```
/**
 * @author Harley Phung
 * Test methods in Market Class
 */
import org.junit.*;
import static org.junit.Assert.*;
import java.util.NoSuchElementException;
public class MarketTester {
  Trader trader1 = new Trader("Harley");
  Trader trader2 = new Trader("Harold");
  Trader trader3 = new Trader("Jason");
  MarketMaker trader4 = new MarketMaker("Jen", 1000, 125);

  LLNode<Order> buyNode = new LLNode<Order>(new MarketBuyOrder('M', 100, 90,
trader3), null);
  LLNode<Order> buyList = new LLNode<Order>(new BuyLimitOrder('N', 120, 253, true,
true, trader1), buyNode);
  LLNode<Order> sellNode = new LLNode<Order>(new MarketSellOrder('O', 500, 200,
trader4), null);
  LLNode<Order> sellList = new LLNode<Order>(new SellLimitOrder('P', 235,100, true,
true, trader2), sellNode);


  Market market1 = new Market('M', buyList, sellList);
  Market market2 = new Market('N', buyList, sellNode);
  Market market3 = new Market('O', buyNode, sellList);
  Market market4 = new Market('P', buyNode, sellNode);

  /**
   * Test the getStockSymbol() method
   */
  @Test
  public void testGetStockSymbol() {
    assertEquals('M', market1.getStockSymbol());
    assertEquals('N', market2.getStockSymbol());
    assertEquals('O', market3.getStockSymbol());
    assertEquals('P', market4.getStockSymbol());
  }

  /**
   * Test the getSellOrder() method
   */
  @Test
  public void testGetSellOrders() {
    assertEquals(sellList, market1.getSellOrders());
    assertEquals(sellNode, market2.getSellOrders());
    assertEquals(sellList, market3.getSellOrders());
    assertEquals(sellNode, market4.getSellOrders());
  }

  /**
   * Test the getBuyOrders() method
   */
  @Test
  public void testGetBuyOrders() {
    assertEquals(buyList, market1.getBuyOrders());
    assertEquals(buyList, market2.getBuyOrders());
    assertEquals(buyNode, market3.getBuyOrders());
    assertEquals(buyNode, market4.getBuyOrders());
```

```
}

/**
 * Test the getOpenOrders() method
 */
@Test
public void testGetOpenOrders() {
  //
  LLNode<Order> b = market1.getOpenOrders(trader2);
  int ok = 1;
  while(b != null) {
    if (b.getElement().getTrader().equals(trader2)) {
      ;
    }
    else {
      ok = 0;
      break;
    }
    b = b.getNext();
  }
  assertEquals(ok, 1);
  //
  b = market1.getOpenOrders(trader1);
  ok = 1;
  while(b != null) {
    if (b.getElement().getTrader().equals(trader1)) {
      ;
    }
    else {
      ok = 0;
      break;
    }
    b = b.getNext();
  }
  assertEquals(ok, 1);
  //
  b = market1.getOpenOrders(trader3);
  ok = 1;
  while(b != null) {
    if (b.getElement().getTrader().equals(trader3)) {
      ;
    }
    else {
      ok = 0;
      break;
    }
    b = b.getNext();
  }
  assertEquals(ok, 1);
  //
  b = market1.getOpenOrders(trader4);
  ok = 1;
  while(b != null) {
    if (b.getElement().getTrader().equals(trader4)) {
      ;
    }
    else {
      ok = 0;
      break;
```

```
        }
        b = b.getNext();
      }
      assertEquals(ok, 1);
    }

    /**
     * Test the getCurrentBuyPrice() method
     * Because all 4 markets works in the same way, just need to test market 1
     */
    @Test
    public void testGetCurrentBuyPrice() {
      assertEquals(253.0, market1.getCurrentBuyPrice(), 0.1);
    }

    /**
     * Test the getCurrentSellBuyPrice() method
     * Because all 4 markets works in the same way, just need to test market 1
     */
    @Test
    public void testGetCurrentSellPrice() {
      assertEquals(100.0, market1.getCurrentSellPrice(), 0.1);
    }

    /**
     * Test the isOpen() method
     */
    @Test
    public void testIsOpen() {
      Market market5 = new Market('S', null, null);
      Market market6 = new Market('T', buyList, null);
      Market market7 = new Market('W', null, sellNode);
      assertTrue(market1.isOpen());
      assertTrue(market2.isOpen());
      assertTrue(market3.isOpen());
      assertTrue(market4.isOpen());
      assertFalse(market5.isOpen());
      assertFalse(market6.isOpen());
      assertFalse(market7.isOpen());
    }

    /**
     * Test the isValidOrder() method
     */
    @Test
    public void testIsValidOrder() {
      //Test in market 1
      assertTrue(market1.isValidOrder(buyNode.getElement()));
      assertFalse(market1.isValidOrder(buyList.getElement()));
      assertFalse(market1.isValidOrder(sellList.getElement()));
      assertFalse(market1.isValidOrder(sellNode.getElement()));

      //Test in market 2
      assertFalse(market2.isValidOrder(buyNode.getElement()));
      assertFalse(market2.isValidOrder(buyList.getElement())); //The price is not
  match
      assertFalse(market2.isValidOrder(sellList.getElement()));
      assertFalse(market2.isValidOrder(sellNode.getElement()));
      //Test in market 3
      assertFalse(market3.isValidOrder(buyNode.getElement()));
      assertFalse(market3.isValidOrder(buyList.getElement()));
      assertFalse(market3.isValidOrder(sellList.getElement()));
      assertTrue(market3.isValidOrder(sellNode.getElement()));

      //Test in market 4
      assertFalse(market4.isValidOrder(buyNode.getElement()));
      assertFalse(market4.isValidOrder(buyList.getElement()));
      assertTrue(market4.isValidOrder(sellList.getElement()));
      assertFalse(market4.isValidOrder(sellNode.getElement()));
    }

    /**
     * Test the addOrder() method
     * Because all 4 markets works in the same way, just need to test market 1
     */
    @Test
    public void testAddOrder() {
      //add order on market1
      Order order1 = new BuyLimitOrder('M', 325, 12,true, true, trader1);
      Order order2 = new SellLimitOrder('M', 214,45, true, true, trader2);
      Order order3 = new MarketBuyOrder('M', 324,53, trader3);
      Order order4 = new MarketSellOrder('M', 6546,134, trader4);
      market1.addOrder(order1);
      market1.addOrder(order2);
      market1.addOrder(order3);
      market1.addOrder(order4);

      boolean thrown = false;
      try {
        //exception on market 1
        market1.addOrder(new BuyLimitOrder('A', 325, 12,true, true, trader1));
        market1.addOrder(new SellLimitOrder('B', 214,45, true, true, trader2));
        market1.addOrder(new MarketBuyOrder('C', 324,53, trader3));
        market1.addOrder(new MarketSellOrder('D', 6546,134, trader4));
      }
      catch(NoSuchElementException e) {
        thrown = true;
      }

      //Test on market 1.
      assertEquals(buyList.getElement(), market1.getBuyOrders().getElement());
      assertEquals(buyNode.getElement(),
  market1.getBuyOrders().getNext().getElement());
      assertEquals(order3, market1.getBuyOrders().getNext().getNext().getElement());
      assertEquals(order1,
  market1.getBuyOrders().getNext().getNext().getNext().getElement());
      assertEquals(order2, market1.getSellOrders().getElement());
      assertEquals(sellList.getElement(),
  market1.getSellOrders().getNext().getElement());
      assertEquals(order4, market1.getSellOrders().getNext().getNext().getElement());
      assertEquals(sellNode.getElement(),
  market1.getSellOrders().getNext().getNext().getNext().getElement());
      assertTrue(thrown);
    }

    /**
     * Test the removeOrder() method
```

```
     * Because all 4 markets works in the same way, just need to test market 1
     */
    @Test
    public void testRemoveOrder() {
      //add order on market1
      Order order1 = new BuyLimitOrder('M', 325, 12,true, true, trader1);
      Order order2 = new SellLimitOrder('M', 214,45, true, true, trader2);
      Order order3 = new MarketBuyOrder('M', 324,53, trader3);
      Order order4 = new MarketSellOrder('M', 6546,134, trader4);
      market1.removeOrder(order1); //Test last
      market1.removeOrder(order2); //Test first
      market1.removeOrder(order3); //Test middle
      market1.removeOrder(order4); //Test middle
      //BuyOrders
      assertEquals(buyList.getElement(), market1.getBuyOrders().getElement());
      assertEquals(buyNode.getElement(),
market1.getBuyOrders().getNext().getElement());
      //SellOrders
      assertEquals(sellList.getElement(), market1.getSellOrders().getElement());
      assertEquals(sellNode.getElement(),
market1.getSellOrders().getNext().getElement());
    }

    /**
     * Test the matchingOrders() method
     * Because all 4 markets works in the same way, just need to test market 1
     */
    @Test
    public void testMatchingOrders() {
      assertTrue(market1.matchingOrders(buyList.getElement(),sellList.getElement()));
      assertTrue(market1.matchingOrders(buyList.getElement(),
sellNode.getElement()));
      assertFalse(market1.matchingOrders(buyNode.getElement(),
sellList.getElement()));
      assertFalse(market1.matchingOrders(buyNode.getElement(),
sellNode.getElement()));
    }

    /**
     * Test the placeOrder() method
     */
    @Test
    public void testPlaceOrder() {
      Order inputOrder1 = new BuyLimitOrder('M', 123, 68, true,true, trader4);
      Order inputOrder2 = new SellLimitOrder('M',100, 100, true, true, trader3);
      Order inputOrder3 = new MarketBuyOrder('M', 210, 192, trader2);
      Order inputOrder4 = new MarketSellOrder('M', 45, 111, trader1);
      LLNode<Transaction> transaction1 = new LLNode<Transaction>(new Transaction('M',
45, 111, trader1, trader1,0),null);
      LLNode<Transaction> transaction2 = new LLNode<Transaction>(new Transaction('M',
75, 100, trader1, trader3,1),null);
      boolean thrown = true;
      //Test the exception
      try {
        market1.placeOrder(buyList.getElement());
        market1.placeOrder(sellList.getElement());
        market1.placeOrder(sellNode.getElement());
        market1.placeOrder(inputOrder3).getElement();
      }
      catch(NoSuchElementException e) {
        thrown = true;
      }
      //Test if there's no match found
      assertEquals(null, market1.placeOrder(inputOrder1));
      assertEquals(null, market1.placeOrder(buyNode.getElement()));

      //Test if there's match found
      assertEquals(transaction1.getElement(),
  market1.placeOrder(inputOrder4).getElement());
      assertEquals(transaction2.getElement(),
  market1.placeOrder(inputOrder2).getElement());
    }

    /**
     * Test the closeMarket() method
     */
    @Test
    public void testClostMarket() {
      market1.closeMarket();
      assertEquals(null, market1.getBuyOrders());
      assertEquals(null, market1.getSellOrders());
    }

    /**
     * Test the toString() method
     */
    @Test
    public void testToString() {
      assertEquals("The market symbol: M", market1.toString());
      assertEquals("The market symbol: N", market2.toString());
      assertEquals("The market symbol: O", market3.toString());
      assertEquals("The market symbol: P", market4.toString());
    }

    /**
     * Test the equals() method
     */
    @Test
    public void testEquals() {
     Market market5 = new Market('M', buyList, sellList);
     assertTrue(market1.equals(market5));
     assertFalse(market2.equals(market5));
     assertFalse(market3.equals(market5));
     assertFalse(market4.equals(market5));
    }
  }
```