follow the academic honor code of Georgia Tech: http://www.deanofstudents.gatech.edu/osi/plugins/content/index.php. Your work in this class is to be your own. Significant coding will be required on the exams, and the homework is intended to help prepare you.

To reiterate, there should be no collaboration on assignments this semester, not even with students from the other 1331 course.

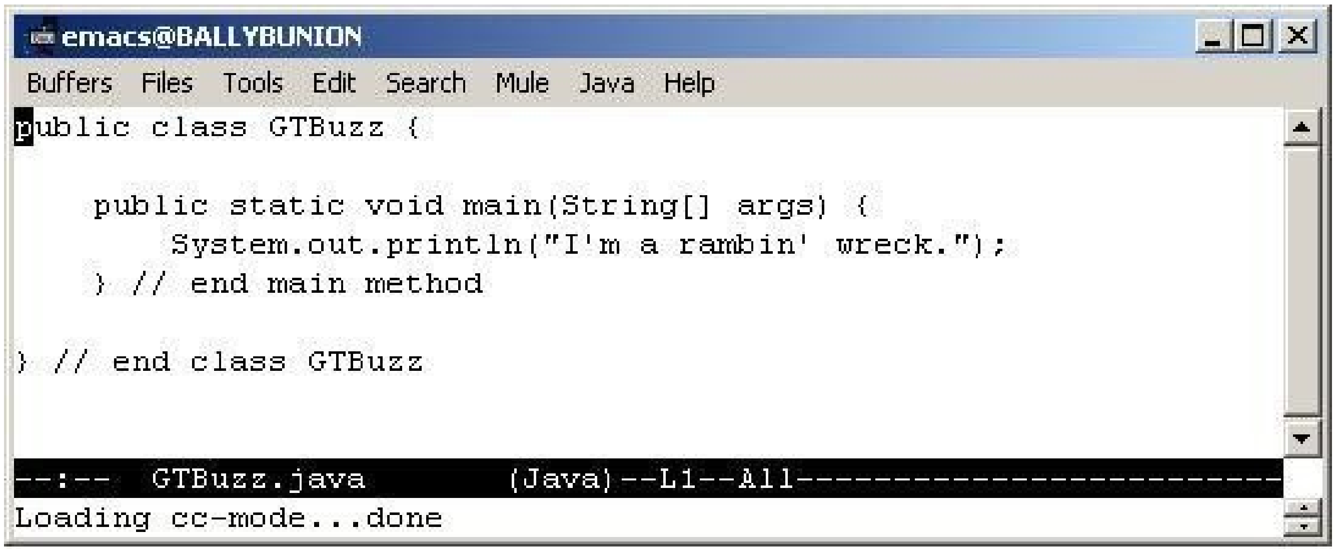
1.2: Installing the Software

The computer language of this course is Java, so before running any programs, we must install the Java compiler and Java Virtual Machine (JVM). We have created a Java Install Guide, as many of you have already seen on T-Square, to guide you through the process. You can find it attached with the assignment and also posted on T-Square, here**.** Please note that we may change the course approved version of java, as prior semesters used jdk 1.6 but the new edition of the book seems to be written with jdk 1.7.

The document also walks you through choosing an **IDE** (Integrated Development Environment) so you can start writing some Java programs. You may choose whichever IDE you want, but it is recommended you avoid more advanced auto-completion and syntax correction features until later in the course. For Mac and Linux, the built-in editors work great by default. For Windows, the guide lists several alternatives as using “notepad” is highly discouraged (a free windows application called notepad++ however is very popular).

1.3 Writing and executing a simple program

First, you will type Java source code using an editor. Next, you will compile the source code. Recall the purpose of the compiler and compilation step is to translate Java source code into bytecode and in the process check the code for compiler errors. Once free of compiler errors, the file is compiled and a file of bytecode is generated. The JVM then interprets and executes the bytecode. First things first – let's create a java source file. In the editor of your choice (shown here is emacs), create a plain text file named GTBuzz.java. IDEs such as jGrasp and Eclipse include an editor for exactly this purpose. In the empty file, type the following code and save it. Please note that indention is also very important because it makes the code more readable and easier to debug.



```
public class GTBuzz {

    public static void main(String[] args) {
        System.out.println("I'm a rambin' wreck.");
    } // end main method

} // end class GTBuzz
```

```
--:--  GTBuzz.java       (Java)--L1--All--------------------------
Loading cc-mode...done
```

Congratulations! You have just written your first java program. Let’s go through the code so you can understand what is going on.

Line Number	Code Segment	Explanation
1	`public class GTBuzz {`	Every java file starts with public class <<filename>> followed by a curly brace. Remember, for every opening brace you need a closing brace.
2	`public static void main(String[] args) {`	The details of this line are not important (for now), but take away two things. First, this segment of code is called a method header. Second, this is a special method called ‘main’.
3	`System.out.println(“I’m a ramblin’ wreck.”);`	Now we are going to make our code do something. For this first homework, all we want to do is print something to the console or command prompt. The ‘System.out.println' tells the computer you want to print something to the screen. What you put inside the parenthesis, in quotes, is the text you want to print out. Note

		that the line ends with a semi-colon.
4	`} //end main method`	This curly brace matches with the one in line 2, which means anything between the braces belongs to the main method. The '//' after the curly brace allows us to place a comment. Anything after the '//' is not considered code to execute. It is good practice to label the end braces so we know what it matches
5	`} // end class GTBuzz`	This brace ends the class. Same idea as above.

After saving your file, there are several ways to compile your code. Remember that compiling will alert you to any syntax errors that you might have.

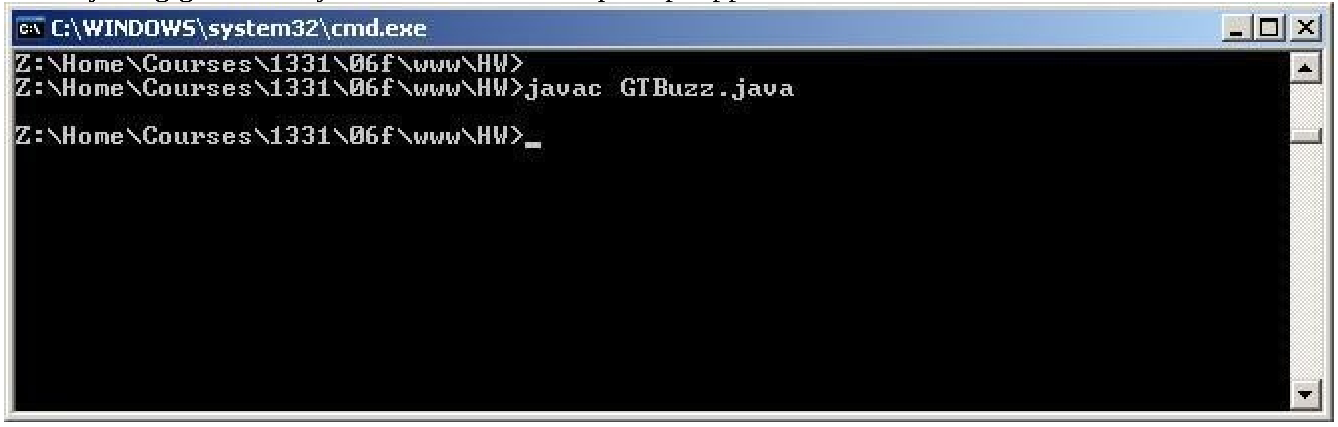
If you are using an IDE like JGrasp, look in the toolbar at the top for a button that says compile.

Alternatively, if you want to compile your code from the command prompt, open a command prompt window (in MS Windows, go to Start->Run and type 'cmd' in the box. A black screen should appear with a prompt; in OS X, go to Applications->Utilities->terminal; In any GNOME environment (including Ubuntu), Applications->Accessories->Terminal) and go to the directory in which your file is saved.

To go to the directory in which your file is saved type:
`cd directory` (for example, "cd Desktop")

Then, to compile your code:
```
javac GTBuzz.java
```

If everything goes well, you should see a new prompt appear:



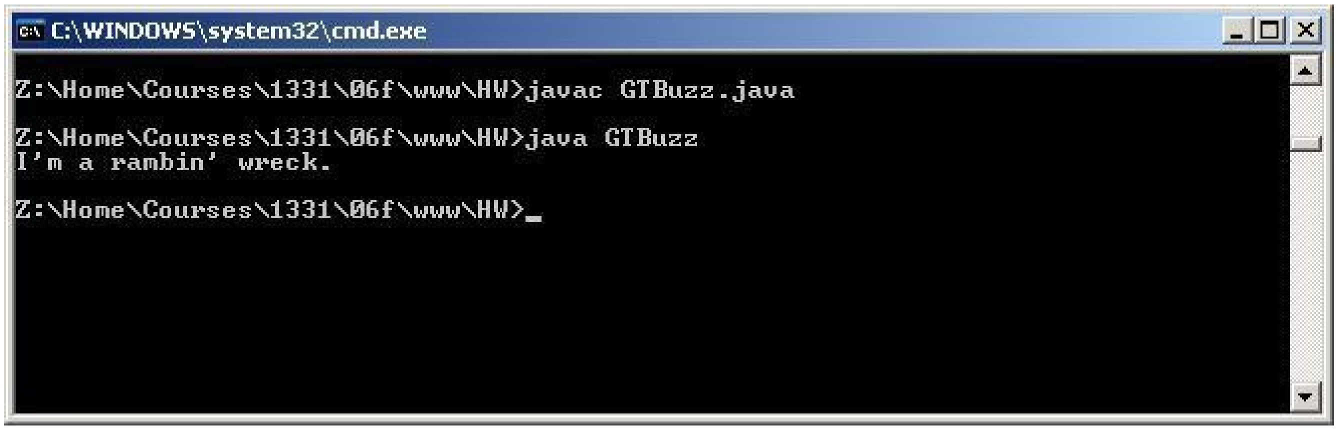
```
Z:\Home\Courses\1331\06f\www\HW>
Z:\Home\Courses\1331\06f\www\HW>javac GTBuzz.java

Z:\Home\Courses\1331\06f\www\HW>_
```

Congratulations! You have just compiled your first java program. Notice that in the same directory a file called GTBuzz.class has appeared. This file has the bytecode we mentioned earlier. Now let's run it and see what happens. If you are using an IDE, find the run button in the toolbar. Alternatively, at the command prompt, type:

```
java GTBuzz
```

You should see something like the following:



```
Z:\Home\Courses\1331\06f\www\HW>javac GTBuzz.java

Z:\Home\Courses\1331\06f\www\HW>java GTBuzz
I'm a rambin' wreck.

Z:\Home\Courses\1331\06f\www\HW>_
```

Again, congratulations. You have run your first java program. Now, let's see what happens when there are errors in the code.

1.4: Experimenting with Errors

Now you will modify the program you created in Part 3 and experiment with dealing with errors. Write the answers for this part in a file called HW1Errors.txt.

For each of the steps listed at the end of this section you will do the following:

1. Start from the original GTBuzz.java file
2. Make the one change described
3. Save the changed file as GTBuzz.java
4. Attempt to recompile your code
5. Note any error message you are given in your text file and state if that error is a compile-time error or a runtime error. (Note to Eclipse users: since eclipse compiles in real-time, it may be difficult to discern the difference between the two types of errors. If you are having trouble with this, consider reverting to a simpler IDE)
6. If no error is given, compilation was a success. Only then should you run the new version.
7. If it does not run, note any message that was given in your text file and state if that error is a compile or run-time error.
8. Undo the change you made so GTBuzz.java is back to its original version. To be positive that it is back to the working version, do the following:
    a. Save
    b. Compile
    c. Run
9. Move on to the next alteration and repeat this process

Here are the changes:

1. Change GTBuzz to GTbuzz (note the lowercase 'b').
2. After reverting to the original code, change wreck to WRECK.
3. After reverting to the original code, remove the first quotation mark in the string.
4. After reverting to the original code, remove the last quotation mark in the string.
5. After reverting to the original code, change main to man.
6. After reverting to the original code, change System.out.println to println.
7. After reverting to the original code, remove the semicolon at the end of the println statement.

8. After reverting to the original code, remove the last brace in the program.

After trying all these changes, revert to the original code, save, compile, and run. What you turn in should be the original code with none of these modifications remaining.

When recording possible errors, make sure to clearly indicate whether the error is compile or run-time, and the error that was thrown. Here are some sample answers:

Runtime error: ArithmeticException on line 5
Runtime error: Tried to divide by zero on line 5.
Runtime error: Exception in thread "main" java.lang.ArithmeticException: / by zero at
    GTBuzz.main(GTBuzz.java:5). [This is known as the stack trace]
No error.

Turn-in Procedure

Turn in the following files on T-Square. When you're ready, double-check that you have *submitted* and not just saved as draft.

- GTBuzz.java
- HW1Errors.txt

Make sure you are submitting your .java file, and not your .class file.

Verify the Success of Your HW Turn-In

Practice "safe submission"! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
    a. It helps insure that you turn in the correct files.
    b. It helps you realize if you omit a file or files.**
    (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
    c. Helps find last minute causes of files not compiling and/or running.

**Note: Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework (past the grace period of 2 am) will not be accepted regardless of excuse. Treat the due date with respect. The real due date and time is 8pm Thursday. Do not wait until the last minute!