import static org.junit.Assert.\*;

import models.Edge;

import models.Node;

import org.junit.Test;

/\*\*

 \* Testing strategy for the Edge class

 \*

 \* Partition the inputs as follows:

 \*

 \* For the Edge constructor:

 \* - Test with different starting and ending nodes.

 \*

 \* For getStartNode() and getEndNode():

 \* - Test getting the start and end nodes and verify they are correct.

 \*

 \* For setWeight(int weight) and getWeight():

 \* - Test setting a new weight and verify that getWeight() returns the correct value.

 \* - Test setting a negative weight and verify that getWeight() returns the correct value.

 \*

 \* For containsNode(Node node):

 \* - Test with a node that is part of the edge and expect true.

 \* - Test with a node that is not part of the edge and expect false.

 \*

 \* For equals(Edge edge):

 \* - Test with an equivalent edge and expect true.

 \* - Test with a non-equivalent edge and expect false.

 \*

 \* For toString():

 \* - Test getting the string representation of the edge and verify it matches the expected format.

 \*

 \* Exhaustive Cartesian coverage of partitions for each method.

 \*/

public class EdgeTest {

    // covers Edge constructor with different starting and ending nodes

    @Test

    public void testEdgeConstructor() {

        Node node1 = new Node(1);

        Node node2 = new Node(2);

        Edge edge = new Edge(node1, node2);

        assertEquals(node1, edge.getStartNode());

        assertEquals(node2, edge.getEndNode());

    }

    // covers getStartNode() and getEndNode()

    @Test

    public void testGetStartAndEndNodes() {

        Node node1 = new Node(1);

        Node node2 = new Node(2);

        Edge edge = new Edge(node1, node2);

        assertEquals(node1, edge.getStartNode());

        assertEquals(node2, edge.getEndNode());

    }

    // covers setWeight(int weight) and getWeight()

    @Test

    public void testSetAndgetWeight() {

        Edge edge = new Edge(new Node(1), new Node(2));

        edge.setWeight(5);

        assertEquals(5, edge.getWeight());

        edge.setWeight(-3);

        assertEquals(-3, edge.getWeight());

    }

    // covers containsNode(Node node)

    @Test

    public void testContainsNode() {

        Node node1 = new Node(1);

        Node node2 = new Node(2);

        Edge edge = new Edge(node1, node2);

        // Test with a node that is part of the edge

        assertTrue(edge.containsNode(node1));

        // Test with a node that is not part of the edge

        assertFalse(edge.containsNode(new Node(3)));

    }

    // covers equals(Edge edge)

    @Test

    public void testEquals() {

        Node node1 = new Node(1);

        Node node2 = new Node(2);

        Edge edge1 = new Edge(node1, node2);

        Edge edge2 = new Edge(node1, node2);

        // Test with an equivalent edge

        assertTrue(edge1.equals(edge2));

        // Test with a non-equivalent edge

        assertFalse(edge1.equals(new Edge(node1, new Node(3))));

    }

    // covers toString()

    @Test

    public void testToString() {

        Node node1 = new Node(1);

        Node node2 = new Node(2);

        Edge edge = new Edge(node1, node2);

        assertEquals("Edge ~ 1 - 2", edge.toString());

    }

}