CSE338: Software Testing, Validation, and Verification

Lab 2 Report

Name: Marc Nagy Nasry

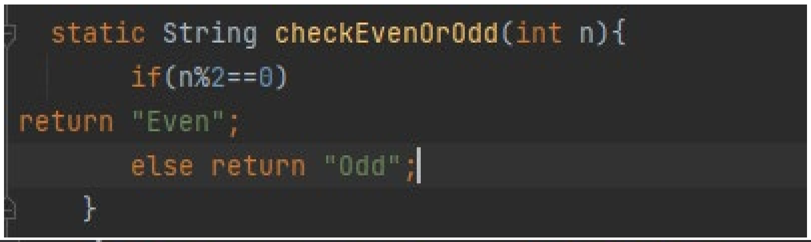
ID: 19P3041

GitHub Repository: https:github.com/marcnagy/lab-2-report.git

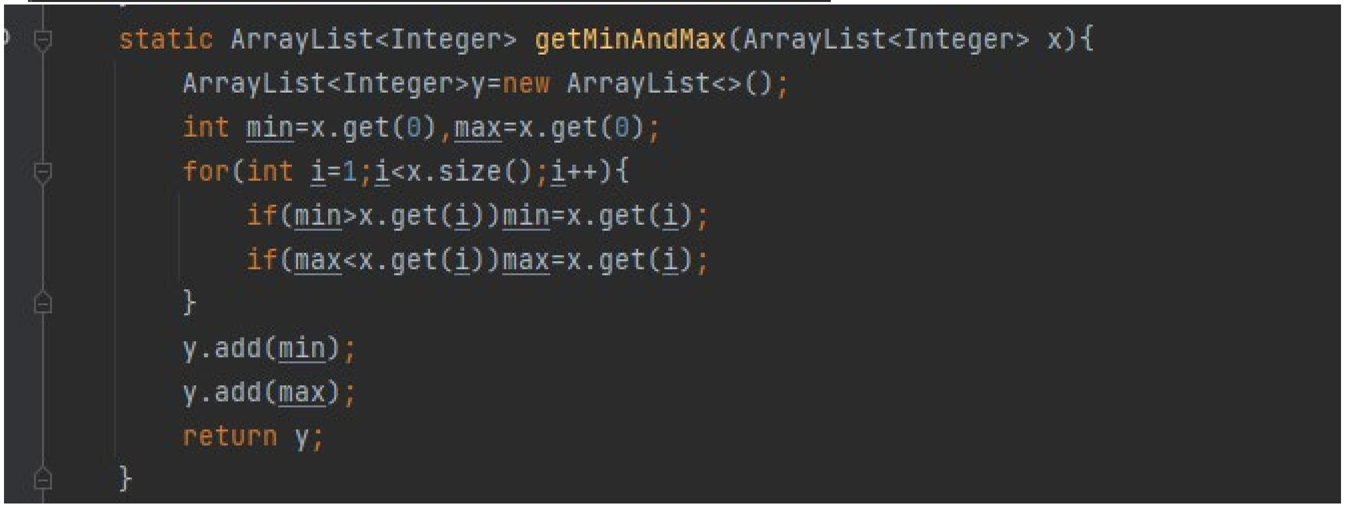
Question 1

Checking even and odd numbers and finding the maximum and minimum value in an array.

Java Function:



```
  static String checkEvenOrOdd(int n){
        if(n%2==0)
return "Even";
        else return "Odd";
    }
```



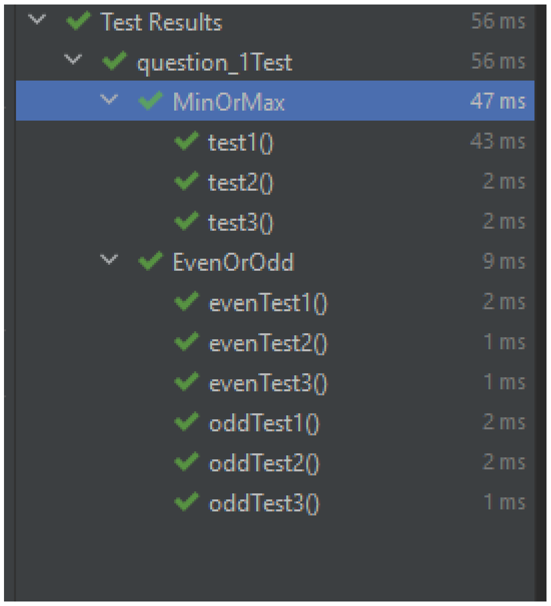
```
    static ArrayList<Integer> getMinAndMax(ArrayList<Integer> x){
        ArrayList<Integer>y=new ArrayList<>();
        int min=x.get(0),max=x.get(0);
        for(int i=1;i<x.size();i++){
            if(min>x.get(i))min=x.get(i);
            if(max<x.get(i))max=x.get(i);
        }
        y.add(min);
        y.add(max);
        return y;
    }
```

<u>Test Junit File:</u>

For the even and odd function, it is being tested on various numbers that the result of it is known by eye to check whether it is functional or not. For finding the maximum and minimum function, they are being tested on various arrays with different elements which is then compared using the built-in functions within Java.

```java
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Nested;
import org.junit.jupiter.api.Test;

import java.util.ArrayList;

import static org.junit.jupiter.api.Assertions.*;

class question_1Test {
    question_1 x;
   @BeforeEach
   void init(){
        x=new question_1();
   }
   @Nested
    class EvenOrOdd{
       @Test
       void evenTest1(){
          assertEquals(question_1.checkEvenOrOdd(4),"Even");
       }
       @Test
       void evenTest2(){
          assertEquals(question_1.checkEvenOrOdd(99990),"Even");
       }
       @Test
       void evenTest3(){
          assertEquals(question_1.checkEvenOrOdd(120323238),"Even");
       }
       @Test
       void oddTest1(){
          assertEquals(question_1.checkEvenOrOdd(3),"Odd");
       }
       @Test
       void oddTest2(){
          assertEquals(question_1.checkEvenOrOdd(2131123125),"Odd");
       }
       @Test
       void oddTest3(){
          assertEquals(question_1.checkEvenOrOdd(2327),"Odd");
       }
   }
   @Nested
    class MinOrMax{
      ArrayList<Integer>y;
      @BeforeEach
      void init(){
        y=new ArrayList<>();
      }
       @Test
       void test1(){
          y.add(-1000);
          y.add(7242);
          y.add(1232);
          y.add(232);
          y.add(2);
          ArrayList<Integer>out=new ArrayList<>();
          out.add(-1000);
          out.add(7242);
          assertEquals(question_1.getMinAndMax(y),out);
       }
       @Test
       void test2(){
          y.add(0);
          y.add(724223);
          y.add(12312);
          y.add(3232);
          y.add(4);
          y.add(213);
          y.add(222222);
          ArrayList<Integer>out=new ArrayList<>();
          out.add(0);
          out.add(724223);
          assertEquals(question_1.getMinAndMax(y),out);
       }
       @Test
       void test3(){
         y.add(123320);
          y.add(72423223);
          y.add(123122);
          y.add(32321);
          y.add(4);
          y.add(21332);
          y.add(222222299);
          ArrayList<Integer>out=new ArrayList<>();
          out.add(4);
          out.add(222222299);
          assertEquals(question_1.getMinAndMax(y),out);}
      @AfterEach
      void cleanUp(){
        y=null;
      }
   }

   @AfterEach
    void cleanUp(){
       x=null;
   }
}
```

Result:



```
Test Results            56 ms
  question_1Test        56 ms
    MinOrMax            47 ms
      test1()           43 ms
      test2()            2 ms
      test3()            2 ms
    EvenOrOdd            9 ms
      evenTest1()        2 ms
      evenTest2()        1 ms
      evenTest3()        1 ms
      oddTest1()         2 ms
      oddTest2()         2 ms
      oddTest3()         1 ms
```

Question 2:

Solving Question 3 on Sheet 3.

Java Class:

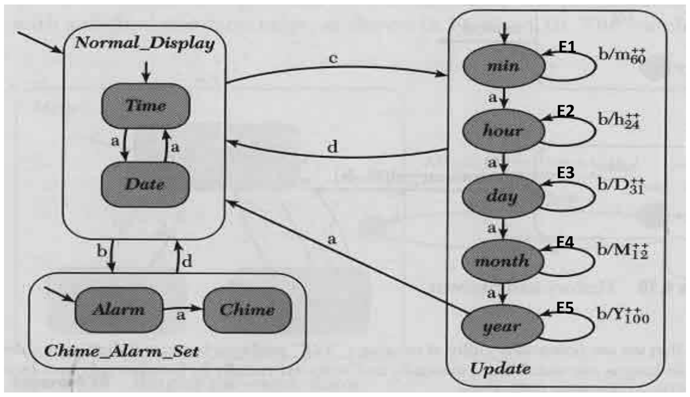
```
import java.util.Scanner;

public class question_2 {
    public String state= "NORMAL";
    public String state1="TIME";
    public int m=0,h=0, D=1,M=1, Y=2000;
    public String[] clock(char input){
        if (state=="NORMAL") {
                    if (input =='c'){ state="UPDATE"; state1 = "min";}
                    if (input =='b') {state="ALARM";  state1 = "Alarm";}
                    if (input =='a')
                    {  if (state1=="TIME")
                            state1="DATE";
                        else
                            state1="TIME";
                    }
                }
        if (state=="UPDATE"){
            if (input =='d') {state="NORMAL";state1 = "TIME";}
            if (input =='a'){
                if (state1.equals("year")) {
                    state = "NORMAL";
                    state1 = "TIME";
                }
                if (state1.equals("month")) {
                    state1 = "year";
                }
                if (state1.equals("day")) {
                    state1 = "month";
                }
                if (state1.equals("hour")) {
                    state1 = "day";
                }
                if (state1.equals("min") ) {
                    state1 = "hour";
                }
            }
            if (input == 'b') {
                if (state1.equals("min")) {
                    m++;
                    if (m >= 60) {
                        m = 0;
                    }
                }
                if (state1.equals("hour")) {
                    h++;
                    if (h >= 24) {
                        h = 0;
```

```java
                    }
                }
                if (state1.equals("day")) {
                    D++;
                    if (D > 31) {
                        D = 1;
                    }
                }
                if (state1.equals("month")) {
                    M++;
                    if (M > 12) {
                        M = 1;
                    }
                }
                if (state1.equals("year")) {
                    Y++;
                    if (Y > 2100) {
                        Y = 2000;
                    }
                }
            }}

                if (state=="ALARM"){
                    if (input =='d') {state="NORMAL";state1="TIME";}
                    if (input =='a') state1="Chime";
                }

        return new String[]{state, state1, String.valueOf(Y) + "-" +
String.valueOf(M) + "-" + String.valueOf(D), String.valueOf(h) + ":" +
String.valueOf(m) };
    }

    public static void main(String[] args) {
        String x[]=new question_2().clock('a');
        for (int i=0;i<x.length;i++)
        System.out.println(x[i]);
    }
}
```

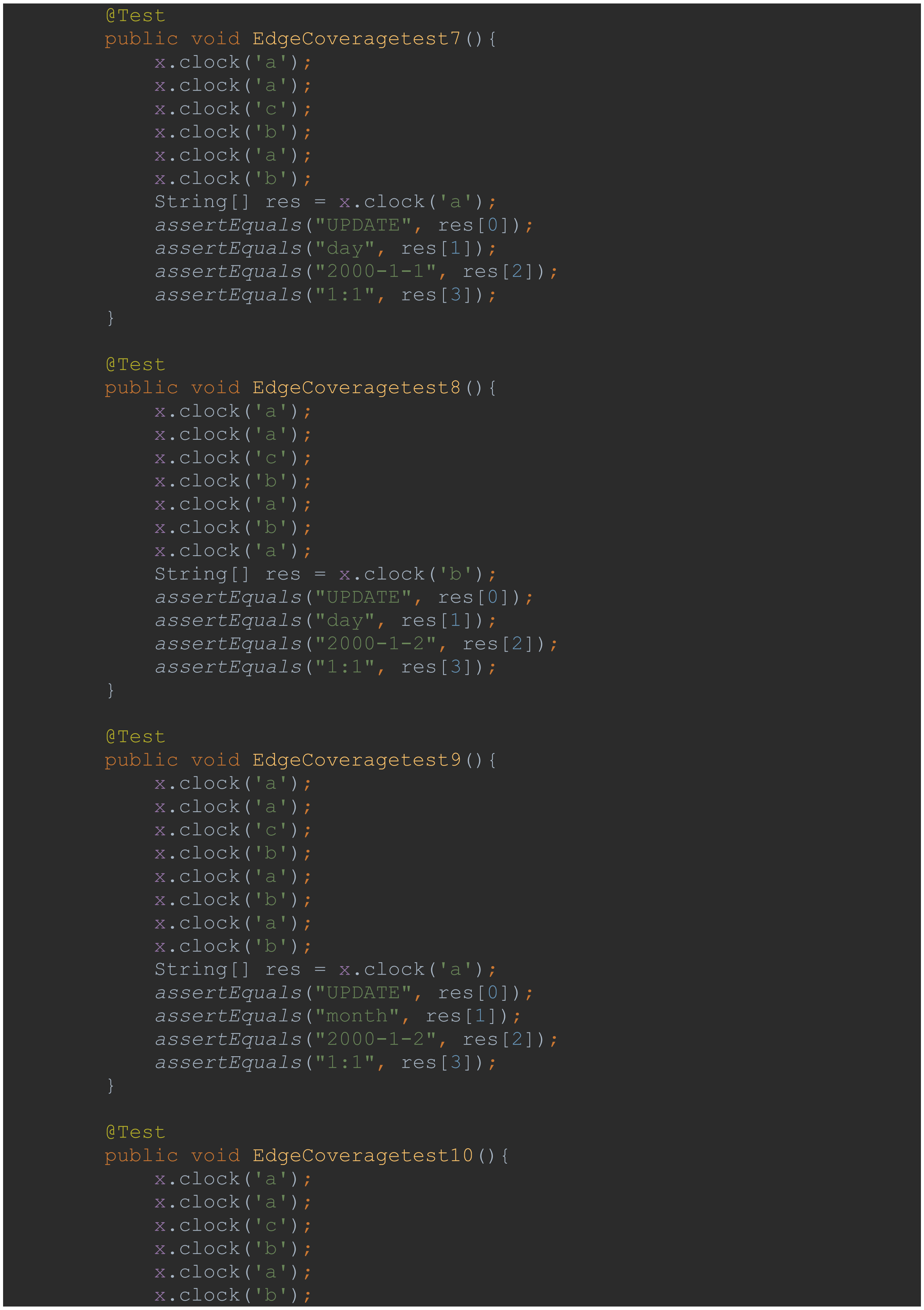
Test Junit File:

Each test coverage is made using 1 test suite each. This test suite is then divided into different tests with a new step added each test. A new object of the clock is created before each test using a the function @beforeEach.

For the ADUP test suite. It's considered that E(1-5) as labelled in the figure are a define and use for their respective variable. In addition to the initializing definition of the variables.



```
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Nested;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class question_2Test {
    @Nested
    class EdgEdgeCoverageoverage{

        question_2 x;

        @BeforeEach
        public void setup(){
            x = new  question_2();
        }

        @Test
        public void EdgeCoveragetest1(){
            String[] res = x.clock('a');
            assertEquals("NORMAL", res[0]);
            assertEquals("DATE", res[1]);
            assertEquals("2000-1-1", res[2]);
            assertEquals("0:0", res[3]);
        }

        @Test
        public void EdgeCoveragetest2(){
            x.clock('a');
            String[] res = x.clock('a');
            assertEquals("NORMAL", res[0]);
            assertEquals("TIME", res[1]);
            assertEquals("2000-1-1", res[2]);
            assertEquals("0:0", res[3]);
        }

        @Test
        public void EdgeCoveragetest3(){
            x.clock('a');
            x.clock('a');
            String[] res = x.clock('c');
            assertEquals("UPDATE", res[0]);
            assertEquals("min", res[1]);
            assertEquals("2000-1-1", res[2]);
            assertEquals("0:0", res[3]);
        }

        @Test
        public void EdgeCoveragetest4(){
            x.clock('a');
            x.clock('a');
            x.clock('c');
            String[] res = x.clock('b');
            assertEquals("UPDATE", res[0]);
            assertEquals("min", res[1]);
            assertEquals("2000-1-1", res[2]);
            assertEquals("0:1", res[3]);
        }

        @Test
        public void EdgeCoveragetest5(){
            x.clock('a');
            x.clock('a');
            x.clock('c');
            x.clock('b');
            String[] res = x.clock('a');
            assertEquals("UPDATE", res[0]);
            assertEquals("hour", res[1]);
            assertEquals("2000-1-1", res[2]);
            assertEquals("0:1", res[3]);
        }

        @Test
        public void EdgeCoveragetest6(){
            x.clock('a');
            x.clock('a');
            x.clock('c');
            x.clock('b');
            x.clock('a');
            String[] res = x.clock('b');
            assertEquals("UPDATE", res[0]);
            assertEquals("hour", res[1]);
            assertEquals("2000-1-1", res[2]);
            assertEquals("1:1", res[3]);
        }
```



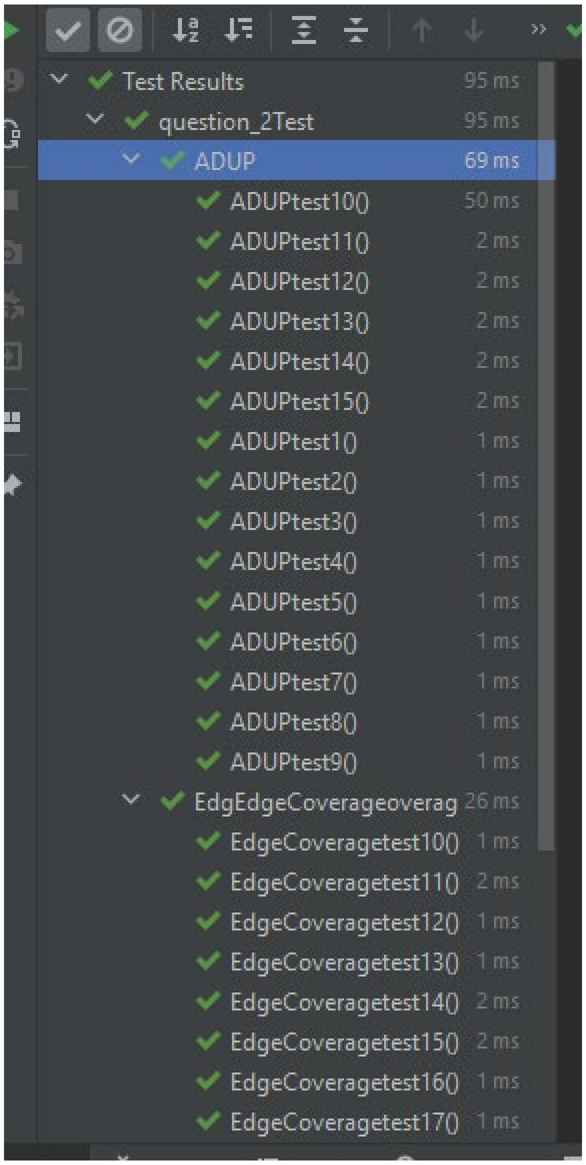
```java
    @Test
    public void EdgeCoveragetest7(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        String[] res = x.clock('a');
        assertEquals("UPDATE", res[0]);
        assertEquals("day", res[1]);
        assertEquals("2000-1-1", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest8(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("day", res[1]);
        assertEquals("2000-1-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest9(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        String[] res = x.clock('a');
        assertEquals("UPDATE", res[0]);
        assertEquals("month", res[1]);
        assertEquals("2000-1-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest10(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("month", res[1]);
        assertEquals("2000-2-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest11(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        String[] res = x.clock('a');
        assertEquals("UPDATE", res[0]);
        assertEquals("year", res[1]);
        assertEquals("2000-2-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest12(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("year", res[1]);
        assertEquals("2001-2-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest13(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        String[] res = x.clock('a');
        assertEquals("NORMAL", res[0]);
        assertEquals("TIME", res[1]);
        assertEquals("2001-2-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest14(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        String[] res = x.clock('c');
        assertEquals("UPDATE", res[0]);
        assertEquals("min", res[1]);
        assertEquals("2001-2-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest15(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('c');
        String[] res = x.clock('d');
        assertEquals("NORMAL", res[0]);
        assertEquals("TIME", res[1]);
        assertEquals("2001-2-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest16(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('c');
        x.clock('d');
        String[] res = x.clock('b');
        assertEquals("ALARM", res[0]);
        assertEquals("Alarm", res[1]);
        assertEquals("2001-2-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest17(){
        x.clock('a');
        x.clock('a');
        x.clock('c');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('a');
        x.clock('c');
        x.clock('d');
        x.clock('b');
        String[] res = x.clock('a');
        assertEquals("ALARM", res[0]);
        assertEquals("Chime", res[1]);
        assertEquals("2001-2-2", res[2]);
        assertEquals("1:1", res[3]);
    }

    @Test
    public void EdgeCoveragetest18(){
        x.clock('a');
        x.clock('a');
```

```
            x.clock('c');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('a');
            x.clock('c');
            x.clock('d');
            x.clock('b');
            x.clock('a');
            String[] res = x.clock('d');
            assertEquals("NORMAL", res[0]);
            assertEquals("TIME", res[1]);
            assertEquals("2001-2-2", res[2]);
            assertEquals("1:1", res[3]);
        }

    }


    @Nested
    class ADUP{

        question_2 x;

        @BeforeEach
        public void setup(){
            x = new  question_2();
        }

        @Test
        public void ADUPtest1(){
            String[] res = x.clock('c');
            assertEquals("UPDATE", res[0]);
            assertEquals("min", res[1]);
            assertEquals("2000-1-1", res[2]);
            assertEquals("0:0", res[3]);
        }

        @Test
        public void ADUPtest2(){
            x.clock('c');
            String[] res = x.clock('b');
            assertEquals("UPDATE", res[0]);
            assertEquals("min", res[1]);
            assertEquals("2000-1-1", res[2]);
            assertEquals("0:1", res[3]);
        }

        @Test
        public void ADUPtest3(){
```

```java
        x.clock('c');
        x.clock('b');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("min", res[1]);
        assertEquals("2000-1-1", res[2]);
        assertEquals("0:2", res[3]);
    }

    @Test
    public void ADUPtest4(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        String[] res = x.clock('a');
        assertEquals("UPDATE", res[0]);
        assertEquals("hour", res[1]);
        assertEquals("2000-1-1", res[2]);
        assertEquals("0:2", res[3]);
    }

    @Test
    public void ADUPtest5(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("hour", res[1]);
        assertEquals("2000-1-1", res[2]);
        assertEquals("1:2", res[3]);
    }

    @Test
    public void ADUPtest6(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("hour", res[1]);
        assertEquals("2000-1-1", res[2]);
        assertEquals("2:2", res[3]);
    }

    @Test
    public void ADUPtest7(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        String[] res = x.clock('a');
        assertEquals("UPDATE", res[0]);
        assertEquals("day", res[1]);
        assertEquals("2000-1-1", res[2]);
        assertEquals("2:2", res[3]);
    }

    @Test
    public void ADUPtest8(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("day", res[1]);
        assertEquals("2000-1-2", res[2]);
        assertEquals("2:2", res[3]);
    }

    @Test
    public void ADUPtest9(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("day", res[1]);
        assertEquals("2000-1-3", res[2]);
        assertEquals("2:2", res[3]);
    }

    @Test
    public void ADUPtest10(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        String[] res = x.clock('a');
        assertEquals("UPDATE", res[0]);
        assertEquals("month", res[1]);
        assertEquals("2000-1-3", res[2]);
        assertEquals("2:2", res[3]);
    }

    @Test
    public void ADUPtest11(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("month", res[1]);
        assertEquals("2000-2-3", res[2]);
        assertEquals("2:2", res[3]);
    }

    @Test
    public void ADUPtest12(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        String[] res = x.clock('b');
        assertEquals("UPDATE", res[0]);
        assertEquals("month", res[1]);
        assertEquals("2000-3-3", res[2]);
        assertEquals("2:2", res[3]);
    }

    @Test
    public void ADUPtest13(){
        x.clock('c');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        x.clock('a');
        x.clock('b');
        x.clock('b');
        String[] res = x.clock('a');
```

```
            assertEquals("UPDATE", res[0]);
            assertEquals("year", res[1]);
            assertEquals("2000-3-3", res[2]);
            assertEquals("2:2", res[3]);
        }

        @Test
        public void ADUPtest14(){
            x.clock('c');
            x.clock('b');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('b');
            x.clock('a');
            String[] res = x.clock('b');
            assertEquals("UPDATE", res[0]);
            assertEquals("year", res[1]);
            assertEquals("2001-3-3", res[2]);
            assertEquals("2:2", res[3]);
        }

        @Test
        public void ADUPtest15(){
            x.clock('c');
            x.clock('b');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            x.clock('b');
            x.clock('a');
            x.clock('b');
            String[] res = x.clock('b');
            assertEquals("UPDATE", res[0]);
            assertEquals("year", res[1]);
            assertEquals("2002-3-3", res[2]);
            assertEquals("2:2", res[3]);
        }


    }
}
```

Result:



```
Test Results                    95 ms
  question_2Test                95 ms
    ADUP                        69 ms
      ADUPtest10()              50 ms
      ADUPtest11()               2 ms
      ADUPtest12()               2 ms
      ADUPtest13()               2 ms
      ADUPtest14()               2 ms
      ADUPtest15()               2 ms
      ADUPtest1()                1 ms
      ADUPtest2()                1 ms
      ADUPtest3()                1 ms
      ADUPtest4()                1 ms
      ADUPtest5()                1 ms
      ADUPtest6()                1 ms
      ADUPtest7()                1 ms
      ADUPtest8()                1 ms
      ADUPtest9()                1 ms
    EdgEdgeCoverageoverag       26 ms
      EdgeCoveragetest10()       1 ms
      EdgeCoveragetest11()       2 ms
      EdgeCoveragetest12()       1 ms
      EdgeCoveragetest13()       1 ms
      EdgeCoveragetest14()       2 ms
      EdgeCoveragetest15()       2 ms
      EdgeCoveragetest16()       1 ms
      EdgeCoveragetest17()       1 ms
```