```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import org.junit.Test;
import static org.junit.Assert.*;
import trafficsimulator.utils.Point;

/**
 *
 * @author snorri
 */
public class LaneTest {

  private Lane lane1;
  private Lane lane2;
  private Road road;

  public void setUp(Point start, Point end, Lane.Direction dir1, Lane.Direction
dir2) {

    road = new Road(start, end);
    lane1 = road.addLane(dir1);
    lane2 = road.addLane(dir2);

  }

  @Test
  public void testLanesHorizontal() {
    System.out.println("Lanes horizontal");

    Point start = new Point(100, 100);
    Point end = new Point(400, 100);

    setUp(start, end, Lane.Direction.IDENTICAL, Lane.Direction.IDENTICAL);

    double expXStartLane1 = start.getX();
    double expYStartLane1 = start.getY() + lane1.LANE_WIDTH/2;
    double expXEndLane1 = end.getX();
    double expYEndLane1 = end.getY() + lane1.LANE_WIDTH/2;
    double expXStartLane2 = start.getX();
    double expYStartLane2 = start.getY() + lane2.LANE_WIDTH + lane2.LANE_WIDTH/2
;
    double expXEndLane2 = end.getX();
    double expYEndLane2 = end.getY() + lane2.LANE_WIDTH  + lane2.LANE_WIDTH/2;

    double resultXStartLane1 = lane1.getStartPoint().getX();
    double resultYStartLane1 = lane1.getStartPoint().getY();
    double resultXEndLane1 = lane1.getEndPoint().getX();
    double resultYEndLane1 = lane1.getEndPoint().getY();
    double resultXStartLane2 = lane2.getStartPoint().getX();
    double resultYStartLane2 = lane2.getStartPoint().getY();
    double resultXEndLane2 = lane2.getEndPoint().getX();
    double resultYEndLane2 = lane2.getEndPoint().getY();

    assertEquals(expXStartLane1, resultXStartLane1, 0.001);
```

```
    assertEquals(expYStartLane1, resultYStartLane1, 0.001);
    assertEquals(expXEndLane1, resultXEndLane1, 0.001);
    assertEquals(expYEndLane1, resultYEndLane1, 0.001);
    assertEquals(expXStartLane2, resultXStartLane2, 0.001);
    assertEquals(expYStartLane2, resultYStartLane2, 0.001);
    assertEquals(expXEndLane2, resultXEndLane2, 0.001);
    assertEquals(expYEndLane2, resultYEndLane2, 0.001);


  }

  @Test
  public void testLanesVertical() {
    System.out.println("Lanes Vertical");

    Point start = new Point(100, 100);
    Point end = new Point(100, 400);

    setUp(start, end, Lane.Direction.IDENTICAL, Lane.Direction.OPPOSITE);

    double expXStartLane1 = start.getX() - lane1.LANE_WIDTH/2;
    double expYStartLane1 = start.getY();
    double expXEndLane1 = end.getX() - lane1.LANE_WIDTH/2;
    double expYEndLane1 = end.getY();
    double expXStartLane2 = end.getX() - lane2.LANE_WIDTH - lane2.LANE_WIDTH/2;
    double expYStartLane2 = end.getY();
    double expXEndLane2 = start.getX() - lane2.LANE_WIDTH  - lane2.LANE_WIDTH/2;
    double expYEndLane2 = start.getY();

    double resultXStartLane1 = lane1.getStartPoint().getX();
    double resultYStartLane1 = lane1.getStartPoint().getY();
    double resultXEndLane1 = lane1.getEndPoint().getX();
    double resultYEndLane1 = lane1.getEndPoint().getY();
    double resultXStartLane2 = lane2.getStartPoint().getX();
    double resultYStartLane2 = lane2.getStartPoint().getY();
    double resultXEndLane2 = lane2.getEndPoint().getX();
    double resultYEndLane2 = lane2.getEndPoint().getY();


    assertEquals(expXStartLane1, resultXStartLane1, 0.001);
    assertEquals(expYStartLane1, resultYStartLane1, 0.001);
    assertEquals(expXEndLane1, resultXEndLane1, 0.001);
    assertEquals(expYEndLane1, resultYEndLane1, 0.001);
    assertEquals(expXStartLane2, resultXStartLane2, 0.001);
    assertEquals(expYStartLane2, resultYStartLane2, 0.001);
    assertEquals(expXEndLane2, resultXEndLane2, 0.001);
    assertEquals(expYEndLane2, resultYEndLane2, 0.001);

  }

}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import org.junit.Test;
import static org.junit.Assert.*;
import trafficsimulator.utils.Point;

/**
 *
 * @author snorri
 */
public class RoadTest {

  private Road road;

  public void setUp(Point start, Point end) {
    road = new Road(start, end);
    Lane lane1 = road.addLane(Lane.Direction.IDENTICAL);
    Lane lane2 = road.addLane(Lane.Direction.OPPOSITE);

  }

  /**
   * This test checks if right parameters are calculated correctly if a
   * horizontal road is created and its direction is to the right, like this: ->
   * .
   */
  @Test
  public void testRoadHorizontalRight() {
    System.out.println("Road horizontal right");

    setUp(new Point(100, 100), new Point(400, 100));

    double expYStart = road.getLeftStartPoint().getY() + road.calculateWidth();
    double expXStart = road.getLeftStartPoint().getX();
    double expYEnd = road.getLeftEndPoint().getY() + road.calculateWidth();
    double expXEnd = road.getLeftEndPoint().getX();

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);

  }

  /**
   * This test checks if right parameters are calculated correctly if a
   * horizontal road is created and its direction is to the left, like this: <-
   * .
   */
```

```
  @Test
  public void testRoadHorizontalLeft() {
    System.out.println("Road horizontal left");

    setUp(new Point(400, 100), new Point(100, 100));

    double expYStart = road.getLeftStartPoint().getY() − road.calculateWidth();
    double expXStart = road.getLeftStartPoint().getX();
    double expYEnd = road.getLeftEndPoint().getY() − road.calculateWidth();
    double expXEnd = road.getLeftEndPoint().getX();

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);

  }

  /**
   * This test checks if right parameters are calculated correctly if a vertical
   * road is created and its direction is to the up, like this: ^ | .
   */
  @Test
  public void testRoadVerticalUp() {
    System.out.println("Road vertical up");

    setUp(new Point(100, 400), new Point(100, 100));

    double expYStart = road.getLeftStartPoint().getY();
    double expXStart = road.getLeftStartPoint().getX() + road.calculateWidth();
    double expYEnd = road.getLeftEndPoint().getY();
    double expXEnd = road.getLeftEndPoint().getX() + road.calculateWidth();

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);

  }

  /**
   * This test checks if right parameters are calculated correctly if a vertical
   * road is created and its direction is to the right, like this: | v .
   */
  @Test
  public void testRoadVerticalDown() {
    System.out.println("Road vertical down");

    setUp(new Point(100, 100), new Point(100, 400));
```

```
    double expYStart = road.getLeftStartPoint().getY();
    double expXStart = road.getLeftStartPoint().getX() - road.calculateWidth();
    double expYEnd = road.getLeftEndPoint().getY();
    double expXEnd = road.getLeftEndPoint().getX() - road.calculateWidth();

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);

  }

  /**
   * This test checks if right parameters are calculated correctly if a road
   * that has a downward slope is created and its direction is to the right,
   * like this: \ v .
   */
  @Test
  public void testRoadDownwardRight() {
    System.out.println("Road downward right");

    setUp(new Point(100, 100), new Point(400, 400));

    double expYStart = road.getLeftStartPoint().getY() + 35;
    double expXStart = road.getLeftStartPoint().getX() - 35;
    double expYEnd = road.getLeftEndPoint().getY() + 35;
    double expXEnd = road.getLeftEndPoint().getX() - 35;

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);

  }

  /**
   * This test checks if right parameters are calculated correctly if a road
   * that has a downward slope is created and its direction is to the left, like
   * this: / v .
   */
  @Test
  public void testRoadDownwardLeft() {
    System.out.println("Road downward left");

    setUp(new Point(400, 100), new Point(100, 400));

    double expYStart = road.getLeftStartPoint().getY() - 35;
    double expXStart = road.getLeftStartPoint().getX() - 35;
```

```
    double expYEnd = road.getLeftEndPoint().getY() - 35;
    double expXEnd = road.getLeftEndPoint().getX() - 35;

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);

  }

  /**
   * This test checks if right parameters are calculated correctly if a road
   * that has an upward slope is created and its direction is to the right, like
   * this: ^ / .
   */
  @Test
  public void testRoadUpwardRight() {
    System.out.println("Road upward right");

    setUp(new Point(100, 400), new Point(400, 100));

    double expYStart = road.getLeftStartPoint().getY() + 35;
    double expXStart = road.getLeftStartPoint().getX() + 35;
    double expYEnd = road.getLeftEndPoint().getY() + 35;
    double expXEnd = road.getLeftEndPoint().getX() + 35;

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);

  }

  /**
   * This test checks if right parameters are calculated correctly if a road
   * that has an upward slope is created and its direction is to the left, like
   * this: ^ \.
   */
  @Test
  public void testRoadUpwardLeft() {
    System.out.println("Road upward left");

    setUp(new Point(400, 400), new Point(100, 100));

    double expYStart = road.getLeftStartPoint().getY() - 35;
    double expXStart = road.getLeftStartPoint().getX() + 35;
    double expYEnd = road.getLeftEndPoint().getY() - 35;
    double expXEnd = road.getLeftEndPoint().getX() + 35;
```

```
    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);

  }

}
```