CSCI 1011 – Lab 12

Learning Outcomes

- Implement code that handles exceptions.
- Read and process data from text files.
- Write output to text files.

Required Reading

Savitch - Sections 9.1, 10.1-10.2

Instructions

1. Start NetBeans.
2. Create a new project called `Lab10` with a main class called `YournameLab10` with your name.
3. Write code to create and write to an empty file.

   (a) Write a public static method called `openFileForWriting` that takes a `String` representing a file name and does the following:

   - Create a new `PrintWriter` object using the given file name.
   - If a `FileNotFoundException` is thrown, display an error message and exit the program.
   - Otherwise, return the `PrintWriter` object.
   - Make sure to add import statements for `java.io.PrintWriter` and `java.io.FileNotFoundException`.

   (b) Write code in the `main` function to test `openFileForWriting`.

   - Use `openFileForWriting` to create and open a text file (you can call it what you like, but it should have a `.txt` file extension.)
   - Display a message to the user that the file was opened for writing.
   - Use the `PrintWriter` object returned by `openFileForWriting` to write some lines to the file.

- Display a message to the user that output was written to the file.
- Close the file.

(c) Test the program to see if it works.

- The output should look something like this:

```
Opened file testfile.txt for writing
Wrote 2 lines to testfile.txt
```

- To inspect the file, click the **File** menu, select **Open File...**, find the `Lab10` folder in your workspace, click on the file you created, and click **Open**.

4. Write code to append to an existing file.

(a) Write a public static method called `openFileForAppending` that takes a `String` representing a file name and does the following:

- Create a new `PrintWriter` object from a new `FileOutputStream` using the given file name, making sure to indicate that the file is being opened for appending.
- If a `FileNotFoundException` is thrown, display an error message and exit the program.
- Otherwise, return the `PrintWriter` object.
- Make sure to add an import statement for `java.io.FileOutputStream`.

(b) Write code in the `main` function to test `openFileForAppending`.

- Use `openFileForAppending` to open the text file from the previous step.
- Display a message to the user that the file was opened for appending.
- Use the `PrintWriter` object returned by `openFileForAppending` to write some lines to the file.
- Display a message to the user that output was written to the file.
- Close the file.

(c) Test the program to see if it works.

- The output should look something like this:

```
Opened file testfile.txt for writing
Wrote 2 lines to testfile.txt
```

```
Opened file testfile.txt for appending
Write 2 lines to testfile.txt
```

- Check the file to see if the additional lines were added.

5. Write code to read from a file.

   (a) Write a public static method called `openFileForReading` that takes a `String` representing a file name and does the following:

   - Create a new `Scanner` object using a new `File` object using the given file name.
   - If a `FileNotFoundException` is thrown, display an error message and exit the program.
   - Otherwise, return the `PrintWriter` object.
   - Make sure to add an import statement for `java.io.File`.

   (b) Write code in the `main` function to test `openFileForReading`.

   - Use `openFileForReading` to open the text file from the previous two steps.
   - Display a message to the user that the file was opened for reading.
   - Use the `Scanner` object returned by `openFileForReading` to read and display all of the lines of the file.
   - Display a message to the user that input was read from the file.
   - Close the file.

   (c) Test the program to see if it works.

   - The output should look something like this:

```
Opened file testfile.txt for writing
Wrote 2 lines to testfile.txt

Opened file testfile.txt for appending
Write 2 lines to testfile.txt

Opened file testfile.txt for reading
first line of file
second line of file
third line of file
fourth line of file
Read 4 lines from testfile.txt
```

   - Make sure the file was not modified.

6. Write a method to read text from a file and display it to the screen.

   (a) Write a method called `readLinesFromFile` that takes a `Scanner` object as a parameter and does the following:

      - Read each of the lines of the file and displays them.
      - Keep track of the number of lines read and return the result.

   (b) Write code in the `main` method to test the `readLinesFromFile` method.

      - Replace the code in the `main` method that displays the contents of the file with a call to `readLinesFromFile`.
      - Use the value returned in the message that displays how many lines were read.
      - Add code to open, read, display, and close the file after the code for writing lines to the empty file but before the code for appending lines to the file.

   (c) Test the program to see if it works.

      - The output should look something like this:

```
Opened file testfile.txt for writing
Wrote 2 lines to testfile.txt

Opened file testfile.txt for reading
first line of file
second line of file
Read 2 lines from testfile.txt

Opened file testfile.txt for appending
Write 2 lines to testfile.txt

Opened file testfile.txt for reading
first line of file
second line of file
third line of file
fourth line of file
Read 4 lines from testfile.txt
```

7. Write a method to read from the keyboard and write it to a file.

   (a) Write a method called `writeLinesToFile` that takes a `PrintWriter` object as a parameter and does the following:

      - Display a prompt to the user to enter lines of text to write to the file and to enter a blank line to stop entering text.

- Read lines from the keyboard and write them to a file until a blank line is entered.
- Keep track of the number of lines written and return the result.

(b) Write code in the `main` method to test the `writeLinesToFile` method.

- Replace the code in the `main` method that writes lines to the empty file with a call to `readLinesFromFile`.
- Use the value returned in the message that displays how many lines were written.
- Do the same for the code that appends lines to the file.

(c) Test the program to see if it works.

- The output should look something like this:

```
Opened file testfile.txt for writing
Enter the text you want to write to the file. Enter a blank line
when you are done.
'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:

Wrote 2 lines to testfile.txt

Opened file testfile.txt for reading
'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
Read 2 lines from testfile.txt

Opened file testfile.txt for appending
Enter the text you want to write to the file. Enter a blank line
when you are done.
All mimsy were the borogoves,
And the mome raths outgrabe.
Wrote 2 lines to testfile.txt

Opened file testfile.txt for reading
'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.
Read 2 lines from testfile.txt
```

8. Hand in your source file, `YournameLab12.java` to the D2L assignment dropbox called **Lab Assignment 12**.