```
/**
 * @author Harley Phung
 * Create a BuyLimitOrder for trader to start transaction
 */

public class BuyLimitOrder extends LimitOrder implements BuyOrder{

    /** The stock symbol of the limit order */
    private char stockSymbol = ' ';

    /** The number of shares the investor wants to buy in the limit order */
    private int numShares = 0;

    /** The price per share the investor wants to pay in the limit order */
    private double numPrice = 0.0;

    /** @param allIn  determine if the investor have to trade all in the limit
order */
    private boolean allIn = true;

    /** @param dayOrder  determine if the trade is happening when market is open */

    private boolean dayOrder = true;

    /** The instance of Trader class */
    private Trader buyTrader = null;

    /**
     * Constructor that take investor's name, Stock Symbol, number of shares, price
per share, all or none,
     * dayOrder, and trader instance as input
     * @param stockSymbol  the stock symbol of the buy limit order
     * @param numShares  the number of shares this buy limit order requested to buy
     * @param numPrice  the price per share this buy limit order requested to buy
     * @param allIn  determine if the trader must buy all the stock or not
     * @param dayOrder  determine if the order is processed in day
     * @param buyTrader  determine that the this is a trader who buy stock
     */
    public BuyLimitOrder (char stockSymbol, int numShares, double numPrice, boolean
allIn, boolean dayOrder, Trader buyTrader) {
        this.stockSymbol = stockSymbol;
        this.numShares = numShares;
        this.numPrice = numPrice;
        this.allIn = allIn;
        this.dayOrder = dayOrder;
        this.buyTrader = buyTrader;
    }

    /**
     * Returns the stock symbol of the order
     * @return stockSymbol  the symbol of buy limit order
     */
    @Override
    public char getStockSymbol() {
        return this.stockSymbol;
    }

    /**
     * Returns the number of shares investor wants to buy
```

```
     * @return numShares  the number of shares the investor requested to buy
     */
    @Override
    public int getNumberShares() {
        return this.numShares;
    }

    /**
     * Changes the number of shares requested in the buy limit order
     * @param numShares  new number of shares requested in the buy limit order
     */
    @Override
    public void setNumberShares(int numShares) {
        this.numShares = numShares;
    }

    /**
     * Returns the price per share requested in the buy limit order
     * @return numPrice  the price per share requested in the buy limit order
     */
    @Override
    public double getPrice() {
        return this.numPrice;
    }

    /**
     * Determine whether trade all the shares of the order or not
     * @return allIn  determine if the trader have to buy all the stock or not.
true means yes, false means no
     */
    @Override
    public boolean isAllOrNone() {
        return this.allIn;
    }

    /**
     * Determines if the transaction is in day order
     * @return dayOrder  determine if the transaction is in day order. true means
yes, false means not in day order.
     */
    @Override
    public boolean isDayOrder() {
        return this.dayOrder;
    }

    /**
     * Returns information of the trader
     * @return buyTrader  the information of the trader
     */
    @Override
    public Trader getTrader() {
        if(this.buyTrader instanceof Trader) {
            return this.buyTrader;
        }
        else {
            return null;
        }
    }
```

```
    /**
     * An override toString method format the returned String
     */
    @Override
    public String toString() {
        return this.getStockSymbol() + ", " + this.getNumberShares()
            + ", " + this.getPrice() + ", " + this.isAllOrNone() + ", " +
this.isDayOrder()
            + ", " + this.getTrader();
    }

    /**
     * An override equals method that compared the two trader's information.
     * @param p  compare the trader
     * @return true  if there's identical buy limit order
     * @return false  if there's no identical buy limit order
     */
    @Override
    public boolean equals(Object p){
        if(p instanceof BuyLimitOrder) {
            BuyLimitOrder newBuyOrder = (BuyLimitOrder)p;
            if(this.getStockSymbol() == newBuyOrder.getStockSymbol()
              && this.getNumberShares() == newBuyOrder.getNumberShares()
              && this.getPrice() == newBuyOrder.getPrice()
              && this.getTrader() == newBuyOrder.getTrader()){
                return true;
            }
        }
        return false;
    }
}
```