Lab 2 report

Name: Mustafa Mohamed zeidan

ID: 19P7998

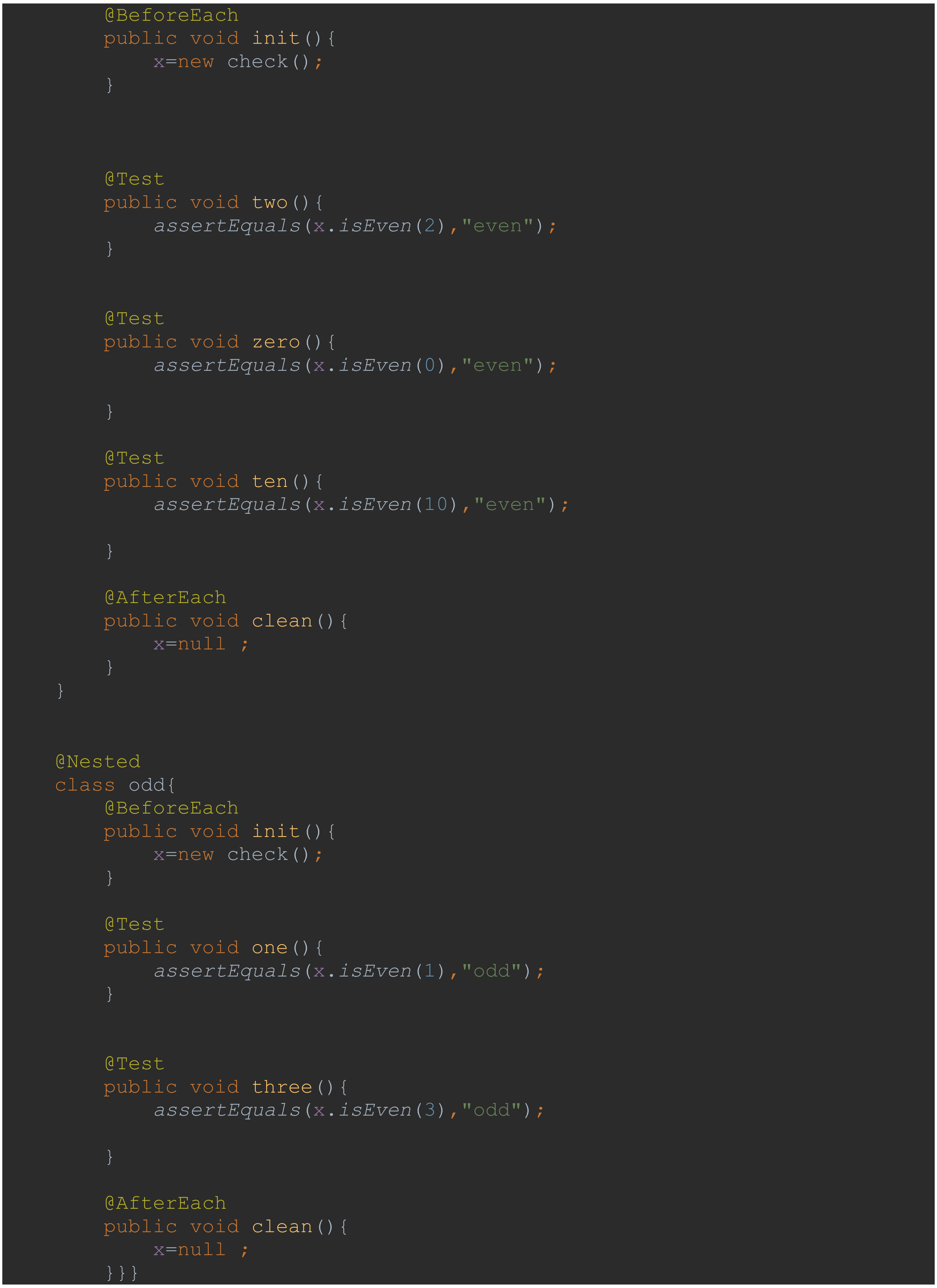
1. Write the code for the following functions then use Junit to apply the unit testing on each of it:
    i. Check even and odd numbers

Code:

```
public class check {

    static String isEven(float x){
        if ( x % 2 ==0)
            return "even";
        return "odd";
    }
```

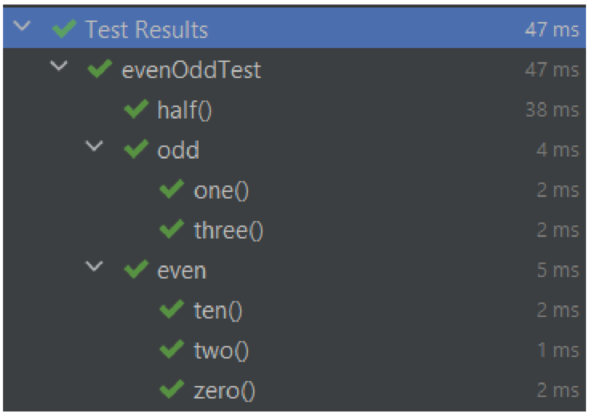
test code:

```
import org.junit.jupiter.api.*;

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

class checkTest {
    check x;

    @BeforeEach
    public void init(){
        x=new check();
    }

    @Test
    public void half(){
        assertEquals(x.isEven(0.5F),"please enter a positive integers
only");
    }

    @AfterEach
    public void clean(){
        x=null ;
    }













    @Nested
    class even{
```



```
        @BeforeEach
        public void init(){
            x=new check();
        }



        @Test
        public void two(){
            assertEquals(x.isEven(2),"even");
        }


        @Test
        public void zero(){
            assertEquals(x.isEven(0),"even");

        }

        @Test
        public void ten(){
            assertEquals(x.isEven(10),"even");

        }

        @AfterEach
        public void clean(){
            x=null ;
        }
    }


    @Nested
    class odd{
        @BeforeEach
        public void init(){
            x=new check();
        }

        @Test
        public void one(){
            assertEquals(x.isEven(1),"odd");
        }


        @Test
        public void three(){
            assertEquals(x.isEven(3),"odd");

        }

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

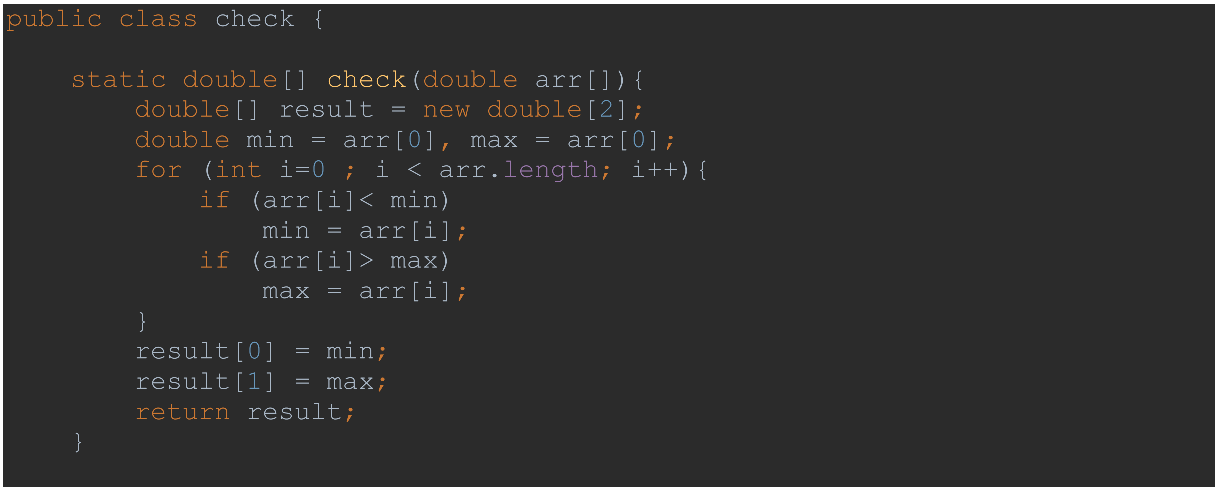
Output:



```
✓ Test Results            47 ms
  ✓ evenOddTest           47 ms
      ✓ half()            38 ms
    ✓ odd                  4 ms
        ✓ one()            2 ms
        ✓ three()          2 ms
    ✓ even                 5 ms
        ✓ ten()            2 ms
        ✓ two()            1 ms
        ✓ zero()           2 ms
```

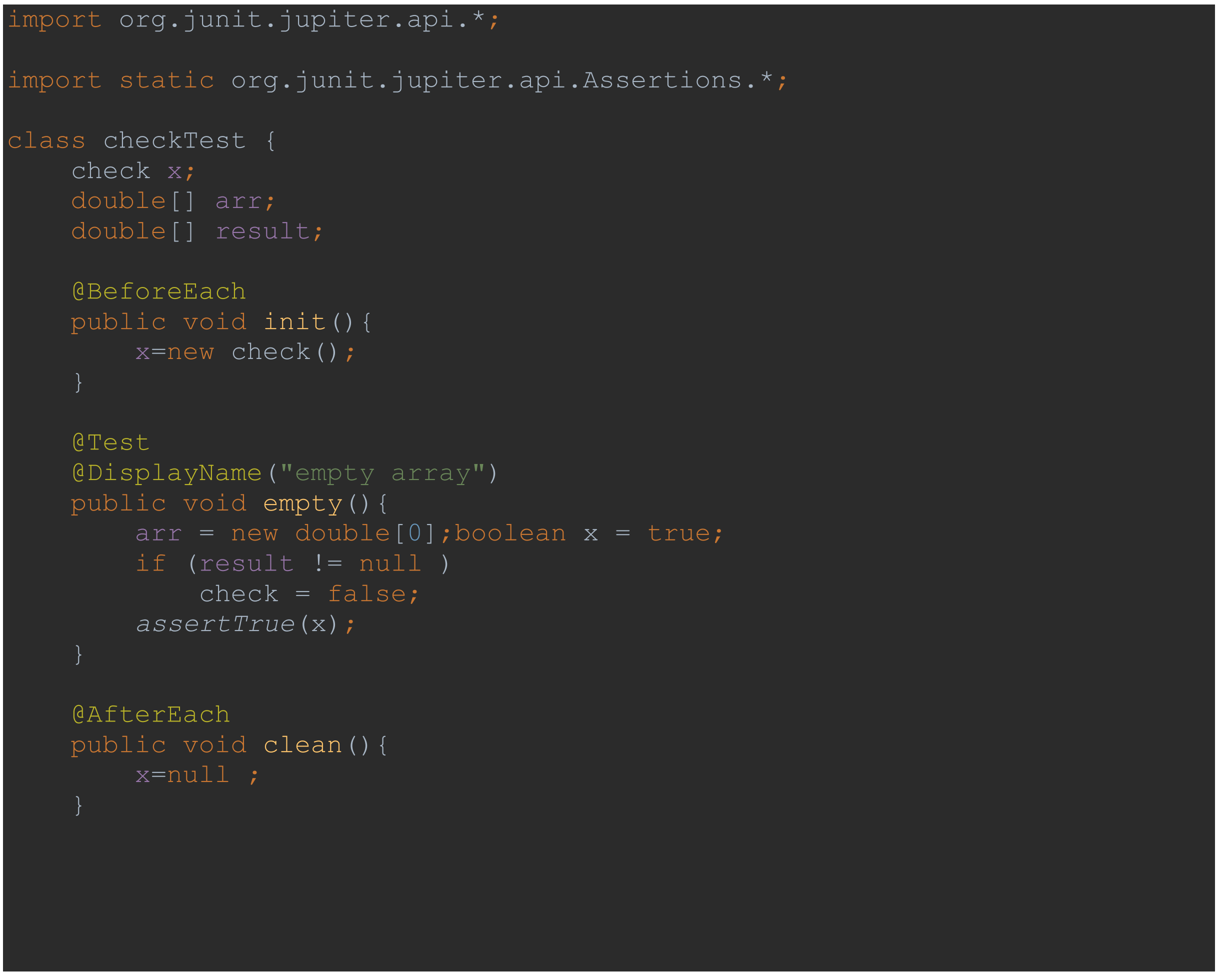
1. Write the code for the following functions then use Junit to apply the unit testing on each of it:
    ii. Finding the maximum and minimum value in an array

Code:

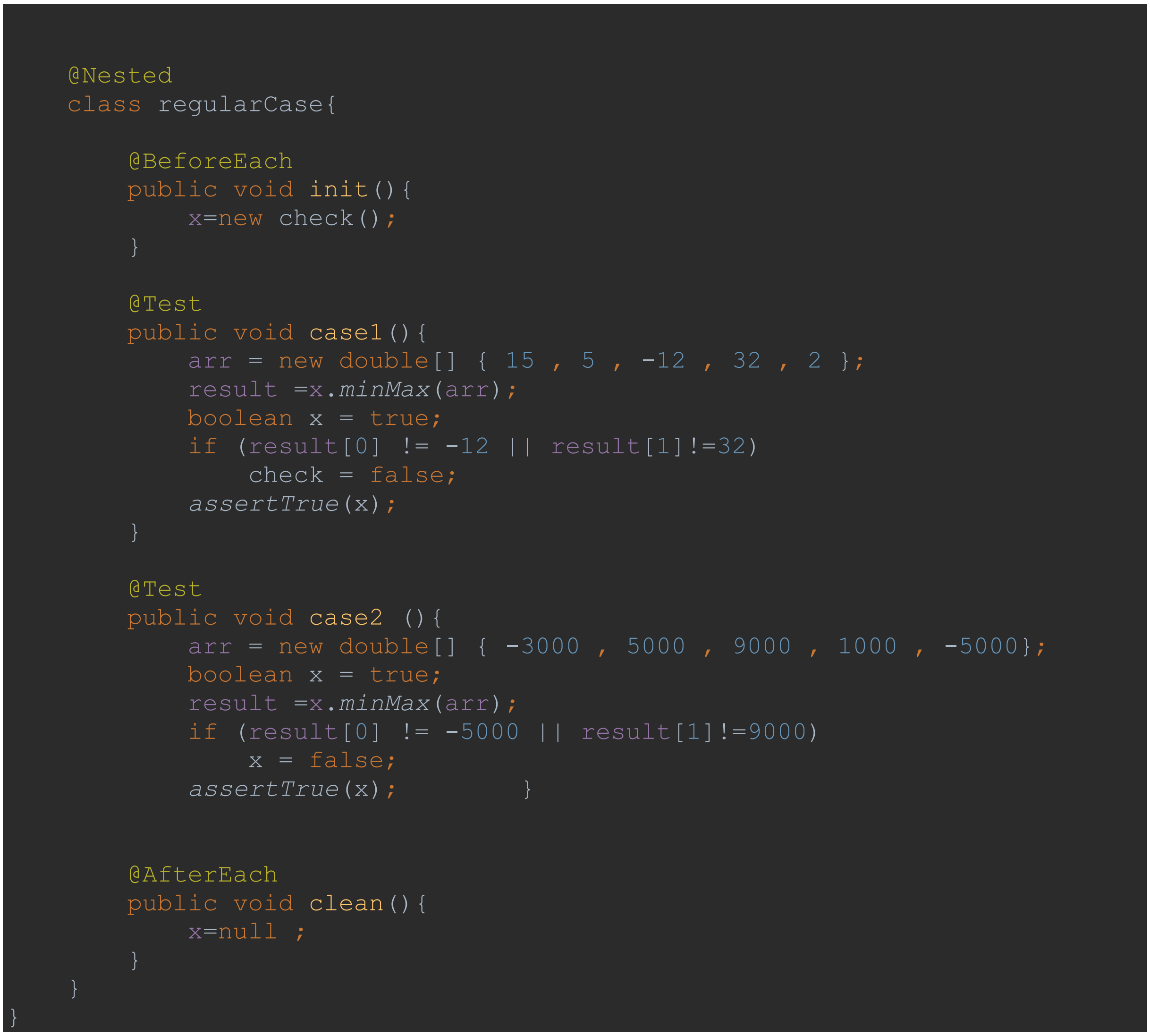


```
public class check {

    static double[] check(double arr[]){
        double[] result = new double[2];
        double min = arr[0], max = arr[0];
        for (int i=0 ; i < arr.length; i++){
            if (arr[i]< min)
                min = arr[i];
            if (arr[i]> max)
                max = arr[i];
        }
        result[0] = min;
        result[1] = max;
        return result;
    }
```

test code:

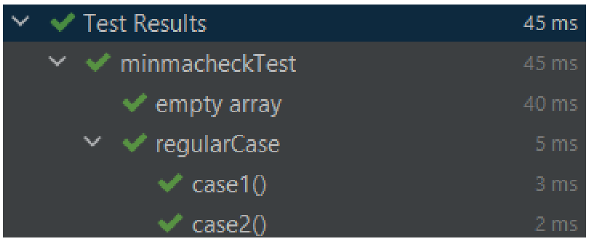


```
import org.junit.jupiter.api.*;

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

class checkTest {
    check x;
    double[] arr;
    double[] result;

    @BeforeEach
    public void init(){
        x=new check();
    }

    @Test
    @DisplayName("empty array")
    public void empty(){
        arr = new double[0];boolean x = true;
        if (result != null )
            check = false;
        assertTrue(x);
    }

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



```
    @Nested
    class regularCase{

        @BeforeEach
        public void init(){
            x=new check();
        }

        @Test
        public void case1(){
            arr = new double[] { 15 , 5 , -12 , 32 , 2 };
            result =x.minMax(arr);
            boolean x = true;
            if (result[0] != -12 || result[1]!=32)
                check = false;
            assertTrue(x);
        }

        @Test
        public void case2 (){
            arr = new double[] { -3000 , 5000 , 9000 , 1000 , -5000};
            boolean x = true;
            result =x.minMax(arr);
            if (result[0] != -5000 || result[1]!=9000)
                x = false;
            assertTrue(x);         }


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

Output:

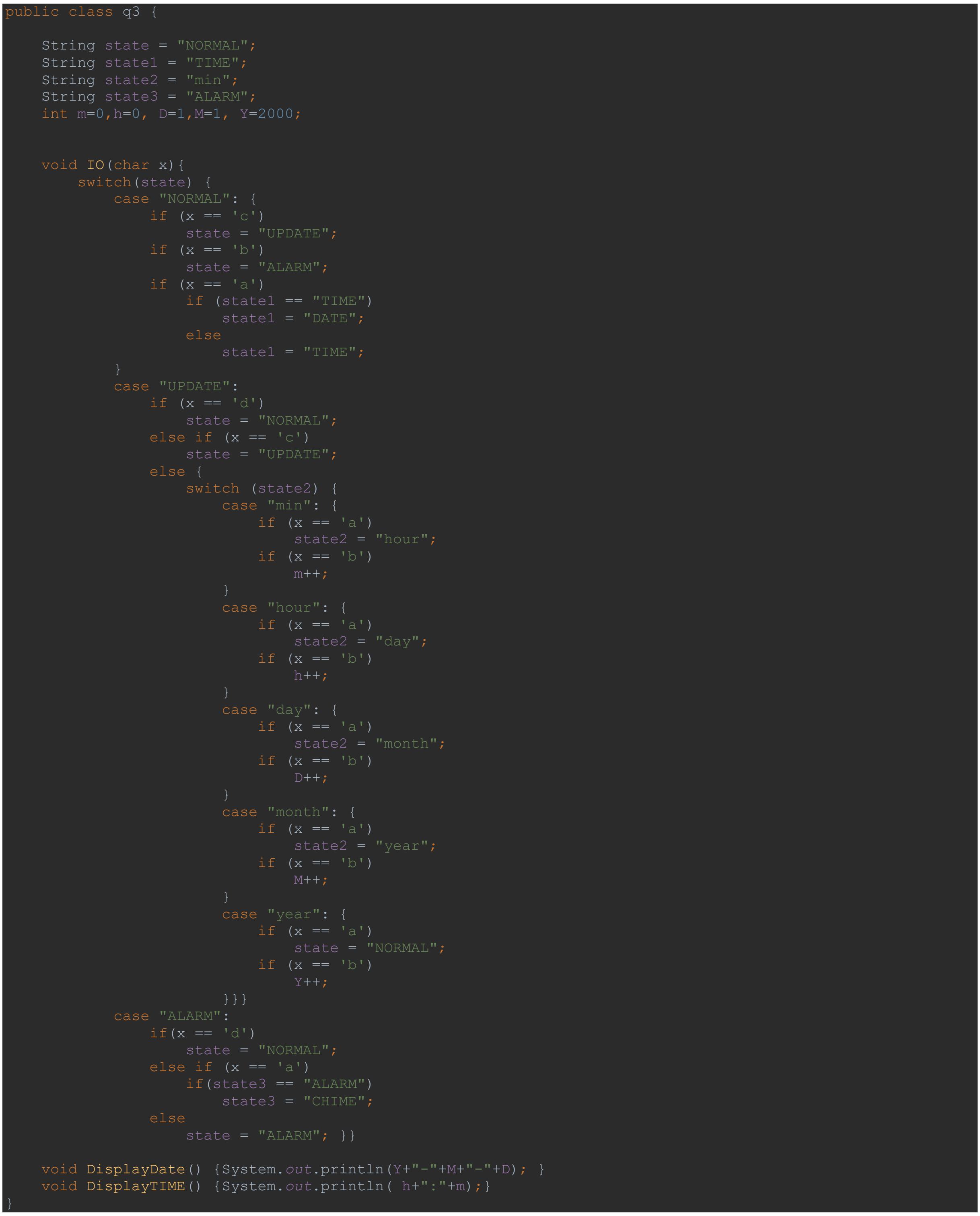


```
✓ Test Results                45 ms
  ✓ minmacheckTest            45 ms
    ✓ empty array             40 ms
    ✓ regularCase              5 ms
      ✓ case1()                3 ms
      ✓ case2()                2 ms
```

Problem3: sheet3 Q3:

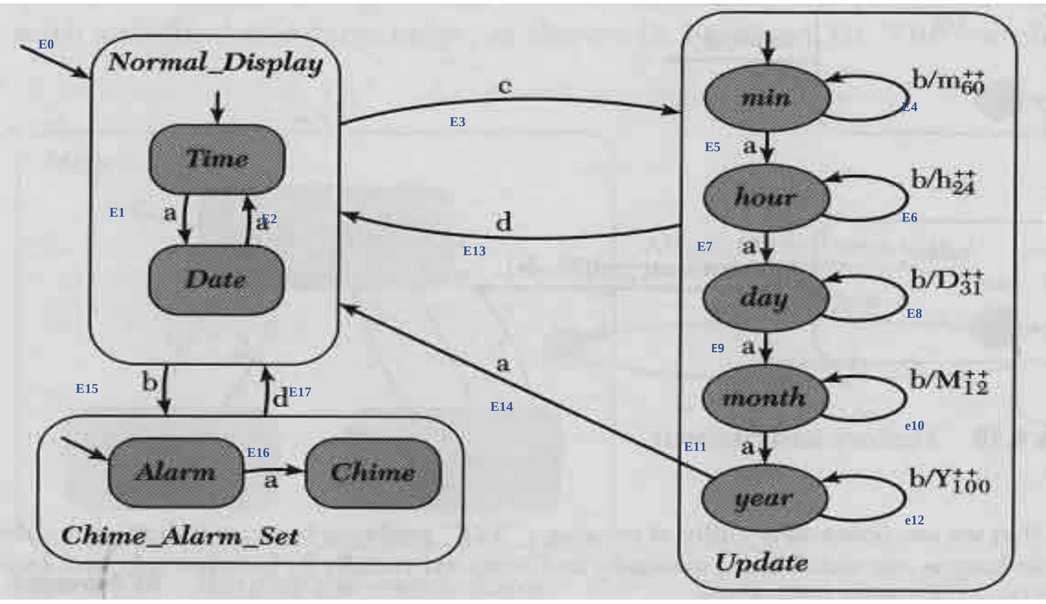
3. **For problem number 2 the following algorithm can be used to model the state chart**
   Input: is either a, b, c, or d output is a variable showing the current state + the values of current time and current date
   Output variables are m=0,h=0, D=1,M=1, Y=2000.
   Output displayed information are Time, Date

1) Complete the algorithm and code it using C/java, so it can be used as a real watch



```
public class q3 {

    String state = "NORMAL";
    String state1 = "TIME";
    String state2 = "min";
    String state3 = "ALARM";
    int m=0,h=0, D=1,M=1, Y=2000;


    void IO(char x){
        switch(state) {
            case "NORMAL": {
                if (x == 'c')
                    state = "UPDATE";
                if (x == 'b')
                    state = "ALARM";
                if (x == 'a')
                    if (state1 == "TIME")
                        state1 = "DATE";
                    else
                        state1 = "TIME";
            }
            case "UPDATE":
                if (x == 'd')
                    state = "NORMAL";
                else if (x == 'c')
                    state = "UPDATE";
                else {
                    switch (state2) {
                        case "min": {
                            if (x == 'a')
                                state2 = "hour";
                            if (x == 'b')
                                m++;
                        }
                        case "hour": {
                            if (x == 'a')
                                state2 = "day";
                            if (x == 'b')
                                h++;
                        }
                        case "day": {
                            if (x == 'a')
                                state2 = "month";
                            if (x == 'b')
                                D++;
                        }
                        case "month": {
                            if (x == 'a')
                                state2 = "year";
                            if (x == 'b')
                                M++;
                        }
                        case "year": {
                            if (x == 'a')
                                state = "NORMAL";
                            if (x == 'b')
                                Y++;
                        }}}
            case "ALARM":
                if(x == 'd')
                    state = "NORMAL";
                else if (x == 'a')
                    if(state3 == "ALARM")
                        state3 = "CHIME";
                else
                    state = "ALARM"; }}

    void DisplayDate() {System.out.println(Y+"-"+M+"-"+D); }
    void DisplayTIME() {System.out.println( h+":"+m);}
}
```

2) find test suite for edge coverage



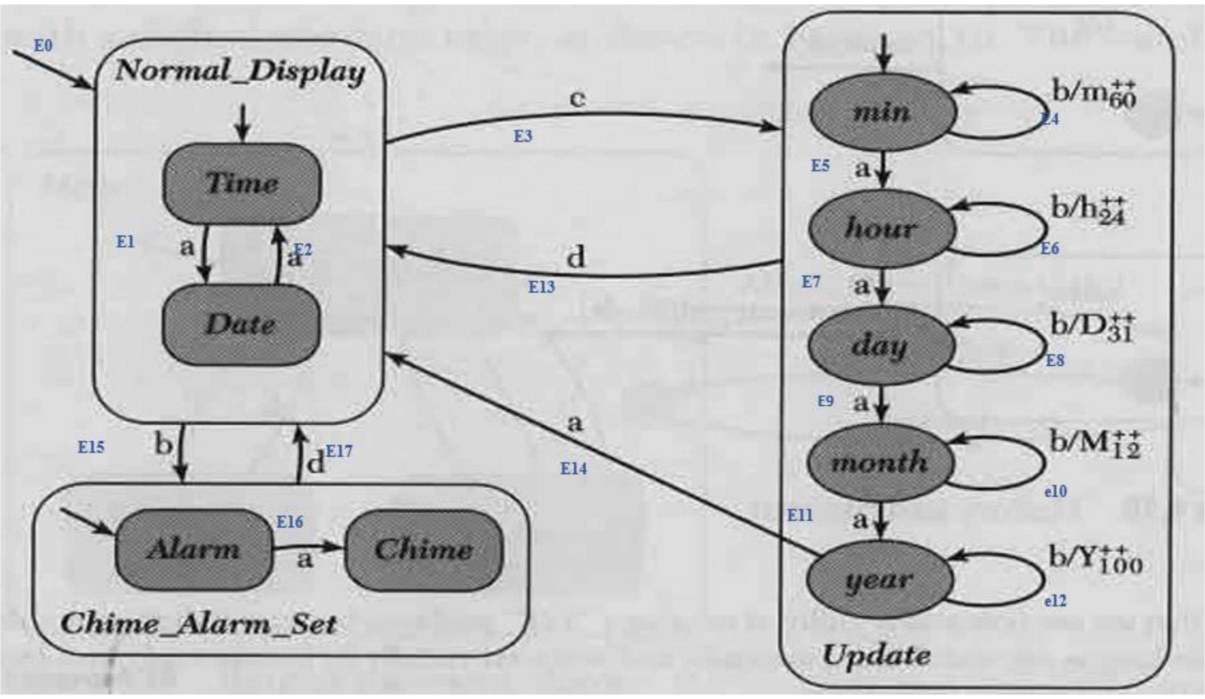
TC1: <a, a, c, b, a, b ,a, b, a, b, a, b, a ,b ,a, d>

TC2:<c,d>

For TC1 the covered edges are: 0,1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,17

For TC1 the covered edges are: 0, 3, 13

3) create an ADUP test suite based on variables: minutes, seconds, hours, day, month and year



edge	d	U
e1		
e2		
e3		
e4	m	m
e5		
e6	h	h
e7		
e8	D	D
e9		
e10	M	M
e11		
e12	Y	Y
e13		
e14		
e15		
e16		
e17		

TC1: <a, a, c, b, b, a, b , b, a, b, b, a, b, b, a, b, b, a, b, b, a, b, b, a ,b , b, a, d>

E4 → E4 (checked)

E6 → E6 (checked)

E8 → E8 (checked)

E10 → E10 (checked)

E12 → E12 (checked)