



**National Institute of Technology, Rourkela**

**CS6474: Software Testing Laboratory**

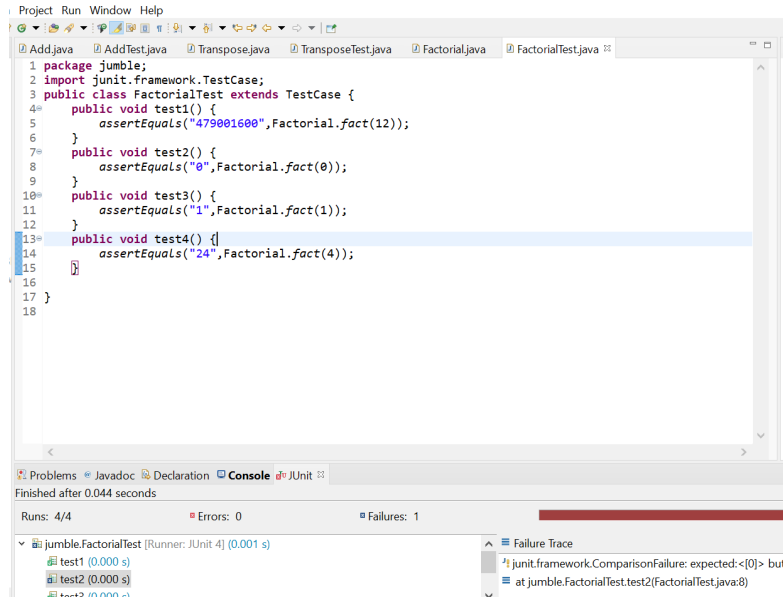
**(Spring 2023)**

**Jumble Assignment**

**Name : Pallavi Priya**  
**Roll number: 222CS3116**

# Solve the below programs in Java

1 Write a program to generate a Factorial of numbers (where stack length should be at 3 (max) ). The numbers should be 5, 3, 8, and 15.



```
1 package jumble;
2 import junit.framework.TestCase;
3 public class FactorialTest extends TestCase {
4     public void test1() {
5         assertEquals("479001600", Factorial.fact(12));
6     }
7     public void test2() {
8         assertEquals("0", Factorial.fact(0));
9     }
10    public void test3() {
11        assertEquals("1", Factorial.fact(1));
12    }
13    public void test4() {
14        assertEquals("24", Factorial.fact(4));
15    }
16 }
17
18
```

Problems Javadoc Declaration Console JUnit

Finished after 0.044 seconds

Runs: 4/4 Errors: 0 Failures: 1

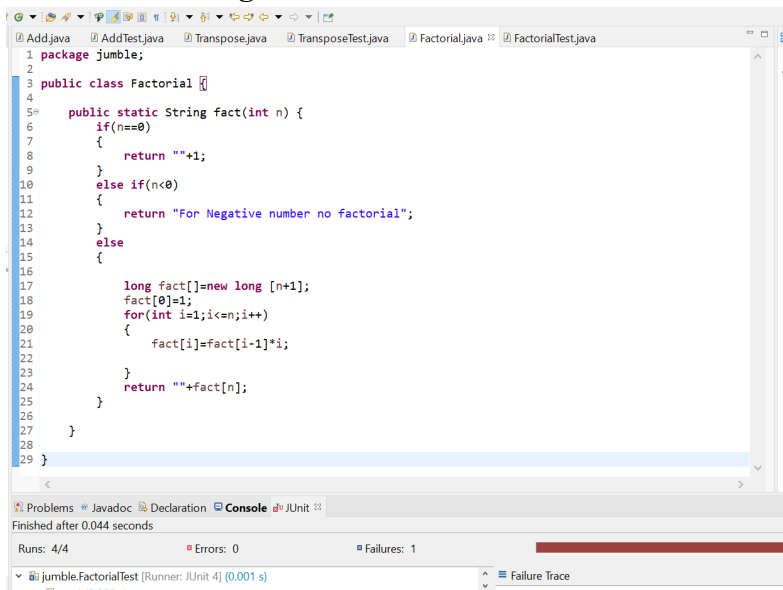
jumble.FactorialTest [Runner: JUnit 4] (0.001 s)

- test1 (0.000 s)
- test2 (0.000 s)
- test3 (0.000 s)

Failure Trace

- junit.framework.ComparisonFailure: expected:<[0]> but was:<[479001600]> at jumble.FactorialTest.test2(FactorialTest.java:8)

Figure 1: Jumble Test Case



```
1 package jumble;
2
3 public class Factorial {
4
5     public static String fact(int n) {
6         if(n==0)
7         {
8             return ""+1;
9         }
10        else if(n<0)
11        {
12            return "For Negative number no factorial";
13        }
14        else
15        {
16            long fact[]=new long [n+1];
17            fact[0]=1;
18            for(int i=1;i<=n;i++)
19            {
20                fact[i]=fact[i-1]*i;
21            }
22            return ""+fact[n];
23        }
24    }
25 }
26
27
28
29
```

Problems Javadoc Declaration Console JUnit

Finished after 0.044 seconds

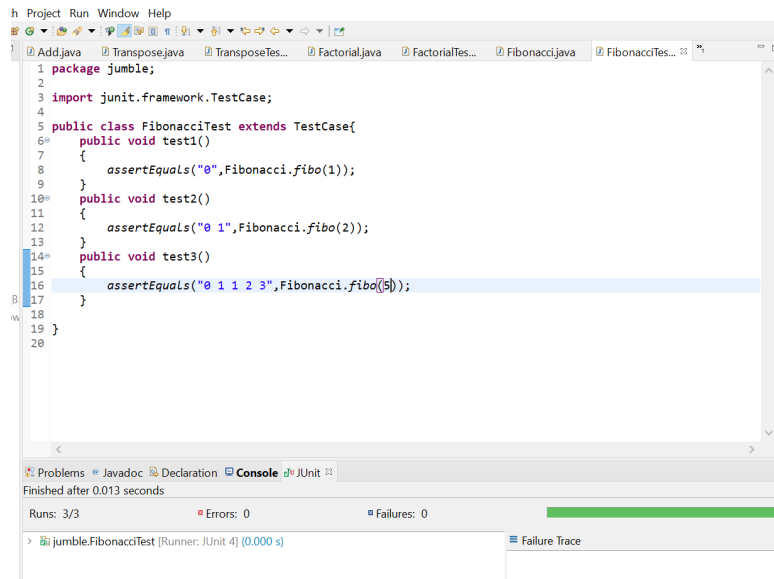
Runs: 4/4 Errors: 0 Failures: 1

jumble.FactorialTest [Runner: JUnit 4] (0.001 s)

Failure Trace

Figure 2: Factorial Code

## 2 Write a program to generate Fibonacci numbers.

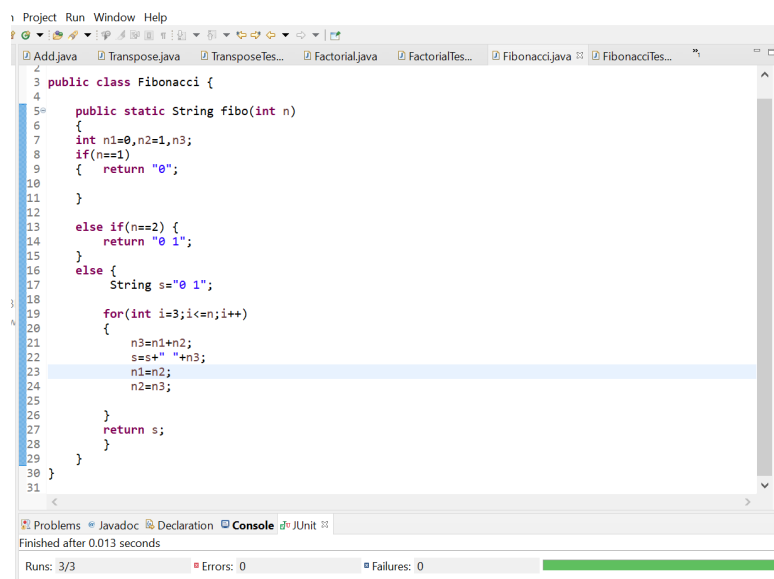


The screenshot shows an IDE window with a Java file named `FibonacciTest.java`. The code is a JUnit test case that tests the `Fibonacci.fibo()` method. It includes three test methods: `test1()`, `test2()`, and `test3()`. The `test3()` method is currently selected and highlighted. The test case is run, and the console shows that it finished after 0.013 seconds with 3/3 runs, 0 errors, and 0 failures.

```
1 package jumble;
2
3 import junit.framework.TestCase;
4
5 public class FibonacciTest extends TestCase{
6     public void test1()
7     {
8         assertEquals("0", Fibonacci.fibo(1));
9     }
10    public void test2()
11    {
12        assertEquals("0 1", Fibonacci.fibo(2));
13    }
14    public void test3()
15    {
16        assertEquals("0 1 1 2 3", Fibonacci.fibo(5));
17    }
18 }
19
20
```

Problems Javadoc Declaration Console JUnit  
Finished after 0.013 seconds  
Runs: 3/3 Errors: 0 Failures: 0  
jumble.FibonacciTest [Runner: JUnit 4] (0.000 s) Failure Trace

Figure 3: Jumble Test Case



The screenshot shows an IDE window with a Java file named `Fibonacci.java`. The code is a public class `Fibonacci` with a static method `fibo(int n)`. The method returns a string of Fibonacci numbers up to the `n`th number. The code is run, and the console shows that it finished after 0.013 seconds with 3/3 runs, 0 errors, and 0 failures.

```
3 public class Fibonacci {
4
5     public static String fibo(int n)
6     {
7         int n1=0, n2=1, n3;
8         if(n==1)
9         {
10            return "0";
11        }
12
13        else if(n==2) {
14            return "0 1";
15        }
16        else {
17            String s="0 1";
18            for(int i=3; i<=n; i++)
19            {
20                n3=n1+n2;
21                s=s+" "+n3;
22                n1=n2;
23                n2=n3;
24            }
25            return s;
26        }
27    }
28 }
29
30
31
```

Problems Javadoc Declaration Console JUnit  
Finished after 0.013 seconds  
Runs: 3/3 Errors: 0 Failures: 0

Figure 4: Fibonacci Code

### 3 Write a program that performs sorting of a group of integer values using the quick sort technique.

```
2 import static org.junit.Assert.assertEquals;
3
4 import junit.framework.TestCase;
5 public class QuickSortTest extends TestCase {
6     public void test1()
7     {
8         int[] arr= {1,4,5,78,2,7,10};
9         int [] output = {1,2,4,5,7,10,78};
10        assertEquals(output, QuickSort.quickSort(arr, 0, 6));
11    }
12    public void test2()
13    {
14        int[] arr= {1,42,78,2,10};
15        int [] output = {1,2,10,42,78};
16        assertEquals(output, QuickSort.quickSort(arr, 0, 4));
17    }
18 }
19 }
20 }
```

Figure 5: Jumble Test Case

```
3 public class QuickSort {
4     public static int partition(int array[], int start, int end)
5     {
6         int pivot = array[end];
7         int i = (start-1);
8         for (int j=start; j<end; j++)
9         {
10            if (array[j] <= pivot)
11            {
12                i++;
13                int temp = array[i];
14                array[i] = array[j];
15                array[j] = temp;
16            }
17        }
18        int temp = array[i+1];
19        array[i+1] = array[end];
20        array[end] = temp;
21        return i+1;
22    }
23    public static int[] quicksort(int arr[], int low, int high)
24    {
25        if (low < high)
26        {
27            int p = partition(arr, low, high);
28            quicksort(arr, low, p-1);
29            quicksort(arr, p+1, high);
30        }
31        return arr;
32    }
33 }
```

Problems @ Javadoc Declaration Console

<terminated> Run Jumble [Java Application] D:\eclipse\workspace\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_15.0.1\bin\java.exe -Dorg.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_15.0.1\bin\java.exe -Dorg.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_15.0.1\bin\java.exe -Dorg.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_15.0.1\bin\java.exe

Mutating soft\_testing\_jumble\_assignment.QuickSort

Tests: soft\_testing\_jumble\_assignment.QuickSortTest

Mutation points = 19, unit test time limit 2.02s

.....M FAIL: (soft\_testing\_jumble\_assignment.QuickSort.java:22): 1 -> 0

.....M FAIL: (soft\_testing\_jumble\_assignment.QuickSort.java:30): 1 -> 0

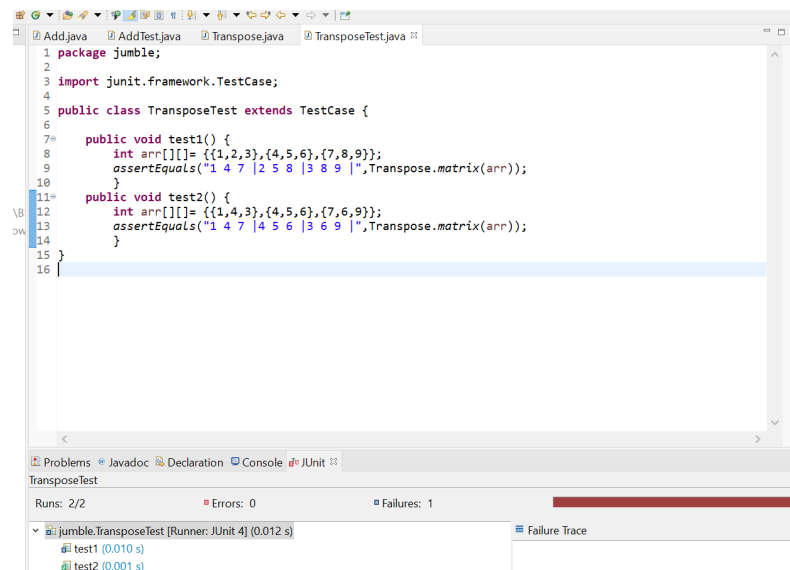
..

Jumbling took 6.661s

Score: 89%

Figure 6: Matrix Transpose Code

4 Write a program that accepts elements of a matrix and displays its transpose.



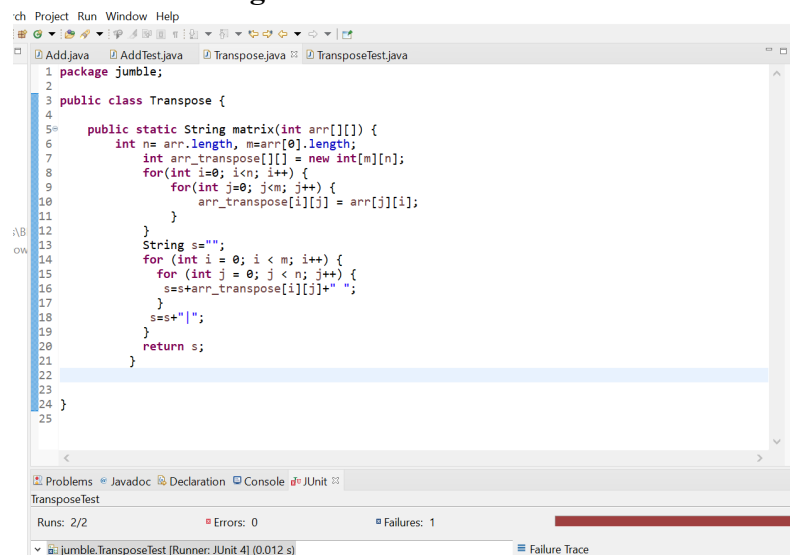
```
1 package jumble;
2
3 import junit.framework.TestCase;
4
5 public class TransposeTest extends TestCase {
6
7     public void test1() {
8         int arr[][] = {{1,2,3},{4,5,6},{7,8,9}};
9         assertEquals("1 4 7 | 2 5 8 | 3 8 9 |", Transpose.matrix(arr));
10    }
11
12    public void test2() {
13        int arr[][] = {{1,4,3},{4,5,6},{7,6,9}};
14        assertEquals("1 4 7 | 4 5 6 | 3 6 9 |", Transpose.matrix(arr));
15    }
16 }
```

Runs: 2/2    Errors: 0    Failures: 1

jumble.TransposeTest [Runner: JUnit 4] (0.012 s)

- test1 (0.010 s)
- test2 (0.001 s)

Figure 7: Jumble Test Case



```
1 package jumble;
2
3 public class Transpose {
4
5     public static String matrix(int arr[][]) {
6         int n = arr.length, m = arr[0].length;
7         int arr_transpose[][] = new int[m][n];
8         for(int i=0; i<n; i++) {
9             for(int j=0; j<m; j++) {
10                 arr_transpose[i][j] = arr[j][i];
11             }
12         }
13         String s="";
14         for (int i = 0; i < m; i++) {
15             for (int j = 0; j < n; j++) {
16                 s+=arr_transpose[i][j]+" ";
17             }
18             s+="| ";
19         }
20         return s;
21     }
22 }
23
24 }
25 }
```

Runs: 2/2    Errors: 0    Failures: 1

jumble.TransposeTest [Runner: JUnit 4] (0.012 s)

Figure 8: Matrix Transpose Code

## 5 Write a program to add two matrices and display the sum matrix.

```

2 import junit.framework.TestCase;
3 import static org.junit.Assert.assertEquals;
4 public class AdditionMatrixTest extends TestCase{
5     public void test1() {
6         int matrix1[][]= {{11,21},{44,58}};
7         int matrix2[][] = {{55,79},{4,5}};
8         int sum[][]= {{66,100},{48,63}};
9         assertEquals(sum,AdditionMatrix.addM(matrix1, matrix2));
10    }
11    public void test2() {
12        int matrix1[][]= {{0,0},{0,0}};
13        int matrix2[][] = {{10,10},{10,10}};
14        int sum[][]= {{10,10},{10,10}};
15        assertEquals(sum,AdditionMatrix.addM(matrix1, matrix2));
16    }
17    public void test3() {
18        int matrix1[][]= {{1,1},{1,1}};
19        int matrix2[][] = {{-2,-2},{2,2}};
20        int sum[][]= {{-1,-1},{3,3}};
21        assertEquals(sum,AdditionMatrix.addM(matrix1, matrix2));
22    }
23 }
24

```

Figure 9: Jumble Test Case

```

3 public class AdditionMatrix {
4     public static int[][] addM(int[][] m1,int[][] m2)
5     {
6         int r = m1.length;
7         int c = m1[0].length;
8         int[][] sum = new int[r][c];
9         for(int i = 0; i < r; i++) {
10             for (int j = 0; j < c; j++) {
11                 sum[i][j] = m1[i][j] + m2[i][j];
12             }
13         }
14         return sum;
15     }
16 }
17

```

Problems @ Javadoc Declaration Console

<terminated> Run Jumble [Java Application] D:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win

Mutating soft\_testing\_jumble\_assignment.AdditionMatrix

Tests: soft\_testing\_jumble\_assignment.AdditionMatrixTest

Mutation points = 9, unit test time limit 2.01s

M FAIL: (soft\_testing\_jumble\_assignment.AdditionMatrix.java:7): 0 -> 1

.....

Jumbling took 3.528s

Score: 88%

Figure 10: Matrix AdditionCode

```

1 public class PrintPrimeTest extends TestCase {
2     public void test1()
3     {
4         assertEquals("2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 ",PrintPrime.testPrime() );
5     }
6 }
7
8
9
10

```

```

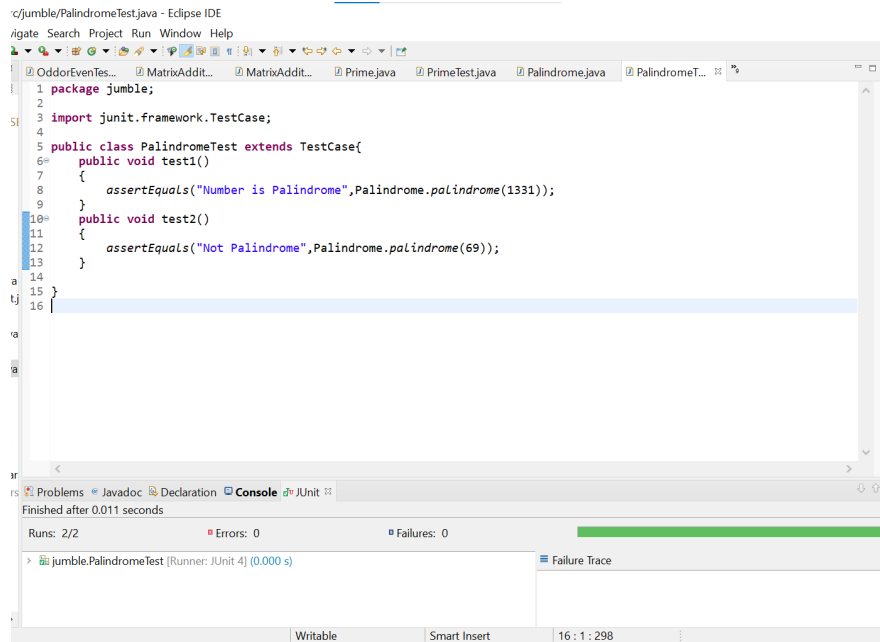
2
3 public class PrintPrime {
4     public static boolean checkPrime(int n)
5     {
6         if(n<=1)
7             return false;
8         for (int i=2;i<=(int)Math.sqrt(n);i++) {
9             if (n % i == 0)
10                return false;
11        }
12        return true;
13    }
14    public static String testPrime()
15    {
16        String result = "";
17        for(int i=1;i <=100;i++)
18        {
19            if (checkPrime(i))
20                result = result + i + " ";
21        }
22        return result;
23    }
24 }
25

```

**Figure 12: Prime Code**

### Figure 12: Prime Code

## 7 Write a program to generate a palindrome of numbers.



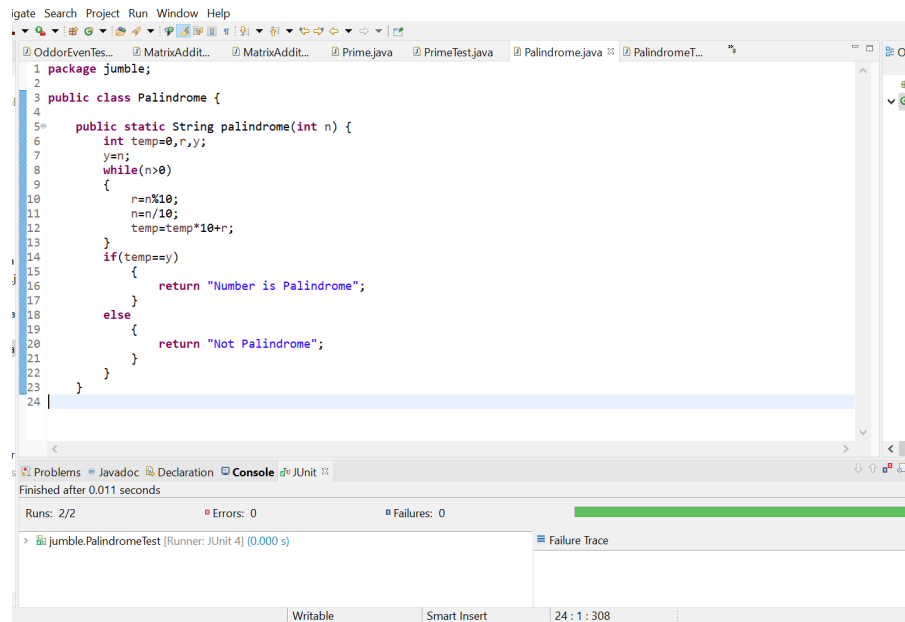
```
c:\jumble\PalindromeTest.java - Eclipse IDE
igate Search Project Run Window Help

package jumble;
import junit.framework.TestCase;

public class PalindromeTest extends TestCase{
    public void test1()
    {
        assertEquals("Number is Palindrome",Palindrome.palindrome(1331));
    }
    public void test2()
    {
        assertEquals("Not Palindrome",Palindrome.palindrome(69));
    }
}

Finished after 0.011 seconds
Runs: 2/2 Errors: 0 Failures: 0
> jumble.PalindromeTest [Runner: JUnit 4] (0.000 s)
Failure Trace
```

Figure 13: Jumble Test Case



```
igate Search Project Run Window Help

package jumble;

public class Palindrome {
    public static String palindrome(int n) {
        int temp=0,r,y;
        y=n;
        while(n>0)
        {
            r=n%10;
            n=n/10;
            temp=temp*10+r;
        }
        if(temp==y)
        {
            return "Number is Palindrome";
        }
        else
        {
            return "Not Palindrome";
        }
    }
}

Finished after 0.011 seconds
Runs: 2/2 Errors: 0 Failures: 0
> jumble.PalindromeTest [Runner: JUnit 4] (0.000 s)
Failure Trace
```

Figure 14: Palindrome Code



## 8 Write a program to find out the sum of two arrays.

```
3 import junit.framework.TestCase;
4
5 public class SumOfArrayTest extends TestCase{
6     public void test1()
7     {
8         int a[]={1,2,3};
9         int b[]={0,1,10};
10        assertEquals(17,SumOfArray.sumArray(a, b));
11    }
12    public void test2()
13    {
14        int a[]={14,89,56};
15        int b[]={5,68,23};
16        assertEquals(255,SumOfArray.sumArray(a, b));
17    }
18    public void test3()
19    {
20        int a[]={4,9,6};
21        int b[]={2,8,3};
22        assertEquals(32,SumOfArray.sumArray(a, b));
23    }
24 }
25
```

Figure 15: Jumble Test Case

```
3 public class SumOfArray {
4     public static int sumArray(int[] a1,int[] a2)
5     {
6         int i=0;
7         int sum=0;
8         while(i< a1.length) {
9             sum += a1[i];
10            i++;
11        }
12        int j=0;
13        while(j< a2.length) {
14            sum += a2[j];
15            j++;
16        }
17        return sum;
18    }
19 }
20
```

Problems @ Javadoc Declaration Console

<terminated> Run Jumble [Java Application] D:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_15.0.1.v20201027-0

Mutating soft\_testing\_jumble\_assignment.SumOfArray

Tests: soft\_testing\_jumble\_assignment.SumOfArrayTest

Mutation points = 10, unit test time limit 2.0s

.....

Jumbling took 3.895s

Score: 100%

Figure 16: Sum of two arrayCode

9 Write a program to check whether the number is even or odd.

```
import junit.framework.TestCase;

public class OddEvenTest extends TestCase{
    public void test1()
    {
        assertEquals("Odd", OddEven.checkOddEven(101));
    }
    public void test2()
    {
        assertEquals("Even", OddEven.checkOddEven(150));
    }
    public void test3()
    {
        assertEquals("Odd", OddEven.checkOddEven(-5));
    }
}
```

Figure 17: Jumble Test Case



```
3 public class OddEven {
4     public static String checkOddEven(int num) {
5         if((num % 2) != 0)
6             return "Odd";
7         else
8             return "Even";
9     }
10 }
11
```

The screenshot shows the Eclipse IDE with the 'Console' tab active. It displays the output of a Jumble test run. The test class is 'soft\_testing\_jumble\_assignment.OddEvenTest'. It reports 7 mutation points and a unit test time limit of 2.0s. The test results show a failure at line 5 of OddEven.java, where the value 2 was expected but 3 was received. The Jumble tool took 2.794s to complete the run and achieved a score of 85%.

<terminated> Run Jumble [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.wir  
Mutating soft\_testing\_jumble\_assignment.OddEven  
Tests: soft\_testing\_jumble\_assignment.OddEvenTest  
Mutation points = 7, unit test time limit 2.0s  
..M FAIL: (soft\_testing\_jumble\_assignment.OddEven.java:5): 2 -> 3  
....  
Jumbling took 2.794s  
Score: 85%

Figure 18: Odd or Even Code

10 Write a program for binary to hexadecimal conversion.

```
import junit.framework.TestCase;

public class BinToHexTest extends TestCase{
    public void test1()
    {
        long b = 10011110;
        String h = "9E";
        assertEquals(h, BinToHex.binToHex(b));
    }
    public void test2()
    {
        long b = 1100011100;
        String h = "31C";
        assertEquals(h, BinToHex.binToHex(b));
    }
    public void test3()
    {
        long b = 1010;
        String h = "A";
        assertEquals(h, BinToHex.binToHex(b));
    }
}
```

Figure 19: Jumble Test Case

```

3 public class BinToHex {
4     public static int binaryToDecimal(long binary)
5     {
6
7         int dec = 0, i = 0;
8         while (binary > 0) {
9             dec += Math.pow(2, i++) * (binary % 10);
10            binary /= 10;
11        }
12        return dec;
13    }
14    public static String binToHex(long bin){
15        int decimalNumber = binaryToDecimal(bin);
16        String hexNumber = Integer.toHexString(decimalNumber);
17        hexNumber = hexNumber.toUpperCase();
18        return hexNumber;
19    }
20 }
21 }
22

```

Problems @ Javadoc Declaration Console

<terminated> Run Jumble [Java Application] D:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotsp  
 Mutating soft\_testing\_jumble\_assignment.BinToHex  
 Tests: soft\_testing\_jumble\_assignment.BinToHexTest  
 Mutation points = 13, unit test time limit 2.0s  
 .....  
 Jumbling took 4.818s  
 Score: 100%

**Figure 20: Binary to Hexadecimal Code**