

LAB 2 REPORT

Name: Osama Ayman Mokhtar Amin

ID: 19P1609

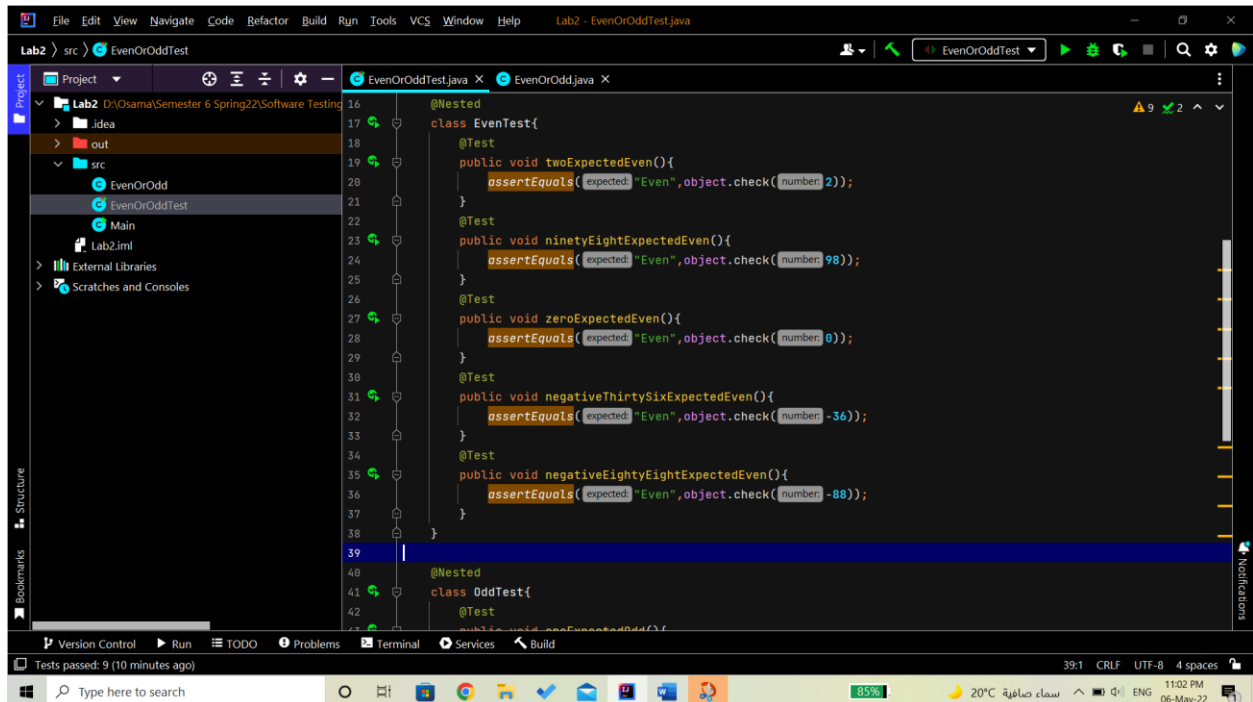
Course Name: Software Testing, Validation, and Verification

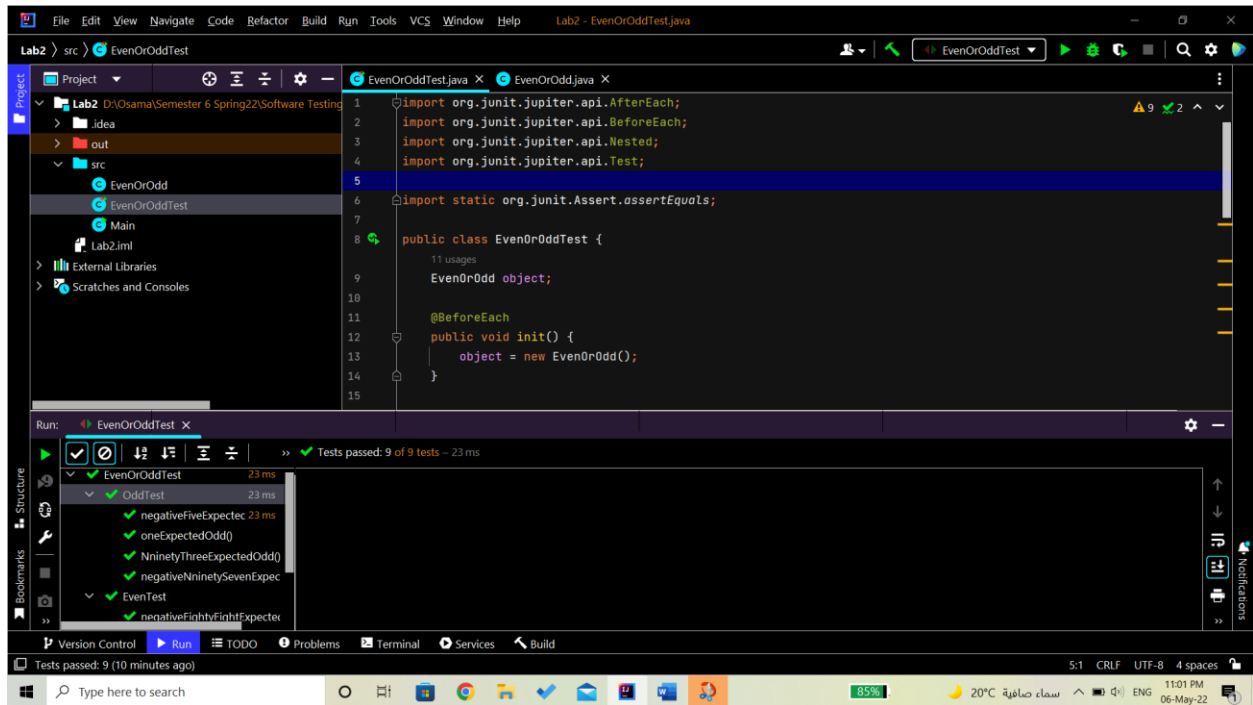
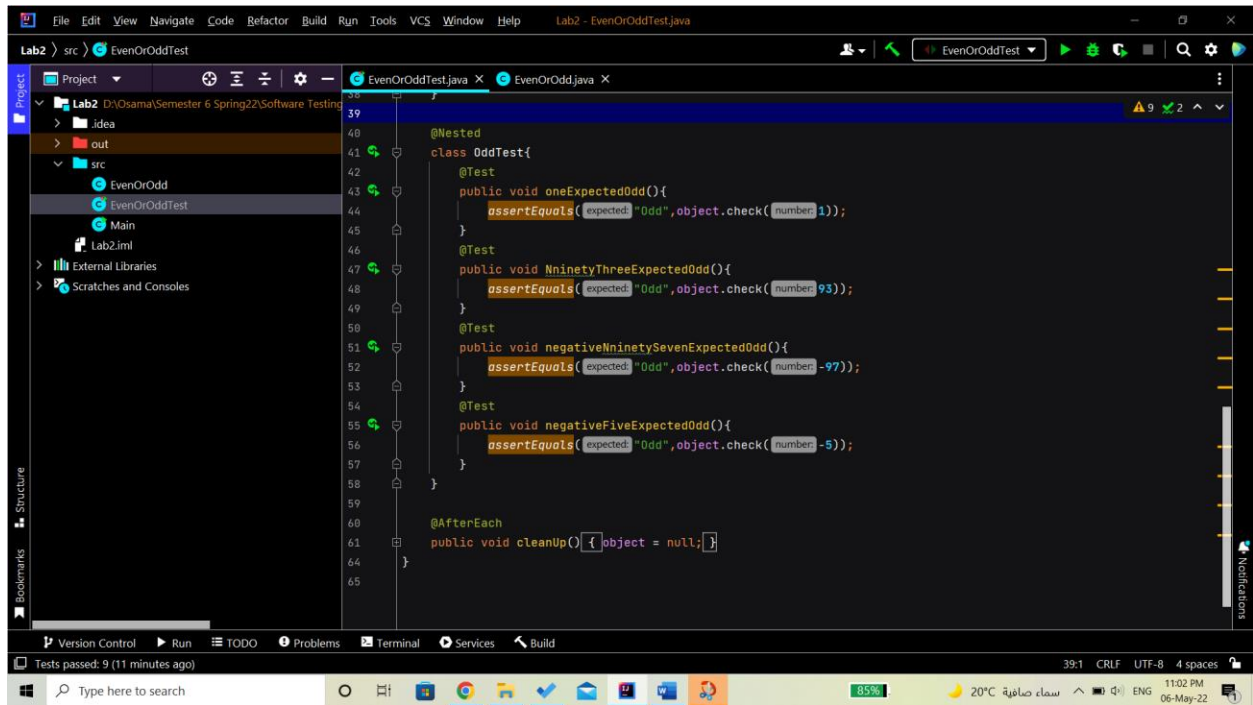
Course Code: CSE 338

1. I) Checking Even and Odd Numbers

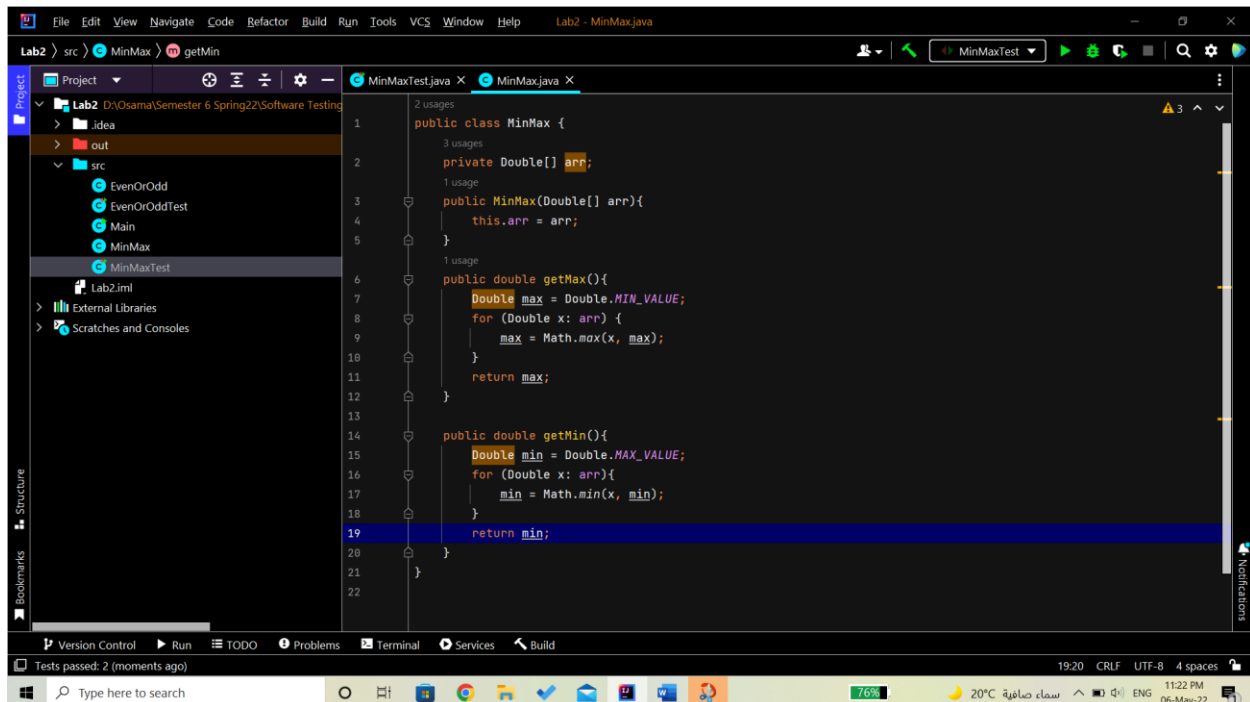


```
4 usages
1  public class EvenOrOdd {
      10 usages
2      public String check(int number){
3          if (number%2 == 0) return "Even";
4          else return "Odd";
5      }
6  }
7
```

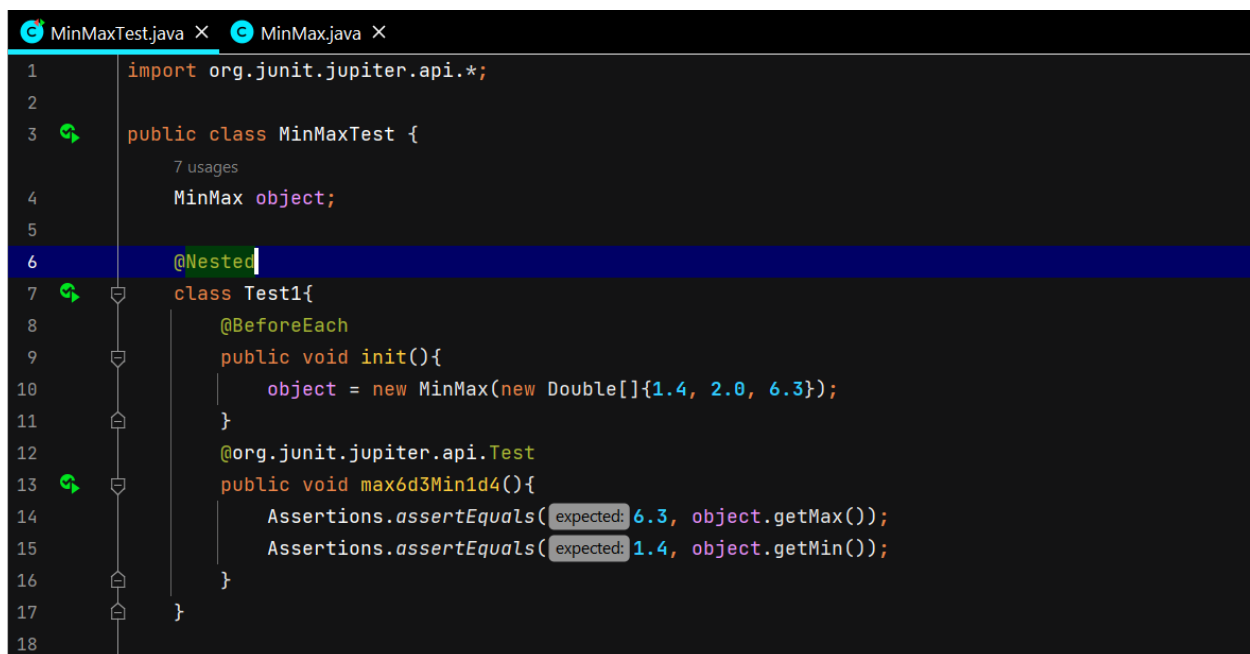




II) Finding the Maximum and Minimum Value in an Array

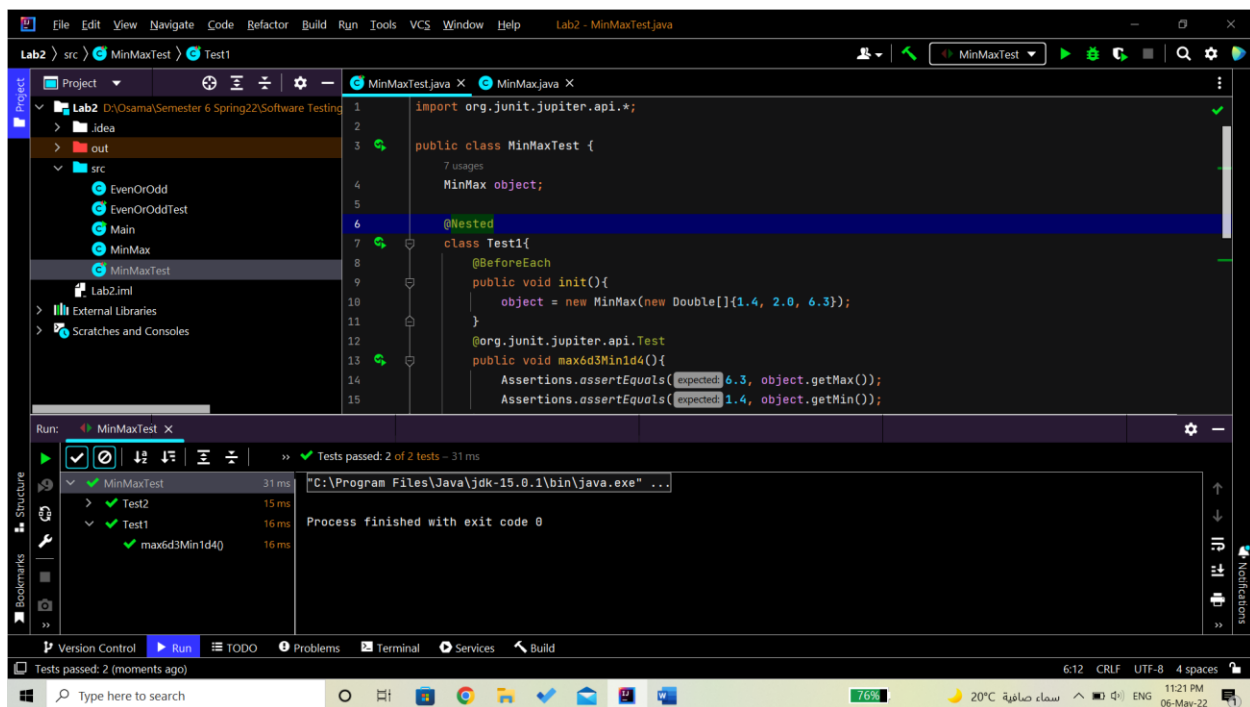


```
1 public class MinMax {
2     private Double[] arr;
3     public MinMax(Double[] arr){
4         this.arr = arr;
5     }
6     public double getMax(){
7         Double max = Double.MIN_VALUE;
8         for (Double x: arr) {
9             max = Math.max(x, max);
10        }
11        return max;
12    }
13    public double getMin(){
14        Double min = Double.MAX_VALUE;
15        for (Double x: arr){
16            min = Math.min(x, min);
17        }
18        return min;
19    }
20 }
21
22
```

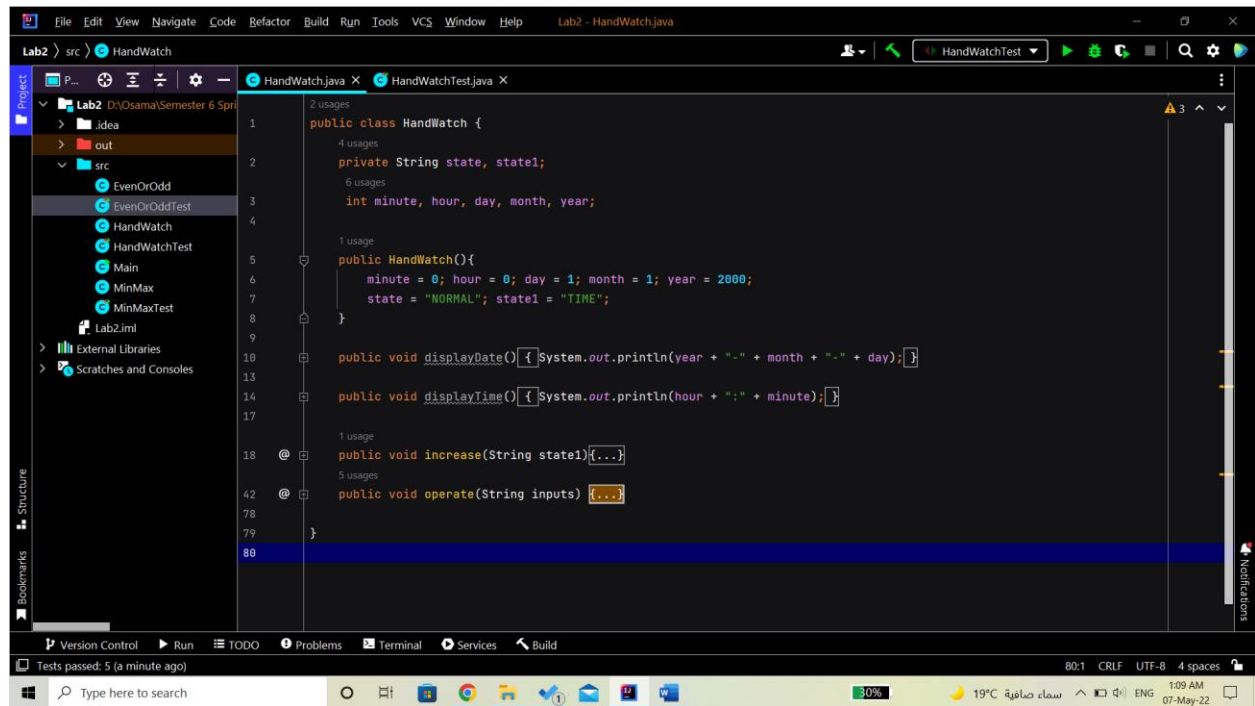


```
1 import org.junit.jupiter.api.*;
2
3 public class MinMaxTest {
4     MinMax object;
5
6     @Nested
7     class Test1{
8         @BeforeEach
9         public void init(){
10             object = new MinMax(new Double[]{1.4, 2.0, 6.3});
11         }
12         @org.junit.jupiter.api.Test
13         public void max6d3Min1d4(){
14             Assertions.assertEquals(expected: 6.3, object.getMax());
15             Assertions.assertEquals(expected: 1.4, object.getMin());
16         }
17     }
18 }
```

```
MinMaxTest.java x MinMax.java x
13 public void maxOddMinId4(){
14     Assertions.assertEquals(expected: 6.3, object.getMax());
15     Assertions.assertEquals(expected: 1.4, object.getMin());
16 }
17 }
18
19 @Nested
20 class Test2{
21     @BeforeEach
22     public void init(){
23         object = new MinMax(new Double[]{-749.4, 0.1, -999.7});
24     }
25     @org.junit.jupiter.api.Test
26     public void maxOddMinNegative999d7(){
27         Assertions.assertEquals(expected: 0.1, object.getMax());
28         Assertions.assertEquals(expected: -999.7, object.getMin());
29     }
30 }
31
32 @AfterEach
33 public void cleanup(){object = null;}
36 }
37
```



2) Question 3 in Sheet 3



```
HandWatch.java X HandWatchTest.java X
1 usage
18 @ public void increase(String state1){
19     switch (state1){
20         case "MINUTE":
21             if (minute < 59) minute++;
22             else minute = 0;
23             break;
24         case "HOUR":
25             if (hour < 23) hour++;
26             else hour = 0;
27             break;
28         case "DAY":
29             if (day < 31) day++;
30             else day = 1;
31             break;
32         case "MONTH":
33             if (month < 11) month++;
34             else month = 1;
35             break;
36         case "YEAR":
37             if (year < 2999 ) year++;
38             else year = 2000;
39     }
40 }
41 }
```

```
HandWatch.java X HandWatchTest.java X
5 usages
42 @ public void operate(String inputs) {
43     for (char input : inputs.toCharArray()) {
44
45         switch (state) {
46             case "NORMAL":
47                 if (input == 'c') {
48                     state = "UPDATE";
49                     state1 = "MINUTE";
50                 }
51                 else if (input == 'b') {
52                     state = "ALARM";
53                     state1 = "ALARM";
54                 }
55                 else if (input == 'a'){
56                     if (state1.equals("TIME")) state1 = "DATE";
57                     else state1 = "TIME";
58                 }
59                 break;
60     }
```

```
HandWatch.java X HandWatchTest.java X
58
59         break;
60
61         case "UPDATE":
62             if (input == 'b') increase(state1);
63             else if (input == 'a') {
64                 if (state1.equals("MINUTE")) state1 = "HOUR";
65                 else if (state1.equals("HOUR")) state1 = "DAY";
66                 else if (state1.equals("DAY")) state1 = "MONTH";
67                 else if (state1.equals("MONTH")) state1 = "YEAR";
68             }
69         }
70         break;
71
72         case "ALARM":
73             if (input == 'a') state1 = "CHIME" ;
74             break;
75     }
76 }
77 }
78
79 }
80
```

```
HandWatch.java X HandWatchTest.java X
1 import org.junit.jupiter.api.AfterEach;
2 import org.junit.jupiter.api.BeforeEach;
3 import org.junit.jupiter.api.Nested;
4
5
6
7 import static org.junit.Assert.assertEquals;
8
9 public class HandWatchTest {
10     HandWatch object;
11
12     @Nested
13     class Test{
14         @BeforeEach
15         public void init(){object = new HandWatch();}
16
17
18
19         @org.junit.jupiter.api.Test
20         public void ExpectedIncMinTo1(){
21             object.operate(inputs: "cb");
22             assertEquals(expected: 1, object.minute);
23         }
24
25         @org.junit.jupiter.api.Test
26         public void ExpectedIncHourTo1(){
27             object.operate(inputs: "cab");
28             assertEquals(expected: 1, object.hour );
29         }
30     }
31 }
```



```
HandWatch.java x HandWatchTest.java x
27 assertEquals(expected: 1, object.hour);
28 }
29 @org.junit.jupiter.api.Test
30 public void ExpectedIncDayTo2(){
31     object.operate(inputs: "caab");
32     assertEquals(expected: 2, object.day);
33 }
34 @org.junit.jupiter.api.Test
35 public void ExpectedIncMonthTo2(){
36     object.operate(inputs: "caaab");
37     assertEquals(expected: 2, object.month);
38 }
39 @org.junit.jupiter.api.Test
40 public void ExpectedIncYearTo2001(){
41     object.operate(inputs: "caaaab");
42     assertEquals(expected: 2001, object.year);
43 }
44 @AfterEach
45 public void cleanup(){ object = null; }
48 }
49
50 }
51
```

