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.

```java
package jumble;
import junit.framework.TestCase;
public class FactorialTest extends TestCase {
    public void test1() {
        assertEquals("479001600",Factorial.fact(12));
    }
    public void test2() {
        assertEquals("0",Factorial.fact(0));
    }
    public void test3() {
        assertEquals("1",Factorial.fact(1));
    }
    public void test4() {
        assertEquals("24",Factorial.fact(4));
    }

}
```

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
at jumble.FactorialTest.test2(FactorialTest.java:8)

Figure 1: Jumble Test Case

```java
package jumble;

public class Factorial {

    public static String fact(int n) {
        if(n==0)
        {
            return ""+1;
        }
        else if(n<0)
        {
            return "For Negative number no factorial";
        }
        else
        {

            long fact[]=new long [n+1];
            fact[0]=1;
            for(int i=1;i<=n;i++)
            {
                fact[i]=fact[i-1]*i;

            }
            return ""+fact[n];
        }

    }

}
```

Finished after 0.044 seconds

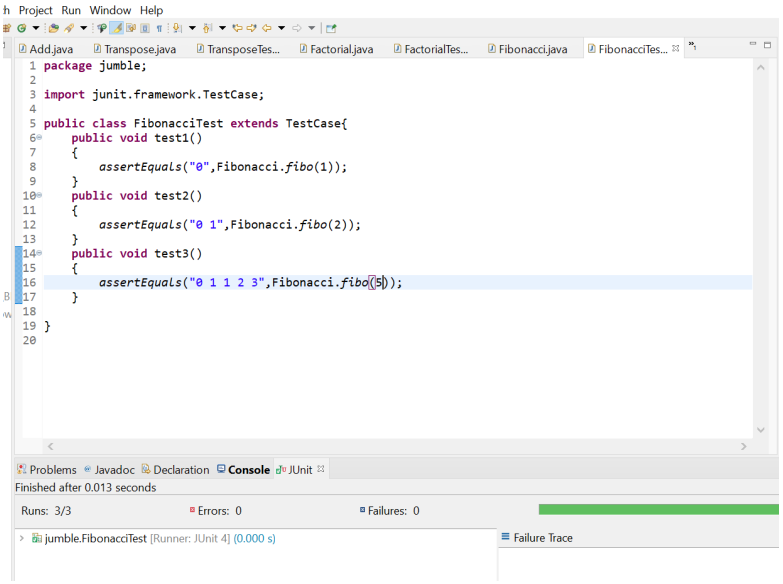
Runs: 4/4 Errors: 0 Failures: 1

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

Failure Trace

Figure 2: Factorial Code

2 Write a program to generate Fibonacci numbers.



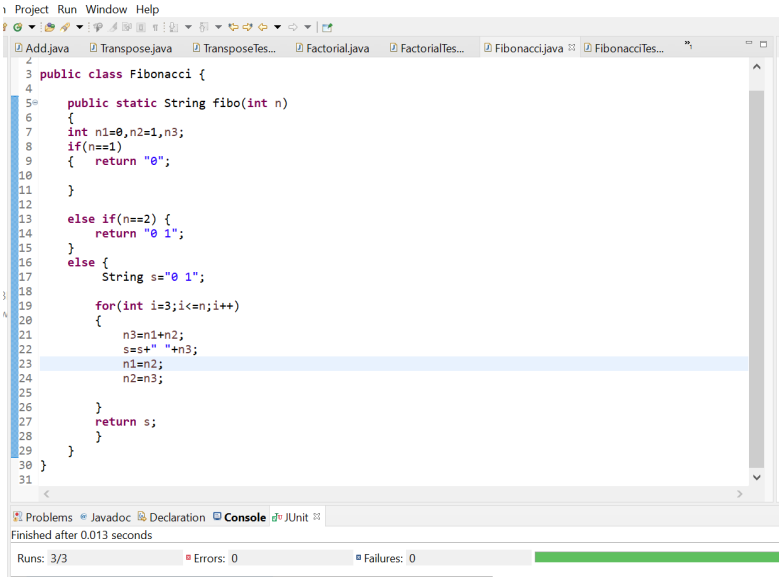
```java
package jumble;

import junit.framework.TestCase;

public class FibonacciTest extends TestCase{
    public void test1()
    {
        assertEquals("0",Fibonacci.fibo(1));
    }
    public void test2()
    {
        assertEquals("0 1",Fibonacci.fibo(2));
    }
    public void test3()
    {
        assertEquals("0 1 1 2 3",Fibonacci.fibo(5));
    }

}
```

Finished after 0.013 seconds

Runs: 3/3 Errors: 0 Failures: 0

jumble.FibonacciTest [Runner: JUnit 4] (0.000 s)

Figure 3: Jumble Test Case



```java
public class Fibonacci {

    public static String fibo(int n)
    {
    int n1=0,n2=1,n3;
    if(n==1)
    {   return "0";

    }

    else if(n==2) {
        return "0 1";
    }
    else {
         String s="0 1";

        for(int i=3;i<=n;i++)
        {
            n3=n1+n2;
            s=s+" "+n3;
            n1=n2;
            n2=n3;

        }
        return s;
        }
    }
}
```

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.

```java
import static org.junit.Assert.assertArrayEquals;

import junit.framework.TestCase;
public class QuickSortTest extends TestCase {
    public void test1()
    {
        int[] arr= {1,4,5,78,2,7,10};
        int [] output = {1,2,4,5,7,10,78};
        assertArrayEquals(output, QuickSort.quicksort(arr, 0, 6));
    }
    public void test2()
    {
        int[] arr= {1,42,78,2,10};
        int [] output = {1,2,10,42,78};
        assertArrayEquals(output, QuickSort.quicksort(arr, 0, 4));
    }

}
```

Figure 5: Jumble Test Case

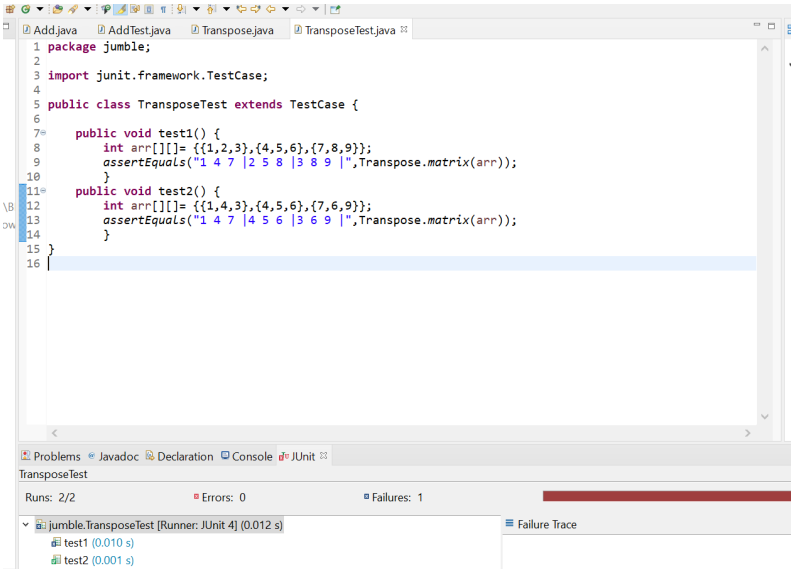
```java
public class QuickSort {
    public static int partition(int array[], int start, int end)
    {
        int pivot = array[end];
        int i = (start-1);
        for (int j=start; j<end; j++)
        {
            if (array[j] <= pivot)
            {
                i++;
                int temp = array[i];
                array[i] = array[j];
                array[j] = temp;
            }
        }
        int temp = array[i+1];
        array[i+1] = array[end];
        array[end] = temp;
        return i+1;
    }
    public static int[] quicksort(int arr[], int low, int high)
    {
        if (low < high)
        {
            int p = partition(arr, low, high);
            quicksort(arr, low, p-1);
            quicksort(arr, p+1, high);
        }
        return arr;
```

Problems @ Javadoc Declaration Console

```
<terminated> Run Jumble [Java Application] D:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.
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.



```java
package jumble;

import junit.framework.TestCase;

public class TransposeTest extends TestCase {

    public void test1() {
        int arr[][]= {{1,2,3},{4,5,6},{7,8,9}};
        assertEquals("1 4 7 |2 5 8 |3 8 9 |",Transpose.matrix(arr));
        }
    public void test2() {
        int arr[][]= {{1,4,3},{4,5,6},{7,6,9}};
        assertEquals("1 4 7 |4 5 6 |3 6 9 |",Transpose.matrix(arr));
        }
}
```

Add.java | AddTest.java | Transpose.java | TransposeTest.java

Problems | Javadoc | Declaration | Console | JUnit

TransposeTest

Runs: 2/2 Errors: 0 Failures: 1

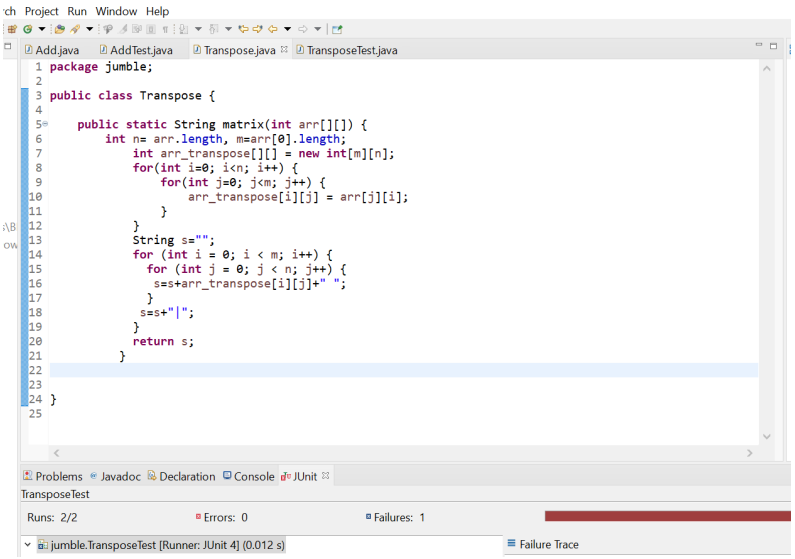
jumble.TransposeTest [Runner: JUnit 4] (0.012 s)

test1 (0.010 s)

test2 (0.001 s)

Failure Trace

Figure 7: Jumble Test Case



```java
package jumble;

public class Transpose {

    public static String matrix(int arr[][]) {
        int n= arr.length, m=arr[0].length;
            int arr_transpose[][] = new int[m][n];
            for(int i=0; i<n; i++) {
                for(int j=0; j<m; j++) {
                    arr_transpose[i][j] = arr[j][i];
                }
            }
            String s="";
            for (int i = 0; i < m; i++) {
              for (int j = 0; j < n; j++) {
               s=s+arr_transpose[i][j]+" ";
              }
             s=s+"|";
            }
            return s;
          }


}
```

Project Run Window Help

Add.java | AddTest.java | Transpose.java | TransposeTest.java

Problems | Javadoc | Declaration | Console | JUnit

TransposeTest

Runs: 2/2 Errors: 0 Failures: 1

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

Failure Trace

Figure 8: Matrix Transpose Code

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

```java
import junit.framework.TestCase;
import static org.junit.Assert.assertArrayEquals;
public class AdditionMatrixTest extends TestCase{
    public void test1() {
        int matrix1[][]= {{11,21},{44,58}};
        int matrix2[][] = {{55,79},{4,5}};
        int sum[][]= {{66,100},{48,63}};
        assertArrayEquals(sum,AdditionMatrix.addM(matrix1, matrix2));
    }
    public void test2() {
        int matrix1[][]= {{0,0},{0,0}};
        int matrix2[][] = {{10,10},{10,10}};
        int sum[][]= {{10,10},{10,10}};
        assertArrayEquals(sum,AdditionMatrix.addM(matrix1, matrix2));
    }
    public void test3() {
        int matrix1[][]= {{1,1},{1,1}};
        int matrix2[][] = {{-2,-2},{2,2}};
        int sum[][]= {{-1,-1},{3,3}};
        assertArrayEquals(sum,AdditionMatrix.addM(matrix1, matrix2));
    }
}
```

Figure 9: Jumble Test Case

```java
public class AdditionMatrix {
    public static int[][] addM(int[][] m1,int[][] m2)
    {
        int r = m1.length;
        int c = m1[0].length;
        int[][] sum = new int[r][c];
        for(int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                sum[i][j] = m1[i][j] + m2[i][j];
            }
        }
        return sum;
    }
}
```

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

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

```java
public class PrintPrimeTest extends TestCase {
    public void test1()
    {
        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() );
    }
}
```

Figure 11: Jumble Test Case

```java

public class PrintPrime {
    public static boolean checkPrime(int n)
    {
        if(n<=1)
            return false;
        for (int i=2;i<=(int)Math.sqrt(n);i++) {
            if (n % i == 0)
                return false;
        }
        return true;
    }
    public static String testPrime()
    {
        String result ="";
        for(int i=1;i <=100;i++)
        {
            if (checkPrime(i))
                result = result + i + " ";
        }
        return result;
    }
}

```

Problems @ Javadoc Declaration Console

```
<terminated> Run Jumble [Java Application] D:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
Mutating soft_testing_jumble_assignment.PrintPrime
Tests: soft_testing_jumble_assignment.PrintPrimeTest
Mutation points = 21, unit test time limit 2.0s
...............M FAIL: (soft_testing_jumble_assignment.PrintPrime.java:17): 1 -> 0
.....
Jumbling took 8.199s
Score: 95%
```

Figure 12: Prime Code

7 Write a program to generate a palindrome of numbers.

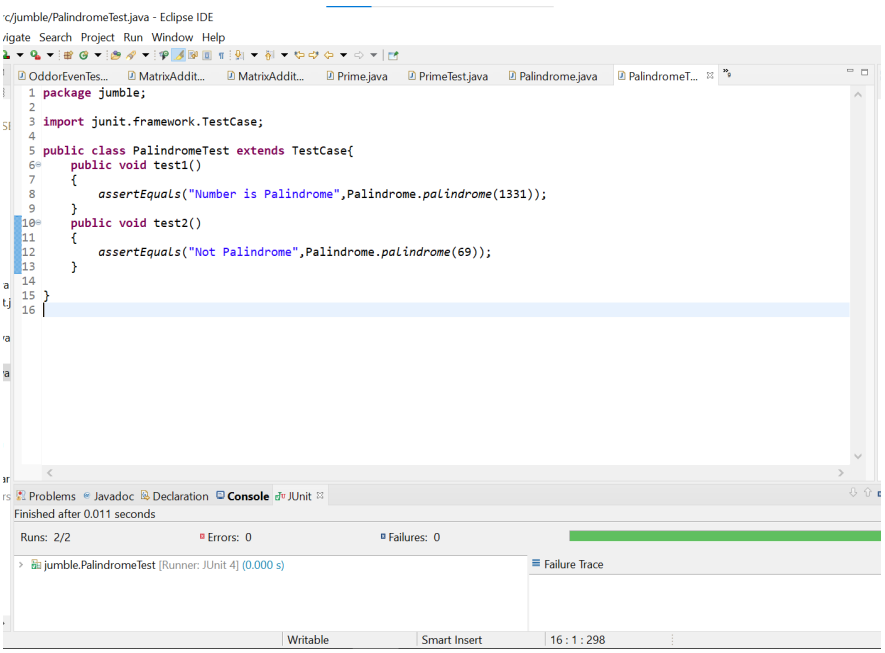


Figure 13: Jumble Test Case

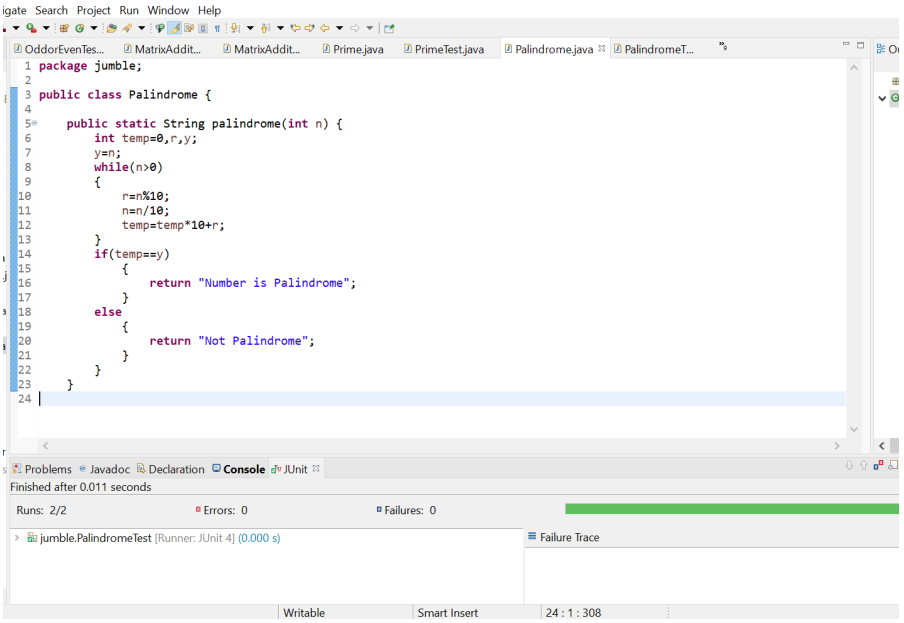


Figure 14: Palindrome Code

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

```java
import junit.framework.TestCase;

public class SumOfArrayTest extends TestCase{
    public void test1()
    {
        int a[]={1,2,3};
        int b[]={0,1,10};
        assertEquals(17,SumOfArray.sumArray(a, b));
    }
    public void test2()
    {
        int a[]={14,89,56};
        int b[]={5,68,23};
        assertEquals(255,SumOfArray.sumArray(a, b));
    }
    public void test3()
    {
        int a[]={4,9,6};
        int b[]={2,8,3};
        assertEquals(32,SumOfArray.sumArray(a, b));
    }
}
```

Figure 15: Jumble Test Case

```java
public class SumOfArray {
    public static int sumArray(int[] a1,int[] a2)
    {
        int i=0;
        int sum=0;
        while(i< a1.length) {
            sum += a1[i];
            i++;
        }
        int j=0;
        while(j< a2.length) {
            sum += a2[j];
            j++;
        }
        return sum;
    }
}
```

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.

```java
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

```java
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
```

Problems @ Javadoc Declaration Console

```
<terminated> Run Jumble [Java Application] D:\eclipse\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

```java
public class BinToHex {
    public static int binaryToDecimal(long binary)
    {

        int dec = 0, i = 0;
        while (binary > 0) {
            dec += Math.pow(2, i++) * (binary % 10);
            binary /= 10;
        }
        return dec;
    }
    public static String binToHex(long bin){
        int decimalNumber = binaryToDecimal(bin);
        String hexNumber = Integer.toHexString(decimalNumber);
        hexNumber = hexNumber.toUpperCase();
         return hexNumber;

    }
}
```

Problems @ Javadoc Declaration Console

```
<terminated> Run Jumble [Java Application] D:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspo
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