```
import org.junit.jupiter.api.Test;

import java.util.NoSuchElementException;

import static org.junit.jupiter.api.Assertions.*;

class DominoAndLinkedListTest {

    @Test
    void testAddToFrontAndIsEmpty() {
        LinkedList list = new LinkedList();
        assertTrue(list.isEmpty());

        list.addToFront(1, 2);

        assertFalse(list.isEmpty());
    }

    @Test
    void testFirstElement() {
        LinkedList list = new LinkedList();
        assertThrows(NoSuchElementException.class, list::firstElement);

        list.addToFront(1, 2);

        assertEquals(2, list.firstElement());
    }

    @Test
    void testLength() {
        LinkedList list = new LinkedList();
        assertEquals(0, list.length());

        list.addToFront(1, 2);
        list.addToFront(3, 4);

        assertEquals(2, list.length());
    }

    @Test
    void testRemoveFromFront() {
        LinkedList list = new LinkedList();
        list.addToFront(1, 2);

        list.removeFromFront();

        assertTrue(list.isEmpty());
    }


    @Test
    void testAddToFrontAndCanAdd() {
        DominoTrain train = new DominoTrain(6);

        train.addToFront(new Domino(5, 6, null));
        //checking if it rotates and adds
        assertEquals("[6|5]", train.getFirstDomino().toString());

        assertFalse(train.isEmpty());
```

```
        //should not be able to add because [6|3] cannot join to [6|5]
        //[6|3] can rotate, but [6|5] is already in the train and should not rotate
        assertFalse(train.canAdd(new Domino(6, 3, null)));
        assertTrue(train.canAdd(new Domino(2, 5, null)));
    }

    @Test
    void testGetEndValue() {
        DominoTrain train = new DominoTrain(3);
        train.addToFront(new Domino(2, 3, null));
        train.addToFront(new Domino(5, 2, null));

        //after adding, we would have [3|2][2|5], so the end value would be 5
        assertEquals(5, train.getEndValue());
    }

}
```