CSCI 241 Assignment 7 Array of Seconds

Due: Thursday, April 8, 2021 *at 11:59 p.m.*

Objective:

Complete a program that uses separate class-level methods to work with arrays of integers.

Description:

Begin by copying the BlueJ project zip file named **`Assign7Times`** from Canvas (it's a zip file). It contains one unfinished class named **`Times`**.

`Times` contains an incomplete `main()` method. It currently declares, creates, and fills an array of 10 integers. Each integer is a random number between 1 and 3599. Each number represents a number of seconds. The other parts of the program will use this array.

Method Tasks and Descriptions

You will complete the `main()` method and write 5 new static methods for this assignment. To help with debugging, write the `printArrayValues()` method first, so that you can print any array at any time as you develop the program. Other methods are listed in order of approximate difficulty.

Each new method should be `public` and `static`. The `main()` method contains some commented-out code, as well as comments indicating which parts should be added. Looking at the `main()` method, it should be obvious where you need to uncomment code, and where you need to complete it.

A working `printArrayValues()` method will help you with debugging. It is described first. The other methods are listed in order of approximate difficulty. For 3 points, email your instructor the lines of code for your **`printArrayValues()`** method (lines copied as text into an email message) by 11:59 p.m. on Tuesday, April 6.

On the next page you will see an example of output printed from a run of the `main()` method in which the user chooses to use the fixed array of 10 values as input (helpful for testing).

The steps include:

1. Asking the user if they wish to use a fixed array of 10 known values, or one of a size entered from the keyboard. If they choose the second option, your code should follow the commented-out instructions and assign the newly allocated space to the array named **`myTimes`**.
2. Printing original array contents, 5 numbers per row
3. Printing the original times as a combination of minutes and seconds (rather than total seconds), one per line
4. Summing the array's *seconds* and printing the total in *minutes*
5. Extracting and printing those values which are **over** a *user-entered* number of minutes.
6. Counting and printing the number of times in each 15-minute quartile

All the methods needed to accomplish these steps are described in detail after the sample output.

Note: in a later step, you will add the option for the user to choose a different-sized array of random values. The output printed below is what you see from the first part of the program that uses the fixed array of 10 values **and** when the user enters the number 30 as the value in step #5. As usual, the input from the user is written **`bold and underlined`** to distinguish it from what the program prints:

```
Enter 1 for fixed 10-element array, 2 for random array: 1

Original array contents
-----------------------
 3398 2067 3064   68   94
 2407  368 1009 1316 2614

Original times in minutes:seconds format
----------------------------------------
Time[00] = 56:38
Time[01] = 34:27
Time[02] = 51:04
Time[03] = 01:08
Time[04] = 01:34
Time[05] = 40:07
Time[06] = 06:08
Time[07] = 16:49
Time[08] = 21:56
Time[09] = 43:34

Times in this array total over 273 minutes

Enter minute threshold for new array: 30

Printing times over 30 minutes
------------------------------
Time[00] = 56:38
Time[01] = 34:27
Time[02] = 51:04
Time[03] = 40:07
Time[04] = 43:34

Printing quartiles ...
  1st  2nd  3rd  4th
 ---- ---- ---- ----
    3    2    3    2
```

- **`void printArrayValues(int [])`**
  This method takes an array of `int`s and prints it in neat format, 5 integers per line. Use **5** positions to print each array value. Print an extra blank line at the very end of the method.

- **`void printTimes(int [])`**
  This method takes an array of `int`s as a parameter. It separates each array value (which represents a total number of seconds) into minutes and seconds. It then prints one time per line, in the format shown here (using 2 positions for each of the numeric values).
  ```
  Time[00] = 56:38
  ```
  Remember, using `"%02d"` will print a leading zero when using `System.out.printf()`.

- **`int sumTimes(int [])`**
  This method takes an array of `int`s as a parameter. It adds all entries in the array, converts the summed value to equivalent minutes and truncates any leftover seconds. It returns the sum that was calculated.

- **`int [] overGivenMinutes(int, int [])`**
  This method is the trickiest! It takes two parameters. The first parameter is an integer representing a threshold of minutes. The second parameter is an array of `int`s holding seconds. The method will examine each entry in the array to see if its time exceeds the number of minutes given in the first parameter. This method will have create and return a new array that holds *only* values from the original array that exceed that minute threshold (the first parameter). The new array should be *exactly the right size* to hold the qualified values, and no longer. The method will return this array to the calling method.
  Hints:
  1. *First, create a new array that is the same length as the original.*
  2. *Use a loop to copy the entries that fit the criteria into the same positions, otherwise the new array slot will hold a zero.*
  3. *After finishing part 2, make a new array of just the right size to hold the non-zero entries; this is the one to return. (Hint: if you keep a count of the number of entries you copy, that will give you the size of the final array.)*
  4. *As you work on copying the entries into the new array (that is only as big as you need), use 2 different indices: one for the array you are copying from, and another for the array you are copying entries into.*

- **`int [] quartiles(int [])`**
  We can picture an hour broken into 4 15-minute sections. Each section is a *quartile*. So, if a time falls within the first 15 minutes, it would be in the first quartile, etc. This method takes an array of `int`s as a parameter, where those integers hold total seconds. None of the total seconds will exceed one hour. This method makes a new array to hold 4 integers. The purpose of this method is to look at the original array and count the number of times that fall into each quartile. In the sample output shown above, there are 3 times in the first quartile (array entries in positions 3, 4 and 6), 2 times in the second quartile (from positions 7 and 8),

3 times in the third quartile (from positions 1, 5 and 9) and 2 times in the fourth quartile (from positions 0 and 2). This method returns the newly created array to the calling method.

Final Step:

Once you have all the methods working with the sample array, uncomment and finish writing the section of code at the beginning of `main()` which allows the user to choose whether to use the original array or a random one of the user's chosen size. Follow the instructions in the comments to finish this part of the code. When you run it, try a larger array and see what kind of results you get. When I test your program, I will try a large array – so make sure you test it as well.
Remember to place a comment describing each method before its code!

Submission Requirements

1. **Electronic:** Use SSH (Secure Shell) to copy your BlueJ project into your CS lab account. Use Remote Desktop Connection to get your lab desktop so you can use the **BlueJ** submission tool to electronically submit your **`Assign7Times`** project from your CS lab account.
2. **Canvas copy:** Submit your **`Times.java`** file to the associated assignment entry in Canvas. I can make notes for you there instead of on paper.

Grading Criteria

Your program must compile without errors to be accepted. **Note: This requirement will be strictly enforced on this and subsequent assignments!!**

All the following criteria will be used to evaluate your program:

Correct submission of project	2 pts
Early submission of `printArrayValues()` method code	3 pts
Appropriate and complete comments	6 pts
Adherence to style and design conventions (identifier names, indentation, etc.)	4 pts
Correct program behavior	15 pts
Total points:	**30 pts**