**JUnit testing:**

Program: Factorial Algorithm

Code:

|  |
| --- |
| package factorial;  public class Factorial {    public static long calculateFactorial(int n){  if(n < 0){  throw new IllegalArgumentException("Factorial is not defined");  }  if(n == 0){  return 1;  }    long result = 1;  for(int i=2; i<=n; i++){  result\*=i;  }  return result;  }    public static void main(String[] args){  try{  int num = -5;  long factorial= calculateFactorial(num);  System.out.println("The Factorial of " + num + "is" + factorial );  }catch(IllegalArgumentException e){  System.err.println(e.getMessage());  }  }  } |

Test Package:

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package factorial;  import org.junit.After;  import org.junit.AfterClass;  import org.junit.Before;  import org.junit.BeforeClass;  import org.junit.Test;  import static org.junit.Assert.\*;  /\*\*  \*  \* @author FA21-BSE-057  \*/  public class FactorialTest {    public FactorialTest() {  }    @BeforeClass  public static void setUpClass() {  }    @AfterClass  public static void tearDownClass() {  }    @Before  public void setUp() {  }    @After  public void tearDown() {  }  /\*\*  \* Test of calculateFactorial method, of class Factorial.  \*/  @Test  public void testCalculateFactorial() {  System.out.println("calculateFactorial");  int n = 5;  long expResult = 120L;  long result = Factorial.calculateFactorial(n);  assertEquals(expResult, result);  // // TODO review the generated test code and remove the default call to fail.  // fail("The test case is a prototype.");  }  @Test(expected = IllegalArgumentException.class)  public void testNegativeInput() {  int n = -5;  Factorial.calculateFactorial(n);  }  /\*\*  \* Test of main method, of class Factorial.  \*/  @Test  public void testMain() {  System.out.println("main");  String[] args = null;  Factorial.main(args);  // // TODO review the generated test code and remove the default call to fail.  // fail("The test case is a prototype.");  }    } |

**Test Cases Through White Box Testing Technique (CFG) :**

**Test Cases for Path 1 (n = 0)**

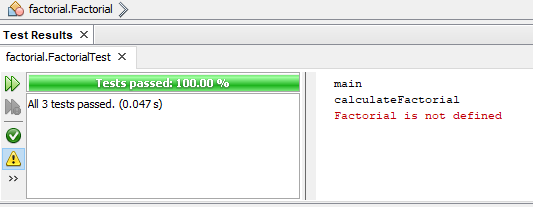
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Result | Actual Outcome (Pass/Fail) | Coverage Criteria |
| TC-1 | Check factorial for zero | 0 | Should return 1 (base case) | Pass | Statement Coverage, Decision Coverage |

**Table 2: Test Cases for Path 2 (n is negative)**

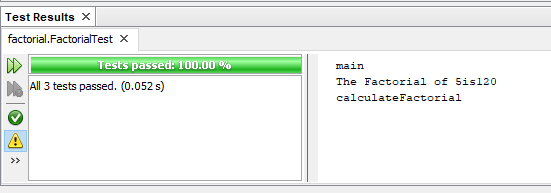
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Result | Actual Outcome (Pass/Fail) | Coverage Criteria |
| TC-2 | Try calculating factorial for a negative number | -3 | Should throw an error message | Pass | Statement Coverage, Decision Coverage |

Outputs WRT to Test Cases:

Test Case Input = - 5



Test Case Input = 5 (Positive Integer)



J-Unit Testing through Black Box Testing Technique:

Equivalence Partitioning:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Input Value | Boundary | Expected Output | Actual Result | Status |
| TC\_01 | 0 | 10 | The Factorial is 1 | 1 | Pass |
| TC\_02 | 3 | 10 | The Factorial is 6 | 6 | Pass |
| TC\_03 | 10 | 10 | The Factorial is 3628800 | 3628800 | Pass |
| TC\_04 | -3 | 10 | IllegalArgumentException.class | Error | Pass |

Error Guessing Testing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Input Value | Expected Output | Actual Result | Status |
| TC\_01 | Hello World | IllegalArgumentException.class | Error | Pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Case Name | Input (n) | Expected Result | Expected Exception | Description |
| TC\_01 | testPositiveFactorial\_Zero | 0 | 1 | - | Tests base case (factorial of 0 is 1) |
| TC\_02 | testPositiveFactorial\_Small | 3 | 6 | - | Tests a small positive integer |
| TC\_03 | testPositiveFactorial\_Large | 10 | 3628800 | - | Tests a (potentially) large positive integer (adjust based on data type) |
| TC\_04 | testNegativeInput | -5 | - | IllegalArgumentException | Tests negative input (expected to throw exception) |
| TC\_05 | testInvalidInputType | "hello" | - | IllegalArgumentException (or appropriate exception) | Tests non-numeric input (expected to throw exception) |

Decision Table Testing: