**Lab Exercise 14- Drool Attribute (Implementing Delayed Rule Execution with the duration Attribute)**

**Objective:**

In this exercise, you will create rules that simulate a delayed execution using the duration attribute in Drools. This attribute allows you to specify a delay before a rule is executed after its conditions have been met.

Scenario: You are developing a system to manage alerts for a temperature monitoring system. The system should send an alert if the temperature remains above a certain threshold for a specified period (e.g., 5 seconds).

**Project Structure**

/src

/main

/java

/com/example

TemperatureSensor.java

TemperatureMonitoringApp.java

/resources

/com/example/rules

temperatureMonitoring.drl

1. **TemperatureSensor.java**

package com.example;

public class TemperatureSensor {

private String sensorId;

private double temperature;

// Constructor

public TemperatureSensor(String sensorId, double temperature) {

this.sensorId = sensorId;

this.temperature = temperature;

}

// Getters and Setters

public String getSensorId() {

return sensorId;

}

public void setSensorId(String sensorId) {

this.sensorId = sensorId;

}

public double getTemperature() {

return temperature;

}

public void setTemperature(double temperature) {

this.temperature = temperature;

}

}

**2. TemperatureMonitoringApp.java**

package com.example;

import org.kie.api.KieServices;

import org.kie.api.runtime.KieContainer;

import org.kie.api.runtime.KieSession;

public class TemperatureMonitoringApp {

public static void main(String[] args) throws InterruptedException {

// Load the knowledge base

KieServices ks = KieServices.Factory.get();

KieContainer kContainer = ks.getKieClasspathContainer();

KieSession kSession = kContainer.newKieSession("ksession-rules");

// Create a TemperatureSensor object



TemperatureSensor sensor = new TemperatureSensor("Sensor-1", 80.0);

// Insert the object into the session

kSession.insert(sensor);



// Fire all rules

kSession.fireAllRules();



// Simulate a delay to observe the duration effect

Thread.sleep(6000);

// Fire all rules again to process any delayed rules

kSession.fireAllRules();



// Dispose of the session

kSession.dispose();

}

}

**temperatureMonitoring.drl**

package com.example.rules;

import com.example.TemperatureSensor;

rule "Temperature Alert"

when

$sensor : TemperatureSensor(temperature > 75)

then

System.out.println("ALERT: High temperature detected by sensor " + $sensor.getSensorId()

+ " with temperature " + $sensor.getTemperature());

end

ule "Delayed Temperature Alert"



duration(5s)

when



$sensor : TemperatureSensor(temperature > 75)

then

System.out.println("ALERT: Temperature has been high for over 5 seconds! Sensor: "

+ $sensor.getSensorId() + " Temperature: " + $sensor.getTemperature());

end

**Explanation:**

1. **First rule ("Temperature Alert")**: Fires immediately when the temperature exceeds 75.
2. **Second rule ("Delayed Temperature Alert")**: Fires after a delay of 5 seconds (duration(5s)) if the temperature has been consistently above 75. This delay simulates the condition where the alert is only triggered if the temperature remains high for a prolonged period.

**Expected Output:**

When you run the program, you should observe that the alert is not printed immediately after inserting the TemperatureSensor object. Instead, it appears in the console after the specified duration (5 seconds in this case).