Санкт-Петербургский государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

Курсовой проект

по Java технологиям:

«Отчет по курсовому проекту»

Выполнил студент группы Клинчарски В.Н.

Преподаватель Пышкин Е.В.

Санкт-Петербург 2012

**Образ и границы проекта:**

**Формулировка задачи:**

Необходимо реализовать приложение, являющееся графическим редактором графов. Редактор должен обеспечивать возможность нахождения кратчайшего пути между вершинами.

**Структура исходных данных:**

Файлы сохранений, содержащие графы сохраненные пользователем. Эти файлы должны представлять собой XML – файлы, для более структурированного и понятного редактирования/прочтения вне редактора графов.

**Основные сценарии функционирования программы:**

При добавлении/удалении узлов необходимо обновлять структуру, реализующую граф. Также обновление должно происходить при соединении узлов дугами. Как было описано ранее для двух выбранных узлов нужно находить кратчайший путь, между ними.

**Техническое задание:**

Необходимо создать структуру реализующую граф, разработать методы работы с этим графом т.е.: удаление/добавление узлов, дуг, вывод графа с помощью реализованного графического способа.

В ходе работы программы необходимо задавать информацию узлам и дугам, в частности задавать название узлов, длину дуг, тип дуги.

Должна быть реализована возможность сохранения графа спроектированного в программе и храниться этот граф будет в XML файле.

Для реализации алгоритма нахождения кратчайшего пути между двумя узлами выбран алгоритм Флойда-Уоршелла.

Вся работа с графом должна происходить в графическом интерфейсе.

**Системная архитектура:**

**Используемые классы:**

Vertice(класс вершин):

X,y - координаты вершины

Name – имя вершины

Edge (класс дуг):

Vert1, vert2 - вершины образующие дугу

Colour – цвет дуги

Distance – длина дуги

Kind – тип дуги(ориентированная/неориентированная)

Graph (класс реализующий граф):

Edge – массив дуг

Vertice – массив вершин

MainFrame(класс реализующий главное окно)

MainPanel(класс реализующий панель для работы с графом)

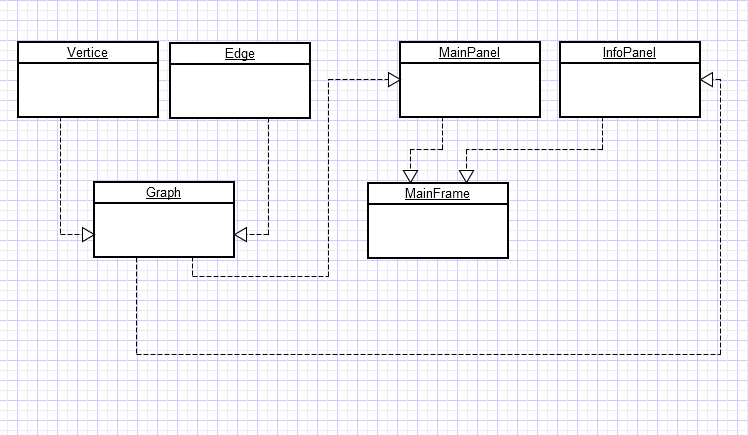
InfoListener(интерфейс обеспечивающий считывание данных)

InfoPanel(класс реализующий панель для редактирования информации графа)

CurrentListener(интерфейс обеспечивающий выбора дуги или вершины)

WorldFileFilter(класс для работы с XML-файлами)

**Взаимоотношение между классами:**

****

**Описание выбранного алгоритма поиска:**

Алгоритм Флойда-Уоршелла:

Находит расстояние от каждой вершины до каждой за количество операций порядка **n^3**. Веса могут быть отрицательными, но у нас не может быть циклов с отрицательной суммой весов рёбер (иначе мы можем ходить по нему сколько душе угодно и каждый раз уменьшать сумму, так не интересно).  
В массиве d[0… n — 1][0… n — 1] на i-ой итерации будем хранить ответ на исходную задачу с ограничением на то, что в качестве «пересадочных» в пути мы будем использовать вершины с номером строго меньше i — 1 (вершины нумеруем с нуля). Пусть идёт i-ая итерация, и мы хотим обновить массив до i + 1-ой. Для этого для каждой пары вершин просто попытаемся взять в качестве пересадочной i — 1-ую вершину, и если это улучшает ответ, то так и оставим. Всего сделаем n + 1 итерацию, после её завершения в качестве «пересадочных» мы сможем использовать любую, и массив d будет являться ответом.  
n итераций по n итераций по n итераций, итого порядка n^3 операций.  
Псевдокод:

прочитать g // g[0 ... n - 1][0 ... n - 1] - массив, в котором хранятся веса рёбер, g[i][j] = 2000000000, если ребра между i и j нет

d = g

for i = 1 ... n + 1

for j = 0 ... n - 1

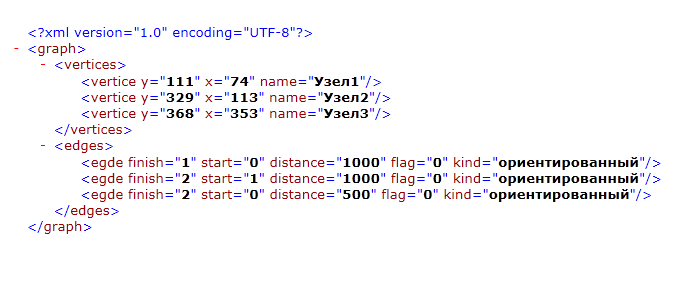
for k = 0 ... n - 1

if d[j][k] > d[j][i - 1] + d[i - 1][k]

d[j][k] = d[j][i - 1] + d[i - 1][k]

вывести d

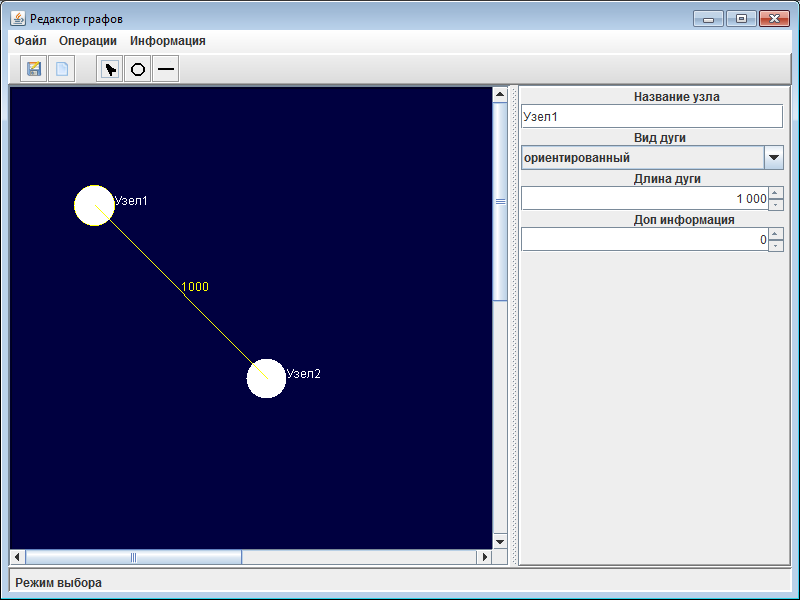
**Структура XML-файла:**

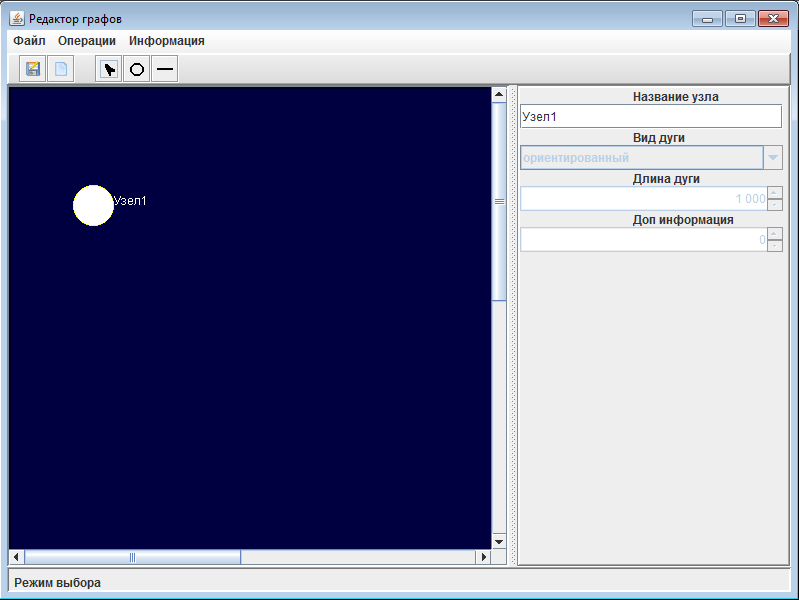
****

В теги **<graph> / </graph>** заключен отдельный граф. В теге **<vertices> / </vertices>** содержится данные обо всех вершинах (**y** – координата узла по y, **x** – координата узла по x, **name** – название узла). В теге **<edges> / </edges>** содержится информация обо всех дугах (**finish** – номер конечного узла, **start** – номер начального узла, **distance** – длина дуги, **flag** – вспомогательная переменная, **kind** – тип дуги).

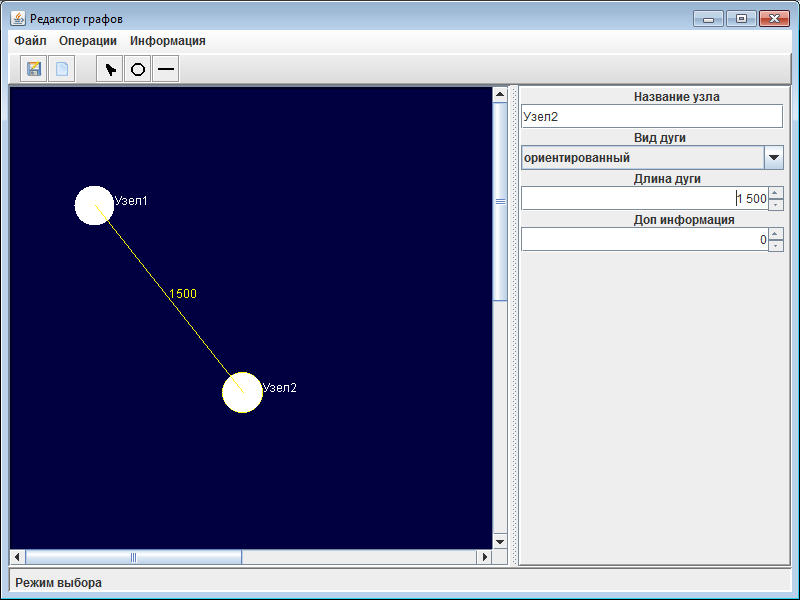
**Тестирование приложения:**

**Основные сценарии работы:**

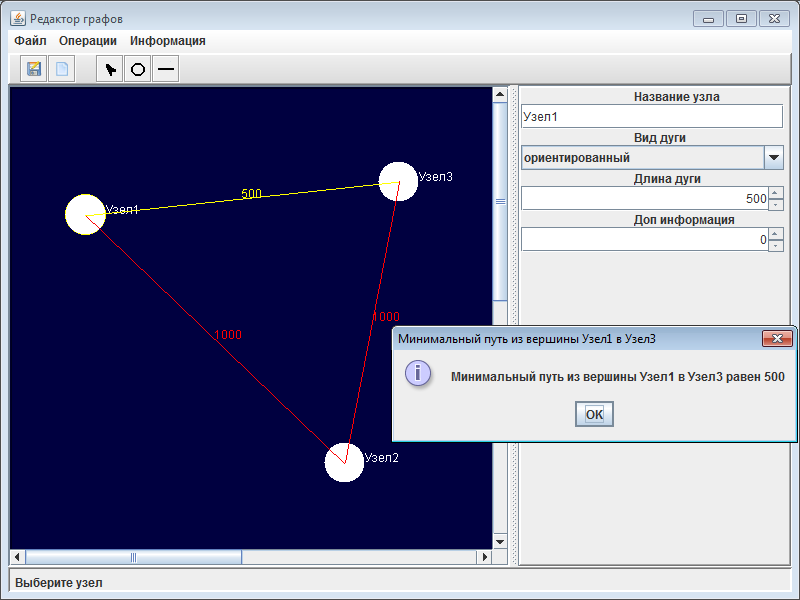
Добавление/удаление вершин (дуг):



Выставление значения длины дуги:

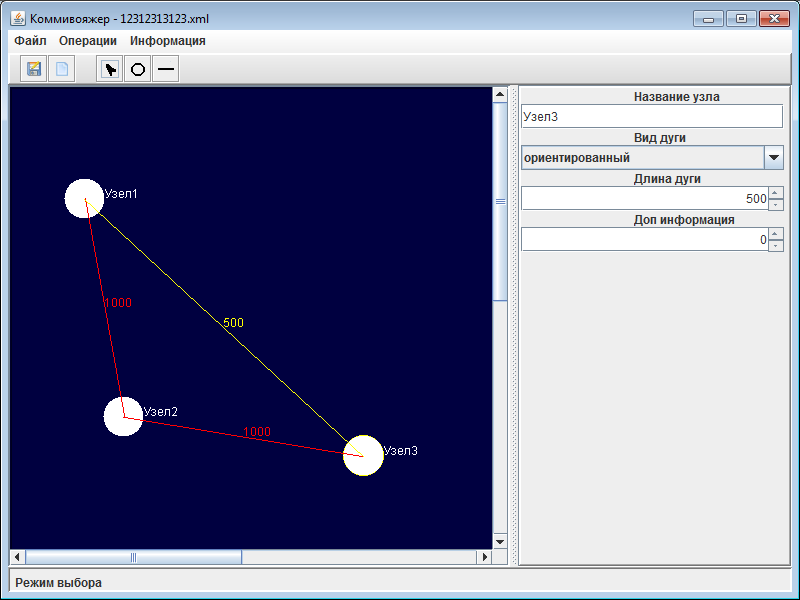


Нахождение кратчайшей длины между вершинами:

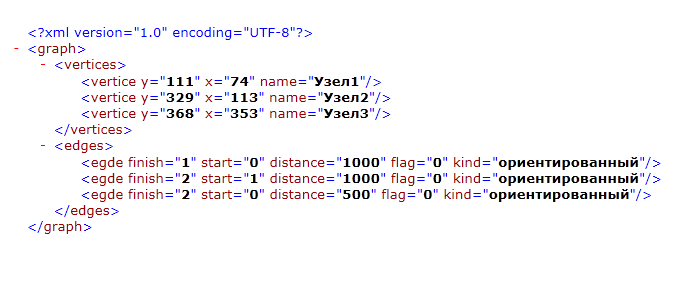


Сохранение графа в файле:

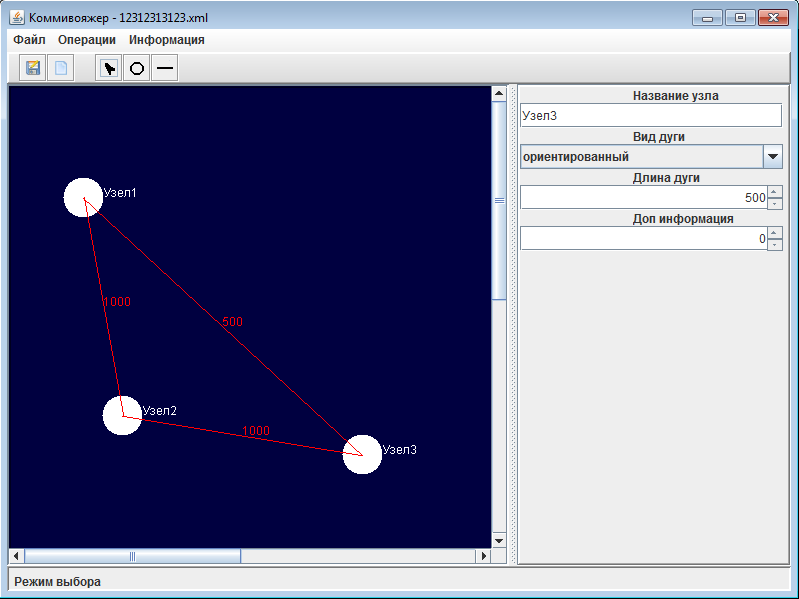
Сохраним след граф



Файл XML содержащий данный граф выглядит следующим образом



Откроем данный файл с помощью нашего редактора графов



**Вывод:**

В ходе выполнения курсовой работы мы научились использовать язык Java для программирования графических приложений. Также для выполнения поставленной задачи необходимо было спроектировать классы реализующие возможность представления графов в программе. Для этого графа необходимо было выбрать и реализовать алгоритм нахождения кратчайшего пути между двумя любыми узлами.

Самой сложной частью курсовой было реализация алгоритма поиска пути и графическая реализация графа и работы с ним. С чем мы в какой-то степени справились.