```java
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.ExpectedException;

import static org.junit.Assert.*;

public class HammingTest {

    @Rule
    public ExpectedException expectedException = ExpectedException.none();

    @Test
    public void testNoDistanceBetweenEmptyStrands() {
        assertEquals(0, new Hamming("", "").getHammingDistance());
    }

    @Test
    public void testNoDistanceBetweenShortIdenticalStrands() {
        assertEquals(0, new Hamming("A", "A").getHammingDistance());
    }

    @Test
    public void testNoDistanceBetweenLongIdenticalStrands() {
        assertEquals(0, new Hamming("GGACTGA", "GGACTGA").getHammingDistance());
    }

    @Test
    public void testCompleteDistanceInSingleNucleotideStrand() {
        assertEquals(1, new Hamming("A", "G").getHammingDistance());
    }

    @Test
    public void testCompleteDistanceInSmallStrand() {
```

```
34          assertEquals(2, new Hamming("AG", "CT").getHammingDistance());
35      }
36
37      @Test
38      public void testSmallDistanceInSmallStrand() {
39          assertEquals(1, new Hamming("AT", "CT").getHammingDistance());
40      }
41
42      @Test
43      public void testSmallDistanceInMediumStrand() {
44          assertEquals(1, new Hamming("GGACG", "GGTCG").getHammingDistance());
45      }
46
47      @Test
48      public void testSmallDistanceInLongStrand() {
49          assertEquals(2, new Hamming("ACCAGGG", "ACTATGG").getHammingDistance());
50      }
51
52      @Test
53      public void testNonUniqueCharacterInFirstStrand() {
54          assertEquals(1, new Hamming("AAG", "AAA").getHammingDistance());
55      }
56
57      @Test
58      public void testNonUniqueCharacterInSecondStrand() {
59          assertEquals(1, new Hamming("AAA", "AAG").getHammingDistance());
60      }
61
62      @Test
63      public void testSameNucleotidesInDifferentPositions() {
64          assertEquals(2, new Hamming("TAG", "GAT").getHammingDistance());
65      }
66
```

```
67      @Test
68      public void testLargeDistanceInPermutedStrand() {
69          assertEquals(4, new Hamming("GATACA", "GCATAA").getHammingDistance());
70      }
71
72      @Test
73      public void testLargeDistanceInOffByOneStrand() {
74          assertEquals(9, new Hamming("GGACGGATTCTG", "AGGACGGATTCT").getHammingDistance());
75      }
76
77      @Test
78      public void testValidatesFirstStrandNotLonger() {
79          expectedException.expect(IllegalArgumentException.class);
80          expectedException.expectMessage("leftStrand and rightStrand must be of equal length.");
81
82          new Hamming("AATG", "AAA");
83      }
84
85      @Test
86      public void testValidatesSecondStrandNotLonger() {
87          expectedException.expect(IllegalArgumentException.class);
88          expectedException.expectMessage("leftStrand and rightStrand must be of equal length.");
89
90          new Hamming("ATA", "AGTG");
91      }
92  }
```

```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class PangramCheckerTest {

    private PangramChecker pangramChecker;

    @Before
    public void setup() {
        pangramChecker = new PangramChecker();
    }

    @Test
    public void emptySentenceIsNotPangram() {
        assertFalse(pangramChecker.isPangram(""));
    }

    @Test
    public void recognizesPerfectLowerCasePangram() {
        assertTrue(pangramChecker.isPangram("abcdefghijklmnopqrstuvwxyz"));
    }

    @Test
    public void pangramWithOnlyLowerCaseLettersIsRecognizedAsPangram() {
        assertTrue(pangramChecker.isPangram("the quick brown fox jumps over the lazy dog"));
    }

    @Test
    public void phraseMissingCharacterXIsNotPangram() {
        assertFalse(pangramChecker.isPangram("a quick movement of the enemy will jeopardize five gunboats"));
```

```
33      }
34
35      @Test
36      public void phraseMissingAnotherCharacterIsNotPangram() {
37          assertFalse(pangramChecker.isPangram("five boxing wizards jump quickly at it"));
38      }
39
40      @Test
41      public void pangramWithUnderscoresIsRecognizedAsPangram() {
42          assertTrue(pangramChecker.isPangram("the_quick_brown_fox_jumps_over_the_lazy_dog"));
43      }
44
45      @Test
46      public void pangramWithNumbersIsRecognizedAsPangram() {
47          assertTrue(pangramChecker.isPangram("the 1 quick brown fox jumps over the 2 lazy dogs")
    );
48      }
49
50      @Test
51      public void phraseWithMissingLettersReplacedByNumbersIsNotPangram() {
52          assertFalse(pangramChecker.isPangram("7h3 qu1ck brown fox jumps ov3r 7h3 lazy dog"));
53      }
54
55      @Test
56      public void pangramWithMixedCaseAndPunctuationIsRecognizedAsPangram() {
57          assertTrue(pangramChecker.isPangram("\"Five quacking Zephyrs jolt my wax bed.\""));
58      }
59
60      @Test
61      public void upperAndLowerCaseVersionsOfTheSameCharacterShouldNotBeCountedSeparately() {
62          assertFalse(pangramChecker.isPangram("the quick brown fox jumps over with lazy FX"));
63      }
64  }
```