15-1-2025

Andrés Layedra, Xavier Camacho, Carlos Ronquillo

Taller 07

JUnit

**Casos de pruebas:**

|  |  |
| --- | --- |
| ID: | 001 |
| Propósito: | Probar que la función MakeFormula() genera una fórmula válida y aleatoria. |
| Precondiciones: |  |
| Entradas y salidas esperadas: | Invocación de MakeFormula(), 47+31 |
| Resultado: | La fórmula generada debe ser válida con 1 o 2 operaciones y números aleatorios. |

|  |  |
| --- | --- |
| ID: | 002 |
| Propósito: | Verificar que la función Solve() resuelve correctamente una suma |
| Precondiciones: |  |
| Entradas y salidas esperadas: | 15+30, 45 |
| Resultado: | 45 |

|  |  |
| --- | --- |
| ID: | 003 |
| Propósito: | Verificar que la función Solve() resuelve correctamente una resta. |
| Precondiciones: |  |
| Entradas y salidas esperadas: | 30-15, 15 |
| Resultado: | 15 |

|  |  |
| --- | --- |
| ID: | 004 |
| Propósito: | Verificar que la función Solve() resuelve correctamente una multiplicacion |
| Precondiciones: |  |
| Entradas y salidas esperadas: | 8\*3, 24 |
| Resultado: | 24 |

|  |  |
| --- | --- |
| ID: | 005 |
| Propósito: | Verificar que la función Solve() resuelve correctamente una division |
| Precondiciones: |  |
| Entradas y salidas esperadas: | 10/2, 5 |
| Resultado: | 5 |

|  |  |
| --- | --- |
| ID: | 006 |
| Propósito: | Verificar que la función Solve() maneja la jerarquia de operaciones. |
| Precondiciones: |  |
| Entradas y salidas esperadas: | 3+5\*2, 13 |
| Resultado: | 13. Se debe realizar primero la multiplicación (5\*2=10) y luego la suma (3+10=13). |

|  |  |
| --- | --- |
| ID: | 007 |
| Propósito: | Verificar que la función Solve() maneja la división por cero. |
| Precondiciones: |  |
| Entradas y salidas esperadas: | 5/0, Exception |
| Resultado: | Exception (la división por cero debe ser manejad como una excepción) |

|  |  |
| --- | --- |
| ID: | 008 |
| Propósito: | Verificar el comportamiento de Solve() con fórmulas complejas generadas aleatoriamente. |
| Precondiciones: |  |
| Entradas y salidas esperadas: | 42+5\*3-9, 48 |
| Resultado: | 48 |

**Código:**

**@Test**

**public** **void** **testMakeFormula**() {   
  
 System.out.println("MakeFormula");   
  
 String expResult = "";   
  
 String result = Operations.MakeFormula();   
  
 assertEquals(expResult, result);   
  
 // TODO review the generated test code and remove the default call to fail.   
  
 fail("The test case is a prototype.");   
  
 }   
  
   
  
 /\*\*   
  
 \* Test of Solve method, of class Operations.   
  
 \*/   
  
 **@Test**   
  
 **public** **void** **testSolve**() {   
  
 System.out.println("Solve");   
  
 String formula = "";   
  
 String expResult = "";   
  
 String result = Operations.Solve(formula);   
  
 assertEquals(expResult, result);   
  
 // TODO review the generated test code and remove the default call to fail.   
  
 fail("The test case is a prototype.");   
  
 }   
  
   
  
 **@Test**   
  
 **public** **void** **testFormulaFormat**(){   
  
 String formula = MakeFormula();   
  
 assertTrue(formula.matches("\\d+[\\+\\-\\\*/]\\d+([\\+\\-\\\*/]\\d+)?"));   
  
 }   
  
   
  
 **@Test**   
  
 **public** **void** **testFormulaNumberRange**(){   
  
 String formula = MakeFormula();   
  
 String[] numbers = formula.split("[\\\\+\\\\-\\\\\*/]");   
  
 **for**(String number : numbers){   
  
 **int** num = Integer.parseInt(number);   
  
 assertTrue(num >= **1** && num <=**99**);   
  
 }   
  
 }   
  
   
  
 **@Test**   
  
 **public** **void** **testFormulaOperatorCount**(){   
  
 String formula = MakeFormula();   
  
 **int** operatorCount = formula.replaceAll("[^\\+\\-\\\*/]", "").length();   
  
 assertTrue(operatorCount == **1** || operatorCount == **2**);   
  
 }   
  
   
  
 **@Test**   
  
 **public** **void** **testSolveSum**(){   
  
 String result = Solve("10+10");   
  
 assertEquals("10+10=20", result);   
  
 }   
  
   
  
 **@Test**   
  
 **public** **void** **testSolveRes**(){   
  
 String result = Solve("10-10");   
  
 assertEquals("10-10=0", result);   
  
 }   
  
   
  
 **@Test**   
  
 **public** **void** **testSolveMul**(){   
  
 String result = Solve("10\*10");   
  
 assertEquals("10\*10=100", result);   
  
 }   
  
   
  
 **@Test**   
  
 **public** **void** **testSolveDiv**(){   
  
 String result = Solve("10/10");   
  
 assertEquals("10/10=1", result);   
  
 }   
  
   
  
 **@Test**   
  
 **public** **void** **testSolveDivZero**(){   
  
 Solve("10/0");   
  
 }   
  
   
  
 **@Test**   
  
 **public** **void** **testSolveMulOperators**(){   
  
 String result = Solve("10\*10/2");   
  
 assertEquals("10\*10/2=50", result);   
  
 }   
  
}