PGCert IT: Programming for Industry

Input and Output

Exercise One: Simple IO Problems

1. Consider the file input1.txt, which is located in the lab folder. What is the output of the following code?

```
private void fileReaderEx01() {
  int num = 0;
  FileReader fR = null;
  try {
    fR = new FileReader("input1.txt");
    num = fR.read();
    System.out.println(num);
    System.out.println(fR.read());
    System.out.println(fR.read());
    System.out.println(fR.read());
    System.out.println(fR.read());
    fR.close();
  } catch(IOException e) {
    System.out.println("IO problem");
  }
}
```

2. Complete the `printNumEsWithFileReader()` method in the `ExerciseOne` class, which should use a `FileReader` to read input2.txt and print out the following information:

a. The total number of characters in the file, and:

b. The number of e's and E's in the file.

3. Repeat step 2, but this time complete the `printNumEsWithBufferedReader()` method, and use a `BufferedReader`.

Exercise Two: PrintWriter & BufferedReader

1. In the ex02 package, you'll find the MyWriter class. Currently, this lets a user enter a file name, then type as many lines of text as they wish, using the keyboard. For this task, modify the start() method so that it properly uses a PrintWriter to write every line the user types to the file that the user specified.
    - **Hint:** Try to only open / close the file once, but you can write to it as many times as you need to.
    - **Hint #2:** You can check that your program is working as intended by looking at the output file in a text editor.
2. Now, complete the MyReader class, which should again prompt the user to enter a file name, then should use a BufferedReader to open the file, read all the text in that file, and print it out. Check that it works by reading one of the provided input text files, and / or your own output from part 1 of this exercise.
3. In the MyScanner class, create another program for printing out the contents of a text file. The functionality should be identical to that in part 2. This time, use a Scanner.

Exercise Three: DataInputStream & DataOutputStream

In the ex03 package you'll see the Movie class from a previous lab. In this exercise, we'll create two programs: the first will create a file containing data for lots of movies, while the second one will read that file and create Movie objects.

1. Complete MovieWriter's saveMovies method. This method should use a DataOutputStream to write the contents of the films array to the given file. **Note:** You should not make any assumptions about the number of movies in the array – your program (and the one in part 2) should work for any number of movies! It might pay to *write the number of movies to the file*, as well as the movies themselves.
2. Complete MovieReader's loadMovies method. This method should use a DataInputStream to read the contents of the given file and use it to create, fill and return an array of Movie objects.

Exercise Four: CSV Files

In this exercise we'll use something other than DataInputStream and DataOutputStream to load and save movies.

1. In the ex04 package you'll see Ex4MovieWriter, which extends exercise three's MovieWriter class, and expects a different implementation of saveMovies. Implement this version of the method so that it uses a PrintWriter to output a CSV-style file, with each line in the file representing a separate movie.

2. In the ex04 package you'll see Ex4MovieReader, which extends exercise three's MovieReader class, and expects a different implementation of loadMovies. Implement this version of the method so that it uses a Scanner to successfully read the files generated by part 1 of this exercise.

For more information on CSV files, you can visit this link.