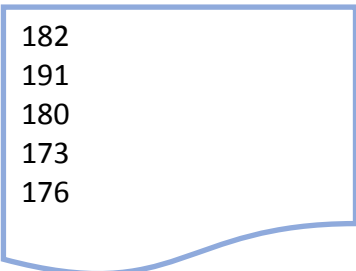


# Assignment #3

## Deadline: Tuesday 20/02, 23:59

**Data management:** This assignment is aimed at developing your basic skills for data management using Java. It also teaches you how to develop your code in a top-down approach: keeping your main method clean and readable and breaking tasks into sub-methods.

We assume that we are holding a simple data set that includes people's heights. This information is stored in a text file with the following format:



```
182
191
180
173
176
```

Please note that the height is an integer value. This file might be edited or viewed in each execution of the code, depending on the user's wish.

To be more precise, in each execution of the code, the following will happen:

- I. The code starts with a welcome message (see the template) and asks the user which option, the user wants to choose among:

```
(1) Data entry
(2) View Data
(3) Data stats
(4) Data cleaning
```

and the user chooses **1**, **2**, **3**, or **4**, accordingly. *Please note that this part is already implemented in the template code.*

- II. If the user chooses Data entry (i.e., enters **1**), the program will call the `newEntry()` method. This method first asks the user (with a proper message, up to you) for the number of new entry heights (call it `noOfNewEntries`) that the user wishes to append to the file. Then, in a loop with the total number of iteration of `noOfNewEntries`, it repeatedly in each iteration:
  - a. takes the `newEntry` height from the user from the console (with a proper message, up to you) and
  - b. uses `newEntry` as the input to call `appendFile(int newEntry)` method, which is a method to append the new entry to the file.
- III. If the user chooses View Data (i.e., enters **2**), the program will call `viewData()` method. This method prints all the contents of the file in the console. To do so, it calls another method, called `readFile()`, which reads the entire file and returns all heights as an array.

This array is used in `viewData()` to be printed in the console with a proper message (up to you).

- IV. If the user chooses Data stats (i.e., enters **3**), the program will call `dataStats()` method. This method first calls `readFile()` method to fetch all heights in the file and stores it in a local array (called it `myArray`). Then, this array (`myArray`) of all heights is used as input to call three methods: `minArray(int[] myArray)`, `maxArray(int[] myArray)`, `meanArray(int[] myArray)`, taking the array of all heights as input and, respectively, returning the *min*, the *max*, and the *average* of the height elements within that array. Finally, the method will print these results with proper messages in the console with a proper message (up to you).
- V. If the user chooses Data Cleaning (i.e., chooses **4**), the program will call `dataCleaning()` method. This method first calls `readFile()` method to fetch all the data in the file and stores it in a local array (called it `heights`). Then, it creates another array of integers that includes only those elements in `heights` that are **NOT larger than 250 and NOT smaller than 0** (called it `myArray`) and overwrite the file with this new array. In other words, here we assume that any entry less than 0 or larger than 250 is a wrong entry and must be deleted from the file. To overwrite the file, the program calls the method `overWriteFile(int[] myArray)` to write back the new array on the file (removing the previous content of the file). See **NOTE4** which gives you a hint how to do it.

**NOTE1:** The format of your output, once not explicitly mentioned, is up to you and will be fine as long as it is reasonable.

**NOTE2:** You MUST follow the 'good programming style'.

**NOTE3:** We assume that the user is smart and only enters choices that are allowed. But as for the heights, the user might enter integer values less than zero or larger than 250.

**NOTE4:** A hint: Overwriting a file means that the previous content of the file will be destroyed and the file will only contain what you overwrite on it. What we covered in L02 slide #41 is to 'append' a file, where the previous content will be kept and anything you write will be added to end of the file. Overwriting a file is similar to appending the file in terms of syntax. The only difference is that in step 3 (L02 slide #41), you should add the `false` option as shown below

```
PrintWriter wr = new PrintWriter( new BufferedWriter( new FileWriter("Heights.txt", false)));
```

**NOTE5:** You should only submit the .java file and not the text file that you create. Please name your files as `A3_<st#>.java`. For example, for a person with student# 000000 the file name would be `A3_000000.java`. Submit your files on your dropbox folder on Canvas.

**Code template:** Please use the following template for your assignment and fill in the blanks

```
import java.util.*;

public class Assignment_3 {
    static Scanner userInput1 = new Scanner(System.in);

    public static void main(String[] args){

        int userChoice = getUserChoice();
        switch (userChoice) {
            case 1:
                newEntry();
                break;
            case 2:
                viewData();
                break;
            case 3:
                dataStats();
                break;
            case 4:
                dataCleaning();
                break;
        }
    }

    // This method asks and returns what the user wants to do
    public static int getUserChoice(){
        System.out.println("*****");
        System.out.println("*****");
        System.out.println("*****          DataI/O          *****");
        System.out.println("*****");
        System.out.println("*****");
        System.out.println("");
        System.out.println("What do you wan to do?");
        System.out.println("(1) Data entry");
        System.out.println("(2) View Data");
        System.out.println("(3) Data stats");
        System.out.println("(4) Data cleaning");
        System.out.println("*****");
        System.out.print("Please enter your choice (1, 2, 3, or 4): ");
        return userInput1.nextInt();
    }

    // This method adds a new entry at the end of the file
    public static void newEntry(){
        // fill in the blank
        // You need to use appendFile(int newEntry) method
    }

    // This method prints out the content of the file in the console
    public static void viewData(){
        // fill in the blank
        // You need to use readFile()
    }

    // This method prints out the min, max, and the average of the heights
    public static void dataStats(){
        // fill in the blank
        // You need to use readFile(), minArray, maxArray, and meanArray methods
    }

    // This method removes all non-reasonable entries from the file
    public static void dataCleaning(){
        // fill in the blank
        // You need to use readFile() and overWriteFile()
    }

    // This method gets myArray as input and returns its minimum value
    public static int minArray(int[] myArray){
        // fill in the blank
    }

    // This method gets myArray as input and returns its maximum value
    public static int maxArray(int[] myArray){
        // fill in the blank
    }
}
```

```
}

// This method gets myArray as input and returns its average
public static double meanArray(int[] myArray){
    // fill in the blank
}

// This method reads the file and returns the values
public static int[] readFile(){
    // fill in the blank
}

// This method writes a new entry to the file
public static void appendFile(int newEntry){
    // fill in the blank
}

// This method writes a new entry to the file
public static void overWriteFile(int[] myArray){
    // fill in the blank
}
}
```