Software Testing Laboratory
(CS6474)
Assignment 06 :Jumble TOOL

Tapas Manna
223CS3152
Master of Technology
223cs3152@nitrkl.ac.in

Department of Computer Science Engineering
NIT, Rourkela

February 19, 2024

Contents

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.

1.1 Java Code

```java
package jumble;



public class Factorial {
    public static int factorial(int n) {

        if (n <=0) {
            return 1;
        }
        return n * factorial(n - 1);
    }
}
```

```java
package jumble;


import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FactorialTest {
    @Test
    public void testFactorialOfFive() {
        assertEquals(120, Factorial.factorial(5));
    }

    @Test
    public void testFactorialOfThree() {
        assertEquals(6, Factorial.factorial(3));
    }

    @Test
    public void testFactorialOfEight() {
        assertEquals(40320, Factorial.factorial(8));
    }
    @Test
    public void testFactorialOfFifteen() {
        assertEquals(1307674368000L, Factorial.factorial(15));
    }

}
```

1.2 Jumble screenshot

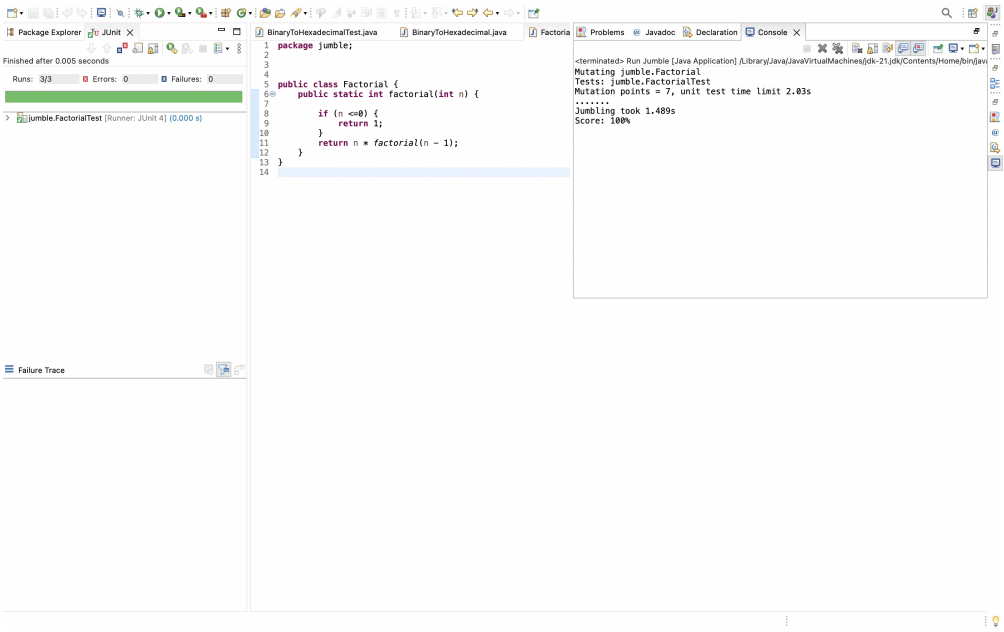

Figure 1: Jumble Test Case

2 Write a program to generate Fibonacci numbers.

2.1 Java Code

```java
package jumble;

public class fibonacci {
    public static int calfibonacci(int n) {
        if (n <= 1)
            return n;
        else
            return calfibonacci(n - 1) + calfibonacci(n - 2);
    }
}
```

```java
package jumble;

import junit.framework.TestCase;

public class fibonacciTest extends TestCase {

  public void testFibonacci() {
    assertEquals(0, fibonacci.calfibonacci(0));
    assertEquals(1, fibonacci.calfibonacci(1));
    assertEquals(1, fibonacci.calfibonacci(2));
    assertEquals(2, fibonacci.calfibonacci(3));
    assertEquals(3, fibonacci.calfibonacci(4));
    assertEquals(5, fibonacci.calfibonacci(5));
    assertEquals(8, fibonacci.calfibonacci(6));
    assertEquals(13, fibonacci.calfibonacci(7));
    assertEquals(21, fibonacci.calfibonacci(8));
    assertEquals(34, fibonacci.calfibonacci(9));
  }
}
```

2.2 Jumble screenshot

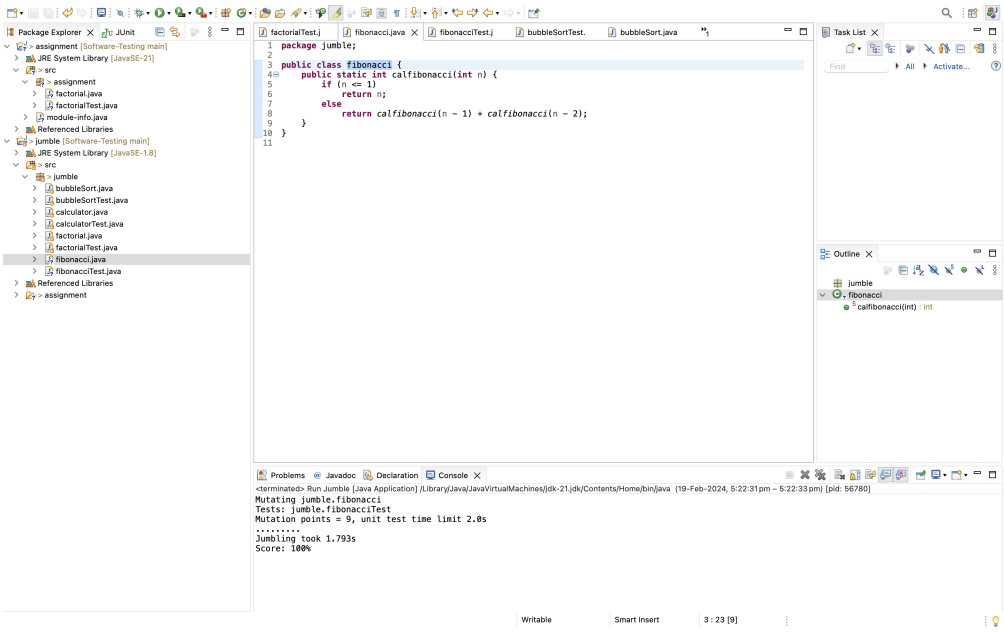

Figure 2: Jumble Test Case

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

3.1 Java Code

```java
package jumble;

public class QuickSort {
    public static void quickSort(int[] arr) {
        if (arr == null || arr.length == 0)
            return;
        sort(arr, 0, arr.length - 1);
    }

    private static void sort(int[] arr, int low, int high) {
        if (low < high) {
            int pivotIndex = partition(arr, low, high);
            sort(arr, low, pivotIndex - 1);
            sort(arr, pivotIndex + 1, high);
        }
    }

    private static int partition(int[] arr, int low, int high) {
        int pivot = arr[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (arr[j] < pivot) {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;
        return i + 1;
    }
}
```

```java
package jumble;

import junit.framework.TestCase;
import java.util.Arrays;

public class QuickSortTest extends TestCase {

  public void testQuickSort() {
        int[] arr1 = {3, 7, 2, 1, 6, 5, 4};
        int[] arr2 = {9, 1, 5, 4, 8, 2, 7, 3, 6};
        int[] arr3 = {5, 4, 3, 2, 1};

        int[] sortedArr1 = {1, 2, 3, 4, 5, 6, 7};
        int[] sortedArr2 = {1, 2, 3, 4, 5, 6, 7, 8, 9};
        int[] sortedArr3 = {1, 2, 3, 4, 5};

        QuickSort.quickSort(arr1);
        QuickSort.quickSort(arr2);
```

```
        QuickSort.quickSort(arr3);

        assertTrue(Arrays.equals(sortedArr1, arr1));
        assertTrue(Arrays.equals(sortedArr2, arr2));
        assertTrue(Arrays.equals(sortedArr3, arr3));
    }

    public void testQuickSortEmptyArray() {
        int[] arr = {};
        int[] sortedArr = {};

        QuickSort.quickSort(arr);

        assertTrue(Arrays.equals(sortedArr, arr));
    }

    public void testQuickSortAlreadySortedArray() {
        int[] arr = {1, 2, 3, 4, 5};
        int[] sortedArr = {1, 2, 3, 4, 5};

        QuickSort.quickSort(arr);

        assertTrue(Arrays.equals(sortedArr, arr));
    }

    public void testQuickSortSingleElementArray() {
        int[] arr = {5};
        int[] sortedArr = {5};

        QuickSort.quickSort(arr);

        assertTrue(Arrays.equals(sortedArr, arr));
    }


}
```

3.2 Jumble screenshot

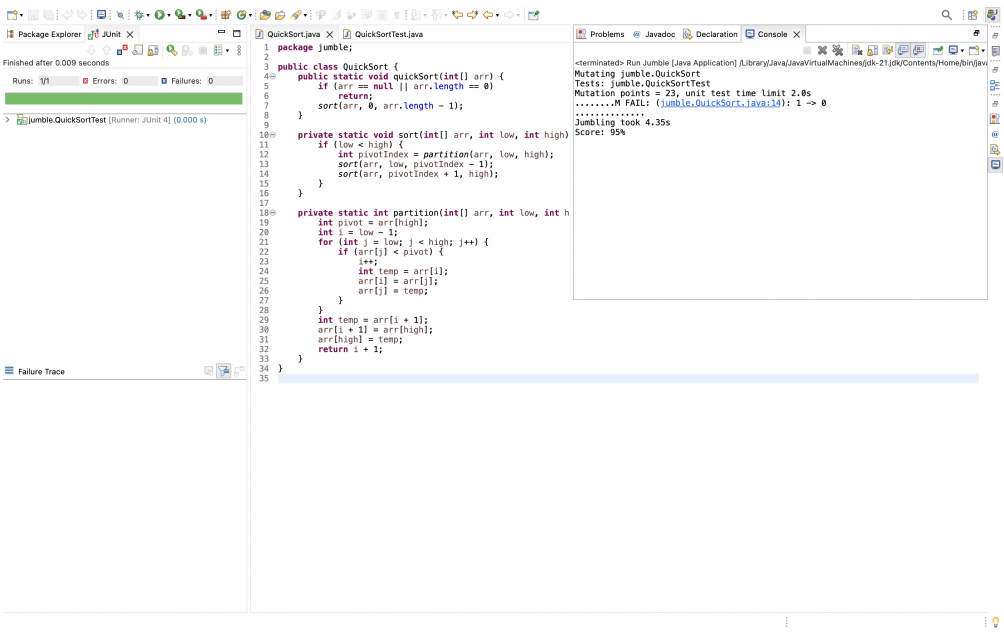

Figure 3: Jumble Test Case

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

4.1 Java Code

```java
package jumble;

public class MatrixTranspose {
    public static int[][] transpose(int[][] matrix) {
        if (matrix == null || matrix.length == 0)
            return null;

        int rows = matrix.length;
        int cols = matrix[0].length;

        int[][] result = new int[cols][rows];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[j][i] = matrix[i][j];
            }
        }

        return result;
    }
}
```

```java
package jumble;

import junit.framework.TestCase;
import java.util.Arrays;

```

```
public class MatrixTransposeTest extends TestCase {

    public void testTranspose() {
         // Test case 1
        int[][] matrix1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        int[][] expectedTranspose1 = {{1, 4, 7}, {2, 5, 8}, {3, 6, 9}};
        assertTrue(Arrays.deepEquals(expectedTranspose1, MatrixTranspose.transpose
    (matrix1)));

        // Test case 2
        int[][] matrix2 = {{1, 2}, {3, 4}, {5, 6}};
        int[][] expectedTranspose2 = {{1, 3, 5}, {2, 4, 6}};
        assertTrue(Arrays.deepEquals(expectedTranspose2, MatrixTranspose.transpose
    (matrix2)));

        // Test case 3
        int[][] matrix3 = {{1}};
        int[][] expectedTranspose3 = {{1}};
        assertTrue(Arrays.deepEquals(expectedTranspose3, MatrixTranspose.transpose
    (matrix3)));

        // Test case 4
        int[][] matrix4 = {{1, 2}, {3, 4}};
        int[][] expectedTranspose4 = {{1, 3}, {2, 4}};
        assertTrue(Arrays.deepEquals(expectedTranspose4, MatrixTranspose.transpose
    (matrix4)));

        // Test case 5
        int[][] matrix5 = {{1, 2, 3, 4}, {5, 6, 7, 8}};
        int[][] expectedTranspose5 = {{1, 5}, {2, 6}, {3, 7}, {4, 8}};
        assertTrue(Arrays.deepEquals(expectedTranspose5, MatrixTranspose.transpose
    (matrix5)));

    }
}
```

4.2 Jumble screenshot

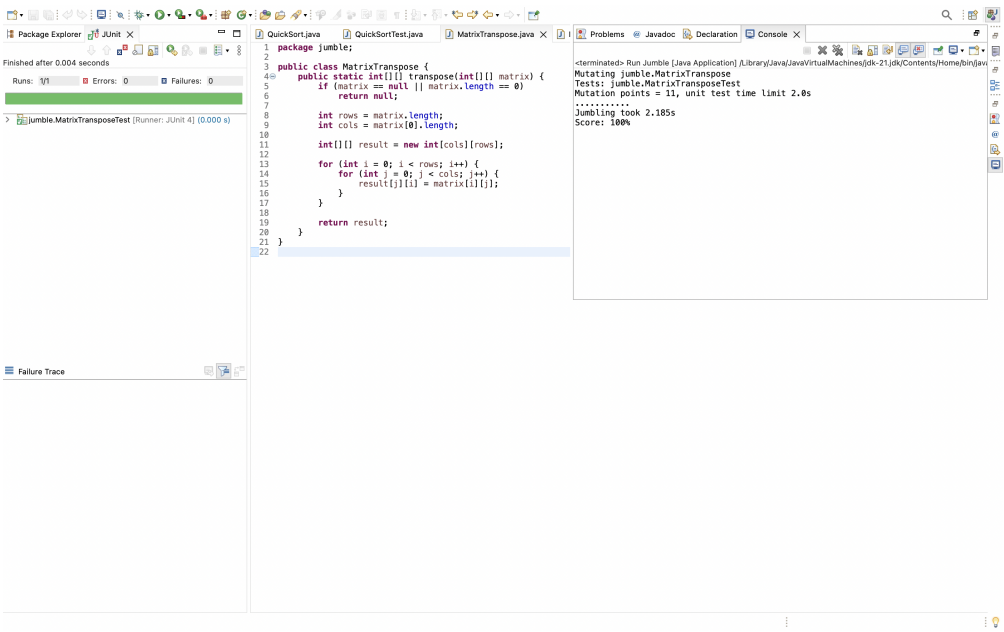

Figure 4: Jumble Test Case

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

5.1 Java Code

```java
package jumble;

public class MatrixAddition {
    public static int[][] addMatrices(int[][] matrix1, int[][] matrix2) {
        if (matrix1 == null || matrix2 == null ||
            matrix1.length == 0 || matrix2.length == 0 ||
            matrix1.length != matrix2.length ||
            matrix1[0].length != matrix2[0].length) {
            return null; // Matrices cannot be added
        }

        int rows = matrix1.length;
        int cols = matrix1[0].length;

        int[][] result = new int[rows][cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = matrix1[i][j] + matrix2[i][j];
            }
        }

        return result;
    }
}
```

```java
package jumble;
```

```
2
3  import junit.framework.TestCase;
4  import java.util.Arrays;
5
6  public class MatrixAdditionTest extends TestCase {
7
8
9
10     public void testAddMatrices() {
11         int[][] matrix1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
12         int[][] matrix2 = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
13         int[][] expectedSum = {{10, 10, 10}, {10, 10, 10}, {10, 10, 10}};
14         int[][] result = MatrixAddition.addMatrices(matrix1, matrix2);
15         assertTrue(Arrays.deepEquals(expectedSum, result));
16     }
17
18     public void testAddMatricesWithZeros() {
19         int[][] matrix1 = {{0, 0}, {0, 0}};
20         int[][] matrix2 = {{0, 0}, {0, 0}};
21         int[][] expectedSum = {{0, 0}, {0, 0}};
22         int[][] result = MatrixAddition.addMatrices(matrix1, matrix2);
23         assertTrue(Arrays.deepEquals(expectedSum, result));
24     }
25
26     public void testAddMatricesNegativeNumbers() {
27         int[][] matrix1 = {{1, -2}, {-3, 4}};
28         int[][] matrix2 = {{-5, 6}, {7, -8}};
29         int[][] expectedSum = {{-4, 4}, {4, -4}};
30         int[][] result = MatrixAddition.addMatrices(matrix1, matrix2);
31         assertTrue(Arrays.deepEquals(expectedSum, result));
32     }
33     public void testAddMatrices2() {
34         int[][] matrix1 = {{1, 2}, {3, 4}};
35         int[][] matrix2 = {{5, 6}, {7, 8}};
36         int[][] expectedSum = {{6, 8}, {10, 12}};
37         int[][] result = MatrixAddition.addMatrices(matrix1, matrix2);
38         assertTrue(Arrays.deepEquals(expectedSum, result));
39     }
40     public void testAddMatrices3() {
41         int[][] matrix1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
42         int[][] matrix2 = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
43         int[][] expectedSum = {{10, 10, 10}, {10, 10, 10}, {10, 10, 10}};
44         int[][] result = MatrixAddition.addMatrices(matrix1, matrix2);
45         assertTrue(Arrays.deepEquals(expectedSum, result));
46     }
47
48
49 }
```

5.2 Jumble screenshot

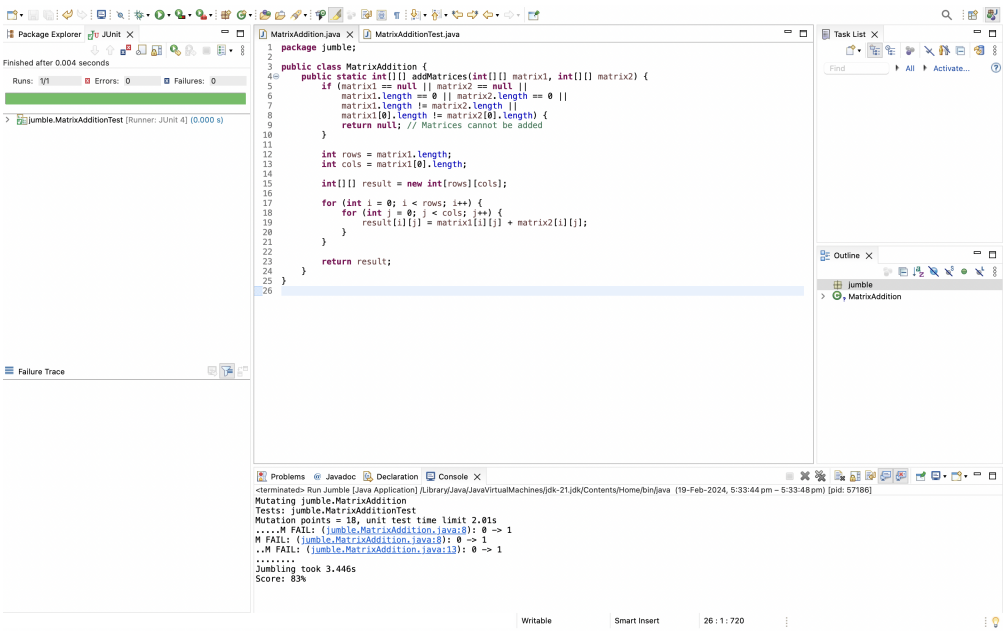

Figure 5: Jumble Test Case

6 Write a program to Print Prime Numbers from 1 to 100 using Scanner Class and For Loop.

6.1 Java Code

```java
package jumble;

import java.util.ArrayList;
import java.util.List;

public class PrimeNumbers {
    public static List<Integer> getPrimes(int upperLimit) {
        List<Integer> primes = new ArrayList<>();

        for (int num = 2; num <= upperLimit; num++) {
            if (isPrime(num)) {
                primes.add(num);
            }
        }

        return primes;
    }

    private static boolean isPrime(int num) {
        if (num <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) {
                return false;
            }
        }
```

```
        return true;
    }
}
```

```
package jumble;

import junit.framework.TestCase;
import java.util.Arrays;
import java.util.List;

public class PrimeNumbersTest extends TestCase {

    public void testGetPrimes() {
        List<Integer> expectedPrimes = Arrays.asList(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);
        List<Integer> actualPrimes = PrimeNumbers.getPrimes(100);
        assertEquals(expectedPrimes, actualPrimes);
    }
}
```

6.2 Jumble screenshot

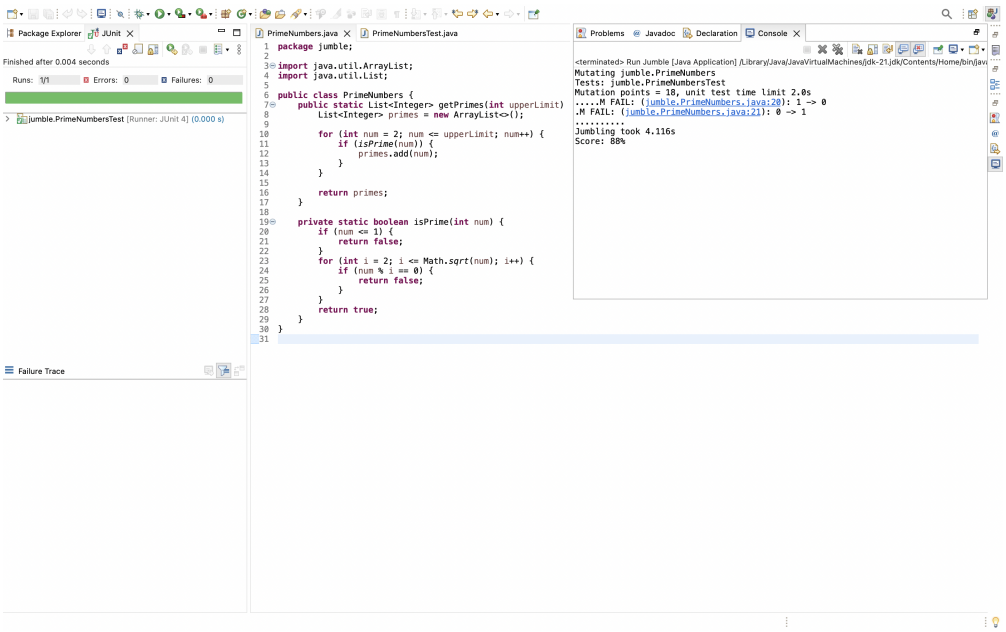

Figure 6: Jumble Test Case

7 Write a program to generate a palindrome of numbers.

7.1 Java Code

```
1
2 package jumble;
3
4 public class PalindromeGenerator {
5     public static int generatePalindrome(int number) {
6
7
8         int originalNumber = number;
9         int reversedNumber = 0;
10
11        while (number != 0) {
12            int remainder = number % 10;
13            reversedNumber = reversedNumber * 10 + remainder;
14            number = number / 10;
15        }
16
17        // Append the reversedNumber to the original number
18        return Integer.parseInt(Integer.toString(originalNumber) + Integer.
      toString(reversedNumber));
19    }
20 }
```

```
1 package jumble;
2
3
4 import org.junit.Test;
5 import static org.junit.Assert.*;
6
7 public class PalindromeGeneratorTest {
8
9     @Test
10    public void testGeneratePalindrome() {
11        assertEquals(1221, PalindromeGenerator.generatePalindrome(12));
12        assertEquals(123321, PalindromeGenerator.generatePalindrome(123));
13        assertEquals(12344321, PalindromeGenerator.generatePalindrome(1234));
14        assertEquals(77, PalindromeGenerator.generatePalindrome(7));
15        assertEquals(88, PalindromeGenerator.generatePalindrome(8));
16
17    }
18
19
20 }
```

7.2 Jumble screenshot

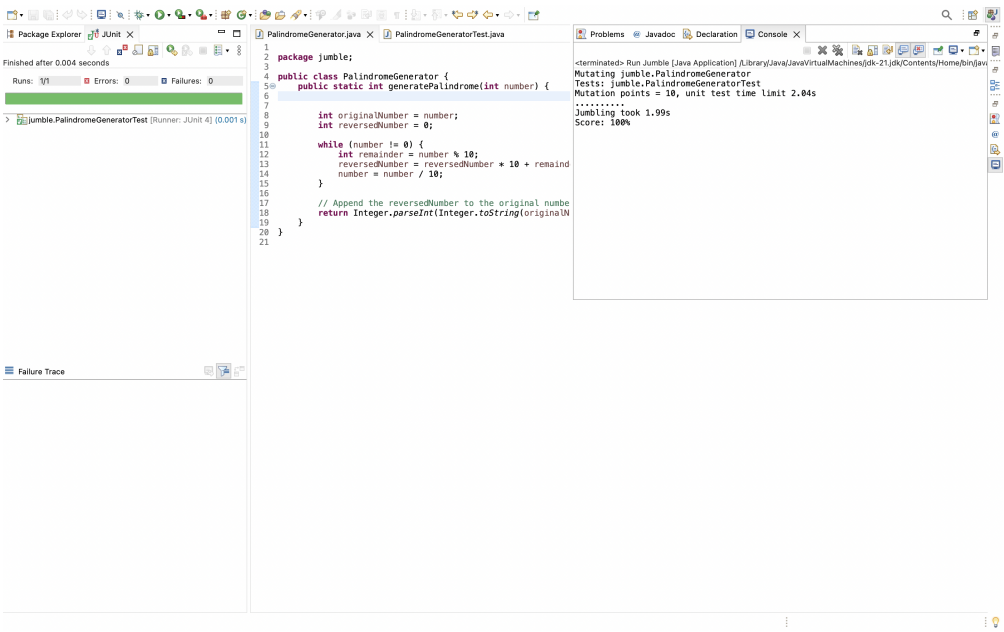

Figure 7: Jumble Test Case

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

8.1 Java Code

```java
package jumble;



public class ArraySum {
    public static int[] sumArrays(int[] array1, int[] array2) {

        int[] sum = new int[array1.length];
        for (int i = 0; i < array1.length; i++) {
            sum[i] = array1[i] + array2[i];
        }
        return sum;
    }
}
```

```java
package jumble;


import org.junit.Test;
import static org.junit.Assert.assertArrayEquals;

public class ArraySumTest {
    @Test
      public void testSumArrays() {
          int[] array1 = {1, 2, 3};
          int[] array2 = {4, 5, 6};
          int[] expected = {5, 7, 9};
          assertArrayEquals(expected, ArraySum.sumArrays(array1, array2));
      }
```

```
15
16      @Test
17      public void testSumArraysEmpty() {
18          int[] array1 = {};
19          int[] array2 = {};
20          int[] expected = {};
21          assertArrayEquals(expected, ArraySum.sumArrays(array1, array2));
22      }
23
24      @Test
25      public void testSumArraysDifferentLength() {
26         int[] array1 = {1, 5, 3};
27            int[] array2 = {0, 0, 0};
28            int[] expected = {1, 5, 3};
29            assertArrayEquals(expected, ArraySum.sumArrays(array1, array2));
30      }
31
32      @Test
33      public void testSumArraysNegativeNumbers() {
34          int[] array1 = {-1, -2, -3};
35          int[] array2 = {-4, -5, -6};
36          int[] expected = {-5, -7, -9};
37          assertArrayEquals(expected, ArraySum.sumArrays(array1, array2));
38      }
39
40      @Test
41      public void testSumArraysLargeNumbers() {
42          int[] array1 = {Integer.MAX_VALUE, Integer.MAX_VALUE};
43          int[] array2 = {1, 1};
44          long sum1 = (long) Integer.MAX_VALUE + 1;
45          long sum2 = (long) Integer.MAX_VALUE + 1;
46          int[] expected = {(int) sum1, (int) sum2};
47          assertArrayEquals(expected, ArraySum.sumArrays(array1, array2));
48      }
49
50
51  }
```

8.2 Jumble screenshot

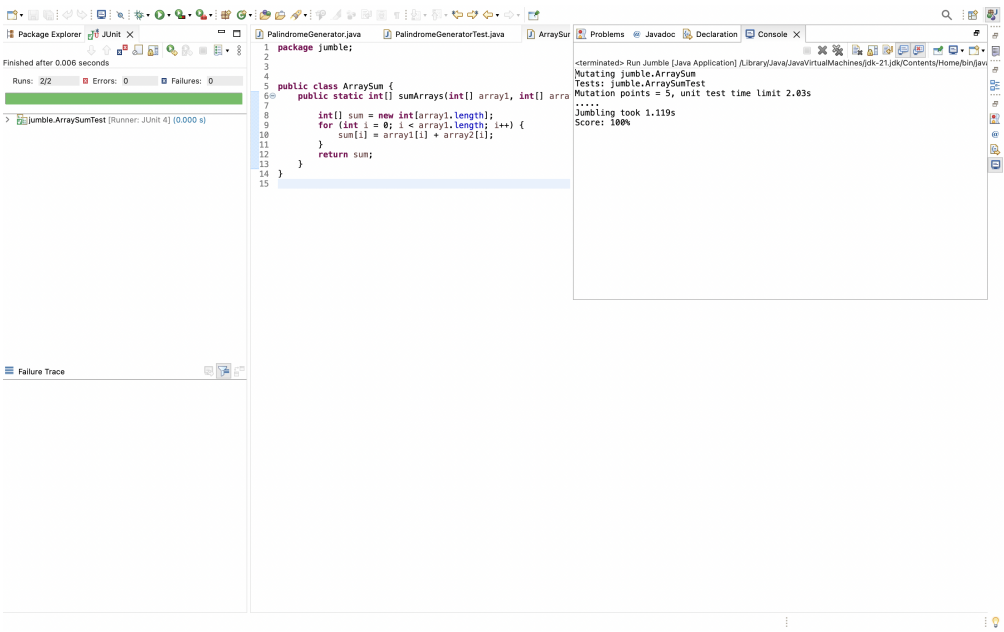

Figure 8: Jumble Test Case

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

9.1 Java Code

```java
package jumble;



public class EvenOrOdd {
    public static String checkEvenOrOdd(int number) {
        return (number % 2 == 0) ? "Even" : "Odd";
    }
}
```

```java
package jumble;



import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class EvenOrOddTest {
    @Test
    public void testCheckEvenOrOdd() {
        assertEquals("Even", EvenOrOdd.checkEvenOrOdd(4));
        assertEquals("Odd", EvenOrOdd.checkEvenOrOdd(7));
        assertEquals("Even", EvenOrOdd.checkEvenOrOdd(0));
        assertEquals("Even", EvenOrOdd.checkEvenOrOdd(-4));
        assertEquals("Odd", EvenOrOdd.checkEvenOrOdd(-7));
    }
}
```

9.2 Jumble screenshot

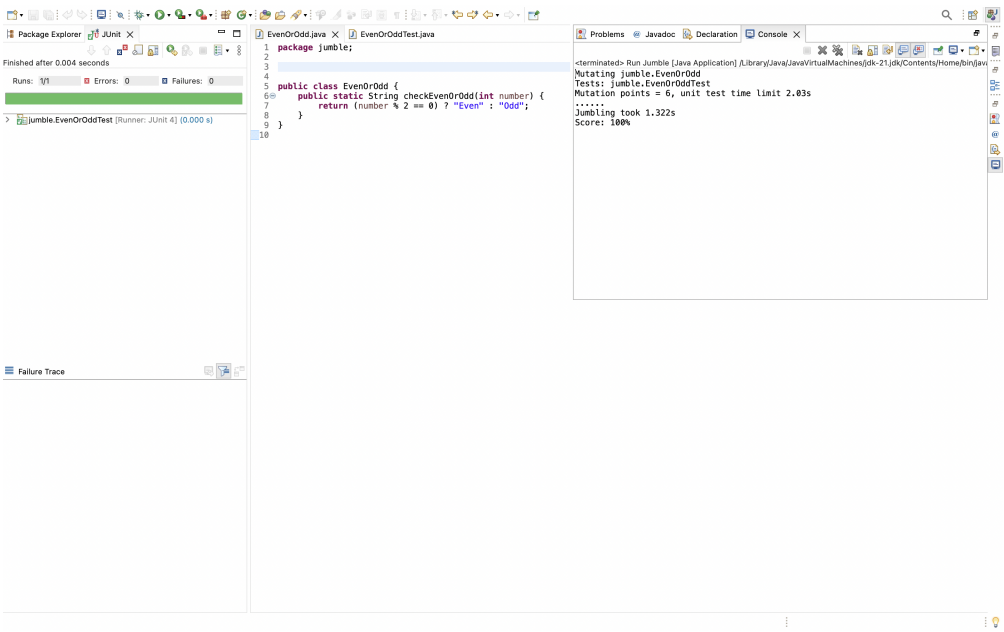

Figure 9: Jumble Test Case

10 Write a program for binary to hexadecimal conversion.

10.1 Java Code

```java
package jumble;



public class BinaryToHexadecimal {
    public static String convertBinaryToHex(String binary) {
        int decimal = Integer.parseInt(binary, 2);
        return Integer.toHexString(decimal).toUpperCase();
    }
}
```

```java
package jumble;



import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class BinaryToHexadecimalTest {
    @Test
    public void testConvertBinaryToHex() {
        assertEquals("A", BinaryToHexadecimal.convertBinaryToHex("1010"));
        assertEquals("1F", BinaryToHexadecimal.convertBinaryToHex("11111"));
        assertEquals("0", BinaryToHexadecimal.convertBinaryToHex("0"));
        assertEquals("1", BinaryToHexadecimal.convertBinaryToHex("1"));
        assertEquals("F", BinaryToHexadecimal.convertBinaryToHex("1111"));
    }
}
```

10.2 Jumble screenshot

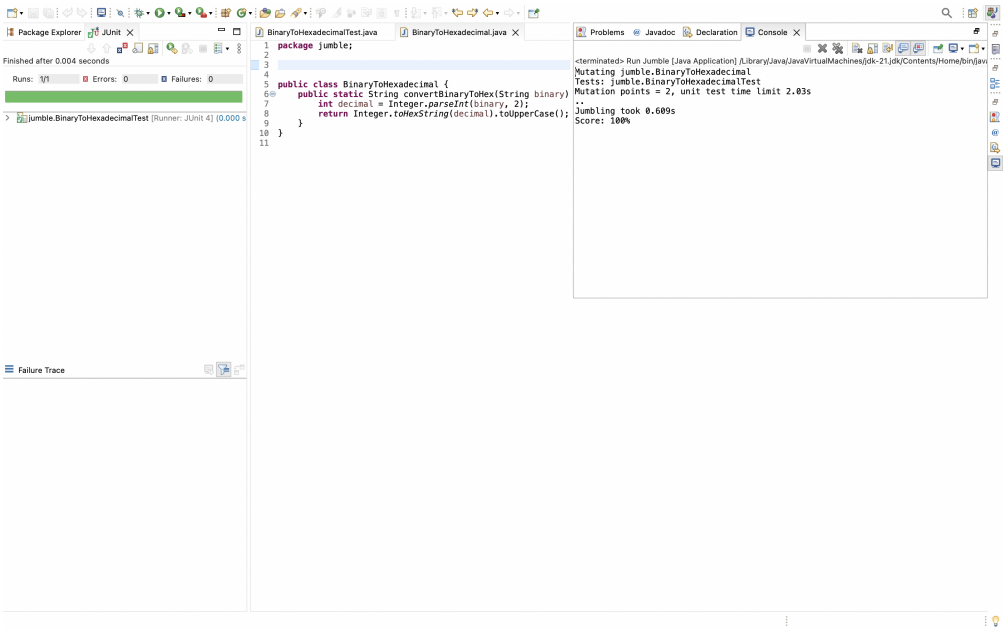

Figure 10: Jumble Test Case