```
/* *********************************
 *   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{

    /** 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 der)
    {
        List<TypeInstance> relations = der.type.instances;
        Weight w = new Weight(1);
        List<Constraint> constraints = List.<Constraint>create();

        Variable[] allvars = new Variable[23];
        char start = 'a';

        // Generate a list of variables to use in creating combinations
        for (int i = 0; i < allvars.length; i++) {
                allvars[i] = new Variable(new ident(String.valueOf((char)(start + i))));
        }

        int[] indices;
        CombinationGenerator comb = new CombinationGenerator(allvars.length, 3);

        int count = 1;
        List<Variable> somevars;
        // Use the combination generator to create a list of constraints
        while (comb.hasMore()) {
                indices = comb.getNext();
                somevars = List.<Variable>create();

                for (int i = 0; i < indices.length; i++) {
                        somevars = somevars.push(allvars[indices[i]]);
                }

                int index = Util.random(relations.length());
                RelationNr r = relations.lookup(index).r;
                Constraint con = new Constraint(w, r, somevars);
                constraints = constraints.push(con);
                if(count == 1)
                {
                        System.out.println(con.display());
                        count++;
                }
        }

        RawMaterialInstance rm = new RawMaterialInstance(constraints);

        System.out.println("<b>Delivering</b>: Raw material with " + constraints.length() + " constraints.");
        System.out.println("  to " + der.optbuyer);

        if (der.isClassic()) {
                return rm;
        } else {
                FinishAgent finishAgent = new FinishAgent();
                FinishedProduct fp = finishAgent.finishDerivative(der.deliver(rm));
                Assignment assign = fp.ip.assignment;
                double qual = fp.quality.val;
                rm = rm.addSecret(assign);
                rm = rm.addSecretQuality(qual);

                return rm;
        }
    }

}
```