SFE 4030

Assignment 1

Roman Numerals Test Cases

1. testSingleLetters: This set of tests ensures that each Roman numeral character (I, V, X, L, C, D, M) is correctly converted to its corresponding integer value (1, 5, 10, 50, 100, 500, 1000).
2. testManyLetters: Checks the conversion of a Roman numeral with multiple characters (XI) to its integer equivalent (11).
3. testSubtractiveNotation: Verifies the correct conversion of a Roman numeral using subtractive notation (IV) to its integer value (4).
4. testWithAndWithoutSubtractiveNotation: Tests the conversion of Roman numerals both with and without subtractive notation (XIV, XV) to their respective integer values (14, 15).
5. testRepetition: Checks the conversion of a Roman numeral with repeated characters (II) to its integer value (2).
6. testFirstNumber: Ensures that the conversion of the first Roman numeral character (I) results in its corresponding integer value (1).
7. testInvalidLetter: Verifies that an IllegalArgumentException is thrown when an invalid Roman numeral character (Z) is provided.
8. testInvalidAndValidLetter: Tests if an IllegalArgumentException is thrown when a mix of valid and invalid Roman numeral characters (XIZ) is provided.
9. testNotValid: Checks if an IllegalArgumentException is thrown when an invalid combination of Roman numeral characters (VV) is provided.
10. testNull: Ensures that passing a null value to the romanToInt method results in an IllegalArgumentException.

Test Case Code:

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class RomanNumeraltest {

@Test

public void testSingleLetters() {

RomanNumeral romanNumeral = new RomanNumeral();

assertEquals(1, romanNumeral.romanToInt("I"));

assertEquals(5, romanNumeral.romanToInt("V"));

assertEquals(10, romanNumeral.romanToInt("X"));

assertEquals(50, romanNumeral.romanToInt("L"));

assertEquals(100, romanNumeral.romanToInt("C"));

assertEquals(500, romanNumeral.romanToInt("D"));

assertEquals(1000, romanNumeral.romanToInt("M"));

}

@Test

public void testManyLetters() {

RomanNumeral romanNumeral = new RomanNumeral();

assertEquals(11, romanNumeral.romanToInt("XI"));

}

@Test

public void testSubtractiveNotation() {

RomanNumeral romanNumeral = new RomanNumeral();

assertEquals(4, romanNumeral.romanToInt("IV"));

}

@Test

public void testWithAndWithoutSubtractiveNotation() {

RomanNumeral romanNumeral = new RomanNumeral();

assertEquals(14, romanNumeral.romanToInt("XIV"));

assertEquals(15, romanNumeral.romanToInt("XV"));

}

@Test

public void testRepetition() {

RomanNumeral romanNumeral = new RomanNumeral();

assertEquals(2, romanNumeral.romanToInt("II"));

}

@Test

public void testFirstNumber() {

RomanNumeral romanNumeral = new RomanNumeral();

assertEquals(1, romanNumeral.romanToInt("I"));

}

@Test(expected = IllegalArgumentException.class)

public void testInvalidLetter() {

RomanNumeral romanNumeral = new RomanNumeral();

romanNumeral.romanToInt("Z");

}

@Test(expected = IllegalArgumentException.class)

public void testInvalidAndValidLetter() {

RomanNumeral romanNumeral = new RomanNumeral();

romanNumeral.romanToInt("XIZ");

}

@Test(expected = IllegalArgumentException.class)

public void testNotValid() {

RomanNumeral romanNumeral = new RomanNumeral();

romanNumeral.romanToInt("VV");

}

@Test(expected = IllegalArgumentException.class)

public void testNull() {

RomanNumeral romanNumeral = new RomanNumeral();

romanNumeral.romanToInt(null);

}

}

Screenshot of test that have passed and failed:

