**Lab Exercise 2- Exploring LHS (When) Syntax in Drools**

**Objective:**

* To create and experiment with various LHS conditions in Drools rules, understanding how to match facts, use logical operators, and bind variables.

**Prerequisites:**

* A basic understanding of Drools and Java.
* Completion of the previous lab exercise on creating a simple Drools project.
* The Maven project setup from the previous lab.

**Step 1: Set Up the Project**

* If you have the project set up from the previous exercise, you can use the same project. If not, follow the steps in the previous lab exercise to set up a basic Drools project.

**Step 2: Define a New Java Model**

**Create a Product Class:**

In the src/main/java/com/example/model directory, create a Product.java file with the following content:

package com.example.model;

public class Product {

private String type;

private int price;

private boolean available;

public Product(String type, int price, boolean available) {

this.type = type;

this.price = price;

this.available = available;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public boolean isAvailable() {

return available;

}

public void setAvailable(boolean available) {

this.available = available;

}

@Override

public String toString() {

return "Product{type='" + type + "', price=" + price + ", available=" + available + '}';

}

}

**Step 3: Create DRL Rules to Explore LHS Syntax**

Create the Rule File:

In the src/main/resources directory, create a file named productRules.drl with the following content:

package com.example.rules

rule "Discount on Electronics"

when

$product : Product(type == "Electronics", price > 1000)

then

$product.setPrice($product.getPrice() - 100);

System.out.println("Applied discount on Electronics: " + $product);

end

rule "Check Availability"

when

$product : Product(available == false)

then

System.out.println("Product not available: " + $product);

end

rule "Promote Cheap Products"

when

$product : Product(price < 500, available == true)

then

System.out.println("Promoting cheap product: " + $product);

end

rule "Bind Variables"

when

$product : Product($type : type, $price : price)

eval($price > 500 && $type == "Furniture")

then

System.out.println("Furniture item eligible for premium promotion: " + $product);

end

**Explanation of the Rules:**

* Discount on Electronics: Applies a discount of 100 if the product type is "Electronics" and the price is greater than 1000.
* Check Availability: Prints a message if the product is not available.
* Promote Cheap Products: Promotes products that are priced below 500 and are available.
* Bind Variables: Demonstrates binding variables from the LHS and using them in conditions.

**Step 4: Update the Main Application**

* Modify the DroolsTest Class:
* Update the DroolsTest.java file to include the new Product class:

package com.example;

import com.example.model.Product;

import org.kie.api.KieServices;

import org.kie.api.runtime.KieContainer;

import org.kie.api.runtime.KieSession;

public class DroolsTest {

public static void main(String[] args) {

// Load the knowledge base

KieServices ks = KieServices.Factory.get();

KieContainer kContainer = ks.getKieClasspathContainer();

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

// Create sample products

Product product1 = new Product("Electronics", 1200, true);

Product product2 = new Product("Clothing", 800, false);

Product product3 = new Product("Furniture", 700, true);

Product product4 = new Product("Grocery", 300, true);

// Insert the products into the session

kSession.insert(product1);

kSession.insert(product2);

kSession.insert(product3);

kSession.insert(product4);

// Fire all rules

kSession.fireAllRules();

// Dispose the session

kSession.dispose();

}

}

**Step 5: Run the Application**

**Compile and Run:**

* Compile and run the DroolsTest class as described in the previous lab exercise.

**Expected Output:**

The console should display output similar to the following:

Applied discount on Electronics: Product{type='Electronics', price=1100, available=true}

Product not available: Product{type='Clothing', price=800, available=false}

Promoting cheap product: Product{type='Grocery', price=300, available=true}

Furniture item eligible for premium promotion: Product{type='Furniture', price=700, available=true}