p0 Java IO Practice Pair or Team Programming: **NOT ALLOWED**
Programming III (CS400) Spring 2020 Due on or before: **Feb** 6^{th}

Overview

Learn or relearn key Java programming language techniques for interacting with the user (input and output) with a console based Java program. For full credit your program must interact with the user using `Scanner` and `PrintWriter` instances and by asking for and using user input from the keyboard and input file.

Learning Objectives

1. Practice your incremental program development skills.
2. Create and develop a Java programming project of your own design from scratch.
3. Create and design an interactive program for the user.
4. Display prompts and program output to the user via the print and println methods of the System.out object.
5. Get and use input that the user types at the keyboard using a `Scanner` connected to System.in
6. Read and use the contents of a text-only file using a `Scanner` .
7. Write program information and results to a text-only file using `PrintWriter`
8. Write clean code with file and method method headers and comments.

Pro Tip : Remember this program and keep it handy when you need to experiment with reading and writing input in other later and larger programs.

Links

Review these links for additional help using `Scanner` and `PrintWriter` for your Java IO needs.

- Deb's Scanner Handout
- Java User Input w3schools
- java.util.Scanner api
- java.io.PrintWriter api

Program Requirements

1. **Creative** Must use user input to compute and produce some type of output to the user. Input and output must be different than other students. Do not share your ideas. It can be short and sweet, just make sure that you meet all requirements and get your name, email, and lecture number on your work. Your submission will be compared against other submissions for originality.

   Example ideas Display a menu and allow the user to select any of several actions to complete, repeat until use select exit. Create a detailed computation based on multiple user inputted values. Play a simple joke. Solve a simple puzzle or text game with the user. This is a chance to show what you know how to make Java do.

2. **Interactive**: Must interact with the user in multiple ways, with a minimum of three inputs and three outputs, including the following:

3. **Standard Output**: Must use `System.out.print` and `System.out.println` to write prompts and output to screen (console window).

4. **Standard Input**: Must use a single `Scanner` instance connected to the standard Java input stream `System.in` to read input typed by user.

5. **File Input**: Must read lines of text from a file using a `Scanner` connected to the file and do something besides echo to screen. Do not hard-code the file name. Do get the file name from the user.

6. **File Output**: Must use a `PrintWriter` to write data and other program results to a local file. May hard-code the output file name as `output.txt`, or prompt user for output file's name.

7. **Comments and Style**: Must include file and method headers comments, be modular (use private methods) to make it easy to read, maintain and improve, and be clean, consistent, not-redundant and make good use of vertical and horizontal white space. Avoid magic numbers and make your program something that you are proud of.
   Follow Google Java Style Guide if you are unsure.

Getting Started

1. Read entire assignment
2. Create a Java project folder named `p0_JavaIOPractice`
3. Add a Java class named `Main`
4. Add a file header that includes your name, your *wisc.edu* email, your lecture number, and a description of what your program does.
5. Add a `private static final Scanner` field to the class and initialize it with a new `Scanner` instance connected to the standard Java input stream `System.in`. Use this instance to get any keyboard input from the user that your program needs.
6. Add a `private final String title` field to the class and initialize it with a string that contains your name, your wisc.edu email address, and your cs400 lecture number.
7. Add a `public static void main(String [] args) method` to the class
8. Add a print statement to your `main` method that prints the contents of your program's title field.
9. **Test and submit to Canvas. Check your submission.**
10. Add a code to ask the user for some information.
11. Display some output to the user based on the information you requested.
12. Run your program and see what happens if user answers with expected information and what happens if the user does something unexpected.
13. Add exception handling so that the user can not crash your program.
14. **Test and submit to Canvas. Check your submission.**
15. Add code to get a file name from the user and read and use the contents of the file for other output. Note: reading from a file using `Scanner` connected to the file.
16. Add code to write some results to a file using `PrintWriter` .
17. Repeat until complete:
    (a) Reread entire assignment and rubric, complete your work.
    (b) **Test and submit to Canvas. Check your submission.**

p0 Questions

After you have practiced creating and using a `Scanner` instance, experiment with the various `Scanner` methods. When you understand how `Scanner` works on keyboard and file input streams, you should be able to answer and explain your answer for each of these questions.

1. What is the *newline character* in Java?
2. What is *white-space* in user input?
3. Answer these questions for each of these `Scanner` input methods: next(), nextLine(), nextInt(), nextDouble()?
   (a) Does the method skip *white-space* to "look" for the next input value?
   (b) Does the method return a single word or the rest of the line?
   (c) Does the method leave the data on the *input stream* or consume it?
   (d) Does the method leave the *newline character* on the input stream?
   (e) Does the method return the *newline character* with the input value?
4. What happens if you connect `PrintWriter` to an existing file? Does it fail, overwrite, append?
5. What happens if you do not have write permissions to the file (and path) used to create the `PrintWriter` instance?

Files to Submit

1. `Main.java` Your main program source.
2. `*.java` Other source files you have written.
3. `input.txt` An example input file your program can read.
4. `output.txt` A file that your program produced.
5. `log.txt` A file showing the input/output sequence that your program produced when you ran it to show the interaction. **Pro Tip** Copy and paste lines from the console window to a new text file to create the `log.txt` file.