|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| https://lh6.googleusercontent.com/QcftzNtI05T0Y6fjdSh1Rr2rt8oqZ1IvnLvbn1jLJ7CCyteVir3k-xBLv4SL1wAgWJsRhmmJSR0UW-RP63_GQenE4vVWv05BRoZTsmIcBccVTnfxwmsnNMvjg599x9SqZd8E3dkd   |  |  |  | | --- | --- | --- | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | |  | | | |  |
|  |  |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИППО)**

|  |  |
| --- | --- |
| **ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2** | |
| **по дисциплине** | |
| «Программное обеспечение интеллектуальных систем» | |
|  | |
| Выполнил студент группы ИКБО-02-17 | *Верба Д.С.* |
| Принял | *Волков М.Ю.* |

|  |  |  |
| --- | --- | --- |
| Работа выполнена | «\_\_»\_\_\_\_\_\_\_2020 г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2020 г. | *(подпись руководителя)* |

Москва 2020

**Цель:** Научиться составлять бизнес правила в Drools и применять их в Java приложениях.

**Задание:**

1. Установить drools.
2. Создать несколько бизнес правил.
3. Продемонстрировать их работу.

**Ход работы**

Было принято решение использовать встроенный в Intellij Idea drools плагин.

В качестве темы была выбрана «рассчитать стоимость покупки с учетом скидки в зависимости от условий». В качестве области применения был выбран магазин продукции Apple.

Сначала необходимо создать классы, которые будут представлять бизнес объекты:

Класс Customer представляет покупателей, у которых есть такие свойства, как money, отвечающее за определение платежеспособности покупателя, и свойство curDevice, на основе которого будет определяться наличие скидки на покупку нового девайса.

Класс Device представляет собой устройство, которое имеется в наличии в магазине. При совпадении определенных условий устройством пользователя, мы высчитываем скидку и выводим ее в правилах. Для выполнения данной задачи в классе Device имеются поля brand, name, price.

Создадим вспомогательный класс Calcutils, который позволяет определить, различаются ли в цене старое и новое устройства.

Далее создадим несколько правил.

Сделаем скидку на покупку устройства при соблюдении следующих обстоятельств: пользователь хочет купить технику Apple, у него уже имеется техника Apple, при этом модель его старого устройства не совпадает с моделью устройства, которое он хочет купить. В данном случае мы делаем скидку, с учетом того, что покупатель может заменить старый девайс на новый. Правило выглядит следующим образом:

rule "Make apple discount for new model"

when

device: Device(brand == "Apple")

user: Customer(hasApple(),

hasOldToNewDiscount(device), money > 0)

then

System.out.println("You can change your old Apple device on New device with 30% discount");

end

Далее, создадим правило, которое позволяет сделать скидку, если соблюдаются следующие условия: у покупателя имеется устройство от компании, отличной от Apple, в магазине имеется устройство от Apple. Скидка делается только в том случае, если устройство покупателя выше цены устройства, имеющегося в магазине. Данное правило приведем ниже:

rule "Make device discount"

when

util: Calcutils()

device: Device(brand == "Apple")

user: Customer(!hasApple(), util.makeDiscountIfOldDeviceExpensive(curDevice, device), money > 0)

then

System.out.println("You can change your another brand on Iphone with 30%")

end

Приведем остальные правила, которые являются дополнением к вышеописанным.

Правило, позволяющее сделать скидку при покупке той же самой модели, что уже есть у покупателя:

rule "Make apple discount for same model"

when

device: Device(brand == "Apple")

user: Customer(hasApple(),

!hasOldToNewDiscount(device), money > 0)

then

System.out.println("You can change your old Apple device on New device with 50% discount");

end

Правило, которое устанавливает, что пользователь не может получить скидку на новое устройство

rule "Don't device discount"

when

util: Calcutils()

device: Device(brand == "Apple")

user: Customer(!hasApple(), !util.makeDiscountIfOldDeviceExpensive(curDevice, device), money > 0)

then

System.out.println("You can't change your another brand on Iphone")

end

Далее попробуем протестировать наши правила. Подключим все необходимые библиотеки и создадим сессии.

import org.drools.core.impl.InternalKnowledgeBase

import org.drools.core.impl.KnowledgeBaseFactory

import org.kie.api.io.ResourceType

import org.kie.internal.builder.KnowledgeBuilderFactory

import org.kie.internal.io.ResourceFactory

import java.io.File

fun main() {

val kbase = *readKnowledgeBase*(*listOf*(File("/Users/macbookpro152019/university/poisLabs/lab22/src/main/resources/rules/Devices.drl")))

val kSession = kbase.newKieSession()

val device = Device("Apple", "Iphone12", 1100)

val Dima = Customer(1000, curDevice = Device("Apple", "Iphone11", 1000))

val Fedor = Customer(1000, curDevice = Device("Apple", "Iphone12", 1100))

val Sasha = Customer(1000, curDevice = Device("Samsung", "S21", 1500))

val Vasya = Customer(1000, curDevice = Device("Samsung", "S10", 1000))

kSession.insert(Calcutils())

kSession.insert(Dima)

kSession.insert(device)

kSession.fireAllRules()

kSession.insert(Fedor)

kSession.insert(device)

kSession.fireAllRules()

kSession.insert(Sasha)

kSession.insert(device)

kSession.fireAllRules()

kSession.insert(Vasya)

kSession.insert(device)

kSession.fireAllRules()

}

fun readKnowledgeBase(files: List<File>): InternalKnowledgeBase {

val knowledgeBuilder = KnowledgeBuilderFactory.newKnowledgeBuilder()

files.*forEach* **{** i: File? **->** knowledgeBuilder.add(ResourceFactory.newFileResource(i), ResourceType.*DRL*) **}**

val errors = knowledgeBuilder.*errors*

if (!errors.isEmpty()) {

for (error in errors) {

*println*("{$error}")

}

throw IllegalArgumentException("Could not parse knowledge.")

}

val kBase = KnowledgeBaseFactory.newKnowledgeBase()

kBase.addPackages(knowledgeBuilder.*knowledgePackages*)

return kBase

}

Проверим работу наших правил:

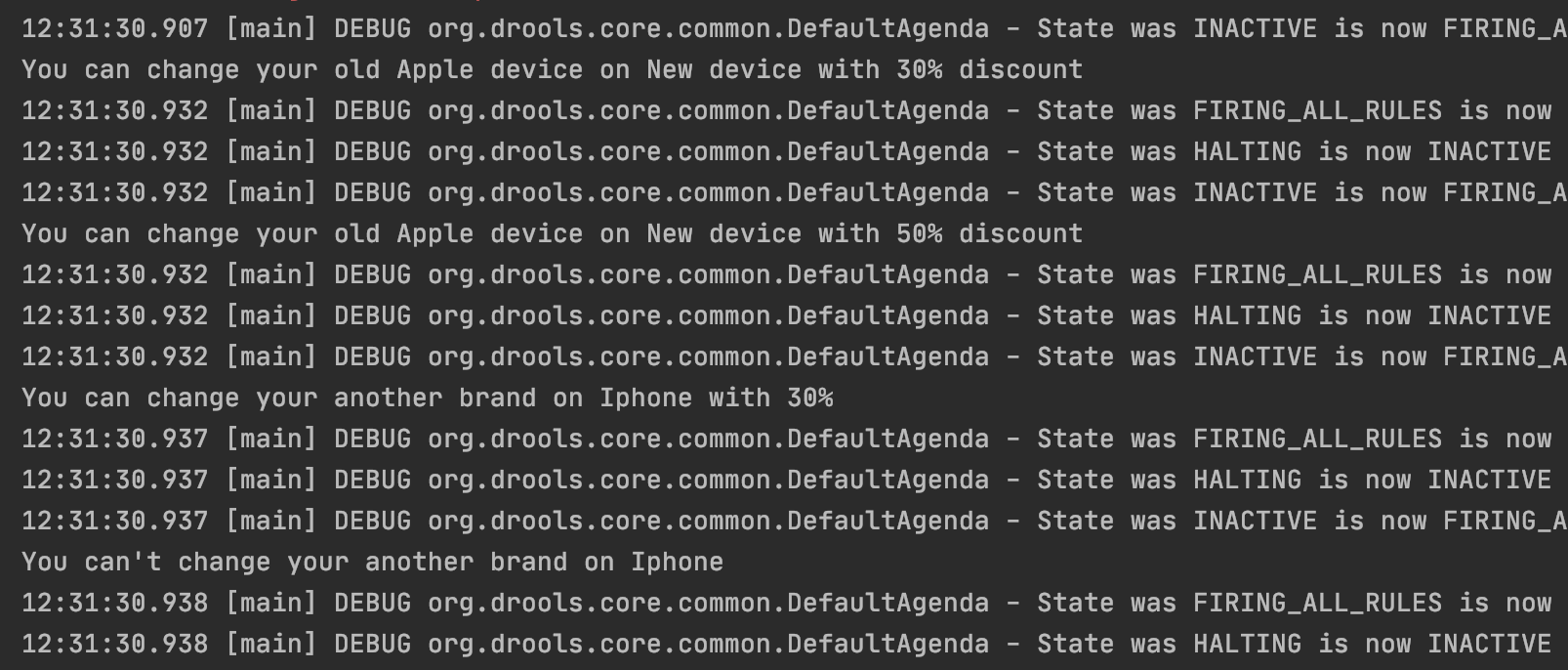


Рисунок 1. Выход программы

Правила сработали так, как ожидалось.

Ссылка на github: https://github.com/GooDiVer/sem7/tree/main/pois/lab2

**Вывод**

В ходе выполнения лабораторной работы мы научились составлять бизнес правила в Drools и применять их в Java приложениях.

Drools – хороший инструмент, чтобы отделить бизнес логику программы от основного кода. Для написания правил можно воспользоваться специальными библиотеками, подключаемыми в intelij idea.

**Список литературы**

1. Mauricio Salatino. Mastering JBoss Drools 6 for Developers/ - М.: Изд-во Книги по требованию, 2016

2. Петер Флах. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / - М.: Изд-во ДМК Пресс, 2019

3. IBM. Реализация бизнес-логики при помощи процессора правил Drools / URL: <https://www.ibm.com/developerworks/ru/library/j-drools/index.html>

4. Материалы лекций