# Programming Project 4 – Planet Stats

*Note: When you turn in an assignment to be graded in this class, you are making the claim that you neither gave nor received assistance on the work you turned in (except, of course, assistance from the instructor or teaching assistants).*

The new Martian settlers know that life is limited on Mars, so they want to explore the moons of Mars and other planets to determine if they can establish any sort of establishment on these moons. Write a program called **MoonSearch.java** that will contain several methods.

Your program will have several parallel arrays in the main method which you will use in the program. The data in each array is related to the other arrays by its position in the array; thus, for example, the moon named Phobos has a radius of 11.3 km. Below are the arrays that you will need. I also attached them in a text file. Please note that these are **sample** arrays. When we test your program, we will be using different arrays.

String array of moon names:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Phobus | Deimos | Adrastea | Aitne | Amalthea | Ananke | Aoede | Arche |

Double array of radii in kms:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11.3 | 6.2 | 68.9 | 33.6 | 71.2 | 26.8 | 255.9 | 47.4 |

Double array of density in kg/m3:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1.8 | 1.4 | 14.2 | 33.3 | 16.4 | 68.1 | 121.3 | 38.2 |

Double array of distance in 103 km from the Mars:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3.7 | 23.4 | 550391.6 | 227894.9 | 778893.6 | 143323.5 | 287223.5 | 449655.1 |

Your program will contain several methods:

* The first method, calcAvg(), calculates the average from all the values in the double array passed into the method. This method passes in a double array and returns the average as double. The method header is:

public static double calcAvg(double [] values)

* The second method, findHighValue(), searches the double array passed in to find the largest value from the array This method passes in a double array and returns the double largest value found in the array. The method header is:

public static double findHighValue(double[] values)

* The third method, findLeastValue(), searches the double array passed in to find the smallest value from the array. This method passes in a double array and returns the double smallest value found in the array. The method header is:

public static double findLeastValue(double[] values)

* The fourth method, findHighestTwo(), passes in a String array of moon names and a double array of values. It will search through the double array to find the two largest values in the double array. It will return a String array that contains the moon names associated with the largest two double values in the double array, with the moon with the highest value listed first. The method header is:

public static String[] findHighestTwo(String[] names, double[] values)

* The fifth method, findLeastTwo(), passes in a String array of moon names and a double array of values. It will search through the double array to find the two smallest values in the double array. It will return a String array that contains the moon names associated with the smallest two double values in the double array, with the moon with the smallest value listed first. The method header is:

public static String[] findLeastTwo(String[] names, double[] values)

* The sixth method, findMoon(), will verify if a given String is a valid moon within the String array of moon names. This method passes in the String array of moon names and the moon being searched for. The method returns a boolean value of **true** or **false** based on whether the moon was found within the moon names array. If the moon name is in the array, the method will return a value of true, and, if the moon name is not in the array, the method will return false. Note, the moon names must match exactly for the method to return true. The method header is:

public static boolean findMoon(String[] names, String moon)

The main method will contain the arrays needed for the program. This method will also do the following:

* Call the calcAvg() method, passing it the double array of radii and print out the results.
* Call the calcAvg() method, passing it the double array of density and print out the results.
* Call the findHighValue() method, passing it the double array of radii and print the results.
* Call the findLeastValue() method, passing it the double array of distance and print the results.
* Call the findHighestTwo() method, passing it the String array of moon names and the double array of radii and print the results. (**Note**: You will need to use a for each loop to print out the values from the array.)
* Call the findLeastTwo () method, passing it the String array of moon names and the double array of density and print the results. (**Note**: You will need to use for each loop to print out the values from the array.)
* Ask the user for a moon name
* Call the findMoon(), passing it the String array of names and the given planet and print the results.
* All outputs should be rounded to **one** decimal spaces using **printf().**

Below are two sample runs.

**The average radius is: 65.2**

**The average density is: 36.8**

**The highest radius is: 255.9**

**The lowest distance is: 3.7**

**The highest two moons for radii are:**

**Aoede**

**Amalthea**

**The lowest two moons for density are:**

**Deimos**

**Phobus**

**Enter a moon:**

***Deimos***

**Deimos is a moon in the array.**

**The average radius is: 65.2**

**The average density is: 36.8**

**The highest radius is: 255.9**

**The lowest distance is: 3.7**

**The highest two moons for radii are:**

**Aoede**

**Amalthea**

**The lowest two moons for density are:**

**Deimos**

**Phobus**

**Enter a moon:**

***Moon X***

**Moon X is not a moon in the array.**

Note for Gradescope Testing:

* Use nextLine().trim() for all String inputs
* You will also need to use printf() for printing and rounding your values.

This and all program files in this course must include a comment block at the beginning (top) of the source code file that contains:

* the Java program name
* project description
* your name
* the date created
* the course number and section

The comment lines should look like this:

/\*

\* Java program name

\*

\* Project description

\*

\* Your name

\* The version date

\* The course number and section

\*/

Before beginning this project, you will document your algorithm as a list of steps to take you from your inputs to your outputs. This algorithm will be due a week before the project submitted in Canvas. This will be graded for completion. It will be your responsibility to understand and correct any errors you might have with your algorithm.

Each step of your algorithm will be added as a comment block within your code. You will have the comment block right above the code that performs the actions specified. For example, before your lines of code that ask the user for inputs, you would have a comment block that states what inputs you are requesting from the user.

You will test your code using the provided JUnit tests. You should use these tests within IntelliJ to make sure that your code is running correctly. You will take a screenshot of your code passing all JUnit tests. Make sure that the screenshot has your header comment block in it so that we know it is your code running. Once it is, then you will submit your code to Gradescope. You will only have 4 submissions to Gradescope, so please make sure your code is running correctly in IntelliJ before submitting to Gradescope.

You will submit your Java source code file (MoonSearch.java) to Gradescope. You will upload your screenshot of the JUnit tests by uploading the file to the Assignment link in Canvas. Please do not submit your files in a zipped folder.

Ask questions about any part of the programming project that is not clear!

**Rubric for Programming Project 4**

|  |  |
| --- | --- |
| **Item** | **Points** |
| Algorithm submitted on time | 15 |
| Comment blocks stating the algorithm step above the code as specified including comment header block | 10 |
| Appropriate choice of variable names and constants | 5 |
| Separate calcAvg() method | 5 |
| Separate findHighValue() method | 5 |
| Separate findLeastValue() method | 5 |
| Separate findHighestTwo() method | 5 |
| Separate findLeastTwo() method | 5 |
| Separate findMoon() method | 5 |
| Used arrays | 10 |
| Program layout and appearance (Coding style is clear and easily understood) | 5 |
| Output is correct | 10 |
| Output is displayed and rounded to one decimal place | 5 |
| Screenshot of JUnit tests passing | 10 |
| **Total** | **100** |