CSC232: Object-Oriented Software Development
Homework 04 – Ecstatic about Statics!

Due: Day 18 – Friday, October 8th @ beginning of class

In this homework, you will (1) learn about several popular classes that contain static (and also non-static) methods including Math, LocalDate, and the Arrays class as well as (2) gain additional practice navigating and using the Java API. Follow the directions below to complete this homework – be sure to read what I am asking for carefully as each component of your solution will be worth points and graded accordingly. If you wish, you may work with a single partner, however, you must email me your partner's name (CC them on the email) and only one member should submit a solution to Moodle.

Startup Steps

- On your computer, create a new folder (i.e., project) named **CSC232_HW4** – this will be the folder that you will save your Java classes in for this homework assignment
- Open Visual Studio Code, and select **File >> Open …** option, then navigate to and select the CSC232_HW4 folder in the dialog window that appears
- At this point, within Visual Studio Code, you should now see a collapsible/expandable section with the label **CSC232_HW4** in the Explorer pane on the left side. If the Explorer pane is not visible, select **View >> Explorer** or click on the icon that looks like "two sheets of paper" in the top-left corner.
- As we discussed in class, every Java program needs a class that contains the main( ) method. Add a **Driver.java** file to your project by hovering your mouse over the collapsible/expandable section with the label CSC232_HW4. You should see an icon of "a sheet of paper with a + sign" that reads **New File** when you hover on top of it. Click on this icon and a new file will appear waiting for you to provide its name – name this new file **Driver.java**
  - **Important**: You must put the **.java** extension on the end of the name, otherwise your computer will not know that Driver.java contains Java code
- Next, add the necessary code to define your Driver class – then, add the necessary code to define the main( ) method.
  - **Important**: In Java, the name of the class must always match the name of the file that it is stored in (excluding the .java extension). Therefore, your class's name must be Driver (lowercase vs uppercase matters)
- When you are finished with the steps above, a **Run** link should appear over the main( ) method. (If a Run link does not appear, be sure to rewatch the videos on Moodle in the Course Resources section for how to download and install the JDK, Visual Studio Code, and the Java Extensions Pack). Once the Run link is visible, click on it to run/execute your program. **Recall**: All Java programs begin executing at the main( ) method that you selected, therefore you can always trace through exactly what your code will do.
  - A window should appear at the bottom with the **Terminal** tab selected – this is the tab where your Java program will print its output. You may wish to add a System.out.println("Hello World") statement to your main( ) method to ensure everything is working correctly before continuing to the next steps
  - **Tip**: For this homework, I recommend using your main( ) method to create objects that test the creation of objects and calling their methods to ensure they are working correctly

Homework Steps

- Add an **Evaluator.java** file to your project's folder within Visual Studio Code and define an Evaluator class

Warning: You must define the methods exactly as specified in the instructions below -- this includes the method's **name**, **return type**, **parameters** (inputs), and **static vs. non-static**. Failure to follow these instructions will result in a significant loss of points on this assignment.

The Math class

- The Math class is a class that contains entirely static variables and methods. Add the following static methods to your Evaluator class:
    - Add a static method named **calcSphereSurface** that has a double parameter named **radius** and returns a double. When executed, this method should return the surface area of a sphere with the provided **radius**.
    - Add a static method named **myMathFunction** that takes three double parameters: **x , y ,** and **theta** and returns a double. When executed, this method should return the result of the following equation:
        - result = $\tan(theta) + 2\sqrt{x} - (3 * \min(x, y)) + y^4$

The LocalDate class

- As we discussed in class, we should not use deprecated classes/methods/etc. as they are no longer actively supported. Therefore, since the java.util.Date class is deprecated, you will instead learn about the LocalDate class in Java which contains both static and non-static methods. To provide you experience with this commonly-used LocalDate class and its methods, I want you to add the following static methods to your Evaluator class
    - Add a static method named **howManyDays** that has **no parameters** and returns an **int**. When executed, this method should obtain the current date and return how many days are in the current date's month. For example, if I were to execute your program in December, it should return 31 … if I were to execute your program in November, it should return 30.
    - Add a static method named **isLeapYear** that has a String parameter named **date** and returns a **boolean**. I will call this method and provide a String date always in the following format "yyyy-mm-dd" – for example, if I wanted to call your method and provide it October 31st, 2021, then I would call it with **"2021-10-31"**. Your method should return true if the date I provide is during a leap year. You do not need to use the String class to solve this problem – look for method(s) in the LocalDate class
    - Add a static method named **hasItHappened** that has three **int** parameters named **month**, **dayNum**, and **year** and returns a **boolean**. I will call this method and provide it with a date (e.g., October 31st, 2021 would be **month = 10** , **dayNum = 31** , and **year = 2021** and your method should return true if the date that I provide is **before** the current date (when executed, this method should obtain the current date).
        - **Important**: Any method in the LocalDate class that takes a ChronoLocalDate object as an input will accept a LocalDate object instead.

The Arrays class

- As we have seen, arrays are a very popular structure to store data contiguously and they are used internally by many of the data structures that Java provides (e.g., ArrayList, HashSet, HashMap, etc.). The Array**s** class in Java contains many useful static methods for working with Java arrays.
    - Add a static method named **isInArray** that takes (a) an **array of integers** named **values** and (b) an **int** named **findMe** as parameters and returns a **boolean**. When executed, this method should return true if **findMe** is in the **values** array, otherwise it should return false. However, there is a restriction – you are **not** allowed to use any loops to implement this method (i.e., while loop, for-loop, do-while loop, etc.). Instead, you should determine which method(s) from the Arrays class would help solve this problem. Be sure to click on and read the descriptions for methods that you intend to use to understand how they work.
        - **Important**: Your method should not create an array object and/or put values in it – I will call your method with an array of integer values that I create (and you should test your method by only creating arrays in your Driver class)
    - Add a static method named **getTheLastHalf** that takes an **array of integers** named **values** as a parameter and returns an **array of integers**. When executed, this method should return the "last half" of the **values** array that it was passed as an input (you can assume that the array will have an even number of cells to make this method simpler to write). However, there is a restriction – you are **not** allowed to use any loops to implement this method (i.e., while loop, for-loop, do-while loop, etc.) Instead, you should determine which method(s) from the Arrays class would help solve this problem. Be sure to click on and read the description for methods that you intend to use to understand how they work.
        - **Important**: Your method should not create an array object and/or put values in it – I will call your method with an array of integer values that I create (and you should test your method by o only creating arrays in your Driver class)

Testing Steps

- Before submitting your solution, you should use your Driver's main( ) method to test each of the static methods in your Evaluator class.

Submission Steps

- Open up the Moodle in a web browser and navigate to our CSC232 course's page
- Upload the **.java** files OR a ZIP file containing your folder to the assignment's submission item on Moodle
    - **Important**: If you worked with a partner, only **one** member should submit a solution – be sure that you have emailed me who you were working with