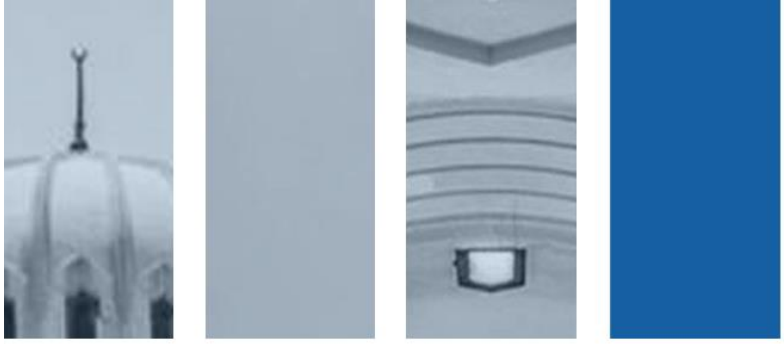


Lab (3)

Program: Computer and Systems Engineering

Course Code: CSE337s

Course Name: Software Testing

Ain Shams University
Faculty of Engineering

Team Number: 7

TEAM MEMBERS:

Name	ID	Level
Mazen Ehab Mohamed Maher	1901120	Senior 2
Ahmed Mahmoud Mohamed Ibrahim	1901143	Senior 2
Mohamed Mostafa Shaban Mohamed	1901650	Senior 2
Mostafa Mohamed Ahmed Abdelaal	1803093	Senior 2
Andrew Samir Kamel Gayed	1900242	Senior 2
AbdAllah Mostafa Mahmoud Alsayed	1900779	Senior 2

1. Implement the FileHandler class that given a text file contains integer values can read these values

FileHandler.java | DataAnalyzer.java | DataAnalyzerTester.java | FileHandlerTester.java

```
9  public class FileHandler {
10     private String filePath;
11
12     public FileHandler(String filePath)
13     {
14         this.filePath = filePath;
15     }
16     public Vector<Integer> GetData()
17     {
18         Vector<Integer> data = new Vector<>();
19         try
20         {
21             File file = new File(filePath);
22
23             FileReader fileReader = new FileReader(file);
24
25             BufferedReader br = new BufferedReader(fileReader);
26
27             String fileContent = "";
28
29             String line;
30             while ((line = br.readLine()) != null)
31             {
32                 fileContent = fileContent + line + "\n";
33             }
34
35             System.out.println(fileContent);
36             br.close();
37
38             String data_string = fileContent;
39
40             // Split the string by newline characters
41             String[] lines = data_string.split("\\r?\\n");
42
43             // Parse each line as an integer and add it to the vector
44             for (String data_line : lines)
45             {
46                 try
47                 {
48                     int num = Integer.parseInt(data_line);
49                     data.add(num);
50                 }
51                 catch (NumberFormatException e)
52                 {
53                     // Handle if the line is not a valid integer
54                     System.err.println("Invalid integer format\n" + data_line);
55                 }
56             }
57         }
58         catch (IOException e)
59         {
60             System.err.println("Exception: " + e);
61
62         }
63         return data;
64     }
65     public void setFilePath(String filePath)
66     {
67         this.filePath = filePath;
68     }
69 }
```

2. Implement the DataAnalyzer class that given a vector of integers can apply analysis on this data and return the min, max and average value

FileHandler.java | DataAnalyzer.java | DataAnalyzerTester.java | FileHandlerTester.java

```
4
5  public class DataAnalyzer {
6      FileHandler fileHandler;
7      public DataAnalyzer(FileHandler fileHandler)
8      {
9          this.fileHandler = fileHandler;
10     }
11     public int GetMin()
12     {
13         Vector<Integer> data = fileHandler.GetData();
14
15         // Check if data is empty
16         if (data.isEmpty())
17         {
18             System.err.println("No data available.");
19             return Integer.MIN_VALUE; // Return some default value indicating no minimum
20         }
21
22         int size = data.get(0);
23
24         if(size != data.size() - 1)
25         {
26             System.err.println("Error in data Given");
27             return Integer.MIN_VALUE; // Return some default value indicating no minimum
28         }
29
30         int min = Integer.MAX_VALUE; // Initialize min value
31
32         // Iterate through the vector to find the minimum value
33         for (int i = 1; i <= size; i++) {
34             int current = data.get(i);
35             if (current < min) {
36                 min = current; // Update min if a smaller value is found
37             }
38         }
39         return min;
40     }
41
```

FileHandler.java | DataAnalyzer.java | DataAnalyzerTester.java | FileHandlerTester.java

```
40     }
41
42     public int GetMax()
43     {
44         Vector<Integer> data = fileHandler.GetData();
45
46         // Check if data is empty
47         if (data.isEmpty())
48         {
49             System.err.println("No data available.");
50             return Integer.MAX_VALUE; // Return some default value indicating no maximum
51         }
52
53
54         int size = data.get(0);
55
56         if(size != data.size() - 1)
57         {
58             System.err.println("Error in data Given");
59             return Integer.MAX_VALUE; // Return some default value indicating no maximum
60         }
61
62         int max = Integer.MIN_VALUE; // Initialize max value
63
64         // Iterate through the vector to find the maximum value
65         for (int i = 1; i <= size; i++) {
66             int current = data.get(i);
67             if (current > max) {
68                 max = current; // Update max if a greater value is found
69             }
70         }
71
72         return max;
73     }
74
```

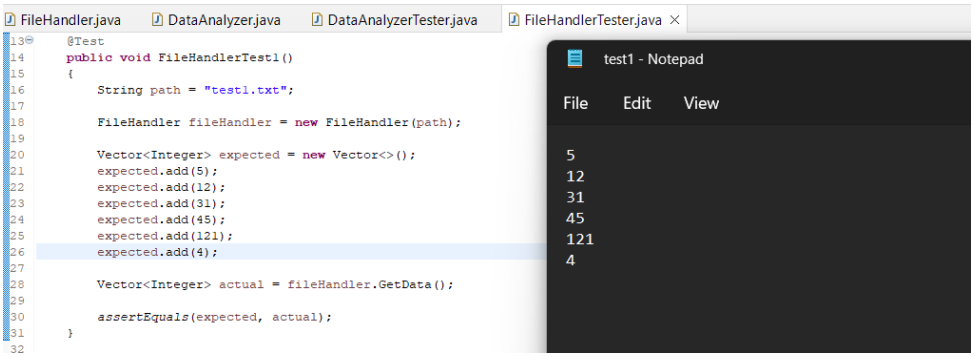
FileHandler.java | DataAnalyzer.java | DataAnalyzerTester.java | FileHandlerTester.java

```
        return max;
    }

    public int GetAverage()
    {
        Vector<Integer> data = fileHandler.GetData();

        // Check if data is empty
        if (data.isEmpty()) {
            System.err.println("No data available.");
            return 0; // Return 0 as default average when there's no data
        }

        int size = data.get(0);

        if(size != data.size() - 1)
        {
            System.err.println("Error in data Given");
            return 0; // Return 0
        }

        int sum = 0;


        // Calculate the sum of all integers
        for(int i = 1; i <= size; i++)
        {
            sum += data.get(i);
        }


        // Calculate the average
        int average = (int) sum / size;

        return average;
    }
}

```

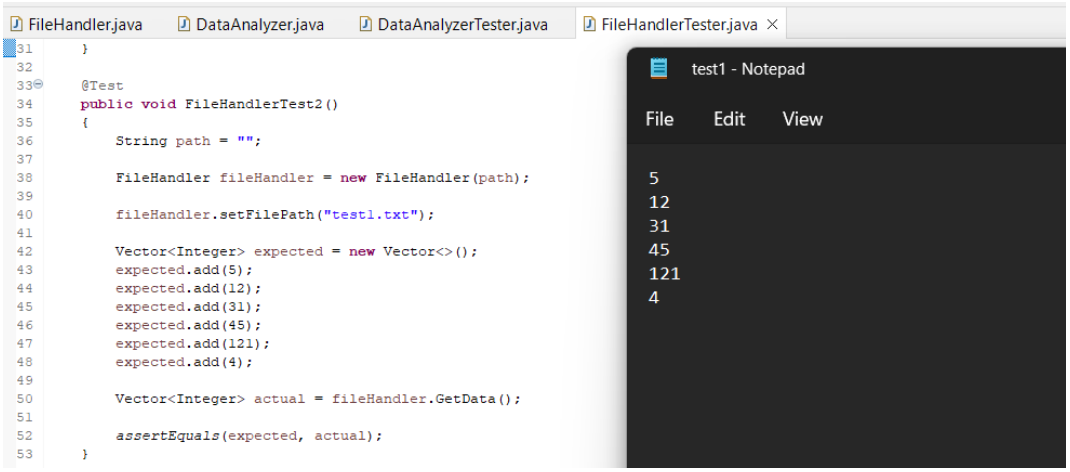
3. Provide unit tests for the FileHandler class

Test cases

1. Test test1.txt contains simple data of 6 integers

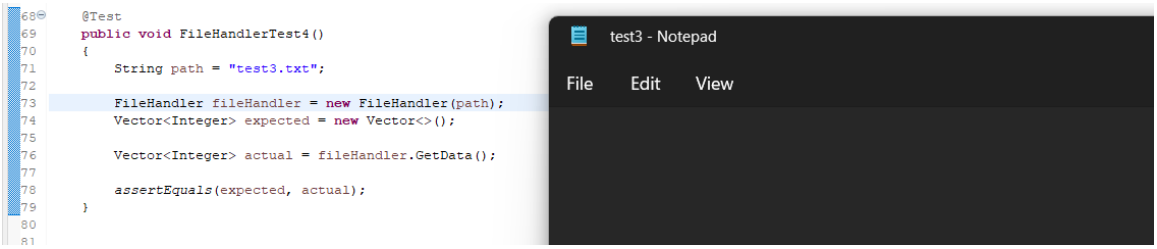


2. Test test1.txt by using setFilePath method

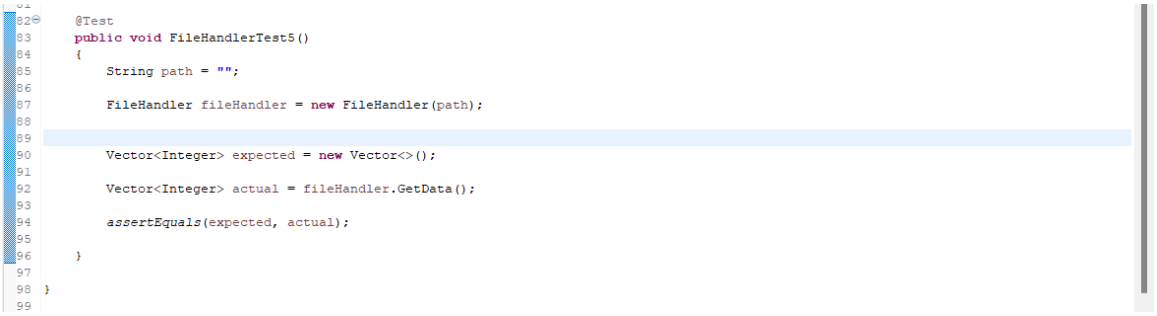


3. Test text2.txt that contains one integer

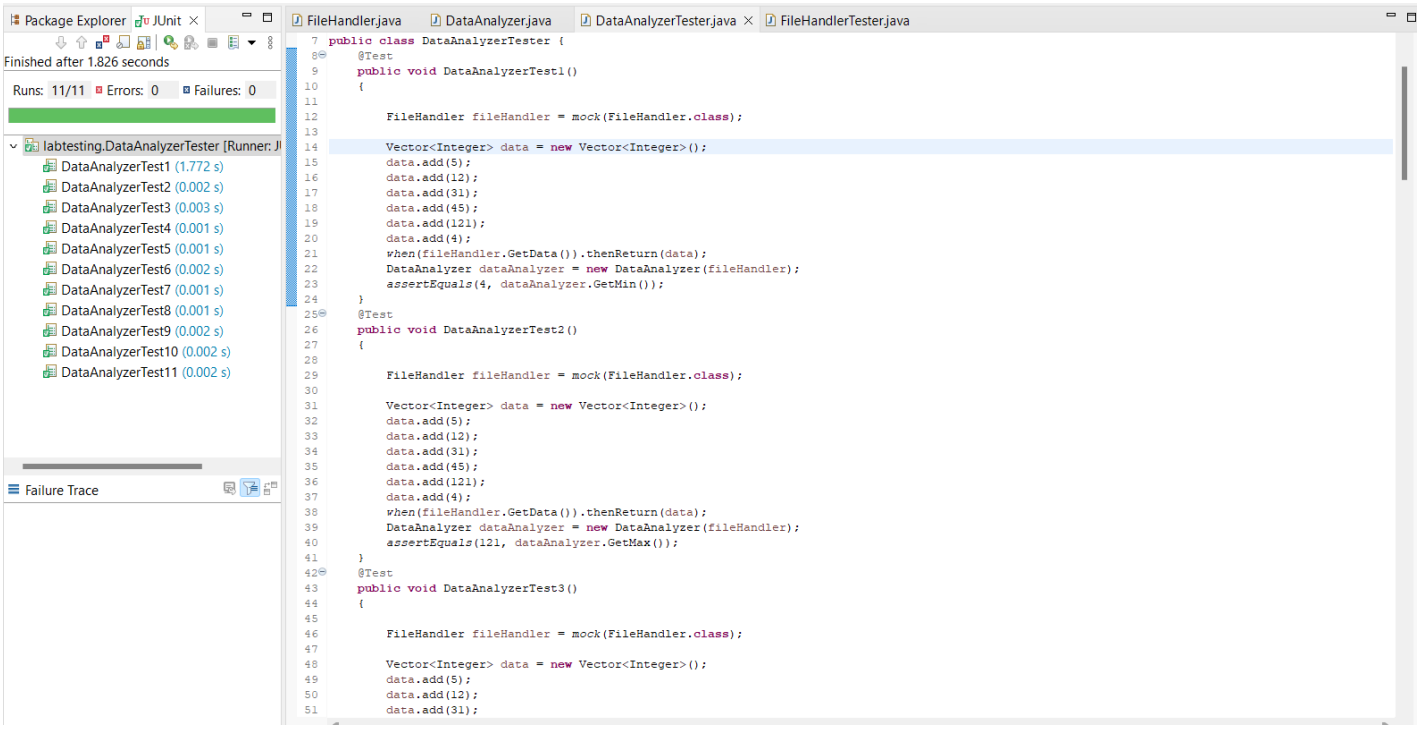
4. Test text3.txt that is empty file



5. Test empty path



4. Provide unit tests for the DataAnalyzer class independently from the FileHandler class (Hint: you can do this using mocks)



Test cases

1. Test GetMin with all +ve numbers

```
@Test
public void DataAnalyzerTest1()
{

    FileHandler fileHandler = mock(FileHandler.class);

    Vector<Integer> data = new Vector<Integer>();
    data.add(5);
    data.add(12);
    data.add(31);
    data.add(45);
    data.add(121);
    data.add(4);
    when(fileHandler.GetData()).thenReturn(data);
    DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
    assertEquals(4, dataAnalyzer.GetMin());
}
```

2. Test GetMax with all +ve numbers

```
25    @Test
26    public void DataAnalyzerTest2()
27    {
28
29        FileHandler fileHandler = mock(FileHandler.class);
30
31        Vector<Integer> data = new Vector<Integer>();
32        data.add(5);
33        data.add(12);
34        data.add(31);
35        data.add(45);
36        data.add(121);
37        data.add(4);
38        when(fileHandler.GetData()).thenReturn(data);
39        DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
40        assertEquals(121, dataAnalyzer.GetMax());
41    }
```

3. Test GetAverage with all +ve numbers

```
42    @Test
43    public void DataAnalyzerTest3()
44    {
45
46        FileHandler fileHandler = mock(FileHandler.class);
47
48        Vector<Integer> data = new Vector<Integer>();
49        data.add(5);
50        data.add(12);
51        data.add(31);
52        data.add(45);
53        data.add(121);
54        data.add(4);
55
56        when(fileHandler.GetData()).thenReturn(data);
57
58        DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
59
60        int sum = 0;
61
62        int size = data.get(0);
63
64        // Calculate the sum of all integers
65        for(int i = 1; i <= size; i++)
66        {
67            sum += data.get(i);
68        }
69
70
71        int expected = sum / size;
72
73        assertEquals(expected, dataAnalyzer.GetAverage());
74    }
75
```

4. Test GetMin with one -ve number and the remaining +ve numbers

```
76    @Test
77    public void DataAnalyzerTest4()
78    {
79
80        FileHandler fileHandler = mock(FileHandler.class);
81
82        Vector<Integer> data = new Vector<Integer>();
83        data.add(5);
84        data.add(-12);
85        data.add(31);
86        data.add(45);
87        data.add(121);
88        data.add(4);
89
90        when(fileHandler.GetData()).thenReturn(data);
91
92        DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
93
94        assertEquals(-12, dataAnalyzer.GetMin());
95    }
```

5. Test GetAverage with one -ve number and the remaining +ve numbers

```
96    @Test
97    public void DataAnalyzerTest5()
98    {
99
100       FileHandler fileHandler = mock(FileHandler.class);
101
102       Vector<Integer> data = new Vector<Integer>();
103       data.add(5);
104       data.add(-12);
105       data.add(31);
106       data.add(45);
107       data.add(121);
108       data.add(4);
109
110       when(fileHandler.GetData()).thenReturn(data);
111
112       DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
113
114       int sum = 0;
115
116       int size = data.get(0);
117
118       // Calculate the sum of all integers
119       for(int i = 1; i <= size; i++)
120       {
121           sum += data.get(i);
122       }
123       int expected = sum / size;
124       assertEquals(expected, dataAnalyzer.GetAverage());
125   }
```

6. Test GetMin with two -ve numbers and one +ve number

```
127   @Test
128   public void DataAnalyzerTest6()
129   {
130
131       FileHandler fileHandler = mock(FileHandler.class);
132
133       Vector<Integer> data = new Vector<Integer>();
134       data.add(2);
135       data.add(-43);
136       data.add(-42);
137
138
139       when(fileHandler.GetData()).thenReturn(data);
140
141       DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
142
143       assertEquals(-43, dataAnalyzer.GetMin());
144   }
```

7. Test GetM with two -ve numbers and one +ve number

```
146    @Test
147    public void DataAnalyzerTest7()
148    {
149
150        FileHandler fileHandler = mock(FileHandler.class);
151
152        Vector<Integer> data = new Vector<Integer>();
153        data.add(2);
154        data.add(-43);
155        data.add(-42);
156
157        when(fileHandler.GetData()).thenReturn(data);
158
159        DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
160
161        assertEquals(-42, dataAnalyzer.GetMax());
162    }
163
```

8. Test GetAverage with two -ve numbers and one +ve number

```
164    @Test
165    public void DataAnalyzerTest8()
166    {
167
168        FileHandler fileHandler = mock(FileHandler.class);
169
170        Vector<Integer> data = new Vector<Integer>();
171        data.add(2);
172        data.add(-43);
173        data.add(-42);
174
175        when(fileHandler.GetData()).thenReturn(data);
176
177        DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
178
179        int sum = 0;
180
181        int size = data.get(0);
182
183        // Calculate the sum of all integers
184        for(int i = 1; i <= size; i++)
185        {
186            sum += data.get(i);
187        }
188
189
190        int expected = sum / size;
191
192        assertEquals(expected, dataAnalyzer.GetAverage());
193    }
```

9. Test 9,10,11 to check that data is more than one integer in text file that analyzed by data analyzer

```
195     @Test
196     public void DataAnalyzerTest9()
197     {
198         FileHandler fileHandler = mock(FileHandler.class);
199         Vector<Integer> data = new Vector<Integer>();
200         data.add(1);
201         data.add(5);
202         data.add(20);
203
204         when(fileHandler.GetData()).thenReturn(data);
205
206         DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
207
208         assertEquals(Integer.MIN_VALUE, dataAnalyzer.GetMin());
209     }
210
211     @Test
212     public void DataAnalyzerTest10()
213     {
214
215         FileHandler fileHandler = mock(FileHandler.class);
216         Vector<Integer> data = new Vector<Integer>();
217         data.add(1);
218         data.add(5);
219         data.add(20);
220         when(fileHandler.GetData()).thenReturn(data);
221         DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
222         assertEquals(Integer.MAX_VALUE, dataAnalyzer.GetMax());
223     }
224     @Test
225     public void DataAnalyzerTest11()
226     {
227         FileHandler fileHandler = mock(FileHandler.class);
228
229         Vector<Integer> data = new Vector<Integer>();
230         data.add(1);
231         data.add(5);
232         data.add(20);
233
234         when(fileHandler.GetData()).thenReturn(data);
235         DataAnalyzer dataAnalyzer = new DataAnalyzer(fileHandler);
236         assertEquals(0, dataAnalyzer.GetAverage());
237     }
238 }
239
```

Link of Source Code

https://drive.google.com/drive/folders/1hBDcJ5qllSXW2hyzIylN7UegZ1OAuEJ5