```
/* *********************************
 *   BuyAgent.java
 *     Handles the buying of Derivatives.
 * *********************************/
package player.playeragent;

import player.*;
import edu.neu.ccs.demeterf.demfgen.lib.List;
import gen.*;

/** Class for buying a derivative */
public class BuyAgent implements PlayerI.BuyAgentI{

    /** Returns the profitable derivatives from those on sale */
    public List<Derivative> buyDerivatives(List<Derivative> forSale, double account){
        return forSale.foldl(new List.Fold<Derivative, BuyChoice>(){
            public BuyChoice fold(Derivative d, BuyChoice choice){
                // Should this one be bought??
                if (shouldBuy(d, choice.account))
                    return choice.buy(d);

                // I guess not...
                return choice;
            }}, new BuyChoice(account)).toBuy;
    }

    /** Is the given Derivative profitable? Do I have enough Money? */
    private boolean shouldBuy(Derivative der, double account){
        return der.price.val < IAmRobot.getBreakEven(der) && der.price.val < account;
    }

    /** Collects buying decisions based on the shouldBuy "predicate" */
    private class BuyChoice{
        double account;
        List<Derivative> toBuy;
        BuyChoice(double acc){ this(acc, List.<Derivative>create()); }
        BuyChoice(double acc, List<Derivative> buy){ account = acc; toBuy = buy; }
        BuyChoice buy(Derivative der){
            return new BuyChoice(account-der.price.val, toBuy.push(der));
        }
    }
}
```