INTRODUCTION TO COMPLEX SYSTEMS, JAVA, MVN, AND GIT

Nicolás Ortega Limas

January 2021

1 Introduction

Some of us have been asking about how was software projects were being managed. Well Apache Maven has basically changed a lot is a software that allows us to manage projects. When creating a Maven project, a very special, very predefined folder structure will automatically be generated.

2 Objectives

- Recognizing and implement a Linked List in order to manage collections
- Implementing basic statistical operations like the mean and the standard deviation and identify its importance.
- Understand the basics of Maven, Git and Java via command line.

3 Problem proposed

We are required to calculate the mean and standard deviation of a set of n real numbers.

It is also a requirement to read the n real numbers from a file and to implement the use a linked list to store the n numbers for the calculations. (Note: You have to write your own implementation of a linked list and it must be compliant with Java's collections API)

4 Solution

We will develop a full java application that solves this problem by implementing this two formulas presented above. [2]

As it says in the guide that i used to help me build my own LinkedList Class "It's good to understand how things work, and understanding linked lists is an

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

Figure 1: formula for calculating the mean

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left(x_i - x_{avg}\right)^2}{n-1}}$$

Figure 2: formula for standard deviation

important step towards understanding more complex data structures, many of which don't exist in the standard libraries." [1]

We are going to use the formula for calculating the mean stated in Figure 1 and the formula for standard deviation stated in Figure 2.

- S is the symbol for summation
- i is an index to the n numbers
- x is the data in the set
- n is the number of items in the set

5 Tests

Now we can start to review the two tests done on the project, they should use the presented data sets found on Figure 3 and 4, we should get the results found on Figure 5. Now we develop our test with those data sets and results on mind, giving us these two tests found on Figures 6 and 7 With the test executing correctly we can now say we finished.

6 Conclusions

Some of us forget how these basics things are done. Then it is always helpful to remember things like Apache Maven, Git, Java and general algorithm design

DATASET 1
186
699
132
272
291
331
199
1890
788
1601

Figure 3: Data set number one

DATASET 2
160
591
114
229
230
270
128
1657
624
1503

Figure 4: Data set number two

TEST	EXPECTED VALUE	
	MEAN	STD. DEVIATION
Data set 1	550.6	572.03
Data set 2	638.9	625.63

Figure 5: Expected tests results

```
@Test
public void testStandardDeviation() {
    try {
        System.out.println("standardDeviation");
        LinkedList lista=new LinkedList();
        Scanner scanner = new Scanner(new File("src/test/input/input-2.txt"));
        ArrayList<Double> number = new ArrayList<>();
        while(scanner.hasNextInt()){
            number.add(scanner.nextDouble());
        }
        for (int j=0;j<number.size();j++){
            Double a = number.get(j);
            lista.add(a);
        }
        double expResult = 625.6339806770231;
        double result = main.standardDeviation(lista);
        assertEquals(expResult, result, 0.0);
    } catch (FileNotFoundException ex) {
        Logger.getLogger(mainTest.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

Figure 6: Proposed test for standard deviation with data set number two

```java
@Test
public void testMean() {
    try {
        System.out.println("mean");
        LinkedList lista=new LinkedList();
        Scanner scanner = new Scanner(new File("src/test/input/input-1.txt"));
        ArrayList<Double> number = new ArrayList<>();
        while(scanner.hasNextInt()){
            number.add(scanner.nextDouble());
        }
        for (int j=0;j<number.size();j++){
            Double a = number.get(j);
            lista.add(a);
        }
        double expResult = 550.6;
        double result = main.mean(lista);
        assertEquals(expResult, result, 0.0);
    } catch (FileNotFoundException ex) {
        Logger.getLogger(mainTest.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

Figure 7: Proposed test for standard deviation with data set number one

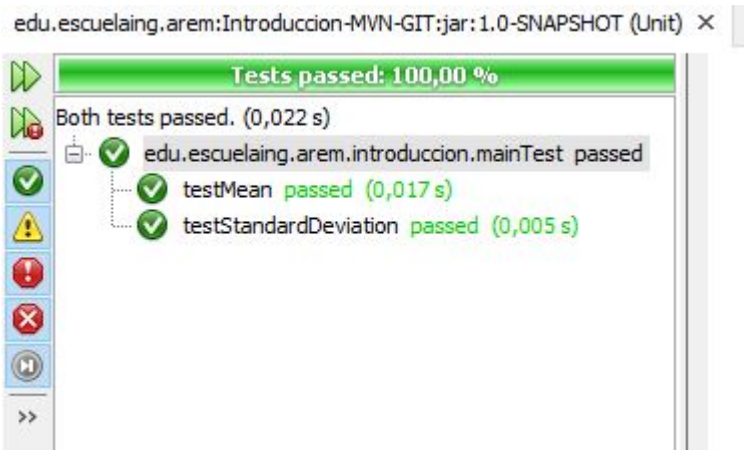

Figure 8: Tests executed correctly

as well. In the same way I have identified some utilities from the Linked Lists like the response time, its implementation is really help full at the time of developing an upper performance data structure that allows us to perform high level actions.

References

[1] How to implement a linkedlist class from scratch in java. https://crunchify.com/how-to-implement-a-linkedlist-class-from-scratch-in-java/. accessed: 29.01.2021.

[2] Read integers from text files in java. https://www.codegrepper.com/code-examples/java/java+read+integer+from+text+file+into+array+scanner. Accessed: 29.01.2021.