**Лабораторная работа №5**

**по курсу «Разработка ПО»**

**Выполнил: Лопатин Илья**

**Проверил: Иванов А.М.**

## Задание

Создайте расширение для Eclipse используя фигуру полученную в процессе лабораторной работы №2.

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**

## Цель задания

Цель задания: познакомит студентов с технологией расширения Eclipse своими плагинами.

## Реализация

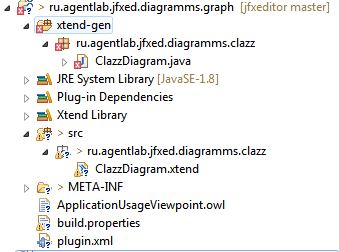


Рисунок 1. Структура папок.

Файл plugin.xml:

<?xml version="1.0" encoding="UTF-8"?>

<?eclipse version="3.0"?>

<plugin>

<extension point="ru.agentlab.jfxed.figure">

<figure name="Component" class="ru.agentlab.jfxed.figures.clazz.ClassFigure"/>

</extension>

</plugin>

## Тестирование

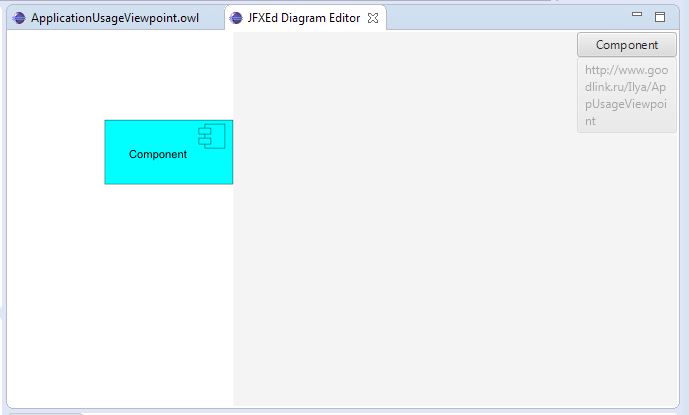


Рисунок 2 - Результат выполнения программы плагин

Исходный код программы:

package ru.agentlab.jfxed.diagramms.clazz

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.ClassFigure

public class ClazzDiagram implements IDiagram {

static String SOURCE = "http://www.goodlink.ru"

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»#BusinessProcess>

}

'''

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 ClassFigure() => [

layoutX = 280

layoutY = 280

name = x.localName

]

jfxDiagram => [

nodes += target

]

}

qe.close()

}

}

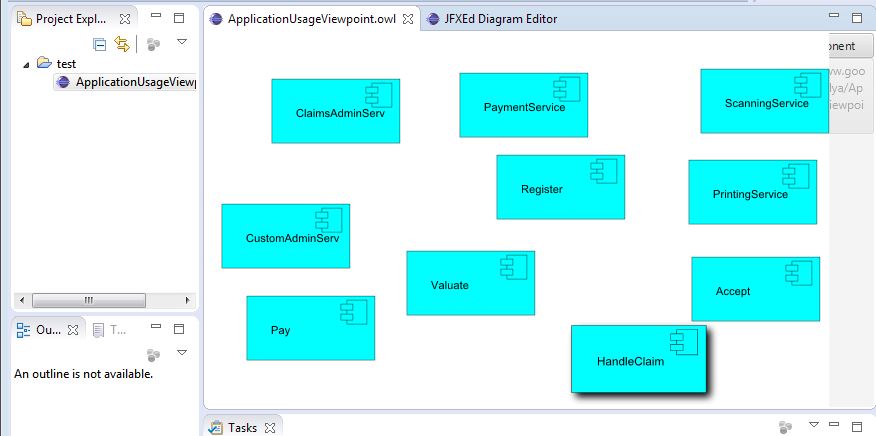


Рисунок 3 – Результат выполнений программы owl файл