МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по учебной практике

Тема: Алгоритм Форда-Фалкерсона

Студент гр. 7304 _____ Сергеев И.Д.

Студент гр. 7304 _____ Нгуен К.Х.

Студентка гр. 7304 _____ Нгуен Т.Т.З

Руководитель _____ Ефремов М.А.

Санкт-Петербург
2019

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Сергеев И.Д. группы 7304

Студент Нгуен К.Х. группы 7304

Студентка Нгуен Т.Т.З группы 7304

Тема практики: выполнение мини-проекта на языке java.

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.

Алгоритм: Форда-Фалкерсона.

Сроки прохождения практики: 01.07.2019 – 14.07.2019

Дата сдачи отчета: 08.07.2019

Дата защиты отчета: 10.07.2019

Студент	_____	Сергеев И.Д.
Студент	_____	Нгуен К.Х.
Студентка	_____	Нгуен Т.Т.З
Руководитель	_____	Ефремов М.А.

АННОТАЦИЯ

В ходе выполнения данной работы был реализован алгоритм Форда Фалкерсона. Практическое задание состоит из 4 частей. В первой части расписаны требования к программе в течении выполнения практического задания. Во второй описано распределение работы на всех студентов и план выполнения работы. В третьей части рассматриваются методы решения задачи и используемые структуры данных. В последнюю часть включено тестирование программы, а также сделано заключение, описаны исходные литературные источники, и прикреплен код проекта.

SUMMARY

In the course of this work, the Ford Fulkerson algorithm was implemented. The practical task consists of 4 parts. The first part describes the requirements for the program during the practical task. The second describes the responsibilities of student for work and the work plan. The third part deals with the methods of implementation of the task and the data structures used. The last part includes testing of the program, as well as a conclusion, describes the source literature, and attached the project code.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Кратко описать цель и задачи практики, а также реализуемый алгоритм и его применение. Цель задачи – создание мини-проекта, который реализует алгоритм Форда-Фалкерсона. В задаче нужно использовать диаграмму всех использующихся классов, реализовать алгоритм и интерфейс к нему. Также нужно связать исходный код с интерфейсом и обработать исключительные ситуации и написать тесты к логике программы. К тому же, необходимо нарисовать диаграмму возможностей пользователя по использованию данной программы. В конце требуется написать отчет по выполненной работе.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. На рисунке 1 изображена use case диаграмма. На старте пользователь может выбрать 4 опции: создать граф самому, загрузить его из файла и показать инструкцию применения и about. Для создания графа вручную пользователь может добавлять или удалять вершины и ребра, а также можно сохранить этот граф в файл. После загрузки графа, следует запустить алгоритм соответствующей кнопкой и посмотреть результат на визуализации.

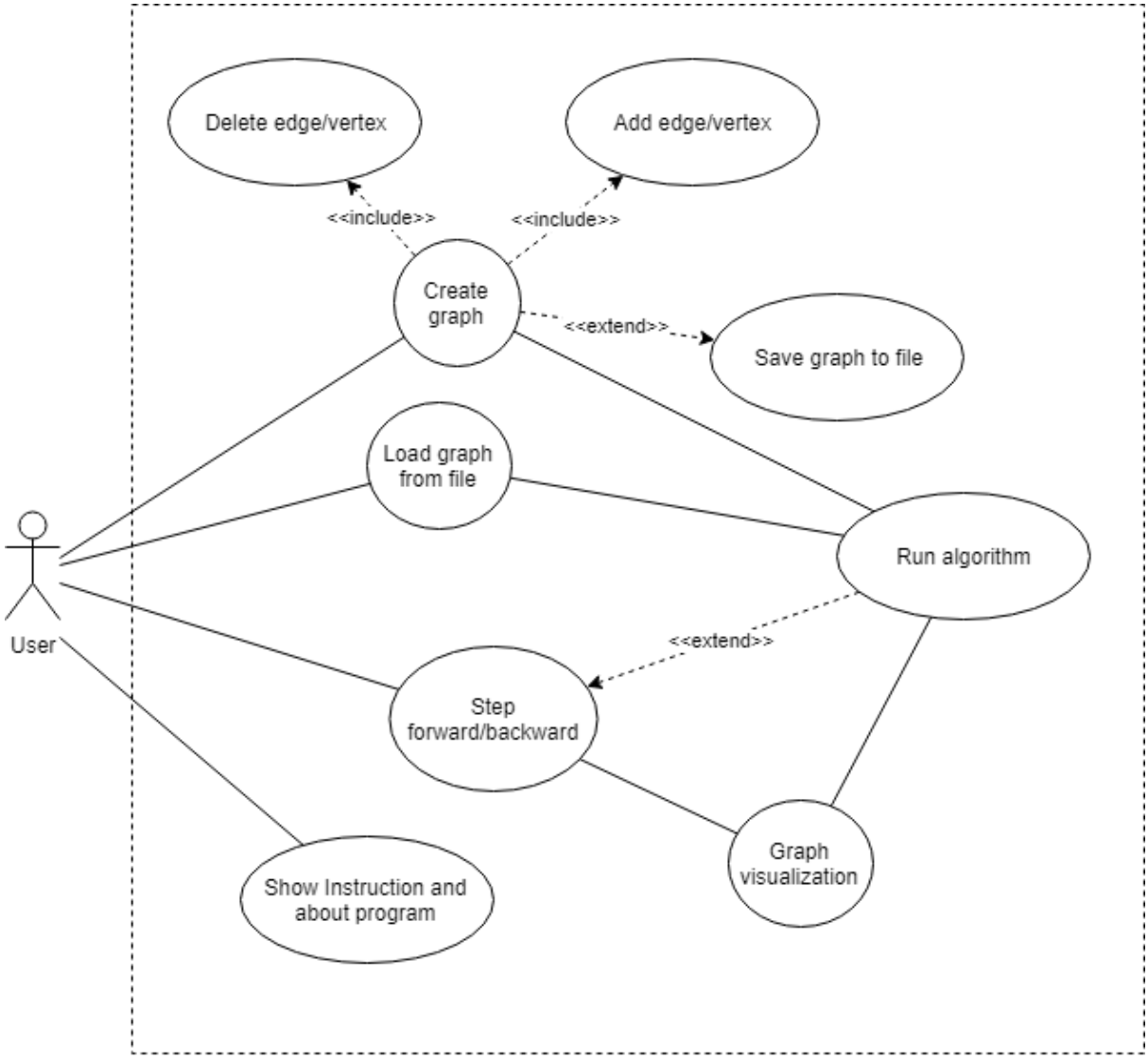

рисунок 1 – use case диаграмма

1.1.2. Отображение вершин. Исток и сток должны быть покрашены определенными цветами. Направление потока показывается с помощью стрелки. Максимальное количество вершин, которые можно вместить от 1 до100, а количество ребер от 1 до 5000. Если в файле введены неверные данные, программа сообщит об этом. Именами для вершин могут быть любые последовательности 1 или более символов, включая !@#$%...

1.2. Уточнение требований после сдачи прототипа.

1.2.1. Заполнить файл с темой задания

1.2.2. Сделать диаграмму классов для интерфейса и добавить возможность добавления вершин и ребер с помощью мышки.

1.2.3. Исправить недочеты в use case и стрелки в графе

1.3. Уточнение требований после сдачи 1-ой версии.

1.3.1. Сделать направления путей в графе и обозначить их цветами при надобности

1.3.2. Установить maven в проект

1.3.3. Сделать unit-тест для модели с проверкой существования экземпляров классов

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

до 04.07: диаграмма классов, use case диаграмма, интерфейс без реализации логики.

до 06.07: прототип с демонстрацией функциональности программы.

до 08.07: 1 версия с обработкой ошибок и всплывающим контекстным меню.

до 10.07: 2 версия, исправление недочетов 1 версии.

2.2. Распределение ролей в бригаде

Сергеев И.Д.: базовые классы, логика алгоритма и отчет.

Нгуен Т.Т.З.: диаграмма классов, контроллер и юнит-тестирование.

Нгуен К.Х.: GUI , use case диаграмма и соединение интерфейса и логики программы.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Использованные структуры данных

На рисунке 2 показана иерархия логики классов. Класс graph – основной класс, который содержит список вершин и ребер и методы работы с ними. Класс Controller служит мостом между интерфейсом и логикой программы. Это запуск основной функции, переопределенные методы добавления/удаления и логика сохранения и загрузки графа из файла. Классы Vertex и Edge реализуют свойства вершин и ребер соответственно(имена, поток, флаги). Классы path нужен для реконструкции путя по ребрам, Dfs и Bfs– реализуют поиски путей. Классы FordFulkerson и PathfindingAlgorithm нужны для поиска минимального потока по всем путям. На рисунке 3 показана иерархия классов интерфейса. GraphEdge и GraphVertex отвечают за отображение объектов на панели интерфейса. VertexPositionChangeListener нужен для перемещения объектов пользователем. GraphElement – нужен для задания параметров вершин и ребер(радиус, Id, флаг, выбран ли объект на данный момент). GraphDisplayFrame принимает список вершин и ребер, инициализирует их и отрисовывает.

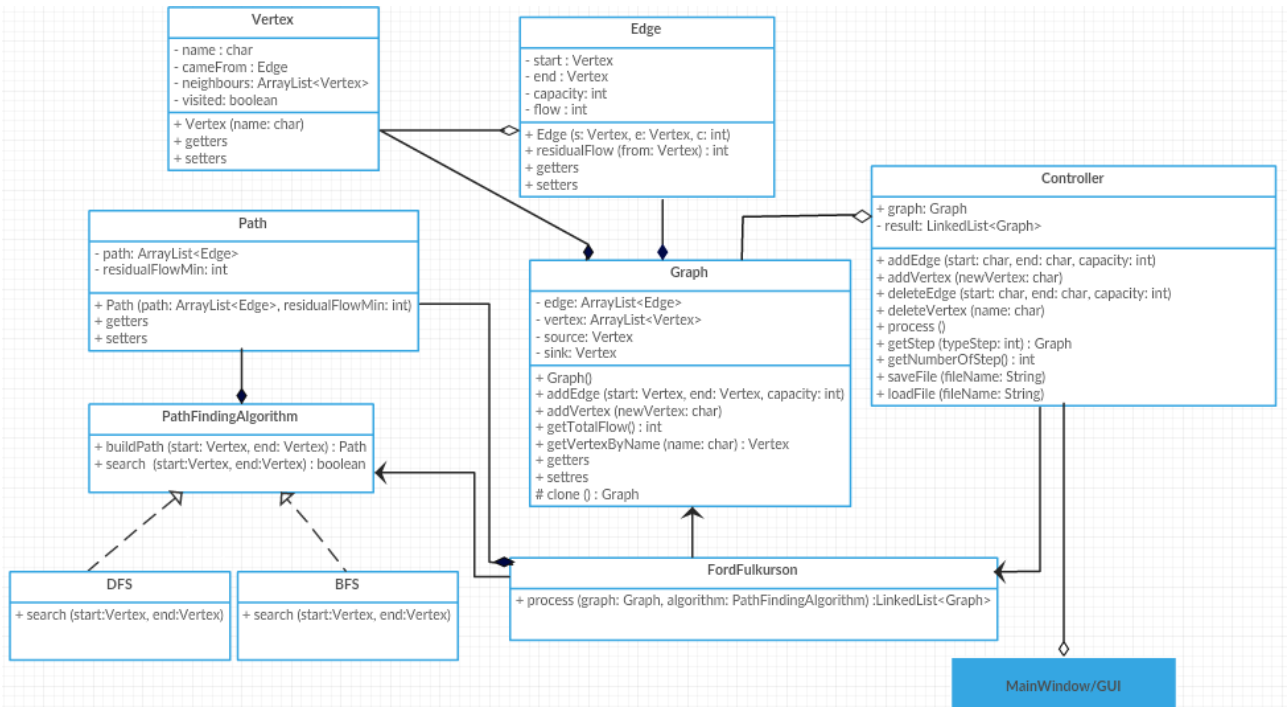

рисунок 2 – классы логики программы.

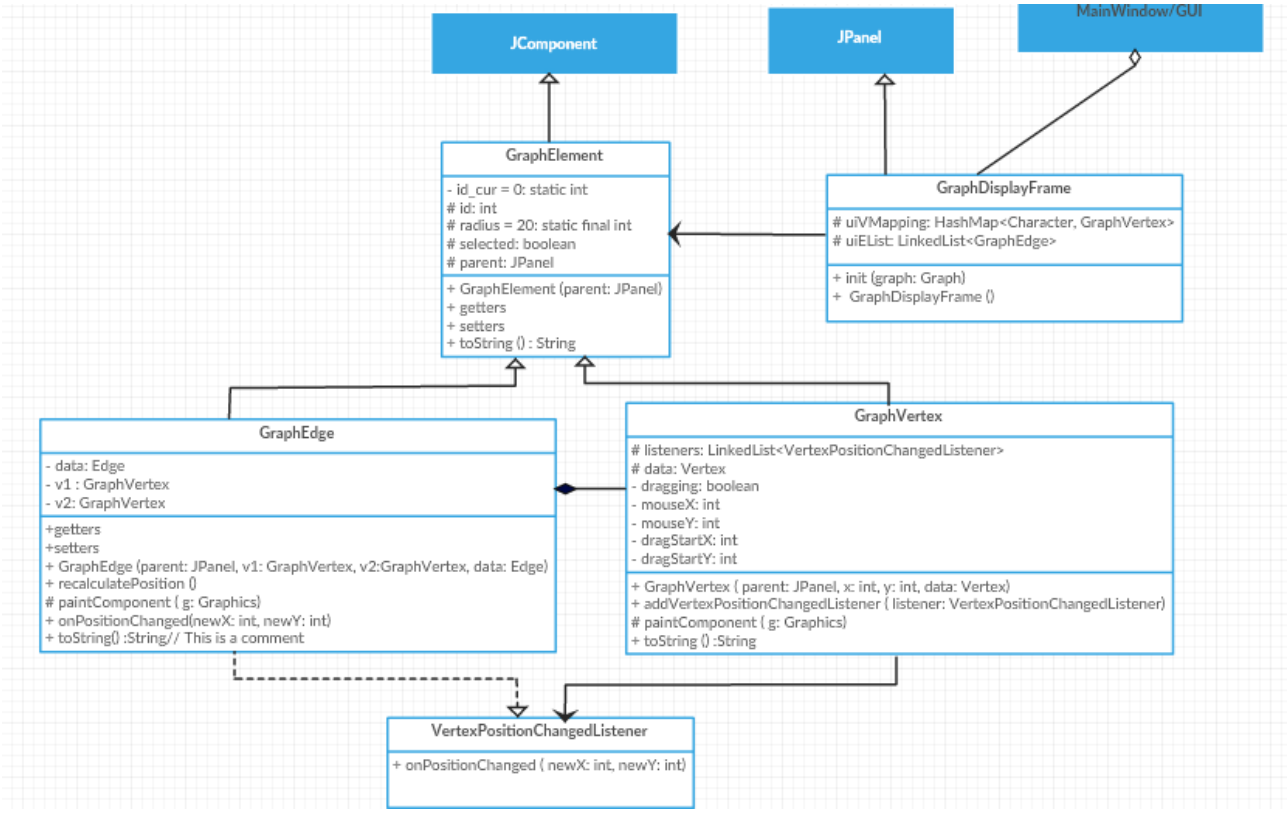

рисунок 3 – классы интерфейса программы

3.2. Основные методы

1) public Graph() – инициализирует массивы для хранения вершин и ребер

2) public Edge(Vertex start, Vertex end, int capacity) – конструктор для ребра

3) public int residualFlow(Vertex from) – получение текущего потока в ребре

4) public Vertex(String name) – конструктор для вершины

5) Path buildPath(Graph graph)- постройка пути в графе по ребрам

6) boolean search(Vertex sourcePeak, Vertex sinkPeak) – поиск в глубину для путей в графе

7) public static LinkedList<Graph> process(Graph graph, PathFindingAlgorhithm pathFinder) – сам алгоритм, то есть поиск всевозможных путей в графе

8) public Path(ArrayList<Edge> path, int residualFlowMin)- конструктор для путей в графе

3.3. Использование интерфейса

На 4 рисунке изображен интерфейс программы. Он состоит из 2 главных окон. В правом окне располагаются кнопки, отвечающие за функциональность

программы. Левое же окно отображает граф и дает пользователю возможность его создавать и редактировать по своему желанию.

В контекстном меню панели пользователь может:

1) Добавлять вершину

В контекстном меню вершины:

1) Удалять вершину
2) Создавать новое ребро
3) Инициализировать вершину как сток/исток

В контекстном меню ребра:

1) Удалять ребро

Также реализована возможность сохранения/загрузки графа в файл/из файла.

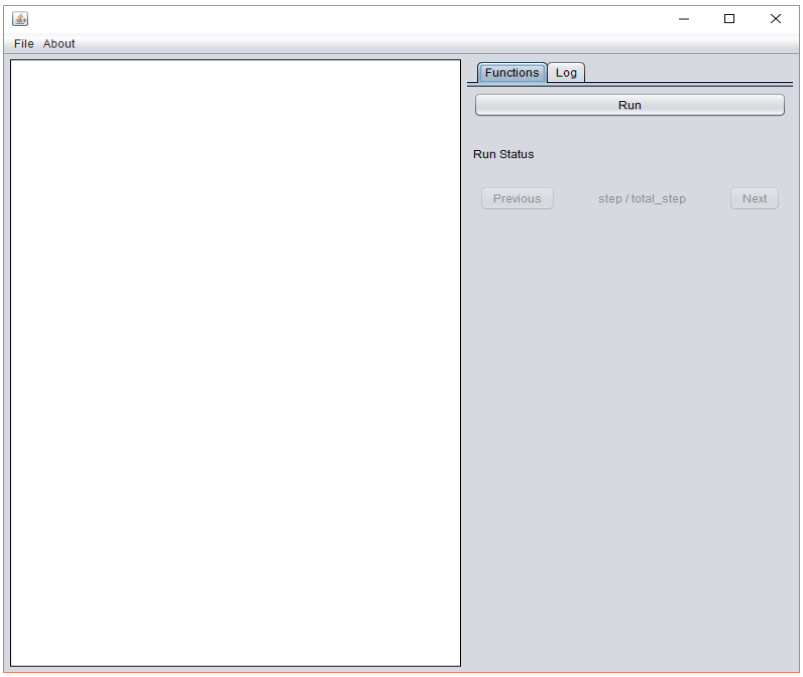

рисунок 4 – интерфейс программы

4. ТЕСТИРОВАНИЕ

4.1. Тестирование графического интерфейса

Создается класс TestGUI для тестирования интерфейса

- В методе *onSetUp()* используется для отображения окна и получения информации об окне.
- Метод *noErrorPressingClearButton*: Проверка правильности работы программы после нажатия кнопки clear.
- Метод *msgBoxPressingRunWithoutGraph()*: Проверка правильности работы программы после нажатия кнопку run без импортирования графа(всплывает окно с соответствующей информацией).
-

4.2. Тестирование кода алгоритма

Создается класс **LogicTest** для создания и хранения модульных тестов.

- В методе *setUp()* создаем graph для использования во всех модульных тестов.
- Метод *testAddVertex()* : Проверка правильности функции *addVertex(String newVertex)* при добавлении вершин в граф – graph. После добавления вершины в граф длину массива вершин увеличится на один и последний элемент массива является этой вершиной.
    - testAddVrtxExists()
- Метод *testAddEdges()* - Проверка правильности функции *addEdge(Vertex start, Vertex end, int capacity)* при добавлении ребро в граф.

  Если сушествует ребро в графе, то тестирование с исключением *(expected = (expected = Exception.class)*, иначе добавлении ребро в граф.
    - *testAddEdgesExists()*
    - *testAddEdgesWithoutStrartVrtx()*
    - *testAddEdgesWithoutEndVrtx()*

- Метод *testDeleteVertex()* - Проверка правильности функции *deleteVertex(Vertex v)*, после удаления возвращает null при поиске.

  Если сушествует вершина в графе, то удаляем эту вершину, иначе тестирование с исключением *(expected = VertexNotFoundException.class)* - исключение при в графе без поиска вершин.

  - *testDelVertexNotExists()*
  - *testDelVertexSource()*
  - *testDelVertexSink()*

- Метод *testDeleteEdge()* - Проверка правильности функции *deleteEdge(Vertex v1, Vertex v2)*, после удаления возвращает null при поиске.

  Если не сушествует ребро в графе, то тестирование с исключением *(expected = VertexNotFoundException.class)*, иначе удаляем ребро в граф.

  - *testDeleteEdgeNotExists()*

- Метод *testIOFile()* – Проверка ввод/вывод доступ к файловой системе с помощью Controller.

- Методы для тестирования алгоритма Форда-Фалкерсона при использавании поиска DFS() и BFS() на граф.

  - *testProcessFFWithDFS()*
  - *testProcessFFWithBFS()*
  - *testProcessWithoutSource()*
  - *testProcessWithoutSink()*

ЗАКЛЮЧЕНИЕ

Таким образом был реализован алгоритм Форда-Фалкерсона. При этом были сделаны диаграммы классов и use case, графический интерефейс, разделение проекта на файлы, обработка ошибок, класс тестирования и сборка при помощи maven. Подводя итог, нужно сказать, что данный проект полностью соответствует требованиям, поставленным в задаче. План работы описан в разделе 2.1, и подробную реализацию можно увидеть в разделах 3 и 4.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шилдт Герберт [Herbert Schildt] Java: The Complete Reference, Tenth Edition: Издательский дом – вильямс, 2015, 155-160с.

2. Кей Хорстманн [*Cay Horstmann*] Java. Библиотека профессионала: Москва • Санкт-Петербург • Киев, 2016, 334-350с.

3. Документация Java™ Platform, Standard Edition 7 API Specification [Электронный ресурс] // Copyright © 1993, 2018, Oracle and/or its affiliates. URL: https://docs.oracle.com/javase/7/docs/api/index.html 04.07.2019

4. Руководство по maven – что такое maven [Электронный ресурс], Apache Maven Project. URL: www.apache-maven.ru 03.07.2019

ПРИЛОЖЕНИЕ А
НАЗВАНИЕ ПРИЛОЖЕНИЯ

```
>>>>>>>>>>>>>>>>>> Controller.java

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package controller;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
import model.BFS;
import model.DFS;
import model.DeleteVertexActionResult;
import model.Edge;
import model.FordFulkerson;
import model.Graph;
import model.Vertex;
import ui.VertexNotFoundException;

/**
 *
 * @author duyenNH
 */
public class Controller {

    public Graph graph;

    private LinkedList<Graph> result = null;

    public Controller() {
```

```
        this.graph = new Graph();
    }

    public Edge addEdge(String start, String end, int capacity) throws
VertexNotFoundException, Exception {
        Vertex v1 = graph.getVertexByName(start);
        Vertex v2 = graph.getVertexByName(end);
        if(v1==null || v2==null) throw new VertexNotFoundException();
        return graph.addEdge(v1,v2, capacity);
    }

    public Vertex addVertex(String newVertex) throws Exception {
        return graph.addVertex(newVertex);
    }

    public void setSource(String name) throws Exception{
        graph.setSource(graph.getVertexByName(name));
    }

    public void setSink(String name) throws Exception{
        graph.setSink(graph.getVertexByName(name));
    }

    public DeleteVertexActionResult deleteVertex(String v) throws
VertexNotFoundException{
        return graph.deleteVertex(graph.getVertexByName(v));

    }

    public void deleteEdge(String start, String end) throws
VertexNotFoundException{
        // TODO;
        Vertex v1 = graph.getVertexByName(start);
        Vertex v2 = graph.getVertexByName(end);

        graph.deleteEdge(v1, v2);
    }

    public void deleteEdge(Edge e) throws VertexNotFoundException{
        graph.deleteEdge(e.getStart(), e.getEnd());
    }
```

```
    public void process() throws Exception {
        result = FordFulkerson.process(graph, new DFS());
    }

    public int getNumberOfStep(){
        if(result == null || result.size()==0) return 0;
        return result.size();
    }
    public Graph getStep(int typeStep) {
        if (typeStep <= result.size() && typeStep > 0) {
            return result.get(typeStep-1);
        } else {
            return null;
        }
    }

    public void saveFile(String fileName) throws IOException {
        FileWriter fw = new FileWriter(fileName);
        fw.write(graph.getVrtx().size() + "\n");
        for (Vertex v : graph.getVrtx()) {
            fw.write(v.getName() + "\n");
        }
        fw.write(graph.getEdges().size() + "\n");
        for (Edge e : graph.getEdges()) {
            fw.write(e.getStart().getName() + " " + e.getEnd().getName() + " " +
e.getCapacity() + "\n");
        }
        if (graph.getSource() != null) {
            fw.write(graph.getSource().getName());
        } else {
            fw.write('#');
        }
        fw.write('\n');
        if (graph.getSink() != null) {
            fw.write(graph.getSink().getName());
        } else {
            fw.write('#');
        }
        fw.close();
    }
```

```
    public void loadFile(String fileName) throws FileNotFoundException,
IOException, Exception {

        FileReader fr = new FileReader(fileName);
        Scanner sc = new Scanner(fr);
        graph = new Graph();

        int V = sc.nextInt();
        for (int i = 0; i < V; i++) {
            graph.addVertex(sc.next());
        }
        int E = sc.nextInt();
        for (int i = 0; i < E; i++) {
            Vertex start = graph.getVertexByName(sc.next());
            Vertex end = graph.getVertexByName(sc.next());
            int capacity = sc.nextInt();
            graph.addEdge(start, end, capacity);
        }
        String srcName = sc.next();
        if (!srcName.equals("#")) {
            //Vertex src = graph.getVertexByName(srcName);
            graph.setSource(graph.getVertexByName(srcName));
        }
        String sinkName = sc.next();
        if (!sinkName.equals("#")) {
            graph.setSink(graph.getVertexByName(sinkName));
        }
        fr.close();
    }
}


>>>>>>>>>>>>>>>>>>> BFS.java

package model;

import java.util.ArrayList;

public class BFS extends PathFindingAlgorhithm{
    @Override
    boolean search(Vertex sourcePeak, Vertex sinkPeak) {
        ArrayList<Vertex> open_set = new ArrayList<>();
```

```
        ArrayList<Vertex> closed_set = new ArrayList<>();
        open_set.add(sourcePeak);
        while (!open_set.isEmpty()) {

            Vertex current = open_set.remove(open_set.size()-1);

            closed_set.add(current);

            if (current.equals(sinkPeak)) {
                // found the path
                return true;
            }

            for (Edge e : current.getNeighbours()) {
                if (e.residualFlow(current) > 0) {
                    Vertex other = e.getEnd().equals(current) ? e.getStart() :
e.getEnd();

                    if (closed_set.contains(other)) {
                        // Already processed (in closed set)
                        continue;
                    }

                    if (!open_set.contains(other)) {
                        // not in open set
                        open_set.add(other);
                        other.setCameFrom(e);
                    }
                }
            }
        }
        return false;
    }
}


>>>>>>>>>>>>>>>>>>> DeleteVertexActionResult.java

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
```

```
 */
package model;

import java.util.List;

/**
 *
 * @author theph
 */
public class DeleteVertexActionResult {

    Vertex deleted;
    List<Edge> affectedEdges;

    public DeleteVertexActionResult(Vertex deleted, List<Edge> affectedEdges) {
        this.deleted = deleted;
        this.affectedEdges = affectedEdges;
    }

    public Vertex getDeleted() {
        return deleted;
    }

    public void setDeleted(Vertex deleted) {
        this.deleted = deleted;
    }

    public List<Edge> getAffectedEdges() {
        return affectedEdges;
    }

    public void setAffectedEdges(List<Edge> affectedEdges) {
        this.affectedEdges = affectedEdges;
    }

}


>>>>>>>>>>>>>>>>>>> DFS.java

package model;
```

```
public class DFS extends PathFindingAlgorhithm{
    @Override
    boolean search(Vertex sourcePeak, Vertex sinkPeak) {
        if (sourcePeak == sinkPeak) {
            return true;
        }
        for (Edge e : sourcePeak.getNeighbours()) {
            if (e.residualFlow(sourcePeak) > 0) {
                Vertex other = e.getEnd().equals(sourcePeak) ? e.getStart() :
e.getEnd();
                if (!other.isVisited()) {
                    other.setVisited(true);
                    other.setCameFrom(e);
                    boolean reached = search(other, sinkPeak);
                    other.setVisited(false);
                    if (reached) {
                        return true;
                    }
                }
            }
        }
        return false;
    }
}
```

>>>>>>>>>>>>>>>>>> Edge.java

```
package model;

import java.io.Serializable;
import java.util.Objects;

public class Edge implements Serializable{
    private Vertex start;
    private Vertex end;
    private int flow;
    private int prevFlow;
    private int capacity;

    public Vertex getStart() {
        return start;
```

```
}

public void setStart(Vertex start) {
    this.start = start;
}

public Vertex getEnd() {
    return end;
}

public void setEnd(Vertex end) {
    this.end = end;
}

public int getFlow() {
    return flow;
}

public int getPrevFlow() {
    return prevFlow;
}

public void setPrevFlow(int prevFlow) {
    this.prevFlow = prevFlow;
}

public void setFlow(int flow) {
    this.prevFlow = this.flow;
    this.flow = flow;
}

public int getCapacity() {
    return capacity;
}

public void setCapacity(int capacity) {
    this.capacity = capacity;
}

public Edge(Vertex start, Vertex end, int capacity) {
    this.start = start;
    this.end = end;
```

```
        this.capacity = capacity;
        this.flow = 0;
    }

    public int residualFlow(Vertex from) {
        if (from.equals(start))
            return capacity - flow;
        else
            return flow;
    }

    @Override
    public int hashCode() {
        int hash = 5;
        hash = 53 * hash + this.start.getName().hashCode();
        hash = 53 * hash + this.end.getName().hashCode();
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Edge other = (Edge) obj;
        if (!this.start.getName().equals(other.start.getName())) {
            return false;
        }
        if (!Objects.equals(this.end.getName(), other.end.getName())) {
            return false;
        }
        return true;
    }
```

```
}


>>>>>>>>>>>>>>>>>>> FordFulkerson.java

package model;

import java.util.ArrayList;
import java.util.LinkedList;

public class FordFulkerson {

    public static LinkedList<Graph> process(Graph graph, PathFindingAlgorhithm
pathFinder) throws Exception {
        if(graph.getSource()==null) throw new Exception("Source is null");
        if(graph.getSink()==null) throw new Exception("Sink is null");

        // cannot find a path from source to sink, algorithm stop;
        LinkedList<Graph> result = new LinkedList<>() ;
        result.add(graph.clone());
        while (pathFinder.search(graph.getSource(), graph.getSink())) {

            //Trace back from sink to get the path;
            Path path = pathFinder.buildPath(graph);

            int cf_min = path.getResidualFlowMin();

            Vertex current = graph.getSource();
            for (Edge e : path.getPath()) {
                if (e.getStart() == current) {
                    // f(u,v) - arc in correct direction
                    e.setFlow( e.getFlow() + cf_min);
                    current = e.getEnd();
                } else {
                    //f(v,u) - arc in opposite direction
                    e.setFlow( e.getFlow() - cf_min);
                    current = e.getStart();
                }
            }
            // TODO: clone the graph
            result.add(graph.clone());
            graph.initPrevFlow();
```

```
        }
        return result;
    }
}
```

>>>>>>>>>>>>>>>>>> Graph.java

```
package model;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import ui.VertexNotFoundException;

public class Graph implements Serializable, Cloneable {

    private ArrayList<Edge> edges;
    private ArrayList<Vertex> vrtx;

    private Vertex source;
    private Vertex sink;

    public Graph() {
        edges = new ArrayList<>();
        vrtx = new ArrayList<>();
    }

    public void initPrevFlow(){
        for(Edge e: edges) e.setPrevFlow(e.getFlow());
    }

    public void resetFlow(){
        for(Edge e: edges) e.setFlow(0);
    }
    public Vertex getSource() {
```

```
    return source;
}

public ArrayList<Edge> getEdges() {
    return edges;
}

public void setEdges(ArrayList<Edge> edges) {
    this.edges = edges;
}

public ArrayList<Vertex> getVrtx() {
    return vrtx;
}

public void setVrtx(ArrayList<Vertex> vrtx) {
    this.vrtx = vrtx;
}

public void setSource(Vertex source) throws Exception {
    if (source.equals(this.sink)) {
        throw new Exception("Source cant be sink");
    }
    if (this.source != null) {
        this.source.setSource(false);
    }
    this.source = source;
    this.source.setSource(true);
}

public Vertex getSink() {
    return sink;
}

public void setSink(Vertex sink) throws Exception {
    if (sink.equals(this.source)) {
        throw new Exception("Sink cant be source");
    }
    if (this.sink != null) {
        this.sink.setSink(false);
    }
    this.sink = sink;
```

```
        this.sink.setSink(true);
    }

    public Edge addEdge(Vertex start, Vertex end, int capacity) throws Exception
{
        Edge edg = new Edge(start, end, capacity);
        for (int i = edges.size() - 1; i >= 0; i--) {
            if (edg.getStart().equals(edges.get(i).getStart())
                    && edg.getEnd().equals(edges.get(i).getEnd())) {
                throw new Exception("ERROR: This edge already exists!");
            }
        }
        edges.add(edg);
        start.getNeighbours().add(edg);
        end.getNeighbours().add(edg);
        return edg;
    }

    public Vertex addVertex(String newVertex) throws Exception {
        if (getVertexByName(newVertex) != null) {
            throw new Exception("Vertex with this name already exist");
        }
        Vertex result = new Vertex(newVertex);

        vrtx.add(result);
        return result;
    }

    public DeleteVertexActionResult deleteVertex(Vertex v) throws
VertexNotFoundException {
        if (v != null && vrtx.contains(v)) {
            LinkedList<Edge> affectedEdges = new LinkedList<>();
            for (int i = edges.size() - 1; i >= 0; i--) {
                if (edges.get(i).getStart().equals(v) ||
edges.get(i).getEnd().equals(v)) {
                    affectedEdges.add(edges.remove(i));
                }
            }
            if (source != null && source.equals(v)) {
                source = null;
            }
            if (sink != null && sink.equals(v)) {
```

```
                sink = null;
            }
            DeleteVertexActionResult res = new
DeleteVertexActionResult(vrtx.remove(vrtx.indexOf(v)), affectedEdges);
            return res;
        } else {
            throw new VertexNotFoundException();
        }
    }

    public void deleteEdge(Vertex v1, Vertex v2) throws VertexNotFoundException
{
        if (v1 != null && v2 != null && vrtx.contains(v1) && vrtx.contains(v2))
{

            for (int i = edges.size() - 1; i >= 0; i--) {
                if (edges.get(i).getStart().equals(v1) &&
edges.get(i).getEnd().equals(v2)) {
                    Edge delete = edges.remove(i);
                    delete.getStart().getNeighbours().remove(delete);
                    delete.getEnd().getNeighbours().remove(delete);
                    break;
                }
            }
        } else {
            throw new VertexNotFoundException();
        }
    }

    public int getTotalFlow() {
        int total_flow = 0;
        for (Edge e : this.source.getNeighbours()) {
            total_flow += e.getFlow();
        }
        return total_flow;
    }

    public Vertex getVertexByName(String name) {
        for (Vertex v : getVrtx()) {
            if (v.getName().equals(name)) {
                return v;
            }
```

```
        }
        return null;
    }

    @Override
    protected Graph clone() {
        try {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(baos);
            oos.writeObject(this);

            ByteArrayInputStream bais = new
ByteArrayInputStream(baos.toByteArray());
            ObjectInputStream ois = new ObjectInputStream(bais);
            return (Graph) ois.readObject();
        } catch (IOException e) {
            return null;
        } catch (ClassNotFoundException e) {
            return null;
        }
    }
}


>>>>>>>>>>>>>>>>>> Path.java

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package model;

import java.util.ArrayList;

/**
 *
 * @author theph
 */
public class Path {
    private ArrayList<Edge> path;
    private int residualFlowMin;
```

```
    public Path(ArrayList<Edge> path, int residualFlowMin) {
        this.path = path;
        this.residualFlowMin = residualFlowMin;
    }

    public ArrayList<Edge> getPath() {
        return path;
    }

    public void setPath(ArrayList<Edge> path) {
        this.path = path;
    }

    public int getResidualFlowMin() {
        return residualFlowMin;
    }

    public void setResidualFlowMin(int residualFlowMin) {
        this.residualFlowMin = residualFlowMin;
    }

}
```

>>>>>>>>>>>>>>>>>> PathFindingAlgorhithm.java

```
package model;

import java.util.ArrayList;

public abstract class PathFindingAlgorhithm {
    Path buildPath(Graph graph)
    {
        Path result = new Path(new ArrayList<>(), Integer.MAX_VALUE);

        Vertex current = graph.getSink();
        while (current.getCameFrom() != null &&
!current.equals(graph.getSource())) {
            Vertex back = current.getCameFrom().getStart().equals(current) ?
current.getCameFrom().getEnd() : current.getCameFrom().getStart();
```

```
            result.getPath().add(0,current.getCameFrom());

            int cfp = current.getCameFrom().residualFlow(back);
            result.setResidualFlowMin(result.getResidualFlowMin() > cfp ? cfp :
result.getResidualFlowMin());

            // clear cameFrom field for next loop
            current.setCameFrom(null);
            current = back;
        }
        return result;
    }

    abstract boolean search(Vertex sourcePeak, Vertex sinkPeak);

}
```

>>>>>>>>>>>>>>>>>> Vertex.java

```
package model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Objects;

public class Vertex  implements Serializable{
    private String name;
    private Edge cameFrom;
    private ArrayList<Edge> neighbours;
    private Boolean visited;

    private boolean isSource;
    private boolean isSink;

    public boolean isSource() {
        return isSource;
    }

    public void setSource(boolean isSource) {
        this.isSource = isSource;
    }
```

```
public boolean isSink() {
    return isSink;
}

public void setSink(boolean isSink) {
    this.isSink = isSink;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public Edge getCameFrom() {
    return cameFrom;
}

public void setCameFrom(Edge cameFrom) {
    this.cameFrom = cameFrom;
}

public ArrayList<Edge> getNeighbours() {
    return neighbours;
}

public void setNeighbours(ArrayList<Edge> neighbours) {
    this.neighbours = neighbours;
}

public boolean isVisited() {
    return visited;
}

public void setVisited(boolean visited) {
    this.visited = visited;
}

public Vertex(String name) {
```

```
        this.name = name;
        neighbours = new ArrayList<>();
        visited = false;
        cameFrom = null;
    }

    @Override
    public int hashCode() {
        int hash = 3;
        hash = 67 * hash + Objects.hashCode(this.name);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Vertex other = (Vertex) obj;
        if (this.name != other.name) {
            return false;
        }
        return true;
    }


}
```

>>>>>>>>>>>>>>>>>>> AboutDialog.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package ui;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.text.Document;
import javax.swing.text.html.HTMLDocument;

/**
 *
 * @author theph
 */
public class AboutDialog extends javax.swing.JDialog {

    /**
     * Creates new form AboutDialog
     */
    public AboutDialog(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
        try {
            File file = new File("about.html");
            FileInputStream fis = new FileInputStream(file);
            byte[] data = new byte[(int) file.length()];
            fis.read(data);
            fis.close();

            String str = new String(data, "UTF-8");
            txtAbout.setContentType("text/html");
            txtAbout.setText(str);
        } catch (FileNotFoundException ex) {
            Logger.getLogger(AboutDialog.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (IOException ex) {
            Logger.getLogger(AboutDialog.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
```

```
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        btnDone = new javax.swing.JButton();
        jScrollPane2 = new javax.swing.JScrollPane();
        txtAbout = new javax.swing.JTextPane();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);

        btnDone.setText("Done");
        btnDone.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDoneActionPerformed(evt);
            }
        });

        jScrollPane2.setViewportView(txtAbout);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jScrollPane2)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(0, 323, Short.MAX_VALUE)
                        .addComponent(btnDone)))
                .addContainerGap())
```

```
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 249, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(btnDone)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void btnDoneActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnDoneActionPerformed
        this.dispose();
    }//GEN-LAST:event_btnDoneActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
```

```
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(AboutDialog.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(AboutDialog.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(AboutDialog.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(AboutDialog.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the dialog */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                AboutDialog dialog = new AboutDialog(new javax.swing.JFrame(),
true);
                dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                    @Override
                    public void windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnDone;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextPane txtAbout;
    // End of variables declaration//GEN-END:variables
}
```

```
>>>>>>>>>>>>>>>>>> GraphDisplayFrame.java

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ui;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.Point;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.geom.Line2D;
import java.util.ArrayList;
import java.util.EventObject;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Random;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JPopupMenu;
import javax.swing.JSeparator;
import javax.swing.event.MouseInputAdapter;
import model.Edge;
import model.Graph;
import model.Vertex;

/**
 *
 * @author theph
 */
public class GraphDisplayFrame extends JPanel implements VertexActionListener {
```

```
    Graph graph;
    HashMap<String, GraphVertex> uiVMapping;

    HashMap<Edge, GraphEdge> uiEMapping;

    Component parent;

    private Line2D tempEdge;

    public GraphDisplayFrame(Component parent) {
        this.setLayout(null);
        this.parent = parent;
        tempEdge = null;
    }

    public void init(Graph graph) {
        this.removeAll();
        this.graph = graph;
        uiVMapping = new HashMap<>();
        uiEMapping = new HashMap<>();
        for (Vertex v : graph.getVrtx()) {
            addVertex(v);
        }
        for (Edge e : graph.getEdges()) {
            addEdge(e);
        }
    }

    public Line2D getTempEdge() {
        return tempEdge;
    }

    public void setTempEdge(Line2D tempEdge) {
        this.tempEdge = tempEdge;
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        if(this.tempEdge!=null){
            g.drawLine((int)this.tempEdge.getX1(), (int)this.tempEdge.getY1(),
(int)this.tempEdge.getX2(), (int)this.tempEdge.getY2());
```

```
        }
    }


    @Override
    public void onVertexPositionChanged() {
        repaint();
    }

    @Override
    public void onSourceChanged(GraphVertex newSource) {
        repaint();
    }

    @Override
    public void onSinkChanged(GraphVertex newSink) {
        repaint();
    }

    @Override
    public void onDelete(GraphVertex v) {
        repaint();
    }

    public void addVertex(Vertex v) {
        addVertex(v, new Point((int) (Math.random() * 400), (int) (Math.random()
* 400)));
    }

    public void addVertex(Vertex v, Point location) {
        GraphVertex gv = new GraphVertex(this, (int) location.getX(), (int)
location.getY(), v);
        //setup listeners
        if (this.parent instanceof VertexActionListener) {
            gv.addVertexChangedListener((VertexActionListener) this.parent);
        }
        gv.addVertexChangedListener(this);

        this.add(gv);
        uiVMapping.put(v.getName(), gv);
        repaint();
    }
```

```
    public void addEdge(Edge e) {
        String nameStart = e.getStart().getName();
        String nameEnd = e.getEnd().getName();
        GraphEdge ge = new GraphEdge(this, uiVMapping.get(nameStart),
uiVMapping.get(nameEnd), e);
        this.add(ge);
        uiEMapping.put(e, ge);
        repaint();
    }

    public void deleteVertex(Vertex v) {
        if (uiVMapping.containsKey(v.getName())) {
            this.remove(uiVMapping.remove(v.getName()));
        }
        revalidate();
        repaint();
    }

    public void deleteEdge(Edge e) {
        if (uiEMapping.containsKey(e)) {
            this.remove(uiEMapping.remove(e));
        }
        revalidate();
        repaint();
    }

    public GraphEdge hasEdgeAt(int x, int y) {
        for (GraphEdge e : uiEMapping.values()) {
            if (Line2D.ptSegDist(e.getV1().getX() + GraphElement.radius,
e.getV1().getY() + GraphElement.radius, e.getV2().getX() + GraphElement.radius,
e.getV2().getY() + GraphElement.radius, x, y) < 15) {
                return e;
            }
        }
        return null;
    }

    public void deleteEdges(List<Edge> edges) {
        for (Edge e : edges) {
            deleteEdge(e);
        }
```

```
    }

    public void loadStepGraph(Graph graph) {
        for (Vertex v : graph.getVrtx()) {
            uiVMapping.get(v.getName()).setData(v);
        }
        for (Edge e : graph.getEdges()) {
            uiEMapping.get(e).setData(e);
        }
    }

    @Override
    public void onVertexSelected(GraphVertex v, EventObject event) {
    }

}
```

>>>>>>>>>>>>>>>>>> GraphEdge.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ui;

import java.awt.Color;
import java.awt.Component;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.Polygon;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.geom.AffineTransform;
import java.util.Random;
import javax.swing.JPanel;
import javax.swing.event.MouseInputAdapter;
import model.Edge;
```

```
/**
 *
 * @author theph
 */
public class GraphEdge extends GraphElement {

    private Edge data;

    private GraphVertex v1, v2;

    public GraphEdge(JPanel parent, GraphVertex v1, GraphVertex v2, Edge data) {
        super(parent);
        this.data = data;
        this.v1 = v1;
        this.v2 = v2;
        //recalculatePosition();
        setLocation(0, 0);
        setSize(9999, 9999);
    }

    @Override
    public void paint(Graphics g) {
        if (data.getPrevFlow() == data.getFlow()) {
            g.setColor(Color.black);
        } else {
            g.setColor(Color.red);
        }
        //draw line
        g.drawLine(v1.getX() + radius, v1.getY() + radius, v2.getX() + radius,
v2.getY() + radius);

        // write text
        g.setFont(new Font("TimesRoman", Font.PLAIN, 20));

        int textX = (v1.getX() + v2.getX()) / 2 + radius;
        int textY = (v1.getY() + v2.getY()) / 2 + radius;

        g.drawString(toString(), textX, textY);
        //draw arrow
        Polygon arrowHead = new Polygon();
        arrowHead.addPoint(0, 5);
        arrowHead.addPoint(-5, -5);
```

```
        arrowHead.addPoint(5, -5);

        double angle = Math.atan2(v2.getY() - v1.getY(), v2.getX() - v1.getX());
        double offsetX = radius * Math.cos(angle);
        double offsetY = radius * Math.sin(angle);

        AffineTransform tx = new AffineTransform();
        tx.setToIdentity();
        tx.translate(v2.getX() + radius - offsetX, v2.getY() + radius -
offsetY);
        tx.rotate((angle - Math.PI / 2));

        Graphics2D g2 = (Graphics2D) g.create();
        g2.setTransform(tx);
        g2.fill(arrowHead);
        g2.dispose();

    }

    public Edge getData() {
        return data;
    }

    public void setData(Edge data) {
        this.data = data;
    }

    @Override
    public String toString() {
        if (data != null) {
            if (data.getPrevFlow() == data.getFlow()) {
                return data.getFlow() + "/" + data.getCapacity();
            } else {
                return data.getPrevFlow() + "→" + data.getFlow() + "/" +
data.getCapacity();
            }

        } else {
            return id + "";
        }
    }
```

```
    public GraphVertex getV1() {
        return v1;
    }

    public void setV1(GraphVertex v1) {
        this.v1 = v1;
    }

    public GraphVertex getV2() {
        return v2;
    }

    public void setV2(GraphVertex v2) {
        this.v2 = v2;
    }

}
```

>>>>>>>>>>>>>>>>>> GraphElement.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ui;

import java.awt.Graphics;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import javax.swing.JComponent;
import javax.swing.JPanel;

/**
 *
 * @author theph
 */
public abstract class GraphElement extends JComponent {

    private static int id_cur = 0;
```

```
    protected int id;

    public static final int radius = 20;

    protected boolean selected;

    protected JPanel parent;



    public GraphElement(JPanel parent) {
        super();
        this.parent = parent;
        this.id = id_cur;
        id_cur++;
        selected = false;
    }

    @Override
    public String toString() {
        return "{" + id + '}';
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public boolean isSelected() {
        return selected;
    }

    public void setSelected(boolean selected) {
        this.selected = selected;
    }

}
```

```
>>>>>>>>>>>>>>>>>>> GraphVertex.java

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ui;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionAdapter;
import java.awt.event.MouseMotionListener;
import java.util.LinkedList;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JPopupMenu;
import javax.swing.JSeparator;
import javax.swing.event.MouseInputAdapter;
import model.Vertex;

/**
 *
 * @author theph
 */
public class GraphVertex extends GraphElement {

    LinkedList<VertexActionListener> listeners;

    private Vertex data;
    //variables for drag and drop vertex
    private boolean dragging = false;

    private int mouseX = 0;
    private int mouseY = 0;
    private int dragStartX = 0;
    private int dragStartY = 0;
```

```
JPopupMenu contextMenu;

public GraphVertex(JPanel parent, int x, int y, Vertex data) {
    super(parent);
    this.data = data;
    this.setLocation(x, y);
    this.setSize(radius * 2, radius * 2);
    listeners = new LinkedList<>();
    this.initContextMenu();
    this.addMouseListener(new MouseInputAdapter() {

        @Override
        public void mousePressed(MouseEvent e) {
            if (e.getButton() == MouseEvent.BUTTON1) {
                mouseX = e.getXOnScreen();
                mouseY = e.getYOnScreen();

                dragStartX = getX();
                dragStartY = getY();

                dragging = true;
            }
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            if (e.getButton() == MouseEvent.BUTTON1) {
                dragging = false;
                for (VertexActionListener listener : listeners) {
                    listener.onVertexSelected(GraphVertex.this, e);
                }
            }
        }

    });
    this.addMouseMotionListener(new MouseMotionAdapter() {
        @Override
        public void mouseDragged(MouseEvent e) {
            if (dragging) {
                int deltaX = e.getXOnScreen() - mouseX;
                int deltaY = e.getYOnScreen() - mouseY;
```

```
                int newX = dragStartX + deltaX;
                int newY = dragStartY + deltaY;
                setLocation(newX, newY);

                for (VertexActionListener listener : listeners) {
                    listener.onVertexPositionChanged();
                }
            }

        }
    });
}

private void initContextMenu() {
    contextMenu = new JPopupMenu();

    JMenuItem addEdgeMI = new JMenuItem("Add edge");
    addEdgeMI.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            System.out.println("selected in graph vertex");
            System.out.println("number of listener: " + listeners.size());
            for (VertexActionListener listener : listeners) {
                listener.onVertexSelected(GraphVertex.this, e);
            }
        }
    });

    JMenuItem deleteVertexMI = new JMenuItem("Remove vertex");
    deleteVertexMI.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            for (VertexActionListener listener : listeners) {
                listener.onDelete(GraphVertex.this);
            }
        }
    });

    JSeparator sep1 = new JSeparator();
    JMenuItem setSourceMI = new JMenuItem("Set source");
    setSourceMI.addActionListener(new ActionListener() {
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            for (VertexActionListener listener : listeners) {
                listener.onSourceChanged(GraphVertex.this);
            }
        }
    });

    JMenuItem setSinkMI = new JMenuItem("Set sink");
    setSinkMI.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            for (VertexActionListener listener : listeners) {
                listener.onSinkChanged(GraphVertex.this);
            }
        }
    });

    contextMenu.add(addEdgeMI);
    contextMenu.add(new JSeparator());
    contextMenu.add(deleteVertexMI);
    contextMenu.add(sep1);
    contextMenu.add(setSourceMI);
    contextMenu.add(setSinkMI);

    this.addMouseListener(new MouseInputAdapter() {
        @Override
        public void mouseReleased(MouseEvent e) {
            showContextMenu(e);
        }

        @Override
        public void mousePressed(MouseEvent e) {
            showContextMenu(e);
        }
    });
    //this.setComponentPopupMenu(contextMenu);

}

private void showContextMenu(MouseEvent e) {
    if (e.isPopupTrigger()) {
```

```
            if (Setting.getInstance().getRunningMode() ==
Setting.MODE_GRAPH_DESIGN) {
                contextMenu.show(e.getComponent(), e.getX(), e.getY());
            }

        }
    }

    public void addVertexChangedListener(VertexActionListener listener) {
        listeners.add(listener);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (!this.selected) {
            if (this.data.isSource()) {
                g.setColor(Color.ORANGE);
            } else if (this.data.isSink()) {
                g.setColor(Color.CYAN);
            } else {
                g.setColor(Color.GREEN);
            }
        } else {
            g.setColor(Color.red);
        }
        g.fillOval(0, 0, radius * 2, radius * 2);

        g.setColor(Color.black);
        g.drawString(toString(), radius, radius);
    }

    @Override
    public String toString() {
        if (data != null) {
            return data.getName() + "";
        } else {
            return id + "";
        }
    }

    public Vertex getData() {
```

```
        return data;
    }

    public void setData(Vertex data) {
        this.data = data;
    }

}
```

>>>>>>>>>>>>>>>>>> MainWindow.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ui;

import controller.Controller;
import java.awt.Color;
import java.awt.Component;
import java.awt.GridLayout;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.awt.geom.Line2D;
import java.io.IOException;
import java.util.EventObject;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JComponent;
import javax.swing.JFileChooser;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPopupMenu;
import javax.swing.JSeparator;
import javax.swing.event.MouseInputAdapter;
```

```
import model.DeleteVertexActionResult;
import model.Edge;
import model.Vertex;

/**
 *
 * @author theph
 */
public class MainWindow extends javax.swing.JFrame implements
VertexActionListener, SettingChangedListener {

    /**
     * Creates new form MainWindow
     */
    GraphDisplayFrame graphDisplay;
    Controller controller;
    int step;
    JPopupMenu contextMenuPanel;
    JPopupMenu contextMenuEdge;

    JComponent selected = null;

    ActionListener runListener;
    ActionListener backToDesignListener;

    public MainWindow() {
        initComponents();
        controller = new Controller();
        drawingPanel.setLayout(new GridLayout(1, 1));
        graphDisplay = new GraphDisplayFrame(this);
        drawingPanel.add(graphDisplay);
        graphDisplay.setBackground(Color.WHITE);
        initContextMenuPanel();
        initContextMenuEdge();
        graphDisplay.init(controller.graph);
        graphDisplay.addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                if (e.getButton() == MouseEvent.BUTTON3) {
                    showPanelContextMenu(e);
                }
                if (e.getButton() == MouseEvent.BUTTON1) {
```

```
                    deselectVertex();
                }
            }

            @Override
            public void mouseReleased(MouseEvent e) {
                if (e.getButton() == MouseEvent.BUTTON3) {
                    showPanelContextMenu(e);
                }
            }

        });
        graphDisplay.addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseMoved(MouseEvent e) {
                if (selected instanceof GraphVertex) {
                    GraphVertex gv = (GraphVertex) selected;
                    graphDisplay.getTempEdge().setLine(gv.getX() +
GraphElement.radius, gv.getY() + GraphElement.radius, e.getX(), e.getY());
                    graphDisplay.repaint();
                }
            }
        });
        runListener = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                runAlgorithm();
            }
        };
        backToDesignListener = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                backToDesign();
            }
        };
        Setting.getInstance().addSettingChangedListener(this);
        Setting.getInstance().setRunningMode(Setting.MODE_GRAPH_DESIGN);

    }

    private void deselectVertex() {
        System.out.println("deselected");
```

```
        selected = null;
        graphDisplay.setTempEdge(null);
        repaint();
    }

    @Override
    public void onSettingChanged() {
        if (Setting.getInstance().getRunningMode() ==
Setting.MODE_ALGORITHM_VISUALIZING) {
            if (btnRun.getActionListeners().length > 0) {
                btnRun.removeActionListener(runListener);
            }
            btnRun.addActionListener(backToDesignListener);
            btnRun.setText("Back to design mode");
            for(Component c : panelStepNav.getComponents()) c.setEnabled(true);
        } else if (Setting.getInstance().getRunningMode() ==
Setting.MODE_GRAPH_DESIGN) {
            if (btnRun.getActionListeners().length > 0) {
                btnRun.removeActionListener(backToDesignListener);
            }
            btnRun.addActionListener(runListener);
            btnRun.setText("Run");
            controller.graph.resetFlow();
            controller.graph.initPrevFlow();
            graphDisplay.loadStepGraph(controller.graph);
            graphDisplay.repaint();
            for(Component c : panelStepNav.getComponents()) c.setEnabled(false);
        }
    }

    private void runAlgorithm() {
        try {
            long time = System.currentTimeMillis();
            controller.process();
            long duration = System.currentTimeMillis() - time;
            lblRunStatus.setText("<html>Run status: Completed in " + (duration /
1000f) + "s. <br>Total flow: " +
controller.getStep(controller.getNumberOfStep()).getTotalFlow() + "</html>");
            goToStep(controller.getNumberOfStep());

Setting.getInstance().setRunningMode(Setting.MODE_ALGORITHM_VISUALIZING);
        } catch (Exception ex) {
```

```
            JOptionPane.showMessageDialog(this, ex.getMessage());
            return;
        }
    }

    private void backToDesign() {
        Setting.getInstance().setRunningMode(Setting.MODE_GRAPH_DESIGN);

    }

    private void initContextMenuEdge() {
        contextMenuEdge = new JPopupMenu();

        JMenuItem deleteEdgeMI = new JMenuItem("Remove edge");
        deleteEdgeMI.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    if (MainWindow.this.selected instanceof GraphEdge) {
                        Edge edge = ((GraphEdge)
MainWindow.this.selected).getData();
                        controller.deleteEdge(edge);
                        graphDisplay.deleteEdge(edge);
                    }
                } catch (VertexNotFoundException ex) {
                    Logger.getLogger(MainWindow.class
                            .getName()).log(Level.SEVERE, null, ex);
                }
            }
        });

        contextMenuEdge.add(deleteEdgeMI);
    }

    private void initContextMenuPanel() {
        contextMenuPanel = new JPopupMenu();
        JMenuItem addVertexMI = new JMenuItem("Add vertex");

        addVertexMI.addActionListener((ActionEvent e) -> {
            Point pos = graphDisplay.getMousePosition();

            String content = JOptionPane.showInputDialog("Input vertex name");
```

```
            if (content.length() > 0) {
                Vertex newV = null;
                try {
                    newV = controller.addVertex(content);
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(this, ex.getMessage());

Logger.getLogger(MainWindow.class.getName()).log(Level.SEVERE, null, ex);
                }
                graphDisplay.addVertex(newV, pos);
            }
        });

        contextMenuPanel.add(addVertexMI);

    }

    private void showPanelContextMenu(MouseEvent e) {
        if (e.isPopupTrigger() && Setting.getInstance().getRunningMode() ==
Setting.MODE_GRAPH_DESIGN) {
            GraphEdge edge = graphDisplay.hasEdgeAt(e.getX(), e.getY());
            if (edge == null) {
                contextMenuPanel.show(e.getComponent(), e.getX(), e.getY());
            } else {
                this.selected = edge;
                contextMenuEdge.show(e.getComponent(), e.getX(), e.getY());
            }
        }
    }

    @Override
    public void onVertexSelected(GraphVertex v, EventObject event) {
        if (event instanceof ActionEvent) {
            System.out.println("selected1");
            this.selected = v;
            graphDisplay.setTempEdge(new Line2D.Double(v.getX() +
GraphElement.radius, v.getY() + GraphElement.radius, getMousePosition().getX(),
getMousePosition().getY()));
        } else if (event instanceof MouseEvent) {
            if (this.selected instanceof GraphVertex) {
                addEdge((GraphVertex) this.selected, v);
                deselectVertex();
```

```
            }
        }
    }

    private void addEdge(GraphVertex v1, GraphVertex v2) {
        try {
            int capacity = Integer.parseInt(JOptionPane.showInputDialog("Input
capacity for the new edge"));
            Edge newEdge = controller.addEdge(v1.getData().getName(),
v2.getData().getName(), capacity);
            graphDisplay.addEdge(newEdge);
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Capacity must be a decimal
number");
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, ex.getMessage());
            Logger.getLogger(MainWindow.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }

    @Override
    public void onSourceChanged(GraphVertex newSource) {
        try {
            controller.setSource(newSource.getData().getName());

        } catch (Exception ex) {
            Logger.getLogger(MainWindow.class
                    .getName()).log(Level.SEVERE, null, ex);
        }
    }

    @Override
    public void onSinkChanged(GraphVertex newSink) {
        try {
            controller.setSink(newSink.getData().getName());

        } catch (Exception ex) {
            Logger.getLogger(MainWindow.class
                    .getName()).log(Level.SEVERE, null, ex);
        }
```

```
    }

    @Override
    public void onDelete(GraphVertex v) {
        String name = v.getData().getName();
        try {
            DeleteVertexActionResult result = controller.deleteVertex(name);
            graphDisplay.deleteVertex(result.getDeleted());
            graphDisplay.deleteEdges(result.getAffectedEdges());

        } catch (VertexNotFoundException ex) {
            Logger.getLogger(MainWindow.class
                    .getName()).log(Level.SEVERE, null, ex);
        }
    }

    @Override
    public void onVertexPositionChanged() {
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        drawingPanel = new javax.swing.JPanel();
        tpanelFunction = new javax.swing.JTabbedPane();
        panelRun = new javax.swing.JPanel();
        btnRun = new javax.swing.JButton();
        lblRunStatus = new javax.swing.JLabel();
        panelStepNav = new javax.swing.JPanel();
        btnPrevStep = new javax.swing.JButton();
        btnNextStep = new javax.swing.JButton();
        lblStep = new javax.swing.JLabel();
        panelLog = new javax.swing.JPanel();
        scrLog = new javax.swing.JScrollPane();
        txtLog = new javax.swing.JTextArea();
```

```
        menuBar = new javax.swing.JMenuBar();
        mnFile = new javax.swing.JMenu();
        mnLoad = new javax.swing.JMenuItem();
        mnSave = new javax.swing.JMenuItem();
        separator1 = new javax.swing.JPopupMenu.Separator();
        mnExit = new javax.swing.JMenuItem();
        mnAbout = new javax.swing.JMenu();
        mnAboutProg = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        drawingPanel.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        javax.swing.GroupLayout drawingPanelLayout = new
javax.swing.GroupLayout(drawingPanel);
        drawingPanel.setLayout(drawingPanelLayout);
        drawingPanelLayout.setHorizontalGroup(

drawingPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGap(0, 445, Short.MAX_VALUE)
        );
        drawingPanelLayout.setVerticalGroup(

drawingPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGap(0, 0, Short.MAX_VALUE)
        );

        btnRun.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnRunActionPerformed(evt);
            }
        });

        lblRunStatus.setText("Run Status");

        btnPrevStep.setText("Previous");
        btnPrevStep.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnPrevStepActionPerformed(evt);
```

```
            }
        });

        btnNextStep.setText("Next");
        btnNextStep.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnNextStepActionPerformed(evt);
            }
        });

        lblStep.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        lblStep.setText("step / total_step");
        lblStep.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        javax.swing.GroupLayout panelStepNavLayout = new
javax.swing.GroupLayout(panelStepNav);
        panelStepNav.setLayout(panelStepNavLayout);
        panelStepNavLayout.setHorizontalGroup(

panelStepNavLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(panelStepNavLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(btnPrevStep)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(lblStep, javax.swing.GroupLayout.DEFAULT_SIZE,
142, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(btnNextStep)
                .addContainerGap())
        );
        panelStepNavLayout.setVerticalGroup(

panelStepNavLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(panelStepNavLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panelStepNavLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
```

```
                    .addComponent(btnPrevStep)
                    .addComponent(btnNextStep)
                    .addComponent(lblStep))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        javax.swing.GroupLayout panelRunLayout = new
javax.swing.GroupLayout(panelRun);
        panelRun.setLayout(panelRunLayout);
        panelRunLayout.setHorizontalGroup(

panelRunLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelRunLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panelRunLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                    .addGroup(panelRunLayout.createSequentialGroup()
                        .addComponent(lblRunStatus,
javax.swing.GroupLayout.PREFERRED_SIZE, 290,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 8, Short.MAX_VALUE))
                    .addComponent(btnRun, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(panelStepNav,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addContainerGap())
        );
        panelRunLayout.setVerticalGroup(

panelRunLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelRunLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(btnRun)
                .addGap(18, 18, 18)
                .addComponent(lblRunStatus,
javax.swing.GroupLayout.PREFERRED_SIZE, 45,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                .addComponent(panelStepNav,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(511, Short.MAX_VALUE))
        );

        tpanelFunction.addTab("Functions", panelRun);

        txtLog.setColumns(20);
        txtLog.setRows(5);
        scrLog.setViewportView(txtLog);

        javax.swing.GroupLayout panelLogLayout = new
javax.swing.GroupLayout(panelLog);
        panelLog.setLayout(panelLogLayout);
        panelLogLayout.setHorizontalGroup(

panelLogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelLogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(scrLog, javax.swing.GroupLayout.DEFAULT_SIZE, 298,
Short.MAX_VALUE)
                .addContainerGap())
        );
        panelLogLayout.setVerticalGroup(

panelLogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelLogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(scrLog, javax.swing.GroupLayout.DEFAULT_SIZE, 623,
Short.MAX_VALUE)
                .addContainerGap())
        );

        tpanelFunction.addTab("Log", panelLog);

        mnFile.setText("File");

        mnLoad.setText("Load");
        mnLoad.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnLoadActionPerformed(evt);
```

```
            }
        });
        mnFile.add(mnLoad);

        mnSave.setText("Save");
        mnSave.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnSaveActionPerformed(evt);
            }
        });
        mnFile.add(mnSave);
        mnFile.add(separator1);

        mnExit.setText("Exit");
        mnExit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnExitActionPerformed(evt);
            }
        });
        mnFile.add(mnExit);

        menuBar.add(mnFile);

        mnAbout.setText("About");

        mnAboutProg.setText("About");
        mnAboutProg.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnAboutProgActionPerformed(evt);
            }
        });
        mnAbout.add(mnAboutProg);

        menuBar.add(mnAbout);

        setJMenuBar(menuBar);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(drawingPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tpanelFunction,
javax.swing.GroupLayout.PREFERRED_SIZE, 323,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(tpanelFunction)
                    .addComponent(drawingPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addContainerGap())
        );

        tpanelFunction.getAccessibleContext().setAccessibleName("Run");

        pack();
    }// </editor-fold>//GEN-END:initComponents

    void reloadStepDisplay() {
        lblStep.setText(step + "/" + controller.getNumberOfStep());
    }

    void goToStep(int step) {
        this.step = step;
        this.reloadStepDisplay();
        graphDisplay.loadStepGraph(controller.getStep(step));
        graphDisplay.repaint();
```

```
    }


    private void mnLoadActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_mnLoadActionPerformed
        JFileChooser chooser = new JFileChooser();
//        FileNameExtensionFilter filter = new FileNameExtensionFilter(
//                "JPG & GIF Images", "jpg", "gif");
//        chooser.setFileFilter(filter);
        int returnVal = chooser.showOpenDialog(this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            try {

controller.loadFile(chooser.getSelectedFile().getAbsolutePath());
                graphDisplay.init(controller.graph);
                graphDisplay.repaint();

            } catch (IOException ex) {
                Logger.getLogger(MainWindow.class
                        .getName()).log(Level.SEVERE, null, ex);

            } catch (Exception ex) {
                Logger.getLogger(MainWindow.class
                        .getName()).log(Level.SEVERE, null, ex);
            }
        }
    }//GEN-LAST:event_mnLoadActionPerformed

    private void mnExitActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_mnExitActionPerformed
        System.exit(0);
    }//GEN-LAST:event_mnExitActionPerformed

    private void mnSaveActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_mnSaveActionPerformed
        JFileChooser chooser = new JFileChooser();
//        FileNameExtensionFilter filter = new FileNameExtensionFilter(
//                "JPG & GIF Images", "jpg", "gif");
//        chooser.setFileFilter(filter);
        int returnVal = chooser.showOpenDialog(this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            try {
```

```
                controller.saveFile(chooser.getSelectedFile().getName());

            } catch (IOException ex) {
                Logger.getLogger(MainWindow.class
                        .getName()).log(Level.SEVERE, null, ex);
            }
        }
    }//GEN-LAST:event_mnSaveActionPerformed

    private void btnPrevStepActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnPrevStepActionPerformed
        if (step > 1) {
            step--;
            goToStep(step);
        }
    }//GEN-LAST:event_btnPrevStepActionPerformed

    private void btnNextStepActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnNextStepActionPerformed
        if (step < controller.getNumberOfStep()) {
            step++;
            goToStep(step);
        }
    }//GEN-LAST:event_btnNextStepActionPerformed

    private void btnRunActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnRunActionPerformed

    }//GEN-LAST:event_btnRunActionPerformed

    private void mnAboutProgActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_mnAboutProgActionPerformed
        new AboutDialog(this, true).setVisible(true);
    }//GEN-LAST:event_mnAboutProgActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">
```

```
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;

                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(MainWindow.class
                    .getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(MainWindow.class
                    .getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(MainWindow.class
                    .getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(MainWindow.class
                    .getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MainWindow().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnNextStep;
```

```
    private javax.swing.JButton btnPrevStep;
    private javax.swing.JButton btnRun;
    private javax.swing.JPanel drawingPanel;
    private javax.swing.JLabel lblRunStatus;
    private javax.swing.JLabel lblStep;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JMenu mnAbout;
    private javax.swing.JMenuItem mnAboutProg;
    private javax.swing.JMenuItem mnExit;
    private javax.swing.JMenu mnFile;
    private javax.swing.JMenuItem mnLoad;
    private javax.swing.JMenuItem mnSave;
    private javax.swing.JPanel panelLog;
    private javax.swing.JPanel panelRun;
    private javax.swing.JPanel panelStepNav;
    private javax.swing.JScrollPane scrLog;
    private javax.swing.JPopupMenu.Separator separator1;
    private javax.swing.JTabbedPane tpanelFunction;
    private javax.swing.JTextArea txtLog;
    // End of variables declaration//GEN-END:variables

}
```

>>>>>>>>>>>>>>>>>> Setting.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ui;

import java.util.LinkedList;

/**
 *
 * @author theph
 */
public class Setting {

    public static Setting instance = null;
```

```
    public static Setting getInstance() {
        if (instance == null) {
            instance = new Setting();
        }
        return instance;
    }

    LinkedList<SettingChangedListener> listeners;

    private Setting() {
        this.listeners = new LinkedList<>();
    }

    public void addSettingChangedListener(SettingChangedListener l) {
        listeners.add(l);
    }

    public static final int MODE_GRAPH_DESIGN = 1;
    public static final int MODE_ALGORITHM_VISUALIZING = 2;
    int runningMode;

    public int getRunningMode() {
        return runningMode;
    }

    public void setRunningMode(int runningMode) {
        this.runningMode = runningMode;
        for (SettingChangedListener scl : listeners) {
            scl.onSettingChanged();
        }
    }
}
```

>>>>>>>>>>>>>>>>>>> SettingChangedListener.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package ui;

/**
 *
 * @author theph
 */
public interface SettingChangedListener {
    void onSettingChanged();
}


>>>>>>>>>>>>>>>>>>> VertexActionListener.java

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ui;

import java.awt.event.MouseEvent;
import java.util.EventObject;
import model.Vertex;

/**
 *
 * @author theph
 */
public interface VertexActionListener {

    void onSourceChanged(GraphVertex newSource);

    void onSinkChanged(GraphVertex newSink);

    void onVertexPositionChanged();

    void onDelete(GraphVertex v);

    void onVertexSelected(GraphVertex v, EventObject event);
}
```

```
>>>>>>>>>>>>>>>>>>> VertexNotFoundException.java

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ui;

/**
 *
 * @author theph
 */
public class VertexNotFoundException extends Exception {


}


>>>>>>>>>>>>>>>>>>> LogicTest.java

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Test;

import controller.Controller;
import java.io.IOException;
import java.util.LinkedList;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.BFS;
import model.DFS;
import model.FordFulkerson;
import model.Graph;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
```

```
import static org.junit.Assert.*;
import ui.VertexNotFoundException;

/**
 *
 * @author duyenNH
 */
public class LogicTest {

    private Graph graph;

    public LogicTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() throws Exception {
        graph = new Graph();

        graph.addVertex("A");
        graph.addVertex("B");
        graph.addVertex("C");
        graph.addVertex("D");
        graph.addVertex("E");
        graph.addVertex("F");

        graph.addEdge(graph.getVertexByName("A"), graph.getVertexByName("B"),
8);
        graph.addEdge(graph.getVertexByName("A"), graph.getVertexByName("C"),
2);
        graph.addEdge(graph.getVertexByName("B"), graph.getVertexByName("D"),
6);
        graph.addEdge(graph.getVertexByName("C"), graph.getVertexByName("E"),
5);
```

```
        graph.addEdge(graph.getVertexByName("D"), graph.getVertexByName("C"),
2);
        graph.addEdge(graph.getVertexByName("E"), graph.getVertexByName("B"),
3);
        graph.addEdge(graph.getVertexByName("D"), graph.getVertexByName("F"),
4);
        graph.addEdge(graph.getVertexByName("E"), graph.getVertexByName("F"),
5);

        graph.setSource(graph.getVertexByName("A"));
        graph.setSink(graph.getVertexByName("F"));

    }

    @After
    public void tearDown() {
    }

    @org.junit.Test
    public void testAddVertex() throws Exception {
        assertEquals(6, graph.getVrtx().size());

        graph.addVertex("G");

        assertEquals(7, graph.getVrtx().size());
        assertEquals("G", graph.getVrtx().get(6).getName());
        assertEquals(0, graph.getVrtx().get(6).getNeighbours().size());
    }

    @org.junit.Test(expected = Exception.class)
    public void testAddVrtxExists() throws Exception {
        graph.addVertex("A");
    }

    @org.junit.Test
    public void testAddEdges() throws Exception {
        assertEquals(8, graph.getEdges().size());

        graph.addEdge(graph.getVertexByName("C"), graph.getVertexByName("F"),
3);
```

```
        assertEquals(4, graph.getVrtx().get(2).getNeighbours().size()); //vertex
C
        assertEquals(9, graph.getEdges().size());
    }

    @org.junit.Test(expected = Exception.class)
    public void testAddEdgesExists() throws Exception {
        graph.addEdge(graph.getVertexByName("C"), graph.getVertexByName("E"),
4);
    }

    @org.junit.Test(expected = NullPointerException.class)
    public void testAddEdgesWithoutStrartVrtx() throws Exception {
        graph.addEdge(graph.getVertexByName("Z"), graph.getVertexByName("B"),
9);
    }

    @org.junit.Test(expected = NullPointerException.class)
    public void testAddEdgesWithoutEndVrtx() throws Exception {
        graph.addEdge(graph.getVertexByName("C"), graph.getVertexByName("I"),
7);
    }

    @org.junit.Test(expected = VertexNotFoundException.class)
    public void testDelVertexNotFoundException() throws VertexNotFoundException
{
        graph.deleteVertex(graph.getVertexByName("S"));
    }

    @org.junit.Test
    public void testDelVertex() throws VertexNotFoundException {
        assertNotNull(graph.getVertexByName("C"));

        graph.deleteVertex(graph.getVertexByName("C"));

        assertEquals(5, graph.getVrtx().size());
        assertNull(graph.getVertexByName("C"));
    }

    @org.junit.Test
    public void testDelVertexSource() throws VertexNotFoundException {
        graph.deleteVertex(graph.getSource());
```

```
        assertNull(graph.getSource());
    }

    @org.junit.Test
    public void testDelVertexSink() throws VertexNotFoundException {
        graph.deleteVertex(graph.getSink());
        assertNull(graph.getSink());
    }

    @org.junit.Test
    public void testDeleteEdge() throws VertexNotFoundException {
        assertEquals(3, graph.getVrtx().get(1).getNeighbours().size());//Vertex
B

        graph.deleteEdge(graph.getVertexByName("B"),
graph.getVertexByName("D"));

        assertEquals(2, graph.getVrtx().get(3).getNeighbours().size());//Vertex
B
        assertEquals(7, graph.getEdges().size());
        assertNotNull(graph.getVertexByName("B"));
        assertNotNull(graph.getVertexByName("D"));
    }

    @org.junit.Test
    public void testInputFile() throws IOException, Exception {
        Controller c = new Controller();

        c.loadFile("file.txt");

        assertEquals(3, c.graph.getVrtx().size());
        assertEquals(2, c.graph.getEdges().size());
        assertEquals(2, c.graph.getVrtx().get(1).getNeighbours().size());
        assertEquals("A", c.graph.getSource().getName());
        assertEquals("C", c.graph.getSink().getName());
    }

    @org.junit.Test
    public void testIOFile() throws IOException, Exception {
        Controller c = new Controller();
        c.graph = graph;
```

```
        c.saveFile("test2.txt");

        // TODO review the generated test code and remove the default call to
fail.
        c.loadFile("test2.txt");
        assertEquals(6, c.graph.getVrtx().size());
        assertEquals(8, c.graph.getEdges().size());
        assertEquals("F", c.graph.getVrtx().get(5).getName());
    }

    @org.junit.Test(expected = Exception.class)
    public void testProcessWithoutSource() throws VertexNotFoundException,
Exception {
        graph.deleteVertex(graph.getSource());
        FordFulkerson.process(graph, new DFS());
    }

    @org.junit.Test(expected = Exception.class)
    public void testProcessWithoutSink() throws VertexNotFoundException,
Exception {
        graph.deleteVertex(graph.getSink());
        FordFulkerson.process(graph, new DFS());
        assertEquals(0, graph.getTotalFlow());
    }

    @org.junit.Test
    public void testProcessFFWithBFS() throws Exception {
        FordFulkerson.process(graph, new BFS());
        assertEquals(8, graph.getTotalFlow());
    }

    @org.junit.Test
    public void testProcessFFWithDFS() throws Exception {
        FordFulkerson.process(graph, new DFS());

        assertEquals(8, graph.getTotalFlow());
    }

    @org.junit.Test
    public void testProcess2() throws Exception {
        LinkedList<Graph> result = FordFulkerson.process(graph, new BFS());
        assertEquals(8, graph.getTotalFlow());
```

```
        }
    }
```