

# SOFTWARE TESTING LAB ASSIGNMENT-6

**Q.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.**

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under "repetatenumber/src".
- FactorialTest.java:** The code defines a class FactorialTest that extends TestCase. It contains six test methods: test1 through test6. The test methods assertEqual values for factorial calculations.
- Console:** Displays the execution results:
  - Runs: 6/6
  - Errors: 0
  - Failures: 1
- Failure Trace:** Shows a junit.framework.ComparisonFailure error for test5, comparing expected value [1307674368000] with actual value [20043].

```
1 package repetatenumber;
2
3 import java.util.ArrayList;
4 import java.util.Arrays;
5 import java.util.List;
6 import junit.framework.TestCase;
7
8 public class FactorialTest extends TestCase {
9     public void test1() {
10         assertEquals("120",Factorial.factor(5));
11     }
12
13     public void test2() {
14         assertEquals("factorial doesn't exist for negative number",Factorial.factor(-2));
15     }
16     public void test3() {
17         assertEquals("1",Factorial.factor(0));
18     }
19     public void test4() {
20         assertEquals("6",Factorial.factor(3));
21     }
22     public void test5() {
23         assertEquals("1307674368000",Factorial.factor(15));
24     }
25     public void test6() {
26         assertEquals("40320",Factorial.factor(8));
27     }
28 }
29 }
```

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under "repetatenumber/src".
- Factorial.java:** The code defines a class Factorial with a static method fact that calculates the factorial of a given integer n. It handles edge cases for n=0 and n<0.
- FactorialTest.java:** The code defines a class FactorialTest that extends TestCase. It contains five test methods: test1 through test5. The test methods assertEqual values for factorial calculations.
- Console:** Displays the execution results:
  - Runs: 6/6
  - Errors: 0
  - Failures: 1
- Failure Trace:** Shows a junit.framework.ComparisonFailure error for test5, comparing expected value [1307674368000] with actual value [20043].

```
1 package repetatenumber;
2
3
4 public class Factorial {
5     public static String fact(int n) {
6         if(n==0)
7         {
8             return ""+1;
9         }
10        else if(n<0)
11        {
12            return "factorial doesn't exist for negative number";
13        }
14        else
15        {
16
17            int fact[] = new int [n+1];
18            fact[0]=1;
19            for(int i=1;i<=n;i++)
20            {
21                fact[i]=fact[i-1]*i;
22            }
23        }
24        return ""+fact[n];
25    }
26
27 }
28 }
```

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under the package `repetatenumber`. The `src` folder contains several Java files: `Addition.java`, `AdditionTest.java`, `BtoH.java`, `BtoHTest.java`, `EvenorOdd.java`, `EvenorOddTest.java`, `Factorial.java`, and `FactorialTest.java`.
- Editor:** The main editor window displays `FactorialTest.java` with the following code:1 package repetatenumber;
2 import java.util.ArrayList;
3 import java.util.Arrays;
4 import java.util.List;
5 import junit.framework.TestCase;
6
7 public class FactorialTest extends TestCase {
8 public void test1() {
9 assertEquals("120",Factorial.fact(5));
10 }
11
12 public void test2() {
13 assertEquals("factorial doesn't exist for negative number",Factorial.fact(-2));
14 }
15 public void test3() {
16 assertEquals("1",Factorial.fact(0));
17 }
18 public void test4() {
19 assertEquals("6",Factorial.fact(3));
20 }
21 public void test5() {
22 assertEquals("1307674368000",Factorial.fact(15));
23 }
24 public void test6() {
25 assertEquals("40320",Factorial.fact(8));
26 }
27 }
28 }
- Console:** The console shows the test results:

```
Finished after 0.02 seconds
Runs: 6/6      Errors: 0      Failures: 0
```
- Failure Trace:** A link labeled "Failure Trace" is present.

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under the package `repetatenumber`. The `src` folder contains several Java files: `Addition.java`, `AdditionTest.java`, `BtoH.java`, `BtoHTest.java`, `EvenorOdd.java`, `EvenorOddTest.java`, `Factorial.java`, and `FactorialTest.java`.
- Editor:** The main editor window displays `Factorial.java` with the following code:1 package repetatenumber;
2
3 public class Factorial {
4 public static String fact(int n) {
5 if(n==0)
6 {
7 return ""+1;
8 }
9 else if(n<0)
10 {
11 return "factorial doesn't exist for negative number";
12 }
13 else
14 {
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 }
- Console:** The console shows the mutation test results:

```
<terminated> Run Jumble [Java Application] /Users/gourav/p2/pool/plugins/org.eclipse.jdt.core/jdt.core_1.1.0.v20221102-1007/jre/bin/java -O4-Mc
Mutating repetatenumber.Factorial
Tests: repetatenumber.FactorialTest
Mutation points = 17, unit test time limit 2.06s
.....
Jumbling took 12.821s
Score: 100%
```

## Q.2 Write a program to generate Fibonacci numbers.

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists several Java files under the package `repetatenumber`, including `Fibonacci.java`, `FibonacciTest.java`, and various test files for addition, factorial, prime, etc. The JUnit view on the right displays the code for `FibonacciTest.java` and the results of the test run. The test results show 3/3 runs successful, 0 errors, and 0 failures. The failure trace section is empty.

```
1 package repetatenumber;
2 import static org.junit.Assert.assertEquals;
3
4 import java.util.ArrayList;
5 import java.util.Arrays;
6 import java.util.List;
7 import junit.framework.TestCase;
8
9 public class FibonacciTest extends TestCase {
10    public void test1()
11    {
12        assertEquals("0",Fibonacci.fibo(1));
13    }
14    public void test2()
15    {
16        assertEquals("0 1",Fibonacci.fibo(2));
17    }
18    public void test3()
19    {
20        assertEquals("0 1 1",Fibonacci.fibo(3));
21    }
22
23
24
25 }
26
```

Console Declaration Javadoc JUnit  
Finished after 0.019 seconds  
Runs: 3/3 Errors: 0 Failures: 0

repetatenumber.FibonacciTest [Runner: JUnit 4] (0.000 s) Failure Trace

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists several Java files under the package `repetatenumber`, including `Fibonacci.java`. The code editor view on the right shows the implementation of the `fibo` method in `Fibonacci.java`. The console view at the bottom shows the output of the `FibonacciTest` class, indicating a mutation score of 100%.

```
1 package repetatenumber;
2
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        else if(n==2)
13        {
14            return "0 1";
15        }
16        else
17        {
18            String s="0 1";
19            for(int i=3;i<=n;i++)
20            {
21                n3=n1+n2;
22                s=s+" "+n3;
23                n1=n2;
24                n2=n3;
25            }
26        }
27        return s;
28    }
29 }
30 }
```

Console Declaration Javadoc JUnit  
<terminated> Run Jumble [Java Application] /Users/gourav/p2/pool/plugins/org.eclipse.jdt.openjdk.hotspot.jre.full.macosx.x86\_64\_19.0.1.v20221102-1007/jre/bin/java (02-Mutating repetatenumber.Fibonacci  
Tests: repetatenumber.FibonacciTest  
Mutation points = 16, unit test time limit 2.08s  
.....T.  
Jumbling took 13.839s  
Score: 100%

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

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists several Java files under the package `repetatenumber`. The `Quicksort.java` file is selected and open in the central editor. The code implements the Quicksort algorithm. Below the editor is the Console view, which displays the output of a JUnit test run. The output shows several failed assertions (FAIL) and a success message. At the bottom of the console, it says "Jumbling took 8.728s" and "Score: 63%".

```
1 package repetatenumber;
2
3 public class Quicksort {
4
5     static void swap(int arr[], int x, int y) {
6         int temp = arr[x];
7         arr[x] = arr[y];
8         arr[y] = temp;
9     }
10
11    public static int partition(int arr[], int start, int end) {
12        int pivot = arr[end];
13        int Index = start;
14        for (int i = start; i < end; i++) {
15            if (arr[i] < pivot) {
16                swap(arr, i, Index);
17                Index++;
18            }
19        }
20        swap(arr, Index, end);
21        return Index;
22    }
23
24
25    public static int[] sort(int arr[], int start, int end) {
26        if (start < end) {
27            int Index = partition(arr, start, end);
28            sort(arr, start, Index - 1);
29            sort(arr, Index + 1, end);
30        }
31        return arr;
32    }
}
<terminated> Run Jumble [Java Application] /Users/gourav/p2/pool/plugins/org.eclipse.jst.jdt.core/JRE/bin/java (04-M
M FAIL: (repetatenumber.Quicksort.java:15): negated conditional
..M FAIL: (repetatenumber.Quicksort.java:14): negated conditional
..M FAIL: (repetatenumber.Quicksort.java:26): negated conditional
..M FAIL: (repetatenumber.Quicksort.java:29): 1 -> 0
Jumbling took 8.728s
Score: 63%
```

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists several Java files under the package `repetatenumber`. The `QuicksortTest.java` file is selected and open in the central editor. The code contains three test methods: `test1`, `test2`, and `test3`, each using the `assertArrayEquals` method to verify the correctness of the `sort` function. Below the editor is the Console view, which shows the test results: 3/3 runs successful, 0 errors, and 0 failures. The Failure Trace view is also visible at the bottom.

```
1 package repetatenumber;
2
3 import static org.junit.Assert.assertArrayEquals;
4
5 public class QuicksortTest extends TestCase {
6
7     public void test1() {
8         int []arr= {1,4,6,3,5,0};
9
10        assertArrayEquals(arr,Quicksort.sort(arr,0,5));
11    }
12
13    public void test2() {
14        int []arr= {10,30,50,40,70,65,34,23,56};
15
16        assertArrayEquals(arr,Quicksort.sort(arr,0,8));
17    }
18
19    public void test3() {
20        int []arr= {1,2,3,4,5,6};
21
22        assertArrayEquals(arr,Quicksort.sort(arr,0,5));
23    }
24
25
26
27
28
29
30
31
32
33
34
35
36
37
}
<terminated> Run Jumble [Java Application] /Users/gourav/p2/pool/plugins/org.eclipse.jst.jdt.core/JRE/bin/java (04-M
QuicksortTest
Runs: 3/3      Errors: 0      Failures: 0
repetatenumber.QuicksortTest [Runner: JUnit 4] (0.009 s)
└─ test1 (0.009 s)
└─ test2 (0.000 s)
└─ test3 (0.000 s)
```

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

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists several Java files under the package `repetatenumber`. The `TransposeTest.java` file is selected. The Editor view on the right contains the code for `TransposeTest`, which includes two test methods using JUnit's `assertEquals` assertion. The Console view at the bottom shows the test results: "Runs: 2/2", "Errors: 0", and "Failures: 0". A green progress bar indicates the tests were run successfully.

```
1 package repetatenumber;
2 import static org.junit.Assert.assertEquals;
3
4 public class TransposeTest extends TestCase {
5     public void test1() {
6         int arr[][] = {{1,4,3},{2,5,8},{3,2,7}};
7         assertEquals("1 2 3 |4 5 2 |3 8 7 ", Transpose.matrix(arr));
8     }
9     public void test2() {
10        int arr[][] = {{2,5,3},{12,66,6},{1,16,42}};
11        assertEquals("2 12 1 |5 66 16 |3 6 42 ", Transpose.matrix(arr));
12    }
13}
14
15
16
17
18
19
20
21
```

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists several Java files under the package `repetatenumber`. The `Transpose.java` file is selected. The Editor view on the right contains the implementation of the `matrix` method, which creates a transpose of the input matrix and returns it as a string. The Console view at the bottom shows the test results for `TransposeTest`: "Runs: 2/2", "Errors: 0", and "Failures: 0". The output also includes the command used to run the tests and the mutation statistics: "Mutation points = 17, unit test time limit 2.05s".

```
1 package repetatenumber;
2
3 public class Transpose {
4     public static String matrix(int arr[][]) {
5         int n= arr.length, m=arr[0].length;
6
7         int arr_transpose[][] = new int[m][n];
8
9         for (int i = 0; i < n; i++) {
10             for (int j = 0; j < m; j++) {
11                 arr_transpose[i][j] = arr[j][i];
12             }
13         }
14
15         String s="";
16         for (int i = 0; i < m; i++) {
17             for (int j = 0; j < n; j++) {
18                 s+=arr_transpose[i][j]+" ";
19             }
20             s=s+"|";
21         }
22         return s;
23     }
24
25
26
27
28
29
30
31
32
33}
```

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

The screenshot shows the Eclipse IDE interface. The left pane displays the Package Explorer with a project named 'repetatenumber' containing several source files like FibonacciTest, FactorialTest, TransposeTest, etc., and a test file 'AdditionTest.java'. The right pane shows the code for 'AdditionTest.java':

```
1 package repetatenumber;
2 import static org.junit.Assert.assertEquals;
3
4 public class AdditionTest extends TestCase{
5
6     public void test1()
7     {
8         int first[][]= {{1,3,3},{2,3,4},{4,4,3}};
9         int second[][]= {{2,3,2},{4,5,3},{5,6,7}};
10        assertEquals("3 6 5 |6 8 7 |9 10 10 |",Addition.Matrix(first, second));
11    }
12
13    public void test2()
14    {
15        int first[][]= {{3,3,3},{10,1,5},{6,4,8}};
16        int second[][]= {{2,12,2},{15,5,3},{21,6,7}};
17        assertEquals("5 15 5 |25 6 8 |27 10 15 |",Addition.Matrix(first, second));
18    }
19
20 }
21
22 }
```

The Console tab at the bottom indicates the tests were run successfully: "Runs: 2/2 Errors: 0 Failures: 0".

The screenshot shows the Eclipse IDE interface. The left pane displays the Package Explorer with a project named 'repetatenumber' containing several source files like FibonacciTest, FactorialTest, TransposeTest, etc., and a main file 'Addition.java'. The right pane shows the code for 'Addition.java':

```
1 package repetatenumber;
2
3 public class Addition {
4
5     public static String Matrix(int first[][],int second[][]){
6
7         int[][] sum = new int[3][3];
8         String s="";
9         for(int i = 0; i < 3; i++) {
10             for (int j = 0; j < 3; j++) {
11                 sum[i][j] = first[i][j] + second[i][j];
12                 s=s+sum[i][j]+" ";
13             }
14             s=s+"\n";
15         }
16     }
17     return s;
18 }
19
20 }
```

The Console tab at the bottom shows the output of the application: "Tests: repetatenumber.AdditionTest Mutation points = 15, unit test time limit 2.07s ...M FAIL: (repetatenumber.Addition.java:8): 3 -> 4 M FAIL: (repetatenumber.Addition.java:8): 3 -> 4 ..... Jumbling took 11.456s Score: 86%".

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "repetatenumber/src". The "PrimeTest.java" file is selected and highlighted in blue.
- Editor:** Displays the code for the "PrimeTest" class, which extends "TestCase". It contains a single test method "test1" that asserts the output of the "Checkprime" static method against a hardcoded string of prime numbers.
- Console:** Shows the output of the JUnit test run. It indicates "Runs: 1/1", "Errors: 0", and "Failures: 0".
- Status Bar:** Shows "Finished after 0.015 seconds".

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "repetatenumber/src". The "Prime.java" file is selected and highlighted in blue.
- Editor:** Displays the code for the "Prime" class. It contains two static methods: "Checkprime" and "checkPrime". The "Checkprime" method iterates through numbers 1 to 100, calling the "checkPrime" method for each and appending the result to a string "s". The "checkPrime" method checks if a number is prime by attempting to divide it by all integers from 2 to its square root. If any division results in a remainder of 0, it returns false; otherwise, it returns true at the end of the loop.
- Console:** Shows the output of the JUnit test run. It includes the command "Run Jumble [Java Application] /Users/gourav/p2/pool/plugins/org.eclipse.jdt.openjdk.hotspot.jre.full.macosx.x86\_64\_19.0.1.v20221102-1007/jre/bin/java (04-Mutating repetatenumber.Prime Tests: repetatenumber.PrimeTest Mutation points = 20, unit test time limit 2.06s ....T..... Jumbling took 16.974s Score: 100%".

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

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under the package `repetatenumber`. The `src` folder contains several test classes like `QuicksortTest.java`, `AdditionTest.java`, etc., and the target class `PalindromeTest.java`.
- Editor:** Displays the `PalindromeTest.java` file containing JUnit test cases for checking palindromes.
- Console:** Shows the test results: "Finished after 0.019 seconds" with 3/3 runs successful, 0 errors, and 0 failures.

```
1 package repetatenumber;
2 import static org.junit.Assert.assertEquals;
3
4 public class PalindromeTest extends TestCase {
5     public void test1()
6     {
7         assertEquals("Number is Palindrome", Palindrome.Checkpalindrome(121));
8     }
9     public void test2()
10    {
11        assertEquals("Not Palindrome", Palindrome.Checkpalindrome(96));
12    }
13    public void test3()
14    {
15        assertEquals("Number is Palindrome", Palindrome.Checkpalindrome(10));
16    }
17 }
```

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under the package `repetatenumber`. The `src` folder contains several test classes and the target class `Palindrome.java`.
- Editor:** Displays the `Palindrome.java` file containing the implementation of the `Checkpalindrome` method.
- Console:** Shows the execution results of the `PalindromeTest` class using JUnit. It includes the command run, mutation statistics, and the output of the test cases.

```
1 package repetatenumber;
2
3 public class Palindrome {
4
5     public static String Checkpalindrome(int n) {
6         int temp=0, r, y;
7         y=n;
8         while(n>0)
9         {
10             r=n%10;
11             n=n/10;
12             temp=temp*10+r;
13         }
14         if(temp==y)
15         {
16             return "Number is Palindrome";
17         }
18         else
19         {
20             return "Not Palindrome";
21         }
22     }
23
24 }
```

```
<terminated> Run Jumble [Java Application] /Users/gourav.p2/pool/plugins/org.eclipse.jdt.openjdk.hotspot.jre.full.macosx.x86_64_19.0.1.v20221102-1007/jre/bin/java (04-M
Mutating repetatenumber.Palindrome
Tests: repetatenumber.PalindromeTest
Mutation points = 14, unit test time limit 2.05s
.....T...
Jumbling took 12.618s
Score: 100%
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under the package `repetatenumber`. The `src` folder contains several test files like `SumofarrayTest.java`.
- Editor:** Displays the `SumofarrayTest.java` file which contains JUnit test cases for the `Sumofarray` class.
- Console:** Shows the JUnit test results:
  - Runs: 3/3
  - Errors: 0
  - Failures: 0A green progress bar indicates successful execution.

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under the package `repetatenumber`. The `src` folder contains several test files like `SumofarrayTest.java`.
- Editor:** Displays the `Sumofarray.java` file which contains the implementation of the `arrsum` method.
- Console:** Shows the output of the `Run Jumble` command:
  - <terminated> Run Jumble [Java Application] /Users/gourav/p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86\_64\_19.0.1.v20221102-1007/jre/bin/java (04-Mutating repetatenumber.Sumofarray)
  - Tests: repetatenumber.SumofarrayTest
  - Mutation points = 13, unit test time limit 2.05s
  - M FAIL: (repetatenumber.Sumofarray.java:6): 0 -> 1
  - .....
  - Jumbling took 10.014s
  - Score: 92%

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

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under the package `repetatenumber`. The `src` folder contains several Java files: `EvenorOddTest.java` (selected), `Factorial.java`, `FactorialTest.java`, `Fibonacci.java`, `FibonacciTest.java`, `Palindrome.java`, `PalindromeTest.java`, `Prime.java`, `PrimeTest.java`, `Quicksort.java`, `QuicksortTest.java`, `Sumofarray.java`, `SumofarrayTest.java`, `Transpose.java`, and `TransposeTest.java`.
- Editor:** Displays the `EvenorOddTest.java` code, which contains four test methods (`test1` through `test4`) using `assertEquals` assertions to verify the output of the `EvenorOdd` class.
- Console:** Shows the results of the JUnit test run:
  - Runs: 4/4
  - Errors: 0
  - Failures: 0A green progress bar indicates 100% completion.

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under the package `repetatenumber`. The `src` folder contains several Java files: `EvenorOdd.java` (selected), `EvenorOddTest.java`, `Factorial.java`, `FactorialTest.java`, `Fibonacci.java`, `FibonacciTest.java`, `Palindrome.java`, `PalindromeTest.java`, `Prime.java`, `PrimeTest.java`, `Quicksort.java`, `QuicksortTest.java`, `Sumofarray.java`, `SumofarrayTest.java`, `Transpose.java`, and `TransposeTest.java`.
- Editor:** Displays the `EvenorOdd.java` code, which defines a static method `Eor0` that returns "Number is Even" if the input `n` is even, and "Number is Odd" if it is odd.
- Console:** Shows the results of the JUnit test run:
  - <terminated> Run Jumble [Java Application] /Users/gourav/.p2/pool/plugins/org.eclipse.jst.jdt.openjdk.hotspot.jre.full.macosx.x86\_64\_19.0.1.v20221102-1007/jre/bin/java (04-Mutating repetatenumber.EvenorOdd)
  - Tests: `repetatenumber.EvenorOddTest`
  - Mutation points = 7, unit test time limit 2.06s
  - .....
  - Jumbling took 5.759s
  - Score: 100%

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

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists several Java files under the package `repetatenumber`. The `BtoHTest.java` file is selected and open in the central editor area. The code implements a `TestCase` with four test methods (`test1`, `test2`, `test3`, `test4`) that assert the output of the `btoh` method against expected hex values. The JUnit console at the bottom shows the results of the run: 4/4 tests passed with 0 errors and 0 failures. The failure trace section is empty.

```
1 package repetatenumber;
2 import static org.junit.Assert.assertEquals;
3
4 public class BtoHTest extends TestCase{
5     public void test1()
6     {
7         assertEquals("A ",BtoH.btoh(1010));
8     }
9     public void test2()
10    {
11        assertEquals("2A ",BtoH.btoh(101010));
12    }
13    public void test3()
14    {
15        assertEquals("26",BtoH.btoh(100110));
16    }
17    public void test4()
18    {
19        assertEquals("3B ",BtoH.btoh(111011));
20    }
21
22 }
```

Console output:  
Finished after 0.019 seconds  
Runs: 4/4 Errors: 0 Failures: 0  
repetatenumber.BtoHTest [Runner: JUnit 4] (0.001 s) Failure Trace

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists several Java files under the package `repetatenumber`. The `BtoH.java` file is selected and open in the central editor area. The code defines a static method `btoh` that converts a binary integer to a hexadecimal string. It uses a while loop to repeatedly divide the binary number by 16 and record the remainder as a decimal digit. The decimal digit is then converted to its corresponding hex character ('0'-'F') and appended to a result string. The Run Jumble console at the bottom shows the mutation analysis and jumbling process. The mutation points are identified, and the mutation score is calculated.

```
1 public class BtoH {
2
3     public static String btoh(int binary) {
4         int i = 1, j = 0, rem, decimal = 0;
5         int hex[] = new int[10];
6
7         while (binary > 0)
8         {
9             rem = binary % 2;
10            decimal = decimal + rem * i;
11            i = i * 2;
12            binary = binary / 10;
13        }
14        i = 0;
15        while (decimal != 0)
16        {
17            hex[i] = decimal % 16;
18            decimal = decimal / 16;
19            i++;
20        }
21        String s="";
22        for (j = i - 1; j >= 0; j--)
23        {
24            if (hex[j] > 9)
25                {s=s+(char)(hex[j] + 55)+" ";}
26            else
27                {s=s+hex[j]+" ";}
28        }
29
30    }
31
32    return s;
33 }
34 }
```

Console output:  
<terminated> Run Jumble [Java Application] /Users/gourav/.p2/pool/plugins/org.eclipse.jdt.openjdk.hotspot.jre.full.macosx.x86\_64\_19.0.1.v20221102-1007/jre/bin/java (04-Mutation points = 31, unit test time limit 2.06s  
...M FAIL: (repetatenumber.BtoH.java:6): 0 -> 1  
.M FAIL: (repetatenumber.BtoH.java:7): 101 (e) -> 102 (f)  
.....M FAIL: (repetatenumber.BtoH.java:20): 16 -> 17  
.....  
Jumbling took 22.791s  
Score: 90%