Li Xi

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

import java.util.Iterator;

public class DequeTests {
   @Test
   void isEmpty() {
      Deque<Double> d = new Deque<Double>();
      Assertions.assertTrue(d.isEmpty());
   }

   @Test
   void zeroSize() {
      Deque<Double> d = new Deque<Double>();
      Assertions.assertEquals(d.size(), 0);
   }

   @Test
   void addFirst() {
      Deque<Double> d = new Deque<Double>();
      d.addFirst(0.1);
      d.addFirst(0.2);
      Assertions.assertFalse(d.isEmpty());
```

```
    Assertions.assertEquals(d.size(), 2);
}

@Test
void addLast() {
    Deque<Double> d = new Deque<Double>();
    d.addLast(0.3);
    d.addLast(0.4);
    Assertions.assertFalse(d.isEmpty());
    Assertions.assertEquals(d.size(), 2);
}

@Test
void removeFirst() {
    Deque<Double> d = new Deque<Double>();
    d.addFirst(0.1);
    d.addFirst(0.2);
    Assertions.assertEquals(d.removeFirst(), 0.2);
    Assertions.assertEquals(d.size(), 1);
    Assertions.assertEquals(d.removeFirst(), 0.1);
    Assertions.assertTrue(d.isEmpty());
    Assertions.assertEquals(d.size(), 0);
}

@Test
```

```
void removeLast() {
    Deque<Double> d = new Deque<Double>();
    d.addFirst(0.1);
    d.addFirst(0.2);
    d.addLast(0.3);
    d.addLast(0.4);
    d.addLast(0.5);
    Assertions.assertEquals(d.removeLast(), 0.5);
    Assertions.assertEquals(d.size(), 4);
    Assertions.assertFalse(d.isEmpty());
    Assertions.assertEquals(d.removeLast(), 0.4);
    Assertions.assertEquals(d.size(), 3);
    Assertions.assertFalse(d.isEmpty());
    d.removeLast();
    d.removeLast();
    d.removeLast();
    Assertions.assertTrue(d.isEmpty());
    Assertions.assertEquals(d.size(), 0);
}

@Test
void iterator() {
    Deque<Double> d = new Deque<Double>();
    d.addFirst(0.1);
    d.addFirst(0.2);
```

```
        d.addFirst(0.25);
        d.addLast(0.3);
        d.addLast(0.4);
        d.addLast(0.45);
        d.removeFirst();
        d.removeFirst();
        d.removeLast();
        d.removeLast();
        Iterator<Double> DI = d.iterator();
        Assertions.assertTrue(DI.hasNext());
        Assertions.assertEquals(DI.next(), 0.1);
    }
}
```