



---

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

<b>1</b>	<b>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.</b>	<b>3</b>
1.1	Java Code . . . . .	3
1.2	Jumble screenshot . . . . .	4
<b>2</b>	<b>Write a program to generate Fibonacci numbers.</b>	<b>5</b>
2.1	Java Code . . . . .	5
2.2	Jumble screenshot . . . . .	6
<b>3</b>	<b>Write a program that performs sorting of a group of integer values using the quick sort technique.</b>	<b>7</b>
3.1	Java Code . . . . .	7
3.2	Jumble screenshot . . . . .	8
<b>4</b>	<b>Write a program that accepts elements of a matrix and displays its transpose.</b>	<b>9</b>
4.1	Java Code . . . . .	9
4.2	Jumble screenshot . . . . .	9
<b>5</b>	<b>Write a program to add two matrices and display the sum matrix.</b>	<b>10</b>
5.1	Java Code . . . . .	10
5.2	Jumble screenshot . . . . .	11
<b>6</b>	<b>Write a program to Print Prime Numbers from 1 to 100 using Scanner Class and For Loop.</b>	<b>12</b>
6.1	Java Code . . . . .	12
6.2	Jumble screenshot . . . . .	12
<b>7</b>	<b>Write a program to generate a palindrome of numbers.</b>	<b>13</b>
7.1	Java Code . . . . .	13
7.2	Jumble screenshot . . . . .	14
<b>8</b>	<b>Write a program to find out the sum of two arrays.</b>	<b>15</b>
8.1	Java Code . . . . .	15
8.2	Jumble screenshot . . . . .	15
<b>9</b>	<b>Write a program to check whether the number is even or odd.</b>	<b>16</b>
9.1	Java Code . . . . .	16
9.2	Jumble screenshot . . . . .	16
<b>10</b>	<b>Write a program for binary to hexadecimal conversion.</b>	<b>17</b>
10.1	Java Code . . . . .	17
10.2	Jumble screenshot . . . . .	17

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

```
1 package jumble;
2
3
4
5 public class Factorial {
6     public static int factorial(int n) {
7
8         if (n <=0) {
9             return 1;
10        }
11        return n * factorial(n - 1);
12    }
13 }
```

```
1 package jumble;
2
3
4 import org.junit.Test;
5 import static org.junit.Assert.assertEquals;
6
7 public class FactorialTest {
8     @Test
9     public void testFactorialOfFive() {
10         assertEquals(120, Factorial.factorial(5));
11     }
12
13     @Test
14     public void testFactorialOfThree() {
15         assertEquals(6, Factorial.factorial(3));
16     }
17
18     @Test
19     public void testFactorialOfEight() {
20         assertEquals(40320, Factorial.factorial(8));
21     }
22 }
23 }
```

## 1.2 Jumble screenshot

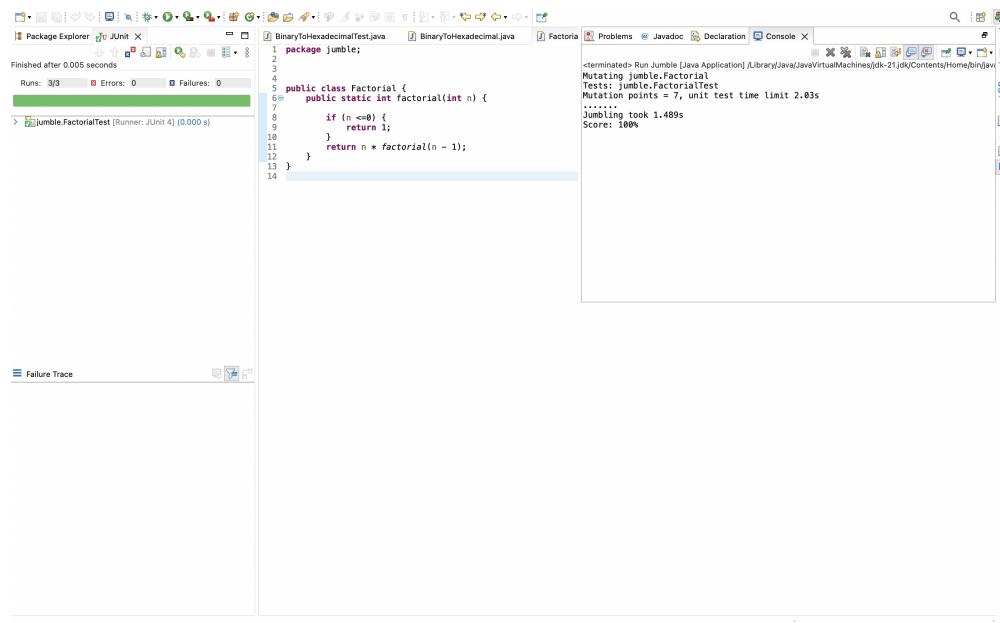


Figure 1: Jumble Test Case

## 2 Write a program to generate Fibonacci numbers.

### 2.1 Java Code

```
1 package jumble;
2
3 public class fibonacci {
4     public static int calfibonacci(int n) {
5         if (n <= 1)
6             return n;
7         else
8             return calfibonacci(n - 1) + calfibonacci(n - 2);
9     }
10 }
```

```
1 package jumble;
2
3 import junit.framework.TestCase;
4
5 public class fibonacciTest extends TestCase {
6
7     public void testFibonacci() {
8         assertEquals(0, fibonacci.calfibonacci(0));
9         assertEquals(1, fibonacci.calfibonacci(1));
10        assertEquals(1, fibonacci.calfibonacci(2));
11        assertEquals(2, fibonacci.calfibonacci(3));
12        assertEquals(3, fibonacci.calfibonacci(4));
13        assertEquals(5, fibonacci.calfibonacci(5));
14        assertEquals(8, fibonacci.calfibonacci(6));
15        assertEquals(13, fibonacci.calfibonacci(7));
16        assertEquals(21, fibonacci.calfibonacci(8));
17        assertEquals(34, fibonacci.calfibonacci(9));
18    }
19 }
```

## 2.2 Jumble screenshot

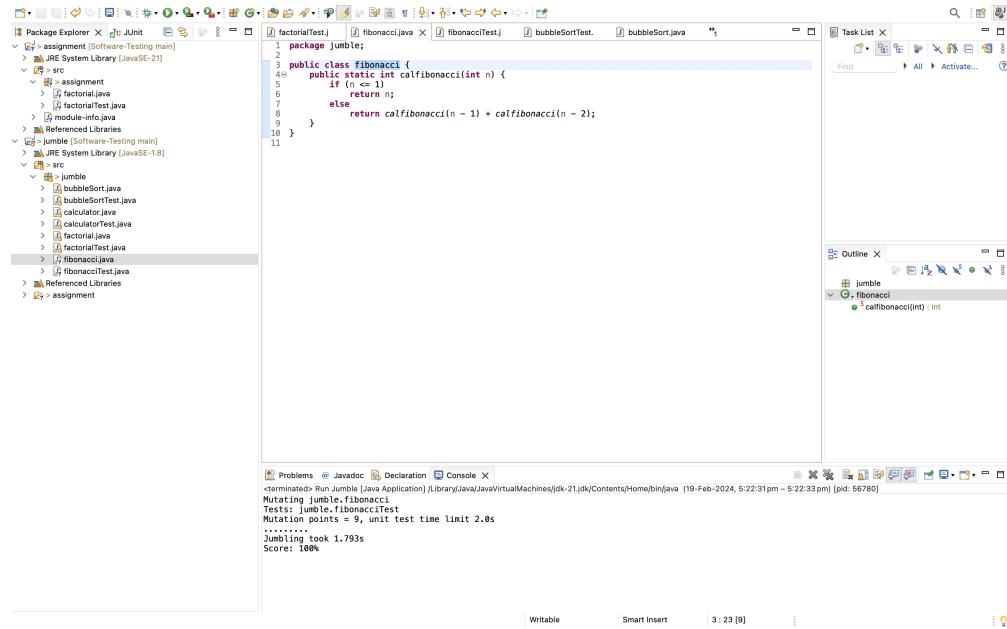


Figure 2: Jumble Test Case

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

#### 3.1 Java Code

```
1 package jumble;
2
3 public class QuickSort {
4     public static void quickSort(int[] arr) {
5         if (arr == null || arr.length == 0)
6             return;
7         sort(arr, 0, arr.length - 1);
8     }
9
10    private static void sort(int[] arr, int low, int high) {
11        if (low < high) {
12            int pivotIndex = partition(arr, low, high);
13            sort(arr, low, pivotIndex - 1);
14            sort(arr, pivotIndex + 1, high);
15        }
16    }
17
18    private static int partition(int[] arr, int low, int high) {
19        int pivot = arr[high];
20        int i = low - 1;
21        for (int j = low; j < high; j++) {
22            if (arr[j] < pivot) {
23                i++;
24                int temp = arr[i];
25                arr[i] = arr[j];
26                arr[j] = temp;
27            }
28        }
29        int temp = arr[i + 1];
30        arr[i + 1] = arr[high];
31        arr[high] = temp;
32        return i + 1;
33    }
34 }
```

```
1 package jumble;
2
3 import junit.framework.TestCase;
4 import java.util.Arrays;
5
6 public class QuickSortTest extends TestCase {
7
8     public void testQuickSort() {
9         int[] arr1 = {3, 7, 2, 1, 6, 5, 4};
10        int[] arr2 = {9, 1, 5, 4, 8, 2, 7, 3, 6};
11        int[] arr3 = {5, 4, 3, 2, 1};
12
13        int[] sortedArr1 = {1, 2, 3, 4, 5, 6, 7};
14        int[] sortedArr2 = {1, 2, 3, 4, 5, 6, 7, 8, 9};
15        int[] sortedArr3 = {1, 2, 3, 4, 5};
16
17        QuickSort.quickSort(arr1);
18        QuickSort.quickSort(arr2);
```

```

19     QuickSort.quickSort(arr3);
20
21     assertTrue(Arrays.equals(sortedArr1, arr1));
22     assertTrue(Arrays.equals(sortedArr2, arr2));
23     assertTrue(Arrays.equals(sortedArr3, arr3));
24 }
25 }
```

### 3.2 Jumble screenshot

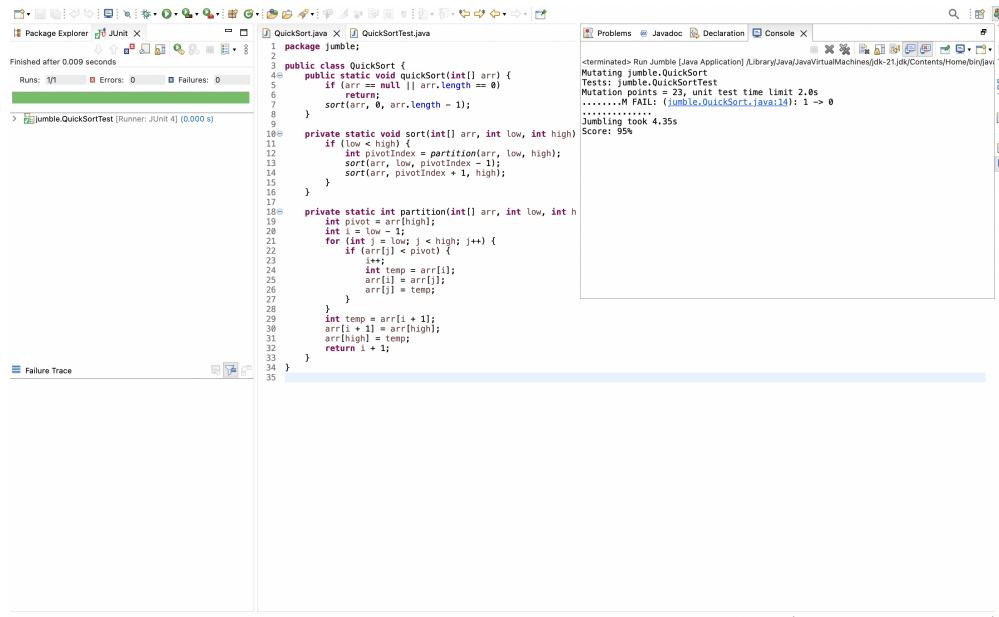


Figure 3: Jumble Test Case

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

### 4.1 Java Code

```
1 package jumble;
2
3 public class MatrixTranspose {
4     public static int[][] transpose(int[][] matrix) {
5         if (matrix == null || matrix.length == 0)
6             return null;
7
8         int rows = matrix.length;
9         int cols = matrix[0].length;
10
11        int[][] result = new int[cols][rows];
12
13        for (int i = 0; i < rows; i++) {
14            for (int j = 0; j < cols; j++) {
15                result[j][i] = matrix[i][j];
16            }
17        }
18
19        return result;
20    }
21 }
```

```
1 package jumble;
2
3 import junit.framework.TestCase;
4 import java.util.Arrays;
5
6 public class MatrixTransposeTest extends TestCase {
7
8     public void testTranspose() {
9         int[][] matrix1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
10        int[][] expectedTranspose1 = {{1, 4, 7}, {2, 5, 8}, {3, 6, 9}};
11
12        int[][] matrix2 = {{1, 2}, {3, 4}, {5, 6}};
13        int[][] expectedTranspose2 = {{1, 3, 5}, {2, 4, 6}};
14
15        int[][] matrix3 = {{1}};
16        int[][] expectedTranspose3 = {{1}};
17
18        assertTrue(Arrays.deepEquals(expectedTranspose1, MatrixTranspose.transpose(matrix1)));
19        assertTrue(Arrays.deepEquals(expectedTranspose2, MatrixTranspose.transpose(matrix2)));
20        assertTrue(Arrays.deepEquals(expectedTranspose3, MatrixTranspose.transpose(matrix3)));
21    }
22 }
```

### 4.2 Jumble screenshot

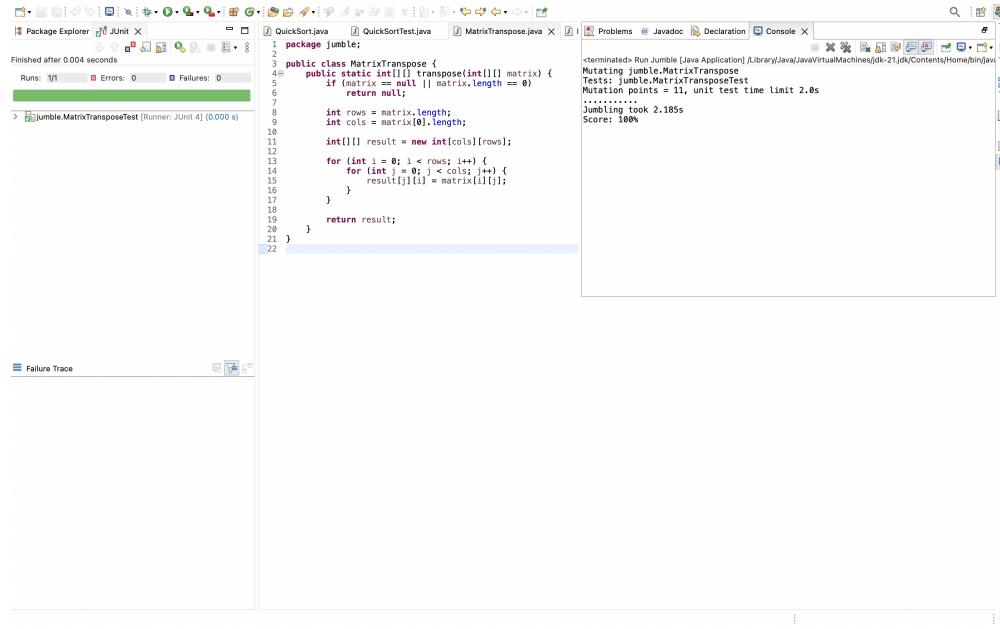


Figure 4: Jumble Test Case

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

### 5.1 Java Code

```

1 package jumble;
2
3 public class MatrixAddition {
4     public static int[][] addMatrices(int[][] matrix1, int[][] matrix2) {
5         if (matrix1 == null || matrix2 == null ||
6             matrix1.length == 0 || matrix2.length == 0 ||
7             matrix1.length != matrix2.length ||
8             matrix1[0].length != matrix2[0].length) {
9                 return null; // Matrices cannot be added
10            }
11
12            int rows = matrix1.length;
13            int cols = matrix1[0].length;
14
15            int[][] result = new int[rows][cols];
16
17            for (int i = 0; i < rows; i++) {
18                for (int j = 0; j < cols; j++) {
19                    result[i][j] = matrix1[i][j] + matrix2[i][j];
20                }
21            }
22
23            return result;
24        }
25    }
  
```

```

1 package jumble;
  
```

```

2
3 import junit.framework.TestCase;
4 import java.util.Arrays;
5
6 public class MatrixAdditionTest extends TestCase {
7
8     public void testAddMatrices() {
9         int[][] matrix1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
10        int[][] matrix2 = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
11        int[][] expectedSum = {{10, 10, 10}, {10, 10, 10}, {10, 10, 10}};
12
13        int[][] result = MatrixAddition.addMatrices(matrix1, matrix2);
14
15        assertTrue(Arrays.deepEquals(expectedSum, result));
16    }
17 }

```

## 5.2 Jumble screenshot

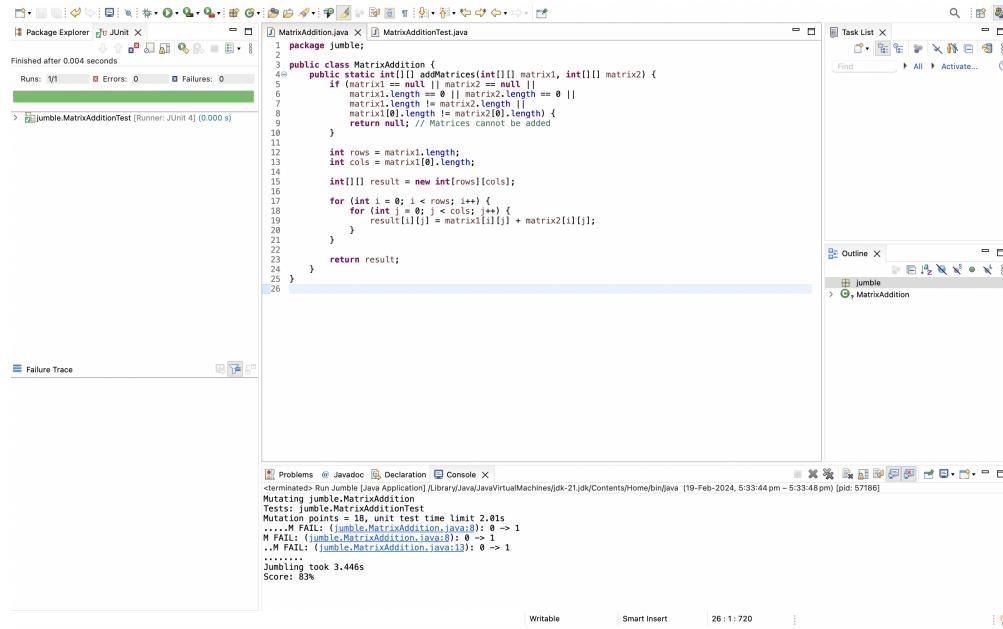


Figure 5: Jumble Test Case

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

### 6.1 Java Code

```
1 package jumble;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class PrimeNumbers {
7     public static List<Integer> getPrimes(int upperLimit) {
8         List<Integer> primes = new ArrayList<>();
9
10        for (int num = 2; num <= upperLimit; num++) {
11            if (isPrime(num)) {
12                primes.add(num);
13            }
14        }
15
16        return primes;
17    }
18
19    private static boolean isPrime(int num) {
20        if (num <= 1) {
21            return false;
22        }
23        for (int i = 2; i <= Math.sqrt(num); i++) {
24            if (num % i == 0) {
25                return false;
26            }
27        }
28        return true;
29    }
30 }
```

```
1 package jumble;
2
3 import junit.framework.TestCase;
4 import java.util.Arrays;
5 import java.util.List;
6
7 public class PrimeNumbersTest extends TestCase {
8
9     public void testGetPrimes() {
10         List<Integer> expectedPrimes = Arrays.asList(2, 3, 5, 7, 11, 13, 17, 19,
11             23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97);
12         List<Integer> actualPrimes = PrimeNumbers.getPrimes(100);
13         assertEquals(expectedPrimes, actualPrimes);
14     }
15 }
```

### 6.2 Jumble screenshot

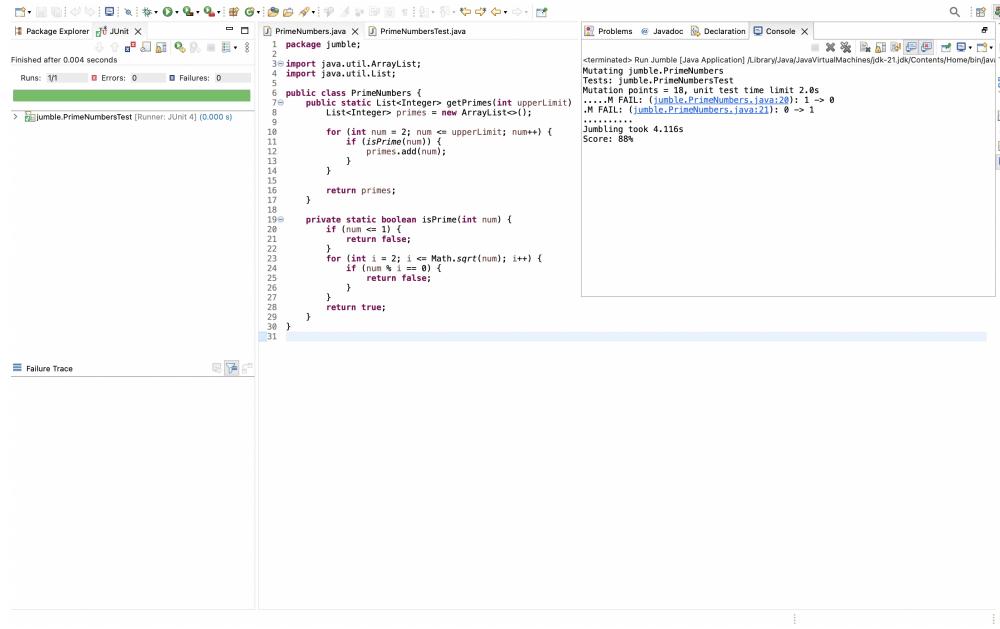


Figure 6: Jumble Test Case

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

### 7.1 Java Code

```

1 package jumble;
2
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.
19         toString(reversedNumber));
20     }
21 }

```

```

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    }
15
16
17 }

```

## 7.2 Jumble screenshot

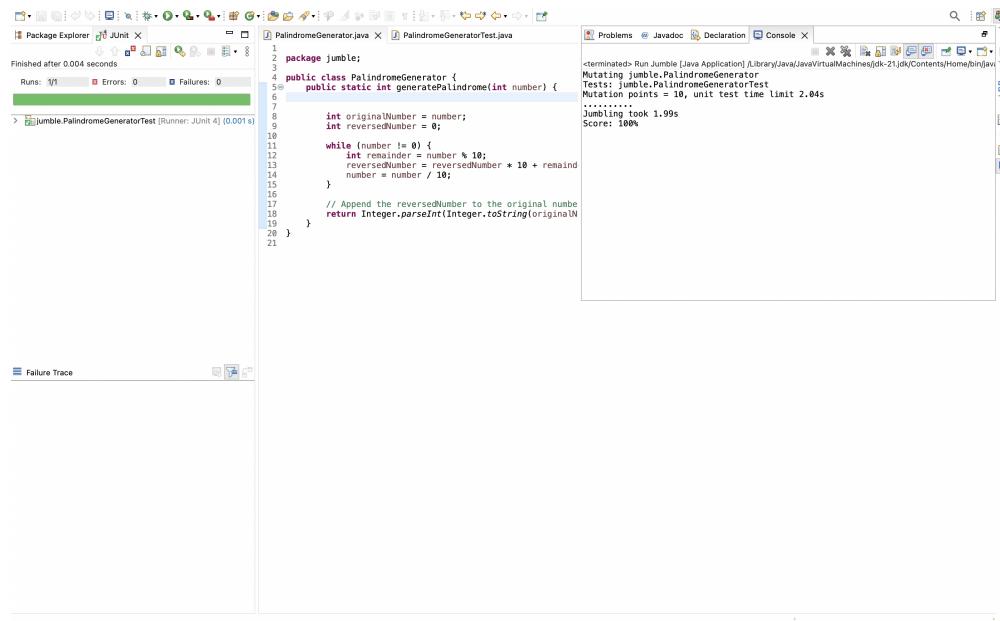


Figure 7: Jumble Test Case

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

### 8.1 Java Code

```
1 package jumble;
2
3
4
5 public class ArraySum {
6     public static int[] sumArrays(int[] array1, int[] array2) {
7
8         int[] sum = new int[array1.length];
9         for (int i = 0; i < array1.length; i++) {
10             sum[i] = array1[i] + array2[i];
11         }
12         return sum;
13     }
14 }
```

```
1 package jumble;
2
3
4 import org.junit.Test;
5 import static org.junit.Assert.assertArrayEquals;
6
7 public class ArraySumTest {
8     @Test
9     public void testSumArrays() {
10         int[] array1 = {1, 2, 3};
11         int[] array2 = {4, 5, 6};
12         int[] expected = {5, 7, 9};
13         assertArrayEquals(expected, ArraySum.sumArrays(array1, array2));
14     }
15
16
17 }
```

### 8.2 Jumble screenshot

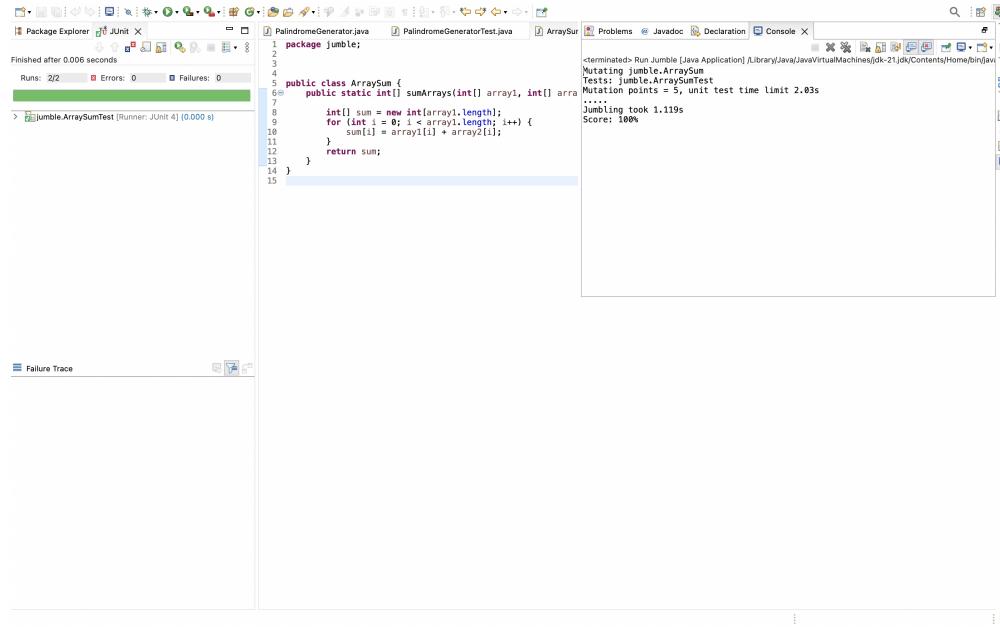


Figure 8: Jumble Test Case

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

### 9.1 Java Code

```

1 package jumble;
2
3
4
5 public class EvenOrOdd {
6     public static String checkEvenOrOdd(int number) {
7         return (number % 2 == 0) ? "Even" : "Odd";
8     }
9 }
```

```

1 package jumble;
2
3
4
5 import org.junit.Test;
6 import static org.junit.Assert.assertEquals;
7
8 public class EvenOrOddTest {
9     @Test
10    public void testCheckEvenOrOdd() {
11        assertEquals("Even", EvenOrOdd.checkEvenOrOdd(4));
12        assertEquals("Odd", EvenOrOdd.checkEvenOrOdd(7));
13    }
14 }
```

### 9.2 Jumble screenshot

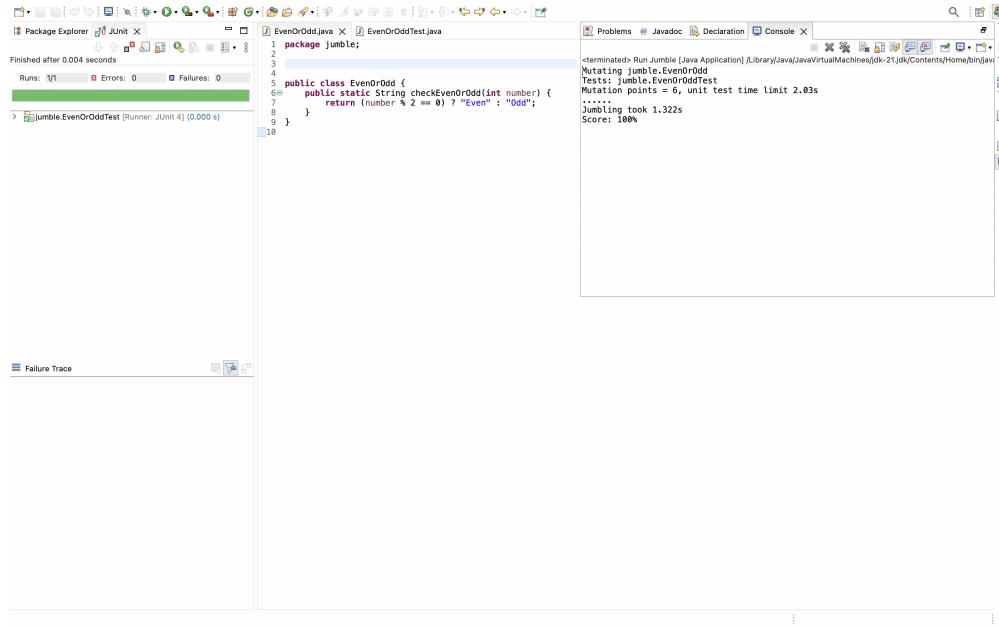


Figure 9: Jumble Test Case

## 10 Write a program for binary to hexadecimal conversion.

### 10.1 Java Code

```

1 package jumble;
2
3
4
5 public class BinaryToHexadecimal {
6     public static String convertBinaryToHex(String binary) {
7         int decimal = Integer.parseInt(binary, 2);
8         return Integer.toHexString(decimal).toUpperCase();
9     }
10 }
```

```

1 package jumble;
2
3
4
5 import org.junit.Test;
6 import static org.junit.Assert.assertEquals;
7
8 public class BinaryToHexadecimalTest {
9     @Test
10     public void testConvertBinaryToHex() {
11         assertEquals("A", BinaryToHexadecimal.convertBinaryToHex("1010"));
12         assertEquals("1F", BinaryToHexadecimal.convertBinaryToHex("11111"));
13     }
14 }
```

### 10.2 Jumble screenshot

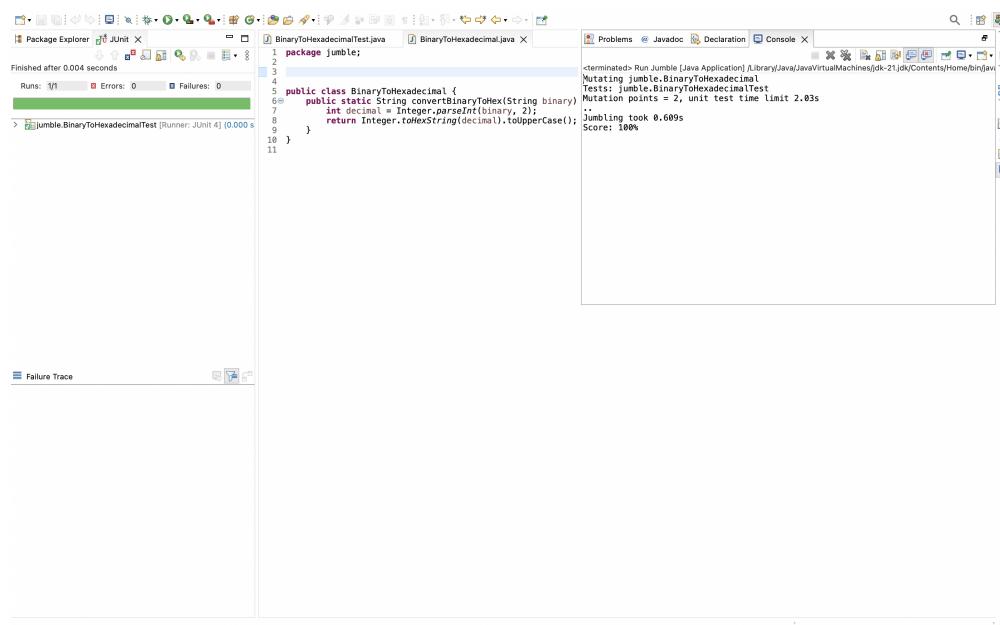


Figure 10: Jumble Test Case