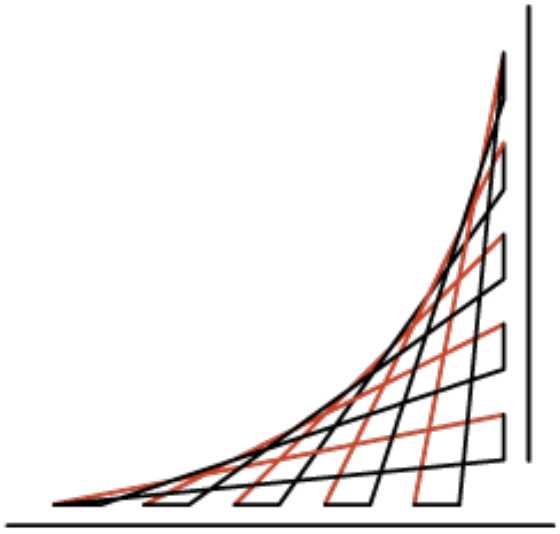

ESCUELA COLOMBIANA DE INGENIERÍA
JULIO GARAVITO

VIGILADA MINEDUCACIÓN

SYSTEMS ENGINEERING

Arquitecturas Empresariales

Workshop 2

Luis Daniel Benavides Navarro

Author:
Juan Camilo Angel Hernandez

Contents

1 Introducction

The aim of this workshop is write a program that allows the calculation of the mean and standard deviation of a set of n real numbers. The program reads the n numbers from a file and uses its own implementation of a linked list to store the numbers on which the respective calculations will be made.

2 Linked List

A linked list is a linear data structure in which each of its elements is a separate object known as a node. The elements in a linked list are not stored in a specific location but are linked together using pointers.

Each node in a linked list consists of two elements, its value and a reference to the next node. The last node has a null reference. The first node in a linked list is called the head and the last one is called the tail.

In this workshop we implemented a doubly linked list(DLL), which differs from the linked list in the addition of a reference to the previous node apart from the reference to the next one. This allows the possibility of reaching any node from both the head and the tail.[1]

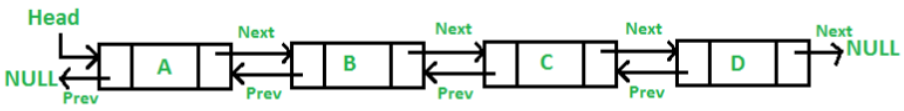

Figure 1: Doubly linked list, *taken from* [2]

3 Concepts

3.1 Mean

The mean is the average of a discrete set of data. The average is the most common measure of location for a set of numbers. The mean is equal to the sum of all the values in the data set divided by the number of values in the data set.

The formula for calculating the mean is

$$x_{avg} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{1}$$

3.2 Standar Deviation

The standard deviation is a measure of the amount of dispersion of a set of values. A low standard deviation indicates that the values tend to be close to the mean of the set,

while a high standard deviation indicates that the values are spread out over a wider range.

The formula for calculating the standard deviation is

$$\delta = \sqrt{\frac{\sum_{i=1}^{n}(x_i - x_{avg})^2}{n-1}} \tag{2}$$

4 Architecture

4.1 Class diagram

In the development of this workshop the iterator design pattern was implemented. According to GoF definition this pattern provides a way to access the elements of a collection in a sequential manner without exposing their underlying representation, this pattern is used to traverse the collection of objects in a defined manner. During iteration, client programs can perform various other operations on the items as required. The key idea is to abstract the responsibility of the access and traversal outside the collection and put it in an Iterator object that defines a standard traversal protocol, in this way achieving a decoupling.[3]

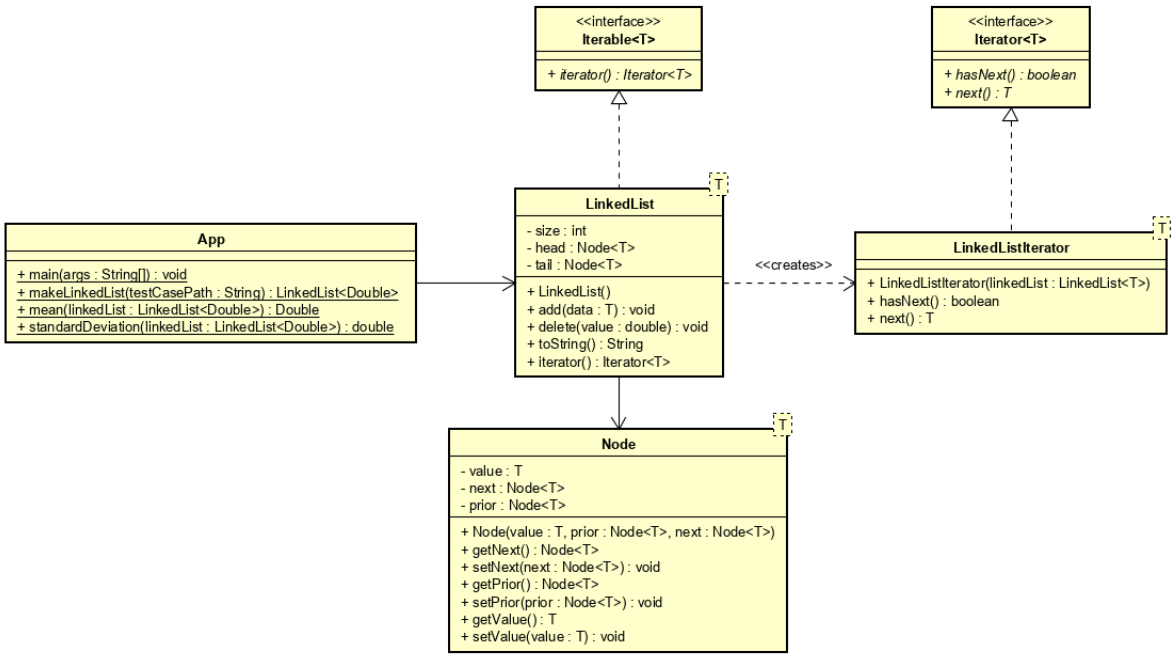

Figure 2: Class diagram

The following are the classes or interfaces that were used in the development of the workshop:

1. The Iterator interface: an interface to access or traversal through the collection of elements. It provides methods that specific iterators must implement.

2. The LinkedListIterator class: implements the Iterator interface methods providing the necessary logic to traverse the collection and access the elements.
3. The Iterable interface: this is a collection interface that defines a method that create an Iterator object.
4. The LinkedList class: Implements Iterable so the objects (nodes) in this class can be easily traversed using forEach. This class has two methods:
   (a) add: this will create a Node and insert it on the tail.
   (b) delete: this will find the first node that match with the given value an remove it.
5. The Node class: contains its value and the reference to the next and previous node to enable traverse the linked list.
6. The App class: Provides the following static methods:
   (a) mean: calculates the average of the data stored in a LinkedList and returns it rounded to 2 decimal places.
   (b) standardDeviation: calculates the standard deviation of the data stored in a LinkedList and returns the response rounded to 2 decimal places.

4.2 Deploy diagram

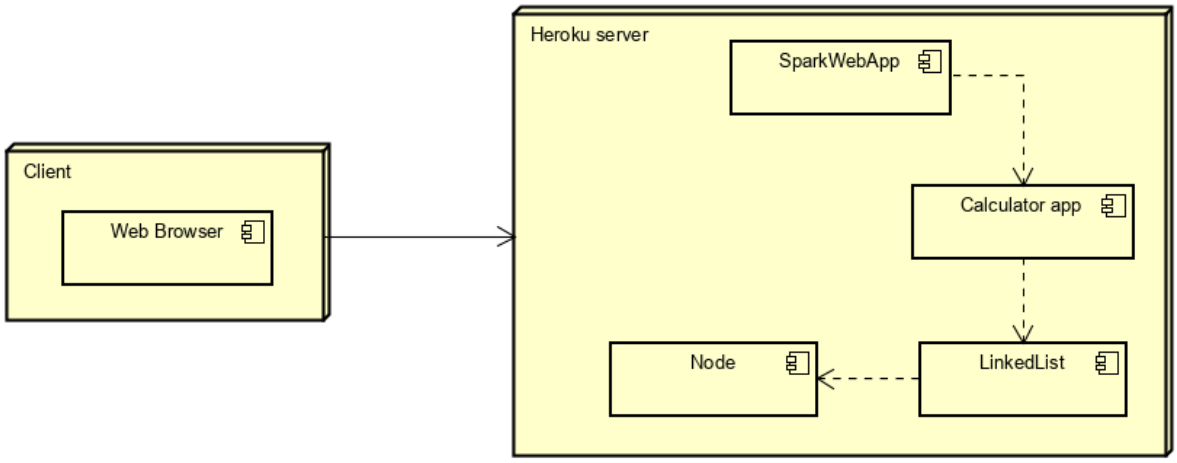

Figure 3: Deploy diagram

In the previous image we can see the deployment diagram, first the user accesses to the application using a web browser where it communicates with the application which is hosted in heroku, once inside he will be shown the web page created, this is thanks to the spark app since it is responsible for determining the behavior of different client requests and to provide the various services, then the user will insert the data to perform and display the calculations using an API along with the help of a LinkedList.

4.3 Web Architecture

In figure 4 we can see the web architecture of the system, in which we can observe the abstraction of the interpreters and together with the abstraction of the links. This design corresponds to a client-server architecture in which the client sends HTTP requests to the server and the server responds thanks to a controller using the Spark micro-framework which offers two services:

- Get/: Return to the user the index.html file, which acts as the graphical interface of the application.

- Post calculator/calculate: Receives the data entered by the user and performs the corresponding calculations and then displays them.

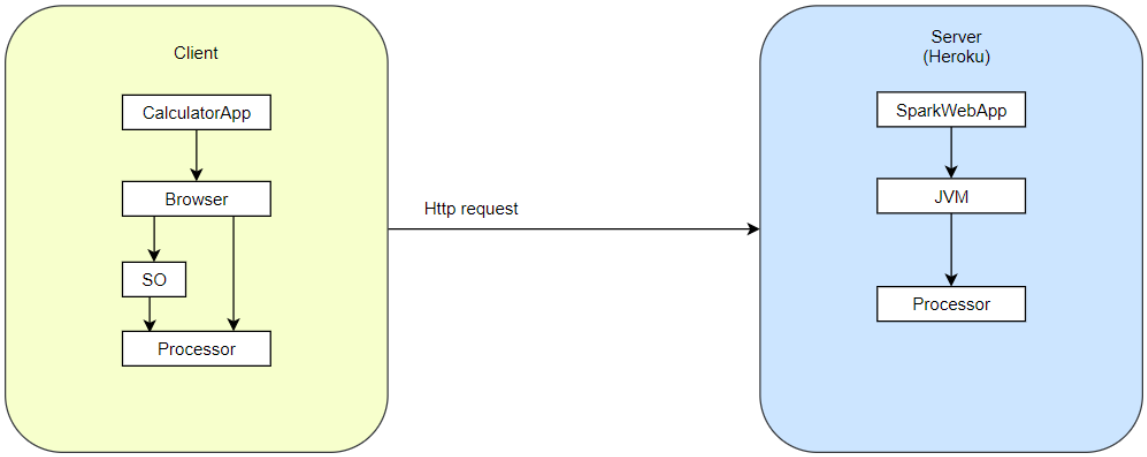

Figure 4: Web Architecture

Respect to the interpreters we can say that the execution of the system begins from the user since this one directs the execution of the program across the browser, this one in turn uses instructions of the operating system and of the processor. On the other hand, in the server the requests are received by the controller and it executes the set of instructions on the virtual machine of java.

The abstraction of the links is represented by HTTP requests since it is in this way that the two parts of the application communicate.

Finally the memory abstraction is represented with the implemented linked list which allows the writing and reading of the data.

5 Tests Cases

In this workshop we have two test cases along with their respective mean and standard deviation in order to check the functioning of the system. In the figure 2 and 3 you can see the test data and the expected results respectively.

Column 1 Estimate Proxy Size	Column 2 Development Hours
160	15.0
591	69.9
114	6.5
229	22.4
230	28.4
270	65.9
128	19.4
1657	198.7
624	38.8
1503	138.2

Figure 5: Test data

Test	**Expected Value**		**Actual Value**	
	Mean	*Std. Dev*	*Mean*	*Std. Dev*
Table 1: Column 1	550.6	572.03		
Table 1: Column 2	60.32	62.26		

Figure 6: Expected results

6 Results

Finally, the following figure shows the results obtained.

Figure 7: Results obtained

Observing the results we can see that these are equal to the expected ones so the program behaves correctly when making the corresponding calculations.

7 References

[1] V. S.Adamchik, *Linked lists*, https://www.cs.cmu.edu/~adamchik/15-121/lectures/Linked%20Lists/linked%20lists.html, Accessed on 2020-08-07, 2009.

[2] *Doubly linked list*, https://www.geeksforgeeks.org/doubly-linked-list/, Accessed on 2020-08-07, 2014.

[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. USA: Addison-Wesley Longman Publishing Co., Inc., 1995, ISBN: 0201633612.