Assignment 6 - Array, ArrayList, and Exceptions

Submit Assignment

Due Wednesday by 11:59pm **Points** 50 **Submitting** a file upload **File Types** java

The program of this assignment can accept any number of *good* or *bad* GPA values entered at command line or at run time and print the highest three, the lowest three, and the average of good GPA. Bad or invalid GPA will be discarded automatically. In case of less than three good GPA are entered or left, the the highest and lowest three print them all in the right order.

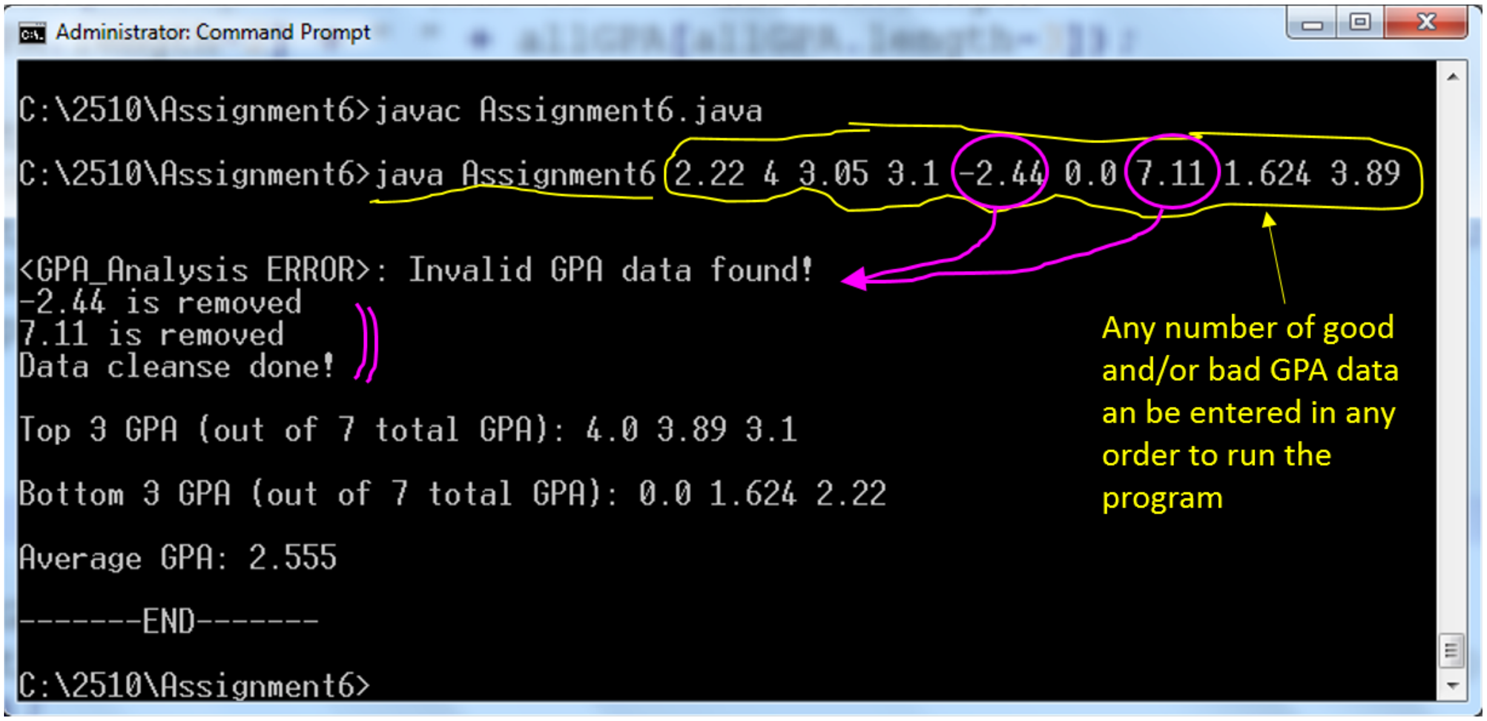
Two files are provided below for your convenience. Please download them to your computer.

- **Assignment6.class**
- **Assignment6.java**

The .java file is where you are going to write your code to complete this assignment. **When finished, just upload Assignment6.java that contains your complete code for grading. No extra folder or zip is needed.**

The .class file is a compiled output from a completed version of Assignment6.java and therefore can be executed by using JDK java command. Executing this file helps you to see what are exactly needed by this program.

The screenshot below shows a typical run of Assignment6.

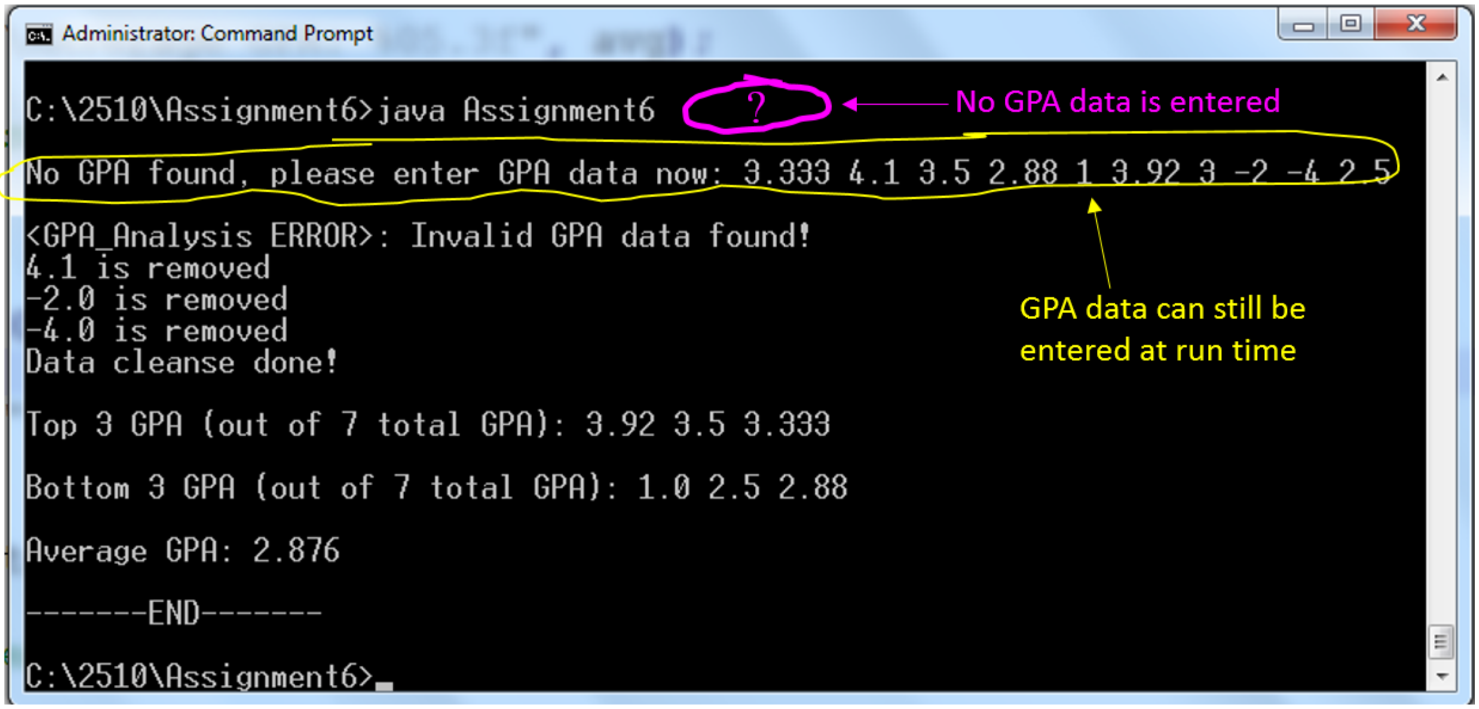

Administrator: Command Prompt
Any number of good and/or bad GPA data an be entered in any order to run the program

```
C:\2510\Assignment6>javac Assignment6.java

C:\2510\Assignment6>java Assignment6 2.22 4 3.05 3.1 -2.44 0.0 7.11 1.624 3.89

<GPA_Analysis ERROR>: Invalid GPA data found!
-2.44 is removed
7.11 is removed
Data cleanse done!

Top 3 GPA (out of 7 total GPA): 4.0 3.89 3.1

Bottom 3 GPA (out of 7 total GPA): 0.0 1.624 2.22

Average GPA: 2.555

--------END-------

C:\2510\Assignment6>
```

This program accepts any number of GPA data, good or bad, to be entered when Assignment6 is entered at the command line to execute. That is, the GPA data are all passed and stored in the 'args' String array of the main() of Assignment6 class.

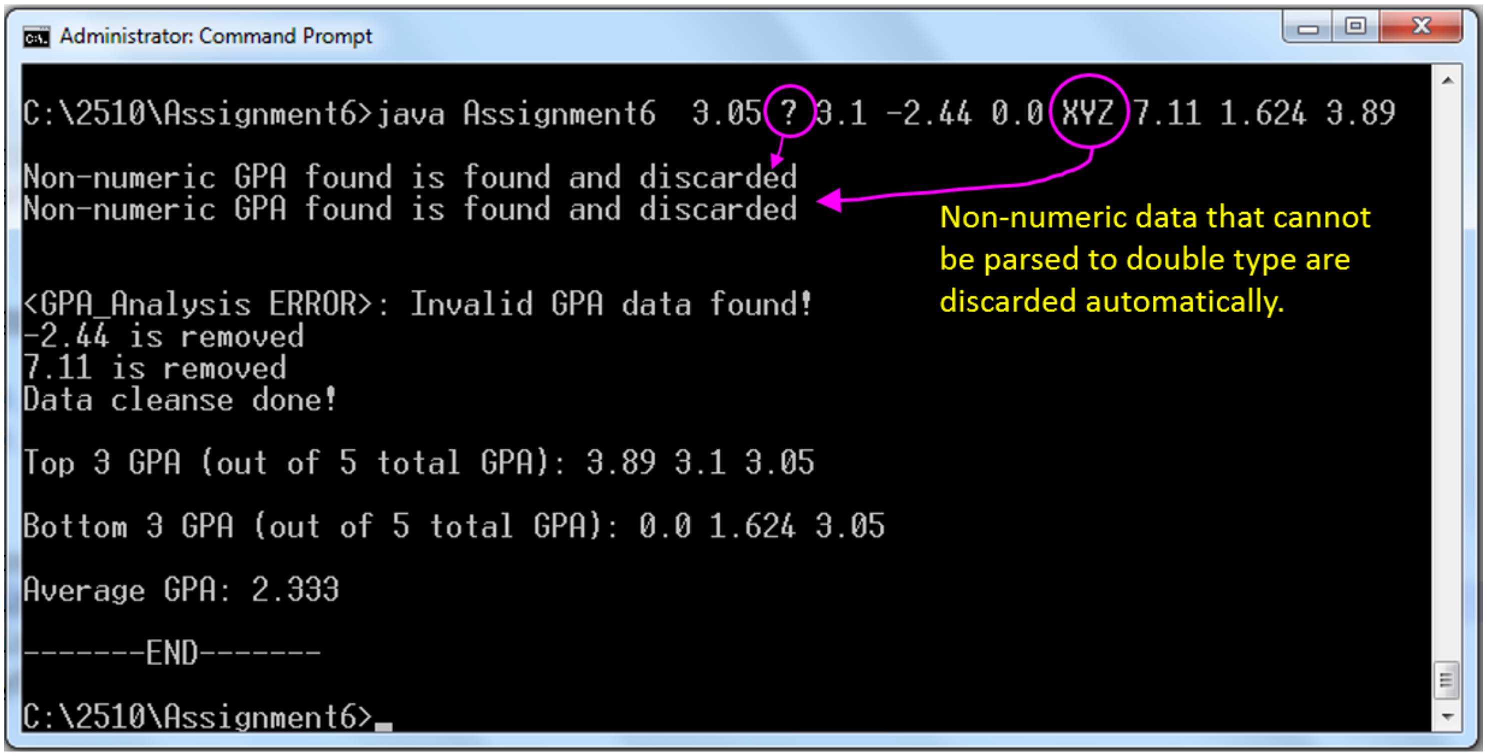
If there are *invalid* GPA (i.e., numeric data but **out of range** of [0.0, 4.0]), they are detected and removed automatically. Then, the program analyzes the good GPA to display the highest three (in descending order), the lowest three (in ascending order), and the overall average with three and only three decimal digits.

In case no data are entered at the command line to run the program, it will prompts user to re-enter as shown below and the rest part of functions remain the same. If user skips data entry, the same prompt of "No GPA found, please enter........" will be repeated until at least one data is entered.


Administrator: Command Prompt
?
No GPA data is entered
GPA data can still be entered at run time

```
C:\2510\Assignment6>java Assignment6
No GPA found, please enter GPA data now: 3.333 4.1 3.5 2.88 1 3.92 3 -2 -4 2.5
<GPA_Analysis ERROR>: Invalid GPA data found!
4.1 is removed
-2.0 is removed
-4.0 is removed
Data cleanse done!

Top 3 GPA (out of 7 total GPA): 3.92 3.5 3.333

Bottom 3 GPA (out of 7 total GPA): 1.0 2.5 2.88

Average GPA: 2.876

-------END-------

C:\2510\Assignment6>
```

No matter entered at command line or at run time, all **non-numeric data** (e.g., "?", "abc", "$2.88") are detected and removed automatically like out-of-range numbers. See the screenshot below for an example.


Administrator: Command Prompt
Non-numeric data that cannot be parsed to double type are discarded automatically.

```
C:\2510\Assignment6>java Assignment6  3.05 ? 3.1 -2.44 0.0 XYZ 7.11 1.624 3.89

Non-numeric GPA found is found and discarded
Non-numeric GPA found is found and discarded


<GPA_Analysis ERROR>: Invalid GPA data found!
-2.44 is removed
7.11 is removed
Data cleanse done!

Top 3 GPA (out of 5 total GPA): 3.89 3.1 3.05

Bottom 3 GPA (out of 5 total GPA): 0.0 1.624 3.05

Average GPA: 2.333

-------END-------

C:\2510\Assignment6>_
```

Program Implementation Instructions and Requirements

You are only required to use the Assignment6.java file given above and follow the instructions/requirements below to complete it.

1. There is no user-defined package(s) needed.
2. The file does not contain import statement(s). You must find which are needed and add them to the file so it can compile and execute.
3. There are three classes, **Assignment6**, **GPA_Analysis**, and **GPAException**, which are contained in the file and only the 2nd class, **GPA_Analysis**, requires you to complete it. No additional classes are needed.
4. **GPAException** is a special version of Exception class which we want it to handle GPA-related exception (e.g., out-of-range GPA values like -2.5 or 4.8 or 6). When such a GPA is detected, we can throw an object of this **GPAException** class as a signal to allow bad GPA to be removed or corrected.
5. **Assignment6** is the main class. Besides the main(), it also implements two more methods defined below, where the first two methods are already completed and tested. No change to them is expected for this assignment.
    - **main()** - it first calls the next method to ensure GPA data are available and contain no non-numeric data such as "x" or "?" or $2.95". When data are entered and parsed to double type in an ArrayList, they are going to be analyzed by an object of **GPA_Analysis** class. This is done by passing the ArrayList to the object when it is constructed. Because out-of-range GPA values will be detected by the object of **GPA_Analysis** class, a try-catch is used to remove these bad GPA values as a result of handling the GPAException. Also, a while-loop is used to enclose this try-catch block so after the exception is handled and data are cleansed by the catch block, the program execution can continue

to analyze the GPA data--finding top 3, bottom 3, and overall average of GPA data. It breaks the while-loop when all three analysis are completed.

- **makeSureThereAreInputData()** - it takes a String array as input, which is the 'args' passed by main() and checks if the array contains data. If not, it prompts to enter GPA data, all in one line. It also detects if the input contains non-numeric data and discard them by using two try-catch blocks. A while-loop is used to ensure at least one data is entered. When data are entered and parsed to double type, it returns them in an ArrayList.
- void **cleanseData(**ArrayList<Double> data**)** - This is the method you need to complete. which removes items in the input ArrayList if their value is not within the range of [0, 4.0]. Its signature (i.e., header) line is already provided and you only need to fill statements into their body to satisfy the following:
  - It defines an object, say, *badData*, with the same type of its input parameter, i.e., ArrayList<Double>.
  - It uses a for-loop to visit every element of the input ArrayList, i.e., data, and add it to the new ArrayList, *badData*, if its value is out of range of [0, 4.0]. At the same time, it prints "xx is removed" for every such element.
  - It uses the statement below to physically delete all bad GPA values.

    data.**removeAll**(badData);

  - It prints "Data cleanse done!" to conclude its job.

6. **GPA_Analysis** is the class that performs the GPA analysis. It defines four methods, including a constructor. They all have their signature line only and you are required to fill statements in the body of all four methods.
   - **GPA_Analysis()** - The class defines a data field 'allGPA' as an array of double type but not initialized. This is because it doesn't know how many GPA data will be passed in for analysis. Therefore, this constructor's first job is to use the 'new' and the size of the input ArrayList together to initialize the allGPA array. Then, it uses a for-loop to check if any value of the input ArrayList is out of the range of [0, 4.0]. If yes, it ***throws a GPAException*** with a message "<GPA_Analysis ERROR>: Invalid GPA data found!" immediately. Otherwise, it simply copies the value from the input ArrayList to the allGPA array. Hint: check divideByZero5.java or divideByZero6.java from one of the zip files in Module 7 for how to "throw" an exception. Also, p.424~7 of Chapter 16 for more about throw statements and custom exceptions.
   - **top3()**- It checks the allGPA array and prints the highest three GPA in descending order as seen in the screenshot above. Because an ArrayIndexOutOfBoundsException could possible occur when there are less than three total GPA values are stored in the allGPA array, it use a try-catch to handle such exception. Instead of showing an error message and stopping the entire program execution, it simply prints all GPA data, which could be two or only one GPA, in descending order.
   - **bottom3()** -Similarly to top3() except it checks and prints the lowest three GPA in ascending order.
   - **average()** - This is a very simple method to calculate the average of all GPA in allGPA array. There is no try-catch needed by this method. To print the average with with a format of three decimal digits, you may use the **printf()** statement shown below without changes, assuming your average is stored in avg:

```
System.out.printf("\nAverage GPA: %05.3f", avg);
```