Regis University CC&IS
CS210 Introduction to Programming
Java Programming Assignment 8: Array Implementation

NOTE: This assignment will be ***more challenging*** than the previous assignments, so make sure to ***start early*** and ***allocate enough time*** to complete it, including time you might need to seek any necessary help.

Since this is the last program, the program **may not be submitted** more than **1 day** late!

Problem Summary

A local city would like you to write a program to help them analyze data on speeding tickets. City police have collected information about the speeding violations. They store data about each ticket issued during a one week period in a text data file. Data collected for each ticket includes the vehicle license plate number, the speed limit, and the clocked speed. From past data, the police department knows that it will not issue more than 200 tickets per week.

You may assume all data in input data file is valid data (i.e. formatted correctly), but you must verify that the file exists (i.e. can be opened). If the file does not exist, the program will loop until you enter the filename for a file that can be opened.

Sample input data file lines:
```
BBB222 50 60
XYZ123 40 45
A1B2C3 30 51
```

The police department would like you to write a program to read the file, calculate the fines to assess, and generate several reports. The program will:

- Read the data from the file, line by line, and store the data into an array that holds ticket data objects. This array will hold a maximum of 200 ticket data objects.
- Calculate the speeding ticket fines to assess, based on the ticket data, and store the fines into a second array that holds ***double*** values. This array will be sized to the number of tickets issued.
- Produce an output file, containing the vehicle license plate numbers and speeding fines.
- Analyze the fines to determine the high, low, and average fines, and display the results.

Here are the formulas the city uses for calculating a speeding ticket fine:

Ticket Fine = Court costs ($45.00)
+ a **fee** for each mile per hour (MPH) by which the speed limit was exceeded

The **fee** will be levied as follows:

MPH Over Speed Limit	Fee per MPH Over Speed Limit (for all MPH over limit)
up to 10	$ 4.25
over 10 and up to 20	$ 6.00
over 20	$ 8.10

You must ***define constants to hold all fixed values*** (court costs, MPH over speed limits, and fees per MPH) in the above chart within your method that calculates the fines.

Program Requirements

This program will implement two different types of arrays. One array will hold ticket objects, and will require an implementation class. The other array will contain primitive data type double values, and will not require an implementation class.

> NOTE: The ticket array and the fines array will be parallel arrays. This means that the fine will be stored into the fines array using the same index as the ticket object in the ticket array.

NOTE: For an example of a Java program implementing arrays, see Online Content section 15.10.

Required Classes and Methods

Three separate classes will be required for this program.

- A class to define a **SpeedingTicket** object
- A class to implement the object array (to hold **SpeedingTicket** objects)
- A ***main*** class to run the program

1. Create a new Java NetBeans project, named as follows:

   `LastnameJavaAssn8`
   For example: **`SmithJavaAssn8`**

2. Define a Java **class** with properties and methods for a speeding ticket object, named:

   `SpeedingTicket`

   The class will have the following **private** data properties:
   - ❖ vehicle license plate number (a 6-character String)
   - ❖ the speed limit
   - ❖ the clocked speed

   Within the **SpeedingTicket** class:

   - Define a **constructor**, with parameters for each data property.
     - o Use the parameters to initialize the values.
   - Define **getters** for each data property.

3. Define a second class that will ***implement the ticket array*** for this program, named:

   `TicketArrayImpl`

   The class will have the following **public** data properties:
   - ❖ A **static** constant array size (maximum items that can be held in the array) set to 200

   The class will have the following **private** data properties:
   - ❖ A ticket data array (to hold **SpeedingTicket** objects)
   - ❖ A count of actual ticket objects stored

   Note: The data property definitions will only ***define*** the array reference variable.
   It will not create the array object.

Within the **TicketArraysImpl** class:

- Define a **constructor** that will:
  - Instantiate a ticket array object (using **new** to initialize the array reference variable and the constant array size)
  - Initialize the count of tickets stored to 0
- Define an **instance** method to **add** one SpeedingTicket object to the ticket data array. The method will:
  - Have one parameter, the ticket object to add
  - Test to see if the array is full
    - If the array is full, will throw an **ArrayIndexOutOfBoundsException**
      - Include a message to pass to the exception handler that states the array is full and stating which license cannot be added.
    - Otherwise, store the ticket object into the array and increase the ticket count
- Define a method to read all the data from the input file and store the ticket data by adding **SpeedingTicket** objects into the array. The method will:
  - Have one parameter: the name of the data input file (String)
  - Return an **int** value
    - If the file was opened successfully, the method will return the number of tickets stored in the array.
    - Otherwise, it will return 0, to indicate the file could not be opened/read.
  - Implement code to try to open the input data file. If the file opened:
    - Display a message that the program is reading the file data
    - In a loop:
      - ❖ Read the three data items from one line of data in the input data file
      - ❖ Call the constructor to create a new **SpeedingTicket** object.
      - ❖ **Try** to call the instance method to add the ticket object to the ticket data array
        (this is an instance method of the **TicketArrayImpl** class)
      - ❖ **Catch** any thrown **ArrayIndexOutOfBoundsException** exceptions.
        - ➢ Display the message passed back via the throw.
        - ➢ Also display a message that no more data will be read from the file.
        - ➢ Set a value to exit the loop (so no more data will be read).

      Loop until there is no more file data or an exception is thrown.
    - Close file.
    - Display the number of SpeedingTicket objects stored in the ticket data array.
  - **Catch** any **FileNotFoundException** exceptions. When caught:
    - ❖ Display a message that explains which file could not be opened.
  - Return the count of tickets stored in the array.

- Define a method to calculate and store values in the **fines** array. The method will:
    - Have one parameter, the empty fines array
    - Define constants for all fixed values in the calculations (from page 1 of this assignment).
    - For each ticket stored in the ticket array:
        - Access the **clockedSpeed** and **speedLimit** data stored in the object using ***getters***.
        - Calculate the ticket fine.
        - Store the calculated fine in the fines array.

- Define an **instance** method to produce a fines report.
  The method will create an **output file** with the fines report. The method should:
    - Have one parameter, the fines array
    - Read the filename for the report output file from the user.
    - **Try** to open the report output file
        - If the file opened successfully, generate a report and write it to the output file.
        - The output file will contain a list of vehicle license plate numbers (accessed via a getter) and the fine associated with that license, with the fines right-aligned on the decimal point.
        - The last line of the output file will contain the total of all fines collected, right-aligned on the decimal point, with the fines listed above it.

          Sample output file
          ```
          BBB222    87.50
          XYZ123    66.25
          A1B2C3   215.10
          Total    368.85
          ```
    - **Catch** any **FileIOException** exceptions. When caught:
        - Display a message that explains which file could not be opened
          and that a fines report will not be generated.

4. Define a third (***main***) class named:

   `TicketAnalysis`

   Within the **TicketAnalysis** class:

- Define three static methods to:
    - Determine and return the lowest fine amount
    - Calculate and return the average fine amount
    - Determine and return the highest fine amount

  Each will have the fines array and count of fines in the array as parameters.

- Define a static method to display a ticket summary, using the previously defined static methods to calculate the low, high, and average fines.
    - Parameters: the fines array and count of fines in the array
    - Ticket summary will be displayed as shown in the sample output on the next page, with all figures will be right-aligned on the decimal point.

- Define a **main** method to:
  - Display a description of what the program will do to the user.
  - In a loop (outer loop):
    - Create a new object of the **TickeArraysImpl** class.
    - In a loop (inner loop):
      - ❖ Read the filename for the input data file from the user.
      - ❖ Using the object, call the instance method to read and store the data from the input file (sends a message to the **TicketArraysImpl** object).

      Loop until the method returns a number other than 0, indicating the file was read (and that the ticket data array now has ticket data stored in it).
    - Display a message to the user, that the program is calculating the fines.
    - Define an array (to hold **double** fines), the same size as the array for ticket objects.
    - Uses the **TicketArraysImpl** object to call:
      - ❖ The instance method to calculate the fines and store them in the array
      - ❖ The instance method to produce a fines report
    - Execute the static method to display a ticket summary.
    - Ask the user whether to run the program again, using a different input file.

    Loop until the user says s/he does not want to run the program again.

Sample Run

```
This program will run analyses on weekly speeding ticket data files

Enter input filename: tickets
    Cannot open input file: tickets

Enter input filename: tickets.txt

Reading ticket data...
Data stored for 3 tickets
Calculating fines...
Done!

Enter name of report output file: report.txt

Week's Ticket Analysis for 3 tickets issued:
    Lowest ticket fine            66.25
    Average ticket fine          122.95
    Highest ticket fine          215.10

Run program again with another file (y/n)? n
```

WARNING: The objects, classes, and methods must be implemented exactly as specified above. If your program produces correct output, but you did not create and use the objects as specified, and implement the required classes and methods, you will lose a significant number of points.

See last page of the file for outline of code for the three required classes.

5. The program must follow the **CS210 Coding Standards** from Content section 6.10.
   Be sure to ***include*** the following comments:
   - Comments at the ***top of each code file*** describing what the class does
     - Include **tags** with the author's name (i.e. your full name) and the version of the code (e.g. Java Assn 4, version 1.0)
   - Comments at the ***top of each method***, describing what the method does

   Include **tags** with names and descriptions of ***each*** parameter and return value.

Testing

You will need to create test data files to test your program. In combination, all your test data files should test every possible execution path within your code, including erroneous data which cause exceptions.

Before you submit your project, add your all files you used to test your program in the top level of your project directory (and add a number to the end of each file, if there are multiple test data files).

File examples: **`datafile1.txt`**
`datafile2.txt` (numbered, if multiple data files tested)

Submission

This programming assignment is due by midnight of the date listed in the **Course Assignments by Week**.

REMINDER: Programs submitted that **do not compile** without errors **will not be accepted**.

Again, you will submit a single **zip** file containing all of the files in your project.

- First export your project from NetBeans, as detailed in previous assignments
  - Name your export file in the following format:
    `<lastname>Assn<x>.zip`

    For example: **`SmithAssn8.zip`**

  NOTE: Save this zip file to some other directory, not your project directory.

- Then submit your **.zip** file to the **Java Prog Assn 8** assignment submission folder (located under the **Assignments/Dropbox** tab in the online course).
  - Warning: Only NetBeans export files will be accepted.
    Do not use any other kind of archive or zip utility.

Grading

Programs will be graded using the **rubric** that is linked on the same assignment page as this file.

WARNING:
*Programs submitted **more than 1 day past the due date** will **not** be accepted,*
and will receive a grade of 0.

```
/**
 * Description of class and author and version tags
 */
public class SpeedingTicket {
      // ticket data fields go here

      // constructor with parameters (and comments) goes here

      // getters (with comments and return tags) go here
}
```

```
/**
 * Description of class and author and version tags
 */
public class TicketArrayImpl {
    // static constant array size goes here

    // data fields go here (ticket array and ticket count)

    // constructor without parameters (nd comments) goes here
    //   (instantiates ticket array object and initializes ticket count)

    // method to add ticket object to ticket array goes here
          (with comments and parameter tag)

    // method to read ticket data from file goes here
           (-includes comments with parameter and return tags
            -creates a ticket object for each ticket and
            -calls previous method to add the object to the ticket array)

    // method to calculate fines/store into array goes here
           (with comments and parameter tag)

    // method to create the fines report goes here
           (with comments and parameter tag)
}
```

```
/**
 * Description of program, and author and version tags
*/
public class TicketAnalysis {

    public static void main(String[] args) {
        // code to define variables & display program description

        // Loop:
            // code to instantiate TicketArrayImpl object
            // Loop:
                  // code to read filename and try to read data from the file
            // code to create fines array
            // code to call other methods
    }

    // three methods to calculate the fines average, low, and high go here
            (includes comments with parameter and return tags)

    // method to display a ticket summary (with comments and parameter tags)
}
```