```
//
//ORIGINAL SAMPLE EXAMPLE FROM LECTURE NOTES
//
Engine engine = new Engine();
engine.setName("CartEngine");

InputVariable inputVariable1 = new InputVariable();
inputVariable1.setEnabled(true);
inputVariable1.setName("angle");
inputVariable1.setRange(-4.000, 4.000);
inputVariable1.addTerm(new Trapezoid("NH", -4.000, -4.000, -2.000, -1.000));
inputVariable1.addTerm(new Triangle("NL", -2.000, -1.000, 0.000));
inputVariable1.addTerm(new Triangle("Z", -1.000, 0.000, 1.000));
inputVariable1.addTerm(new Triangle("PL", 0.000, 1.000, 2.000));
inputVariable1.addTerm(new Trapezoid("PH", 1.000, 2.000, 4.000, 4.000));
engine.addInputVariable(inputVariable1);

InputVariable inputVariable2 = new InputVariable();
inputVariable2.setEnabled(true);
inputVariable2.setName("angular_velocity");
inputVariable2.setRange(-8.000, 8.000);
inputVariable2.addTerm(new Trapezoid("NH", -8.000, -8.000, -7.000, -3.000));
inputVariable2.addTerm(new Triangle("NL", -6.000, -3.000, 0.000));
inputVariable2.addTerm(new Triangle("Z", -1.000, 0.000, 1.000));
inputVariable2.addTerm(new Triangle("PL", 0.000, 3.000, 6.000));
inputVariable2.addTerm(new Trapezoid("PH", 3.000, 7.000, 8.000, 8.000));
engine.addInputVariable(inputVariable2);

OutputVariable outputVariable = new OutputVariable();
outputVariable.setEnabled(true);
outputVariable.setName("force");
outputVariable.setRange(-32.000, 32.000);
outputVariable.fuzzyOutput().setAccumulation(new Maximum());
outputVariable.setDefuzzifier(new Centroid(200));
outputVariable.setDefaultValue(Double.NaN);
outputVariable.setLockValidOutput(false);
outputVariable.setLockOutputRange(false);
outputVariable.addTerm(new Trapezoid("NH", -32.000, -32.000, -20.000, -8.000));
outputVariable.addTerm(new Triangle("NL", -16.000, -8.000, 0.000));
outputVariable.addTerm(new Triangle("Z", -8.000, 0.000, 8.000));
outputVariable.addTerm(new Triangle("PL", 0.000, 8.000, 16.000));
outputVariable.addTerm(new Trapezoid("PH", 8.000, 20.000, 36.000, 36.000));
engine.addOutputVariable(outputVariable);

RuleBlock ruleBlock = new RuleBlock();
ruleBlock.setEnabled(true);
ruleBlock.setName("");
ruleBlock.setConjunction(new Minimum());
ruleBlock.setDisjunction(new Maximum());
ruleBlock.setActivation(new Minimum());
ruleBlock.addRule(Rule.parse("if (angle is Z) and (angular_velocity is Z) then force is Z",
engine));
```

```
ruleBlock.addRule(Rule.parse("if angle is Z and angular_velocity is NH then force is NH",
engine));
ruleBlock.addRule(Rule.parse("if angle is Z and angular_velocity is NL then force is NL",
engine));
ruleBlock.addRule(Rule.parse("if angle is Z and angular_velocity is PL then force is PL",
engine));
ruleBlock.addRule(Rule.parse("if angle is Z and angular_velocity is PH then force is PH",
engine));
ruleBlock.addRule(Rule.parse("if angle is NH and angular_velocity is Z then force is NH",
engine));
ruleBlock.addRule(Rule.parse("if angle is NL and angular_velocity is Z then force is NL",
engine));
ruleBlock.addRule(Rule.parse("if angle is PL and angular_velocity is Z then force is PL",
engine));
ruleBlock.addRule(Rule.parse("if angle is PH and angular_velocity is Z then force is PH",
engine));
ruleBlock.addRule(Rule.parse("if angle is NL and angular_velocity is PL then force is Z",
engine));
ruleBlock.addRule(Rule.parse("if angle is PL and angular_velocity is NL then force is Z",
engine));
engine.addRuleBlock(ruleBlock);


//
//B00060572 DAVID KELLY VERSION IMPLEMENTATION
//
Engine engine = new Engine();
engine.setName("Cart");

InputVariable inputVariable1 = new InputVariable();
inputVariable1.setEnabled(true);
inputVariable1.setName("angle");
inputVariable1.setRange(-4.000, 4.000);
inputVariable1.addTerm(new Trapezoid("N", -4.000, -4.000, -1.000, 0.000));
inputVariable1.addTerm(new Triangle("Z", -2.000, 0.000, 2.000));
inputVariable1.addTerm(new Trapezoid("P", 0.000, 1.000, 4.000, 4.000));
engine.addInputVariable(inputVariable1);

InputVariable inputVariable2 = new InputVariable();
inputVariable2.setEnabled(true);
inputVariable2.setName("angular_velocity");
inputVariable2.setRange(-8.000, 8.000);
inputVariable2.addTerm(new Trapezoid("N", -8.000, -8.000, -3.000, -2.000));
inputVariable2.addTerm(new Trapezoid("Z", -3.000, -2.000, 2.000, 3.000));
inputVariable2.addTerm(new Trapezoid("P", 2.000, 3.000, 8.000, 8.000));
engine.addInputVariable(inputVariable2);

OutputVariable outputVariable = new OutputVariable();
outputVariable.setEnabled(true);
outputVariable.setName("force");
outputVariable.setRange(-32.000, 32.000);
outputVariable.fuzzyOutput().setAccumulation(new Maximum());
```

```
outputVariable.setDefuzzifier(new Centroid(200));
outputVariable.setDefaultValue(Double.NaN);
outputVariable.setLockValidOutput(false);
outputVariable.setLockOutputRange(false);
outputVariable.addTerm(new Trapezoid("N", -32.000, -32.000, -5.000, -1.000));
outputVariable.addTerm(new Trapezoid("Z", -3.000, -1.000, 1.000, 3.000));
outputVariable.addTerm(new Trapezoid("P", 1.000, 5.000, 32.000, 32.000));
engine.addOutputVariable(outputVariable);

RuleBlock ruleBlock = new RuleBlock();
ruleBlock.setEnabled(true);
ruleBlock.setName("");
ruleBlock.setConjunction(new Minimum());
ruleBlock.setDisjunction(new Maximum());
ruleBlock.setActivation(new Minimum());
ruleBlock.addRule(Rule.parse("if angle is N and angular_velocity is N then force is N",
engine));
ruleBlock.addRule(Rule.parse("if angle is Z and angular_velocity is N then force is N",
engine));
ruleBlock.addRule(Rule.parse("if angle is P and angular_velocity is N then force is Z",
engine));
ruleBlock.addRule(Rule.parse("if angle is N and angular_velocity is Z then force is N",
engine));
ruleBlock.addRule(Rule.parse("if angle is Z and angular_velocity is Z then force is Z",
engine));
ruleBlock.addRule(Rule.parse("if angle is P and angular_velocity is Z then force is P",
engine));
ruleBlock.addRule(Rule.parse("if angle is N and angular_velocity is P then force is Z",
engine));
ruleBlock.addRule(Rule.parse("if angle is Z and angular_velocity is P then force is P",
engine));
ruleBlock.addRule(Rule.parse("if angle is P and angular_velocity is P then force is P",
engine));
engine.addRuleBlock(ruleBlock);
```