Java Practical Exercise 2 – Spring 2021

This is a description of the first Java practical exercise for CSCU9T4 in Spring 2021. It is expected that you will work on this over approximately 2 weeks, on your own and in groups during weekly live sessions.
The aim of this practical sessions is to exercise your skills with structured data, use of inheritance, recognising good and bad coding practices, and working with a version control system (Git).

1. The Program

Your task in this practical is to implement a command line program in Java that takes an input text file, transforms it, and outputs it to another file. The program should take the following command line arguments:

- An optional -u flag to indicate all upper case instead of Title case
- The input filename
- The output filename

For example, running

```
Java FormatNames input.txt formatted.txt
```

takes the lines from **input.txt**, formats the data using Title case and places the results in the file **formatted.txt.** (In BlueJ this is equivalent to typing the arguments when you invoke a call to main to run the code. Strings need to be enclosed in quotes.)

The input has the following format:

```
allison wesley 28011990
peter smith 05071992
```

(A list of names and dates of birth, in lowercase and without formatting. See **input.txt**)

The output has the following format:

```
Allison Wesley      28/01/1990
Peter Smith         05/07/1992
```

This is Title case (i.e Allison Wesley), and the date formatted as 'dd/mm/yyyy'.

If the –u flag is indicated then running the program looks like

```
Java FormatNames –u input.txt formattedu.txt
```

This takes the lines from **input.txt**, formats the data using UPPER case and places the results in the file **formatted.txt**. For example, the first two lines of **formatted.txt** should look like:

```
ALLISON WESLEY      28/01/1990
PETER SMITH         05/07/1992
```

2. Git: Setting up the exercise and submitting the checkpoint.

This part is just about accessing the initial reduced version of the program which you will extend.

a) If you haven't already, sign up for a GitHub account. Preferably using your Stirling student email address so that we can be sure it is your account.
b) Go to this repository: https://github.com/saemundur-haraldsson/CSCUT4Practical2

c) Fork the repository to your own account or create a new repository and upload the java file and csv files.
d) [Recommended] Create a new branch

Checkout a copy of the new branch to your machine and start working on the implementation

To submit the checkpoint, you find the relevant assignment/checkpoint on Canvas and submit the URL to your repository on GitHub and then you provide peer review when that option becomes available after the deadline. Same as last checkpoint.

3. The specific tasks

1. Start by implementing the simple Title case version, then add the upper case version.

2. Twist: some people have middle initials. See inputm.txt :

```
allison m wesley 28011990
peter smith 05071992
```

Create another version of your program named `FormatNamesm`. Modify it to handle middle initials by capitalising them and adding a full stop. For example, the lines in inputm.txt listed above should be formatted in the output file as follows (assuming running without the u flag):

```
Allison M. Wesley      28/01/1990
Peter Smith            05/07/1992
```

As you see from this example, not everyone has a middle initial, so your program needs to handle both cases.

Running the command

```
Java FormatNamesm –u inputm.txt formattedmu.txt
```

should produce the following content in the formattedmu.txt file

```
ALLISON  M. WESLEY   28/01/1990
PETER SMITH          05/07/1992
```

3. (optional, but recommended) Instead of using the command line, ask the user for the file name for input and for output. This is straightforward, using `System.out.println`. But what if the file for input does not exist, or if the file for output cannot be opened? The code overleaf is supplied for reading and for writing. Adapt this code so that the user is asked for a file name until she supplies one that works, both for input and output.

4. (optional) Modify the command line program so that you take an alternative flag:

- An optional -h flag to indicate that the output should be formatted as HTML, so that it can be viewed in a browser. The output file name should be appended .html in this case.

This is a matter of appending the right HTML tags to the output you have already. As above, start simple: can you create a basic web page that displays the information on separate lines? Then you might get more creative with display colours and layout. You may also limit the user to a certain number of entries.

Code for filenames:

```
System.out.println("supply filename for input:");
try {
      inputFileName= in.nextLine();
      File inputFile = new File(inputFileName) ;
      Scanner inFile = new Scanner(inputFile);
      } catch (IOException e) {
           System.err.println("IOException: " + e.getMessage()
                                + "not found");
           }

System.out.println("supply filename for output:");
try {
      outputFile = new PrintWriter(filename);
      } catch (FileNotFoundException e) {
             System.err.println("FileNotFoundException: " + e.getMessage()
             + " not openable");
           System.exit(0);
           }
```

Or alternatively:

```
try {
  File inputFile = new File(inputFileName);
  if(inputFile.exists() ==false || inputFile.isDirectory() == true) {
       System.out.println("Please provide a valid input file name.");
       return;
}
Scanner sc = new Scanner(inputFile);
File outputFile = new File (outputFileName);
if(outputFile.exists() ==true && outputFile.isDirectory() == true) {
        System.out.println("Please provide a valid output file name: A
directory with same name exist");
       return;
       }
PrintWriter printWriter = new PrintWriter (outputFile);
```