```
/* *********************************
 *   DeliverAgent.java
 *     Delivers Raw Materials
 * *********************************/
package player.playeragent;

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

/** Class for delivering raw material for a derivative */
public class DeliverAgent implements PlayerI.DeliverAgentI{

  public static int NUM_VARIABLES = 6;

    /** Adds raw materials to the given derivatives */
    public Derivative deliverRawMaterial(Derivative needRM){
        return needRM.deliver(rawMaterialInst(needRM));
    }

    /** Compute a RawMaterial Instance for the given Derivative */
    private RawMaterialInstance rawMaterialInst(Derivative d){
      List<Variable> variables = List.create();
      for (int i = 0; i < NUM_VARIABLES; ++i) {
        variables = variables.append(new Variable(new ident("v" + i)));
      }


      /* TODO: write a general function to do this rather than manaully
       * list possibilities for 4 variables with 3 constraints per variable
       */
      return new RawMaterialInstance(constraints(d.type, variables));
    }

    private static List<Constraint> constraints(Type t, List<Variable> vs) {
      if (vs.length() == 3) {
        return constraints(t, vs.lookup(0), vs.lookup(1), vs.lookup(2));
      } else {
        List<Constraint> result = List.<Constraint>create();
        for (Variable v : vs) {
          result = result.append(constraints(t, vs.remove(v)));
        }
        return result;
      }
    }

    private static List<Constraint> constraints(
        Type t,
        Variable v1,
        Variable v2,
        Variable v3) {
      return List.create(
          makeConstraint(t, v1, v2, v3),
          makeConstraint(t, v1, v3, v2),
          makeConstraint(t, v3, v2, v1),
          makeConstraint(t, v2, v1, v3),
          makeConstraint(t, v2, v3, v1),
          makeConstraint(t, v3, v1, v2)
      );
    }

    private static Constraint makeConstraint(Type t, Variable... s){
      return new Constraint(new Weight(1), new RelationNr(Util.relationNumber(t)),List.<Variable>create
(s));
    }

}
```