```java
package tests;

import static org.junit.Assert.*;

import java.util.Set;

import org.junit.Before;
import org.junit.Test;

import experiment.BoardCell;
import experiment.IntBoard;

public class IntBoardTests
{
  IntBoard board;
  @Before
  public void beforeAll()
  {
    board = new IntBoard();

  }

  @Test
  public void testAdjacency0()
  {
    beforeAll();
//    board = new IntBoard();
    //IntBoard board = new IntBoard();


    BoardCell cell = board.getCell(0,0);
    System.out.println(cell);
    Set<BoardCell> testList = board.getAdjList(cell);
    //assertTrue(testList.contains(board.getCell(1, 0)));
    //assertTrue(testList.contains(board.getCell(0, 1)));

    assertEquals(2, testList.size());

  }

  @Test
  public void testAdjacency1()
  {
    beforeAll();
    BoardCell cell = board.getCell(3,3);
    Set<BoardCell> testList = board.getAdjList(cell);
    assertTrue(testList.contains(board.getCell(3, 2)));
    assertTrue(testList.contains(board.getCell(2, 3)));
    assertEquals(2, testList.size());

  }
  @Test
  public void testAdjacency2()
  {
    beforeAll();
    BoardCell cell = board.getCell(1,3);
    Set<BoardCell> testList = board.getAdjList(cell);
    assertTrue(testList.contains(board.getCell(2, 3)));
    assertTrue(testList.contains(board.getCell(1, 2)));
    assertTrue(testList.contains(board.getCell(0, 3)));

    assertEquals(3, testList.size());

  }
  @Test
  public void testAdjacency3()
  {
    beforeAll();
    BoardCell cell = board.getCell(3,0);
    Set<BoardCell> testList = board.getAdjList(cell);
    assertTrue(testList.contains(board.getCell(2, 0)));
    assertTrue(testList.contains(board.getCell(3, 1)));
    assertEquals(2, testList.size());

  }
  @Test
  public void testAdjacency4()
  {
    beforeAll();
    BoardCell cell = board.getCell(1,1);
    Set<BoardCell> testList = board.getAdjList(cell);
    assertTrue(testList.contains(board.getCell(0, 1)));
    assertTrue(testList.contains(board.getCell(2, 1)));
    assertTrue(testList.contains(board.getCell(1, 2)));
    assertTrue(testList.contains(board.getCell(1, 0)));


    assertEquals(4, testList.size());

  }
  @Test
  public void testAdjacency5()
  {
    beforeAll();
    BoardCell cell = board.getCell(2,2);
    Set<BoardCell> testList = board.getAdjList(cell);
    assertTrue(testList.contains(board.getCell(2, 1)));
    assertTrue(testList.contains(board.getCell(3, 2)));
    assertTrue(testList.contains(board.getCell(2, 3)));
    assertTrue(testList.contains(board.getCell(1, 2)));


    assertEquals(4, testList.size());

  }
  ///////////////////////////////////////////////////////////////////////
  @Test
  public void testTargets0_0_3()
  {
    BoardCell cell = board.getCell(0, 0);
    board.calcTargets(cell, 3);
    Set<BoardCell> targets = board.getTargets();
  //  assertEquals(6, targets.size());
    assertTrue(targets.contains(board.getCell(3, 0)));
    assertTrue(targets.contains(board.getCell(2, 1)));
    assertTrue(targets.contains(board.getCell(0, 1)));
    assertTrue(targets.contains(board.getCell(1, 2)));
    assertTrue(targets.contains(board.getCell(0, 3)));
    assertTrue(targets.contains(board.getCell(1, 0)));
  }
```

```
 @Test
 public void testTargets0_3_2()
 {
  BoardCell cell = board.getCell(0, 3);
  board.calcTargets(cell, 2);
  Set<BoardCell> targets = board.getTargets();
 // assertEquals(4, targets.size());
  assertTrue(targets.contains(board.getCell(2, 3)));
  assertTrue(targets.contains(board.getCell(0, 1)));
  assertTrue(targets.contains(board.getCell(1, 2)));
  assertTrue(targets.contains(board.getCell(1, 2)));

 }
 @Test
 public void testTargets3_1_2()
 {
  BoardCell cell = board.getCell(3, 1);
  board.calcTargets(cell, 2);
  Set<BoardCell> targets = board.getTargets();
// for (BoardCell target : targets)
// {
//  System.out.println(target.column + " " +target.row);
// }
// System.out.println("Balls " + targets);
  assertEquals(4, targets.size());
  assertTrue(targets.contains(board.getCell(3, 3)));
  assertTrue(targets.contains(board.getCell(2, 2)));
  assertTrue(targets.contains(board.getCell(1, 1)));
  assertTrue(targets.contains(board.getCell(2, 0)));

 }
 @Test
 public void testTargets1_0_3()
 {
  BoardCell cell = board.getCell(1, 0);
  board.calcTargets(cell, 3);
  Set<BoardCell> targets = board.getTargets();
  //assertEquals(6, targets.size());
  assertTrue(targets.contains(board.getCell(3, 1)));
  assertTrue(targets.contains(board.getCell(2, 2)));
  assertTrue(targets.contains(board.getCell(1, 3)));
  assertTrue(targets.contains(board.getCell(0, 2)));
  assertTrue(targets.contains(board.getCell(0, 0)));
  assertTrue(targets.contains(board.getCell(1, 1)));


 }

 @Test
 public void testTargets2_3_2()
 {
  BoardCell cell = board.getCell(2, 3);
  board.calcTargets(cell, 2);
  Set<BoardCell> targets = board.getTargets();
 // assertEquals(3, targets.size());
  assertTrue(targets.contains(board.getCell(3, 2)));
  assertTrue(targets.contains(board.getCell(2, 1)));
  assertTrue(targets.contains(board.getCell(1, 2)));
  assertTrue(targets.contains(board.getCell(0, 3)));

```

```java
  }

  @Test
  public void testTargets2_1_2()
  {
    BoardCell cell = board.getCell(2, 1);
    board.calcTargets(cell, 2);
    Set<BoardCell> targets = board.getTargets();
    //assertEquals(6, targets.size());
    assertTrue(targets.contains(board.getCell(3, 2)));
    assertTrue(targets.contains(board.getCell(3, 0)));
    assertTrue(targets.contains(board.getCell(0, 1)));
    assertTrue(targets.contains(board.getCell(1, 0)));
    assertTrue(targets.contains(board.getCell(1, 2)));
    assertTrue(targets.contains(board.getCell(2, 3)));

  }
}
```