COP3503 Project 2 – Data Preprocessing

Objectives:

- Show an understanding of how to
  - Use an input stream to read in data from a delimited text file
  - Use existing methods to convert dates into different formats
  - Use an output stream to create a delimited text file
  - Catch and recover from exceptions that can occur

Submission Requirements:

- Submit your project via the Canvas course
  - Upload Java file containing your source code for the project
    - Filename: Project2_n#
    - Example: Project2_n00123456
  - Upload Flowchart file for main() method
    - Filename: Project2_Flowchart_n#
    - Example: Project2_Flowchart_n#
- Project requires 2 files to be submitted

Design Documentation Requirements: 10 points

- Flowchart of the main() method
  - Should reflect the method calls made in the main() method
  - Does not need to reflect any exception handling

Design Specification Requirements: 90 points

Use the Project2_template.java file as an outline for the project.

1. Display the project title, followed by a blank line
   a. This should only be displayed 1 time when the program executes
2. 20 points: Prompt the user to input the file name and location
   a. If a FileNotFoundException is thrown the file does not exist or path is incorrect inform the user and prompt for new input file
   b. Entering incorrect input should not cause the program to crash
   c. The file Speed_Data.csv must be used for this project
   d. The file contains 6 columns:
      i. Date, Time, Sensor_2278, Sensor_3276, Sensor_4689, Sensor_5032
3. 25 points: Read the data in from the file and store each column in its own ArrayList
   a. If a NumberFormatException is thrown inform the user the file contains bad number input and prompt the user for a new input file
4. 20 points: Convert the date column format from MM/DD/YYYY to YYYY/MM/DD
   a. If a ParseException is thrown inform the user the file contains bad date data and prompt the user for a new input file
5. Subtract each row of sensor3276 speed data from sensor2278 speed data and store the difference in an ArrayList
6. Subtract each row of sensor5032 speed data from sensor4689 speed data and store the difference in an ArrayList
7. Calculate the average speed for each row of data and store the average in an ArrayList
   a. Columns Sensor_2278, Sensor_3276, Sensor_4689, Sensor_5032 store the speed data

8. 25 points: Output the data from all 9 ArrayList into a new csv file
    a. The new file name must be "fileNameInput"_Difference.csv
        i. Example: If file name entered is Speed_Data.csv the new file should be named Speed_Data_Difference.csv
    b. The data must be comma separated
    c. The order of the columns must be:
        i. Date, Time, Sensor_2278, Sensor_3276, Sensor_4689, Sensor_5032, Section1_Diff, Section2_Diff, Total_Avg
        ii. Column headers are required

Note: Refer to the sample output in the **Example Output** section below.

Minimum Requirements: -100 points

- Use must use at least 1 try-catch in your project
    - Your try-catch/s should catch, handle, and recover from FileNotFoundExceptions, NumberFormatExceptions, and ParseExceptions
- Use must use ArrayList to store the file data and calculated data

Additional Notes:

- Use the FileReader class wrapped with the Scanner class to read data in from file
    - You may want to read in each line and then split the data on the comma to load them into their respective ArrayList
- Use the SimpleDateFormat class to convert the date format
    - Make sure to use the Date class from the Java utility package not the Date class from the Java SQL package
- Use the FileWriter class wrapped with the PrintWriter class to output data to the new file
    - Use the println() method to write to the file line by line
- If an exception is thrown make sure to clear any data from all the ArrayList
- When using Scanner(System.in) to get input from the user do not close the input stream until the program exits
- Follow the commenting and programming guidelines outlined in the Commenting and Programming guide documents
    - Failure to do so will result in up to a 25-point reduction

Example Output

Good Input Example:

```
Project 2 Data Preprocessing

Enter file name & location.
Speed_Data.csv
Reading in Data from the file Speed_Data.csv
Converting Dates from MM/DD/YYYY to YYYY/MM/DD
Calculating Speed Difference
Calculating Speed Average
Writing data to file Speed_Data_Difference.csv
Done! Exiting Program
```

Bad Input Example 1:

```
Project 2 Data Preprocessing

Enter file name & location.
Speed_Data.csv
Reading in Data from the file Speed_Data.csv
*Bad Number Data in CSV File.*
Check CSV file data and try again.
Enter file name & location.
```

Bad Input Example 2:

```
Project 2 Data Preprocessing

Enter file name & location.
SpeedData.csv
Reading in Data from the file SpeedData.csv
*File does not exist or path was entered incorrectly.*
Please try again.
Enter file name & location.
```

Bad Input Example 3:

```
Project 2 Data Preprocessing

Enter file name & location.
Speed_Data.csv
Reading in Data from the file Speed_Data.csv
Converting Dates from MM/DD/YYYY to YYYY/MM/DD
*Bad Date Data in CSV File.*
Check CSV file data and try again.
Enter file name & location.
```