CS 1331 Homework 1

Due Friday January 17, 2014 8:00pm

Introduction

Welcome to CS1331, Introduction to Object Oriented Programming! This assignment will help you write, compile, run, and even break a program with errors.

Java Installation

The programming language used in this course is Java. Make sure that Java is installed before running any programs. You can find instructions on how to install Java on Windows and Mac at the course site:

http://www.cc.gatech.edu/~stasko/1331/started/index.html

Note: Make sure that you install the Java JDK and NOT the JRE. The JDK is what we need to compile and build Java.

If you have any problems installing Java, then feel free to talk to any of the TAs via email, office hours, etc.

Text Editor/IDE Installation

After you install Java, you'll need to install a text editor or IDE to actually write your programs in. We don't have any real restrictions on text editors except that we want students to avoid using full-blown IDEs like Eclipse or IntelliJ until later on in the semester *if you haven't used them before*. If you don't know what Eclipse or IntelliJ are, no problem! We will introduce those environments later this year. Suggestions from the TAs for text editors/IDEs:

Sublime Text (used by almost all the TAs, a real fan favorite, and it's really colorful)

Notepad++ (a bare bones, basic, but solid editor for students just starting out)

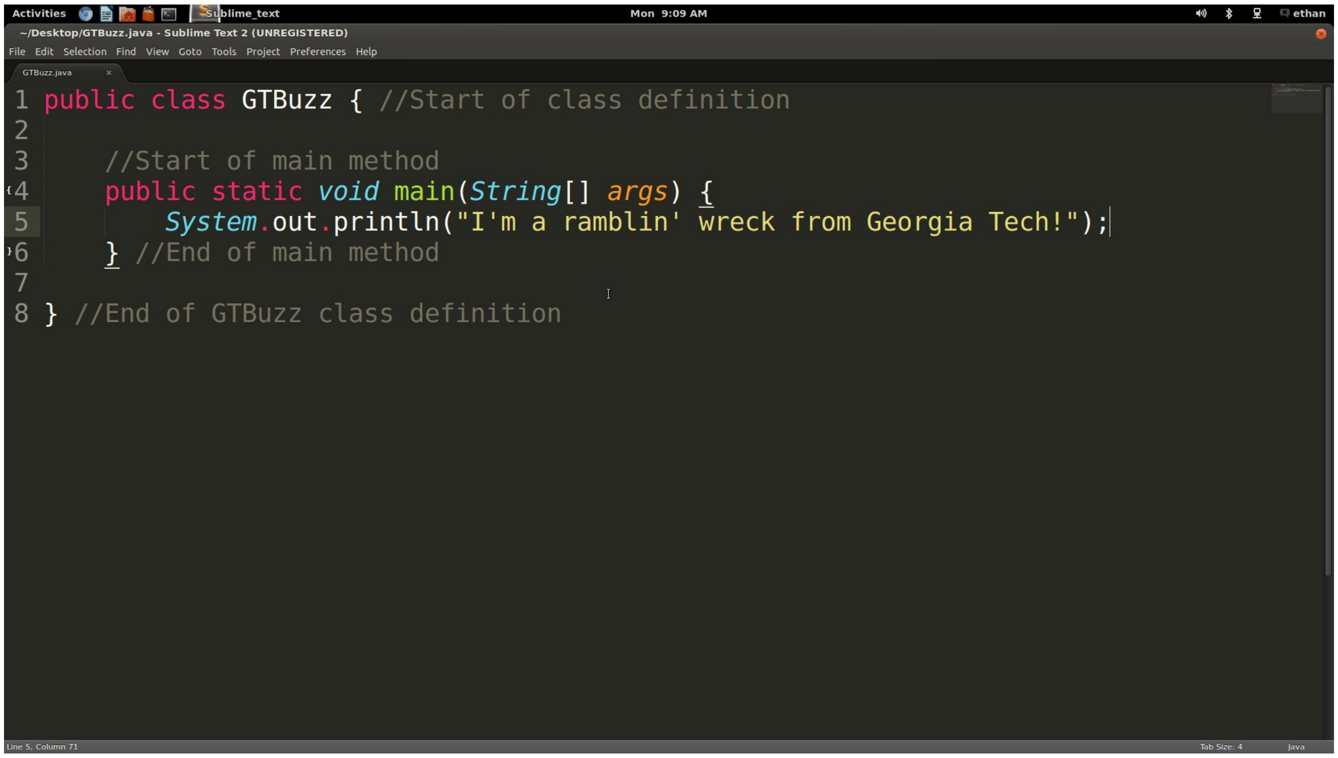
VIM/emacs (for those of you who have *lots* of experience, these are very powerful editors used widely in industry. Definitely not a first timers tool.)

If you'd like to use an IDE, Professor Stasko recommends jGrasp to start. BlueJ is another one students have used in the past.

You should be able to find free (legal) installations for these around the Internet if you search around. Ask a TA for help if you need it!

Let's get started! - Writing 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 Sublime Text), create a plain text file named GTBuzz.java.


Activities
Sublime_text
Mon 9:09 AM
ethan
~/Desktop/GTBuzz.java - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
GTBuzz.java
Line 5, Column 71
Tab Size: 4
Java

```java
public class GTBuzz { //Start of class definition

    //Start of main method
    public static void main(String[] args) {
        System.out.println("I'm a ramblin' wreck from Georgia Tech!");
    } //End of main method

} //End of GTBuzz class definition
```

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 easy to debug.

Wooo, We Did It!

What exactly did we do...? Let's take a look!

public class GTBuzz {
This is the class definition. Every Java file starts with the class definition which HAS to match the same name as the file. For example, GTBuzz.java starts with public class GTBuzz {.

public static void main(String[] args) {
We'll get to the keywords later, but this is the main method header/definition. The main method is the driving method. This means that the main method is the first method to be called and will start everything.

System.out.println("I'm a ramblin' wreck from Georgia Tech.");
Now we want to do something. System.out.println prints "I'm a ramblin' wreck from Georgia Tech." to the screen. Note that the " symbols aren't printed when this happens.

} //End of main method
This is the matching curly brace to the main method header. The curly braces define a block of code, and in this instance the main method.

} //End of GTBuzz class definition
This is the matching curly brace to the class definition header for GTBuzz. Like the one above, this defines and ends the block of code that GTBuzz defines.
Compiling and executing a simple program
Now we want to compile this file into bytecode and then run it with the Java JVM.

Mac/Linux
Open up the Terminal app. To do this, browse out to the Applications folder then click on Utilities. You will find the Terminal app under this folder. Go ahead and open it.

Windows Not 8
Go to Start and press run. Then type cmd into this box.

Windows 8
Press the Windows key, type "Command Prompt" and press enter

We now need to find your Java files with this handy terminal.
Mac and Linux terminals usually start in the home directory for the logged in user. For example, a home directory of Bob is /Users/Bob
Windows command line would start in C:\Users\Bob

Now we need to get from the home directory to the Java files using the cd command (cd means “Change Directory” directory meaning folder).

Example:
Starting directory: /Users/Bob
Java files are on the Desktop.
So we need to do: cd /Desktop/

Command Line Protip: To see which files are in your current directory type “ls” on Linux/Mac or “dir” on Windows. This will help you figure out if you are in the right spot or not!

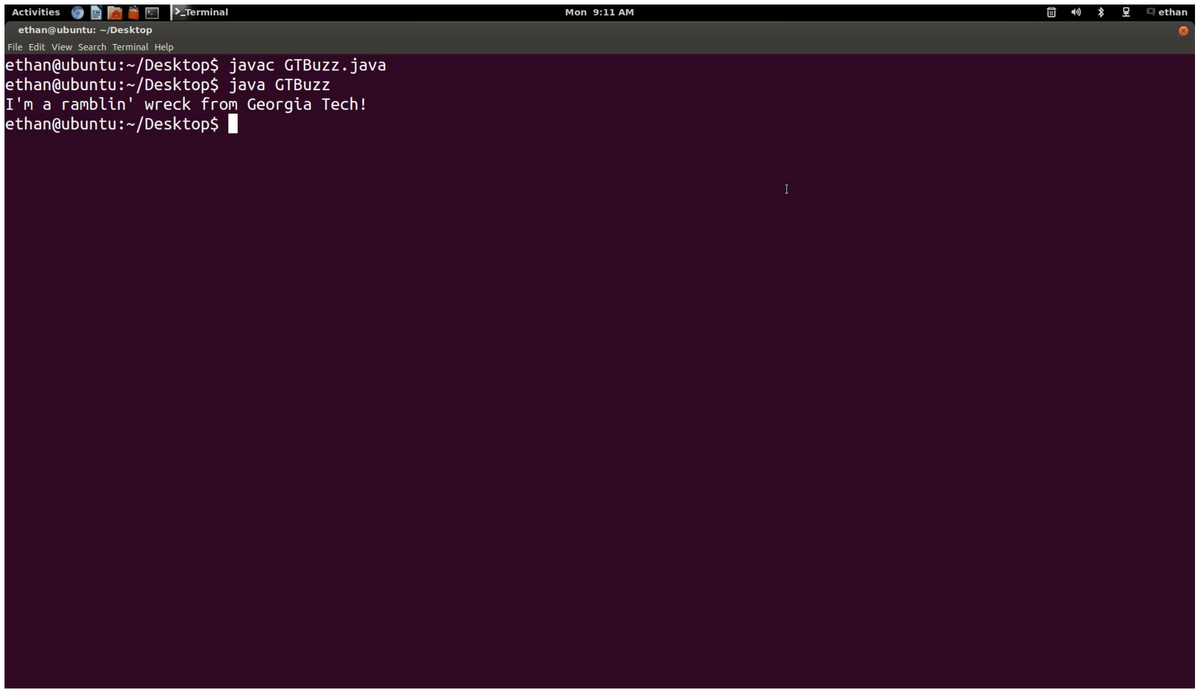
Now we are ready to compile and run our Java code.

To compile Java files type javac GTBuzz.java

Then if all goes well you will see another prompt. If text is outputted, then an error occurred.

This will generate a GTBuzz.class file which is the bytecode we mentioned before.

To run your application type Java GTBuzz and press enter. You should see the following text. Don’t type Java GTBuzz.class.



```
ethan@ubuntu:~/Desktop$ javac GTBuzz.java
ethan@ubuntu:~/Desktop$ java GTBuzz
I'm a ramblin' wreck from Georgia Tech!
ethan@ubuntu:~/Desktop$
```

Good job! You have finished your first Java program. The next part of the assignment involves changing things around and figuring out errors.

Note: If you are using an IDE like jGrasp or BlueJ, there will most likely be a button to compile and run your code. I won't give specific instructions here but you should be able to find these buttons in the documentation of the IDE.

Experimenting with Errors

We will be using the GTBuzz.java file that you just created and experiment with a few errors.
For this, put your answers in a HW1Errors.txt file.

For each error you need to do the following:
1. Start fresh from the GTBuzz.java file.
2. Make each change independently from the others.
3. Save the file as GTBuzz.java
4. Attempt to compile your code.
5. If you have an error message then note it in the HW1Errors.txt file.
6. If no error was given, compilation was successful. Now try to run the program.
7. If there was no error then state that there is no error.
8. If there was an error running the program, then state what the error is the HW1Errors.txt file.
9. IMPORTANT: Fix what you changed in each experiment to be a fresh and working GTBuzz.java file.
10. Repeat the process for all experiments.

Experiments

1. Change public class GTBuzz { to public class GTbuzz {
2. Insert the line: int x = 3 / 0; underneath System.out.println
3. Change System.out.println to println
4. Remove the first quotation in the string "I'm a ramblin … "
5. Change "I'm a rambling wreck from Georgia Tech." to "How bout them dawgs."
6. Remove the semicolon at the end of the System.out.println statement.
7. Remove the last curly brace at the end of the program.

Remember to revert to the original code, save, compile, and run to make sure the program is working before going on to the next experiment.

When recording possible errors, make sure to clearly indicate whether the error is a compile error or runtime error, and record the error that was thrown.

Here are some sample answers:
Runtime error: ArithmeticException on line 5
Compile-time error: ';' expected on line 3
No error.

Turn in Procedures

Turn in the following files on T-Square. When you're ready, double check that you have submitted and not just saved as draft. You should get an email confirming your submission. If you don't get the email, then you didn't submit.

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 time is 8 pm. Do not wait until the last minute!