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

import java.util.PriorityQueue;

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){
    PriorityQueue<Profit> goodDeals = new PriorityQueue<Profit>();
    List<Derivative> result = List.create();
    for (Derivative d : forSale) {
      double profit = Util.breakEvenValue(d.type) - d.price.val;
      if (profit > 0) {
        goodDeals.add(new Profit(profit, d));
      }
    }
    // since this is a priority queue, we will iterate through the derivatives
    // in order from most to least profit
    while (!goodDeals.isEmpty()) {
      Derivative d = goodDeals.remove().getDerivative();
      if (d.price.val < account) {
        result = result.append(d);
        account -= d.price.val;
      }
    }
    return result;
  }

  private static class Profit implements Comparable {

    private final double profit;
    private final Derivative derivative;

    public Profit(final double profit, final Derivative derivative) {
      super();
      this.profit = profit;
      this.derivative = derivative;
    }

    public Derivative getDerivative() {
      return derivative;
    }

    public double getProfit() {
      return profit;
    }

    public int compareTo(Object other) {
      return Double.compare(((Profit) other).profit, profit);
    }

  }
}
```