CSE 8B Homework 3:
Arrays

Learning Goals:

- Write code to iterate through Arrays
- Calculate statistics based on real-world data stored in Objects
- Write test cases to test your methods
- Write code to build classes in Java including:
    - Constructor
    - Getters and Setters
- Learn how file I/O works in Java (optional, will be covered next week)

NOTE: This PA uses content from Zybook assignments 1, 2, and 3. **You should finish these chapters before attempting to start this PA.**

Before you start: Download and Understand the Starter Code

Unlike previous PAs, in this PA, you will start with some code that we have written for you. **Please download the starter code provided here.**

This is what your file structure should look like:

```
+-- PA3/
|     +-- COVID_19_Data.csv
|     +-- COVID_19_Data_Raw.csv
|     +-- DataPoint.java              Edit this file
|     +-- PA3Library.java
|     +-- CovidCalculator.java        Edit this file
|     +-- PA3Tester.java              Edit this file
```

It is very important to organize the files like this in order for the provided Library methods to work correctly.

Here is a description of each file in your `PA3` directory:

1. `COVID_19_Data_Raw.csv` - contains a raw dataset of COVID-related statistics for how people of various ethnicities have been impacted by COVID. **This file should not be changed in any way. You do not even have to use this file.** (If you wish to work with this dataset, then you will receive star points, but the data will need to be *significantly* cleaned).

2. `COVID_19_Data.csv` - a cleaned up version of the above dataset. This file should not be changed in any way. This is the file with the data you will use in PA3.
3. `PA3Library.java` - code to read the data from the file into an array of DataPoints. This file should not be changed in any way. The code that uses this library is already implemented. You will not have to call any of these methods.
4. `DataPoint.java` - a class to represent an entry in the provided dataset. This class contains a stub implementation of the methods (see zyBooks 6.8). **You will complete these methods.**
5. `CovidCalculator.java` - computes various statistics about how certain groups have been impacted by COVID. See below for further information. **You will complete this class.**
6. `PA3Tester.java` - a user-implemented tester for the above methods. You will complete this. As this will be the "driver" that will use the other classes, it will be the only class with a main method.

When you get the starter code, the code will compile and run, but the code will not do anything useful until you implement the code for the assignment. You should verify that the code does compile and run using the following commands:

```
1. javac *.java
2. java PA3Tester.java
```

Reminder: Coding Style (0.5 points)

When coding in Java, there are several style rules that people usually follow to make the code clean and readable. In this course, you are asked to follow rules specified in link below:

https://cseweb.ucsd.edu/classes/fa20/cse8B-a/styleguide.html

(as linked on Canvas). Read the coding style guide carefully and refine your code for this and all future assignments. Sometimes, coding style can help debug your code, so it's a win-win!

Part 1: Finish Implementing the DataPoint class (Pair Programming allowed) (3.5 points, class and tests)

A `DataPoint` object will represent a row in the provided CSV file. As such, the DataPoint class should have the following member variables:

- `String date`: The date in the form of `year month day` (*e.g.* January 13, 2021 will be formatted as `20210113`)
- `String state`: The state abbreviation
- `int totalCases`: Total number of cases reported that day
- `int[] casesByRace`: The number of cases per race on a given day

This class should contain the following *public* methods. **You must name your methods *exactly* as specified below:**

One Constructor

1. `public DataPoint(String dateIn, String stateIn, int totalCasesIn, int[] casesByRaceIn)`
   This constructor should be used to initialize all the member variables listed above. It is OK here for `casesByRace` *not* to make a copy of the array passed in, but to simply set `casesByRace` to refer to the same array as `casesByRaceIn`.

4 Getter Methods

1. `public String getDate()`
   This method returns the `date` exactly as it is stored in the `DataPoint` object

2. `public String getState()`
   This method returns the `state` exactly as it is stored in the `DataPoint` object

3. `public int getTotalCases()`
   This method returns the total number of cases (`totalCases`)

4. `public int getCasesByRace(int index)`
   This method returns the number of cases for the race stored at the given index position.

IMPORTANT: Compile and run your code often. We recommend writing and running your DataPoint class to ensure that it works before moving onto the next step.

Test your DataPoint class

For your testing, it is important that you NOT just use the full CSV file we have provided. Instead, you will test your code first by creating two `DataPoint` objects directly in your code and then calling methods on them.

We have provided a method in the PA3Tester class named `testDataPoint`. This method is called from the `main` method in `PA3Tester`. **Here is what you need to do:**

1. Understand the provided code. Notice that the code creates a made-up `DataPoint` object for the purpose of testing, and then runs tests to be sure the `DataPoint` constructor and getter methods are working.

2. Add a second `DataPoint` object with different data, and then add code to test this second object. You may copy and paste the code provided and modify the code to create this second test.
3. Compile and run `PA3Tester` to ensure that all of the tests pass (instructions for compiling and running are below). You need to manually check to make sure that the actual values match the expected values. Fix any errors and repeat this step as necessary.

To compile and run:

1. Change directory to the directory where the files `DataPoint.java` and `PA3Tester.java` are located.
2. Compile both files together (**HINT:** you can list both filenames after `javac`, or just the one with the `main` method. Alternatively, you can run `javac *.java` to compile ALL Java files).
3. Run `PA3Tester` (you cannot run your `DataPoint` class as that class has no `main` method) and manually verify that your `DataPoint` class works as expected.
4. Fix any bugs and repeat as necessary.

Part 2: Implement CovidCalculator (Pair Programming allowed) (6 Points, class and tests)

In `CovidCalculator.java`, you will be calculating different statistics about the dataset. **You will be writing a total of 4 methods and can write additional methods for a star point.**

You are provided a mostly empty `CovidCalculator` class in the file `CovidCalculator.java`. Complete or add the methods specified below.

Methods to be implemented

1. `public CovidCalculator(DataPoint[] inputData)`

   The constructor should initialize the member variable (the array of DataPoints) that statistics will be calculated for.

   Note: this constructor should *copy* the data in `inputData` into the existing array (which is created when points is declared) referenced by points. Therefore, it should NOT contain the following line:

   ```
           this.points = inputData; // THIS IS WRONG
   ```

   Instead it should use a loop to copy the data from `inputData` to `points`.

2. `public double findAverageCases(String date)`

Return the average number of total cases/infections (as a double) across all states on the specified date. *Be careful to cast `int`s to `double`s when taking the average, or your average will **not** be correct.*

3. `public String findRaceWithHighestCases(String date, String state)`
   Return the String representing the name of the race that has the highest number of infections on the given `date` in the given `state`.

4. `public [retType] myStat([args])`
   This method is up to you. You will implement one more method that calculates a statistic of your choice. Your method can return any type that you like (but not void), and your method can take any number and type of parameters (including none at all). **The only requirements are that this method contain at least a loop (either `for` or `while`) and an `if`-statement, and that this method uses both of these structures meaningfully in calculating the statistic.** The comment for this method must clearly describe what this method calculates. We will not autograde this method. **It will be up to you to demonstrate its functionality (see below).**

   Here are some ideas of what you might calculate:
   - The number of distinct states that have a total number of cases on a given data above some threshold.
   - The state with the biggest average difference between two given races across all dates.
   - The state with the fewest total cases in the data.

Test your CovidCalculator class and explore your data

It is important that you test your code both using COVID_19_Data.csv and a smaller `DataPoint` array. This will help you make sure that your program is working correctly.

We have provided you with the `testCovidCalculator` method in `PA3Tester.java` to help you do this. In the provided starter code, you are given `ccTest1` which is a `CovidCalculator` object created using a small `DataPoint` array. This smaller DataPoint array is populated using `allPoints` which is an array of `DataPoint` objects that represent the whole COVID_19_Data.csv. Then, `ccTest1` is used to test 2 of the methods in your CovidCalculator class.

To further test your CovidCalculator class and explore your data, do the following:

1. In the `testCovidCalculator` method:

   a. Create *at least 2 more tests* using `ccTest1` that test your `myStat` method.
   b. Create another test subset by creating another CovidCalculator object with a different DataPoint array. Then, write at least 2 tests for each method in the CovidCalculator class.
2. In `main`, below the existing calls to `findAverageCases` and `findRaceWithHighestCases` on the full data set, add one or more calls to `myStat` on the full dataset to explore this dataset. The only requirement here is to call `myStat` on the full dataset to calculate something that is interesting to you.

Screenshot your tests for submission

Finally, **take a screenshot** that clearly shows the complete results of running main in `PA3Tester`, **including the output of `testDataPoint()` from Part 1, `testCovidCalculator(DataPoint[] allPoints)`, and the results of running the three methods in `CovidCalculator` on the full data set.** You might need more than one screenshot. Paste your screenshot into a document that you can save as a PDF.

Grading Information

- 2.5 points for DataPoint correctness
- 1 point for DataPoint test implementation
- 5 points for CovidCalculator correctness
- 1 point for CovidCalculator test implementation
- 0.5 points for code style.

Submission

Please follow **ALL** of the instructions below to completely submit your assignment.

- Submit your code from Parts 1 and 2 (`DataPoint.java`, `CovidCalculator.java` and `PA3Tester.java`) on Gradescope using the "PA3 - Code" submission. Since you may have modified `PA3Library.java` to handle the entire dataset, please upload it along with the data files you used (`COVID_19_Data_Raw.csv`, `COVID_19_Data.csv`).
- Submit the PDF containing the screenshot(s) demonstrating your tests of DataPoint and CovidCalculator to Gradescope using the "PA3 - Tests" submission using this template