```
/* *********************************
 *   FinishAgent.java
 *     Finish a given Raw Material
 * *********************************/
package player.playeragent;

import player.*;
import edu.neu.ccs.demeterf.demfgen.lib.*;
import edu.neu.ccs.evergreen.ir.Relation;
import gen.*;

/** Class for finishing a list of derivatives */
public class FinishAgent implements PlayerI.FinishAgentI{

    /** Calculate the finished product for a given Derivative */
    public FinishedProduct finishDerivative(Derivative d){
        List<Variable> vars = List.create();

        //For every constraint add that constraint's variables to the list of variables
        for (Constraint c : d.optraw.inner().instance.cs) {
                for (Variable var : c.vs) {
                        if (!vars.contains(var))
                                vars = vars.append(var);
                }
        }

        double bias = IAmRobot.getBMax(d);

        Assignment asgn = generateAssignment(vars, bias);
        double quality = getQuality(d, asgn);

        for (int i=0; i<1000; i++) {
                Assignment asgnTemp = generateAssignment(vars, bias);
                double tempQuality = getQuality(d, asgnTemp);
                if (tempQuality > quality) {
                        asgn = asgnTemp;
                        quality = tempQuality;
                }
        }

        return new FinishedProduct(new IntermediateProduct(asgn),
                                           new Quality(quality));
    }

    /** Generate a random assignment based on the flip of a biased coin */
    Assignment generateAssignment(List<Variable> vars, double bias) {
        List<Literal> lits = List.create();

        //Assign a value to each variable and add it to the list of literals
        for (Variable var : vars) {
                Sign sign;
                if(Util.coinFlip(bias))
                        sign = new Pos();
                else
                        sign = new Neg();
                lits = lits.append(new Literal(sign, var));
        }

        return new Assignment(lits);
    }

    /** Calculate how well the given assignment satisfies the given derivative */
    double getQuality(Derivative d, Assignment asgn) {
        //Reduce the derivative with each of the variable's values
        int relationNum = d.type.instances.top().r.v;
        RawMaterialInstance rawMat = d.optraw.inner().instance;
        int satisfied = 0;

        for (Constraint c : rawMat.cs) {
                Relation currentRel = new Relation(3, relationNum);

                for (int i = 0; i < 3; i++) {
                        int value = 0;
                        for (Literal lit : asgn.literals) {
                                if (lit.var.v.equals(c.vs.lookup(i).v)) {
                                        value = lit.value.sign();
                                        break;
                                }
                        }

                        currentRel.setRelationNumber(currentRel.reduce(i, value));
```

```
                        if (currentRel.getRelationNumber() == 0 ||
                                currentRel.getRelationNumber() == 255)
                                break;
                }
                if (currentRel.getRelationNumber() == 255)
                        satisfied++;
        }

        int numConstraints = rawMat.cs.length();
        if (numConstraints == 0)        // Should not occur if robots are following the rules
                return 1;

        return satisfied / (double) numConstraints;
    }
}
```