Факультет ИУ "Информатика и системы управления"

Кафедра ИУ-3 "Информационные системы и телекоммуникации"

Отчет к лабораторной работе №5

по курсу "**Разработка программного обеспечения**"

направления 2304000062

# " [Расширение Eclipse своими плагинами](http://www.agentlab.ru/confluence/pages/viewpage.action?pageId=55771143)"

Продолжительность 4 часа.

Выполнил:

студент группы ИУ3-62

Шуклин А.В.

Проверил:

Иванов А.М.

# Цели лабораторной работы

* Углубление навыков работы с системой контроля версий
* Ознакомление на практике с основами графовых БД и технологий Semantic Web

# Задание

1. Чтобы познакомиться с технологией выполните указания обучающего материала

* [Extending the Eclipse IDE - Plug-in development](http://www.vogella.com/tutorials/EclipsePlugIn/article.html)

2. Откройте Eclipse и обновите локальную копию репозитария

* В перспективе Git, надо сделать Pull

3. Переключитесь на стандартную целевую платформу Eclipse

* Настройки Eclipse > Plug-in Development > Target Platform, установить **Running Platform**

4. Импортируйте в Workspace проекты:

* **ru.agentlab.jfxed**
* **ru.agentlab.jfxed.figures**
* **ru.agentlab.jfxed.figures.class**

4. Запустите редактор

* Конфигурация запуска **Eclipse Jfxed**

Исходный код программы:

package ru.agentlab.jfxed.diagramms.umlclassdiagram;

import java.io.FileWriter;

import java.io.IOException;

import com.hp.hpl.jena.ontology.Individual;

import com.hp.hpl.jena.ontology.ObjectProperty;

import com.hp.hpl.jena.ontology.OntClass;

import com.hp.hpl.jena.ontology.OntModel;

import com.hp.hpl.jena.rdf.model.ModelFactory;

import com.hp.hpl.jena.rdf.model.SimpleSelector;

import com.hp.hpl.jena.rdf.model.StmtIterator;

public class UMLClassDiagram {

static final String SCHEMA = "http://www.agentlab.ru/jfxed/onto/UMLClassDiagram";

static final String NS = SCHEMA + "#";

public static void main (String[] args) {

OntModel m = ModelFactory.createOntologyModel();

OntClass concept = m.createClass(NS + "concept");

OntClass relation = m.createClass(NS + "relation");

OntClass objectClass = m.createClass(NS + "object");

objectClass.addSuperClass(concept);

OntClass associationClass = m.createClass(NS + "association");

associationClass.addSuperClass(relation);

OntClass compositionClass = m.createClass(NS + "composition");

compositionClass.addSuperClass(relation);

OntClass aggregationClass = m.createClass(NS + "aggregation");

aggregationClass.addSuperClass(relation);

OntClass generalizationClass = m.createClass(NS + "generalization");

generalizationClass.addSuperClass(relation);

ObjectProperty propAssociationFrom = m.createObjectProperty(NS + "associationFrom");

propAssociationFrom.addDomain(objectClass);

propAssociationFrom.addRange(objectClass);

ObjectProperty propAssociationTo = m.createObjectProperty(NS + "associationTo");

propAssociationTo.addDomain(objectClass);

propAssociationTo.addRange(objectClass);

ObjectProperty propCompositionFrom = m.createObjectProperty(NS + "compositionFrom");

propCompositionFrom.addDomain(objectClass);

propCompositionFrom.addRange(objectClass);

ObjectProperty propCompositionTo = m.createObjectProperty(NS + "compositionTo");

propCompositionTo.addDomain(objectClass);

propCompositionTo.addRange(objectClass);

ObjectProperty propAggregationFrom = m.createObjectProperty(NS + "aggregationFrom");

propAggregationFrom.addDomain(objectClass);

propAggregationFrom.addRange(objectClass);

ObjectProperty propAggregationTo = m.createObjectProperty(NS + "aggregationTo");

propAggregationTo.addDomain(objectClass);

propAggregationTo.addRange(objectClass);

ObjectProperty propGeneralizationFrom = m.createObjectProperty(NS + "generalizationFrom");

propGeneralizationFrom.addDomain(objectClass);

propGeneralizationFrom.addRange(objectClass);

ObjectProperty propGeneralizationTo = m.createObjectProperty(NS + "generalizationTo");

propGeneralizationTo.addDomain(objectClass);

propGeneralizationTo.addRange(objectClass);

Individual tutor = m.createIndividual(NS + "tutor", objectClass);

Individual program = m.createIndividual(NS + "program", objectClass);

Individual transaction = m.createIndividual(NS + "transaction", objectClass);

Individual member = m.createIndividual(NS + "member", objectClass);

Individual vipMember = m.createIndividual(NS + "vipMember", objectClass);

Individual stadium = m.createIndividual(NS + "stadium", objectClass);

Individual booking = m.createIndividual(NS + "booking", objectClass);

Individual session = m.createIndividual(NS + "session", objectClass);

Individual linkAssociation1 = m.createIndividual(NS + "program", associationClass);

linkAssociation1.addProperty(propAssociationFrom, tutor);

linkAssociation1.addProperty(propAssociationTo, program);

Individual linkAssociation11 = m.createIndividual(NS + "tutor", associationClass);

linkAssociation11.addProperty(propAssociationFrom, program);

linkAssociation11.addProperty(propAssociationTo, tutor);

Individual linkAssociation2 = m.createIndividual(NS + "member", associationClass);

linkAssociation2.addProperty(propAssociationFrom, program);

linkAssociation2.addProperty(propAssociationTo, member);

Individual linkAssociation22 = m.createIndividual(NS + "program", associationClass);

linkAssociation22.addProperty(propAssociationFrom, member);

linkAssociation22.addProperty(propAssociationTo, program);

Individual linkAssociation3 = m.createIndividual(NS + "booking", associationClass);

linkAssociation3.addProperty(propAssociationFrom, stadium);

linkAssociation3.addProperty(propAssociationTo, booking);

Individual linkAssociation33 = m.createIndividual(NS + "stadium", associationClass);

linkAssociation33.addProperty(propAssociationFrom, booking);

linkAssociation33.addProperty(propAssociationTo, stadium);

Individual linkAssociation4 = m.createIndividual(NS + "session", associationClass);

linkAssociation4.addProperty(propAssociationFrom, booking);

linkAssociation4.addProperty(propAssociationTo, session);

Individual linkAssociation44 = m.createIndividual(NS + "booking", associationClass);

linkAssociation44.addProperty(propAssociationFrom, session);

linkAssociation44.addProperty(propAssociationTo, booking);

Individual linkComposition = m.createIndividual(NS + "member", compositionClass);

linkComposition.addProperty(propCompositionFrom, booking);

linkComposition.addProperty(propCompositionTo, member);

Individual linkAggregation = m.createIndividual(NS + "member", aggregationClass);

linkAggregation.addProperty(propAggregationFrom, transaction);

linkAggregation.addProperty(propAggregationTo, member);

Individual linkGeneralization = m.createIndividual(NS + "member", generalizationClass);

linkGeneralization.addProperty(propGeneralizationFrom, vipMember);

linkGeneralization.addProperty(propGeneralizationTo, member);

System.out.println("\n");

for(StmtIterator it = linkAggregation.listProperties(); it.hasNext();)

{

System.out.println(it.nextStatement());

}

System.out.println("\n");

for(StmtIterator it = member.listProperties(); it.hasNext();)

{

System.out.println(it.nextStatement());

}

System.out.println("\n");

//вывод всего и всех где есть класс passCheck (goal) для всей модели

OntClass c = tutor.getOntClass();

SimpleSelector s = new SimpleSelector(null, null, c);

for(StmtIterator it = m.listStatements(s); it.hasNext();)

{

System.out.println(it.nextStatement());

}

try {

m.write(new FileWriter("mm.owl"));

} catch (IOException e){

e.printStackTrace();

}

}

}

**package** ru.agentlab.jfxed.diagramms.umlclassdiagram

**import** com.hp.hpl.jena.ontology.OntModel

**import** com.hp.hpl.jena.query.QueryExecutionFactory

**import** com.hp.hpl.jena.query.QueryFactory

**import** com.hp.hpl.jena.query.QuerySolution

**import** com.hp.hpl.jena.rdf.model.Resource

**import** de.fxdiagram.core.XDiagram

**import** ru.agentlab.jfxed.IDiagram

**import** ru.agentlab.jfxed.figures.clazz.Researcher

**public** **class** UML **implements** IDiagram {

**static** String *SOURCE* = "http://www.agentlab.ru/jfxed/onto/UMLClassDiagram"

**static** String *NS* = *SOURCE* + "#"

**override** createJfx(OntModel jenaModel, XDiagram jfxDiagram) {

**val** queryString ='''

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

select ?uri

where {

?uri rdf:type <«*SOURCE*»#concept>

}

'''

**val** query = QueryFactory.*create*(queryString)

// Execute the query and obtain results

**val** qe = QueryExecutionFactory.*create*(query, jenaModel)

**val** results = qe.execSelect()

**for** ( ; results.hasNext() ; )

{

**val** QuerySolution soln = results.nextSolution()

**val** Resource x = soln.getResource("uri") // Get a result variable by name.

**val** target = **new** Researcher() => [

layoutX = 280

layoutY = 280

name = x.localName

]

jfxDiagram => [

nodes += target

]

}

qe.close()

}

}

