CSE/IT 213: Homework 0

Writing Java as C
or how to write really awful Java programs

Due: Wednesday, Jan. 31 @ 11:59pm
This homework in individual.

The goal of this homework is three-fold: introduce C features that you already know in Java, Java I/O, and Eclipse. However, the way you are going to write the programs – everything in the main function – is not good programming style, but serves the purpose for the first homework.

Preliminaries

Projects and Packages

For each homework, you create one project and multiple packages within the project. Each package corresponds to a homework problem.

Package Naming Scheme

For purposes of this class, you will use the following package naming convention:

com.first_initial + last_name.hwX.number

where first_initial is the initial of your first name, X is the homework number, and number is the problem number spelled out. For example, Java Jones would name her packages for the first problem of this homework as:

com.jjones.hw0.one

We will follow this convention throughout the course as it makes grading easier. Package names are always lower case.

File Organization

Every file is organized as:

package name

import statements

class javadoc (see below)

class

See the Google style guide for more information (item 3).

Class Javadoc

Every class is preceded with a Javadoc comment. The format is:

```
/**
* A brief description of what the class does. 80 characters or less.
*
* <p>If needed, a new line and a longer description of the class.
* Javadoc recognizes html tags. Use <p> to separate paragraphs, etc.
*
* @author Your Name
* @version hw X, #Y //where X is the homework number and Y is the problem number
* @bugs list any bugs in your code here. Javadoc does not recognize
* this tag. Use it to indicate any issues with your code. If there
* are no bugs with your code, write None.
*/
```

The Javadoc occurs after the import statements and before the class declaration.

Methods of classes need to have Javadoc comments as well, but are not needed for this homework and will be described in the next homework.

Class Names

Unless specified, you are free to name class names as you see fit.

Problems

For each problem, follow the format for file layout, package naming, and commenting your classes. Write everything inside the main method. Chapter 3 of *Core Java* provides the relevant background.

1. FizzBuzz

   Write a program that prints the numbers from 1 to 100. But for multiples of three print Fizz instead of the number and for the multiples of five print Buzz. For numbers which are multiples of both three and five print FizzBuzz.

   For numbers and Fizz use the `println` method. For Buzz use the `print` method. And for FizzBuzz use the `printf` method.

   Initialize the array with numbers from 1 to 100 using a C style for loop. The loop for printing use a “for each” loop.

   Sample Output

   ```
   1
   2
   Fizz
   4
   Buzz
   Fizz
   7
   8
   Fizz
   Buzz
   11
   Fizz
   13
   14
   FizzBuzz
   ....
   ```

   This problem comes from
   `http://www.codinghorror.com/blog/2007/02/why-cant-programmers-program.html`.
   Before you read the blog post, solve the problem first.

2. The Guessing Game

   Implement the guessing game, “I am thinking of a number between x and y” and have the computer, using a binary search, correctly guess the number. Use a Scanner object to get input from the user. Prompt the user for the range their number lies in. Error check that $x \leq y$. After getting the range, the computer prompts the user with questions to determine the number. Print out the number when it is found.

3. Alice

   On Canvas is the file `alice.txt`, which contains the text of *Alice in Wonderland*. Place this file in the root of your project directory.

   Section 3.7.3 of *Core Java* describes how to perform basic file I/O in Java. Using a Scanner object for input and a PrintWriter object for output, transform the lines of the file so the first line is all lower case, the second line is all upper case, the third line is lower case, the fourth line is upper case, and so on. Ignore blank lines in your conversion. If the line before the blank line is converted to lower case, the blank line is ignored, and the next line is converted to upper case and vice versa. While blanks line are ignored for case conversion, they are outputted, keeping the formatting between the original and the copy of the file the same. Name your output file `alice_in_mixed_case.txt`.

If your Scanner object is named `in`, you can loop through the file using a while loop:

```
while (in.hasNextLine()) {
    ...
}
```

Read the Java API documentation on the `Scanner.nextLine()` method. Since the method returns a line without line separators, this means if you use the `nextLine()` method to get the next line in the file, and the line is blank, the string returned by the method is empty.

Submission

Create a jar file of all your packages. To do this in Eclipse, goto File->Export. In the dialog box, navigate to Java and select JAR file. This opens up another dialog box. Select the resources to export. The only resources you need to select are all the packages of your project. Do not include any `.classpath` or `.project` files or `.settings` directory. Uncheck "Export generated files and resources." and select "Export Java source files and resources". Name your jar file:

`cse213_firstname_lastname_hw0.jar`

Click Finish.

You can test that the jar file was successfully created by creating a new project and import the jar file into the project. (File->Import, General->Archive File). Make sure to import it into the src directory of your project.

Upload the jar file to Canvas. Test your jar file before you upload it to Canvas.

For more information on jar files, essentially Java's version of a tar file, see:

`http://docs.oracle.com/javase/tutorial/deployment/jar/basicsindex.html`