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

ОТЧЕТ

по учебной практике

Тема: Поиск мостов в графе

Студент гр. 8304 _____ Масалыкин Д.Р.

Студент гр. 8304 _____ Чешуин Д.И.

Студент гр. 8304 _____ Матросов Д.В.

Руководитель _____ Размочаева Н.В.

Санкт-Петербург

2020

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Масалыкин Д.Р. группы 8304

Студент Чешуин Д.И. группы 8304

Студент Матросов Д.В. группы 8304

Тема практики: Поиск мостов в графе

Задание на практику:

Командная итеративная разработка визуализатора алгоритма(ов) на Java с графическим интерфейсом.

Алгоритм: Модифицированный поиск в глубину для поиска мостов в неориентированном графе.

Сроки прохождения практики: 29.06.2020 – 12.07.2020

Дата сдачи отчета: 10.07.2020

Дата защиты отчета: 10.07.2020

Студент _____ Масалыкин Д.Р.

Студент _____ Чешуин Д.И.

Студент _____ Матросов Д.В.

Руководитель _____ Размочаева Н.В.

АННОТАЦИЯ

Целью работы является получения навыков работы с такой парадигмой программирования, как объектно-ориентированное программирование. Для получения данных знаний выполняется один из вариантов мини-проекта. В процессе выполнения мини-проекта необходимо реализовать графический интерфейс к данной задаче, организовать ввод и вывод данных с его помощь, реализовать сам алгоритм, научиться работать в команде. В данной работе в качестве мини-проекта выступает задача поиска мостов в неориентированном невзвешенном графе. Также при разработке выполняется написание тестирования, для проверки корректности алгоритма.

SUMMARY

The aim of the work is to gain skills in working with such a programming paradigm as object-oriented programming. To obtain this knowledge, one of the mini-project options is performed. In the process of implementing a mini-project, it is necessary to implement a graphical interface to this task, organize the input and output of data with its help, implement the algorithm itself, learn how to work in a team. In this paper, the task of finding bridges in an undirected unweighted graph is a mini-project. Also during development, testing is written to verify the correctness of the algorithm.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Основная цель практики – реализация мини-проекта, который является визуализацией алгоритма. В данной работе такой алгоритм – это поиск мостов в неориентированном невзвешенном графе. Для выполнения этой цели были поставлены задачи: разработка GUI к проекту, разработка алгоритма и создание формата файла для хранения графа.

Алгоритм является модифицированным поиском в глубину, который использует время входа в вершину для нахождения мостов. Для ввода графа используется редактор с возможностью добавления и удаления вершины или ребра. Также показывается стек с вершинами. Помимо этого граф можно загрузить из файла, а также сохранить созданный граф в файл. Для этого используется файл с расширением .graph.

1 ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1 Исходные требования

1.1.1 Требования к входным данным

Для корректной работы алгоритма требуется:

– Имя файла, содержащего граф, корректно записанный

– Ввод графа вручную через редактор

1.1.2 Требования к визуализации

Программа должна обладать простым и интуитивно понятным интерфейсом. Прототип интерфейса представлен на рисунках 1- 6.

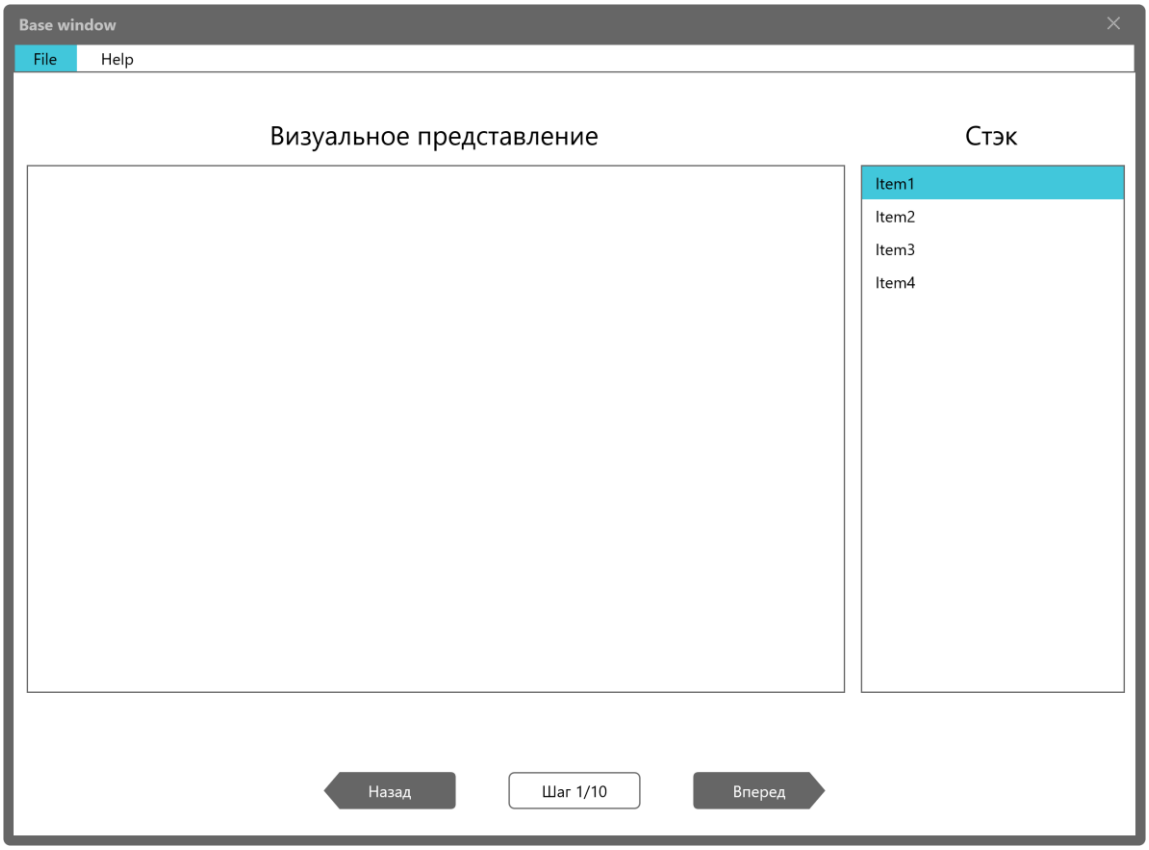

Рисунок 1 – Главное окно.

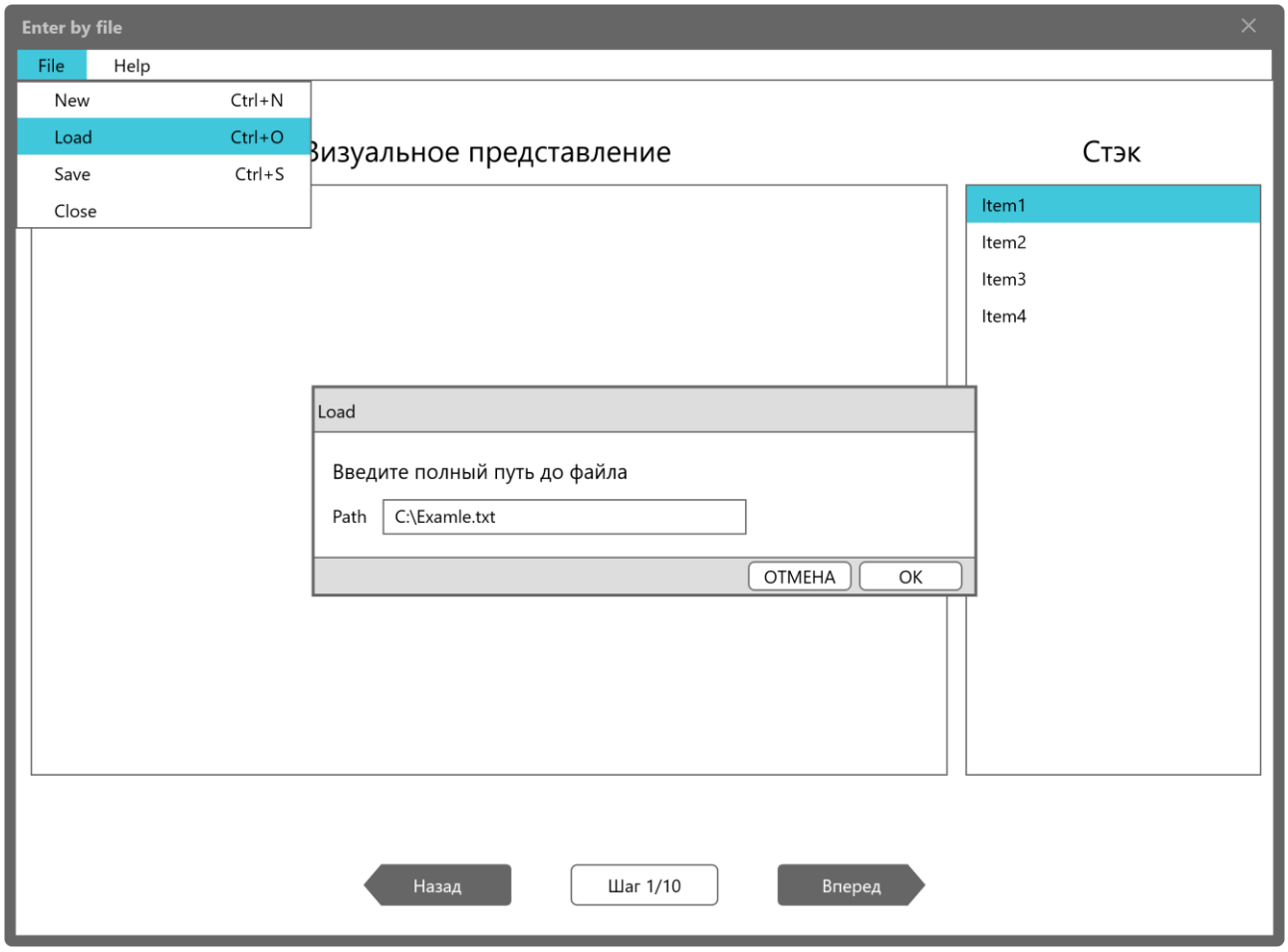

Рисунок 2 – Диалоговое окно ввода файла

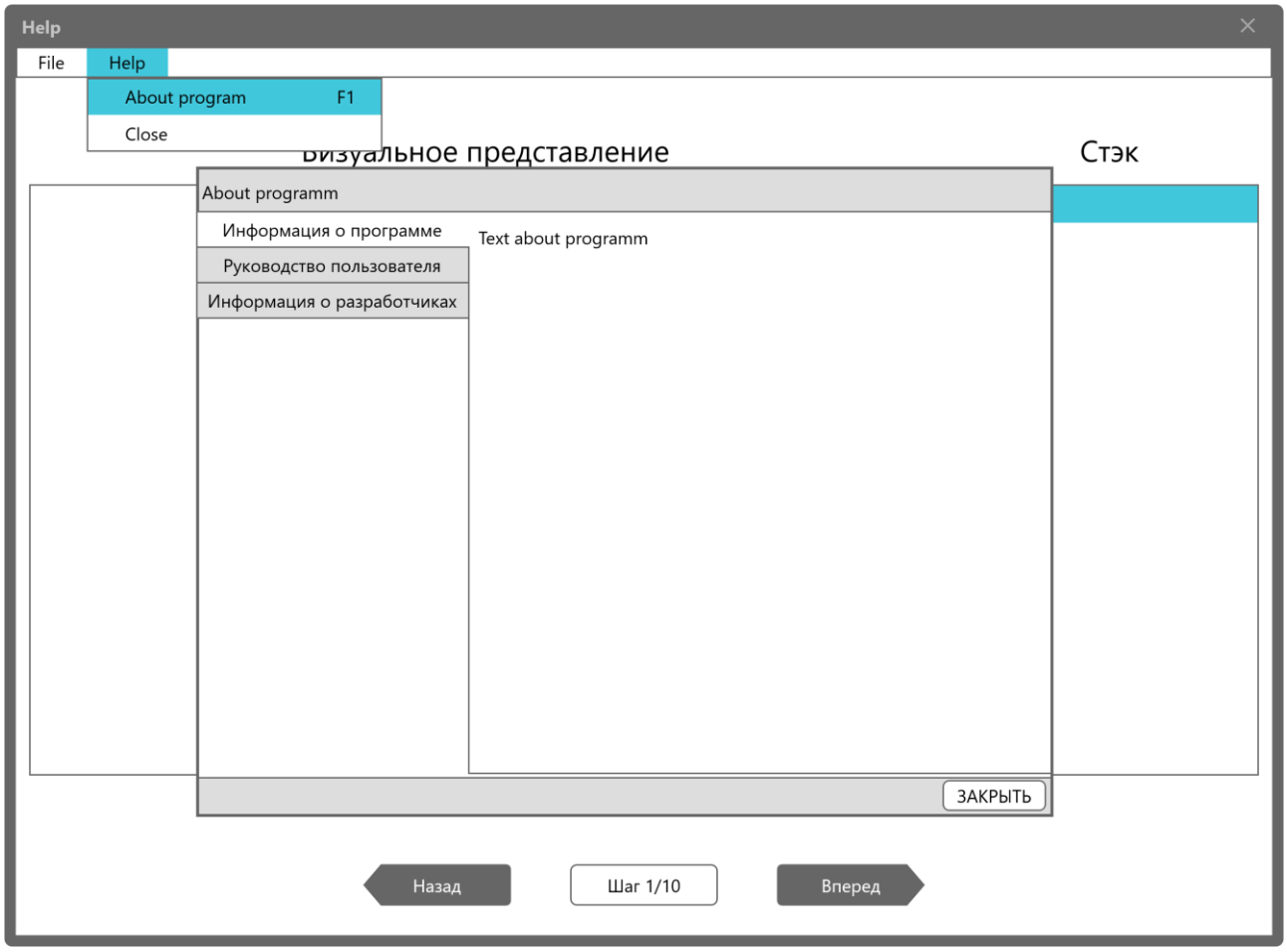

Рисунок 3 – Окно “О программе”

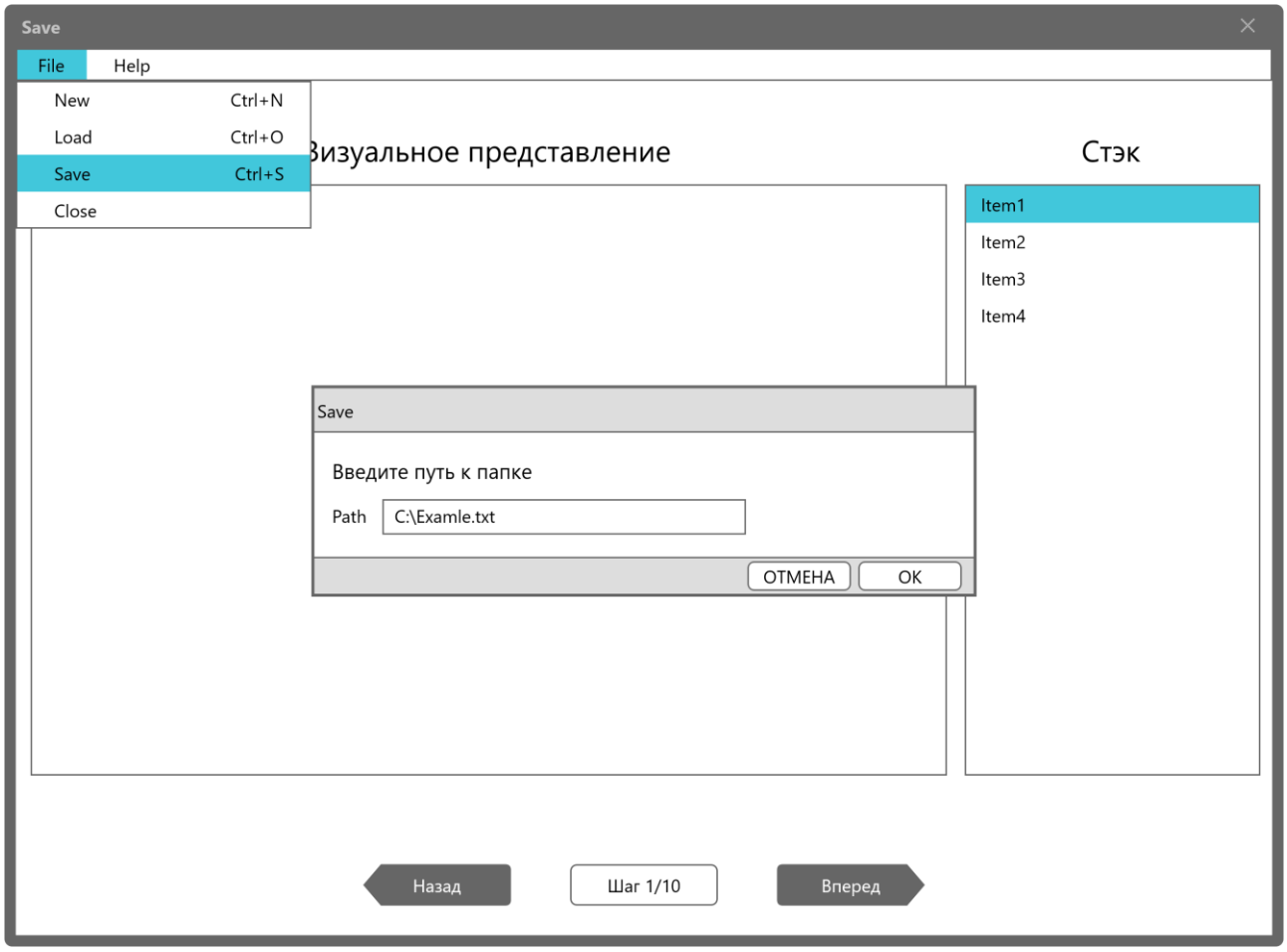

Рисунок 4 – Диалоговое окно сохранения файла

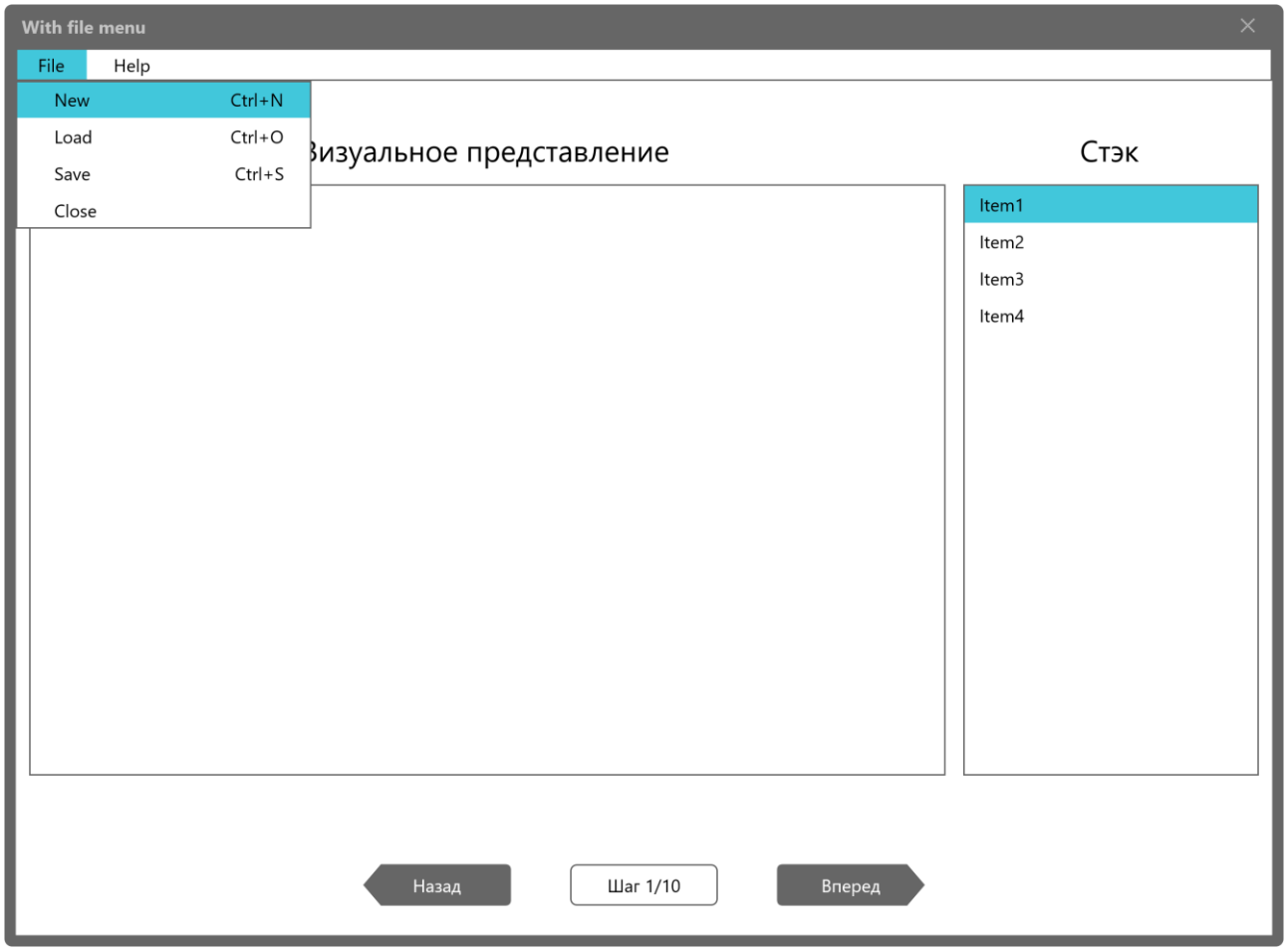

Рисунок 5 – Главное окно с файловым меню

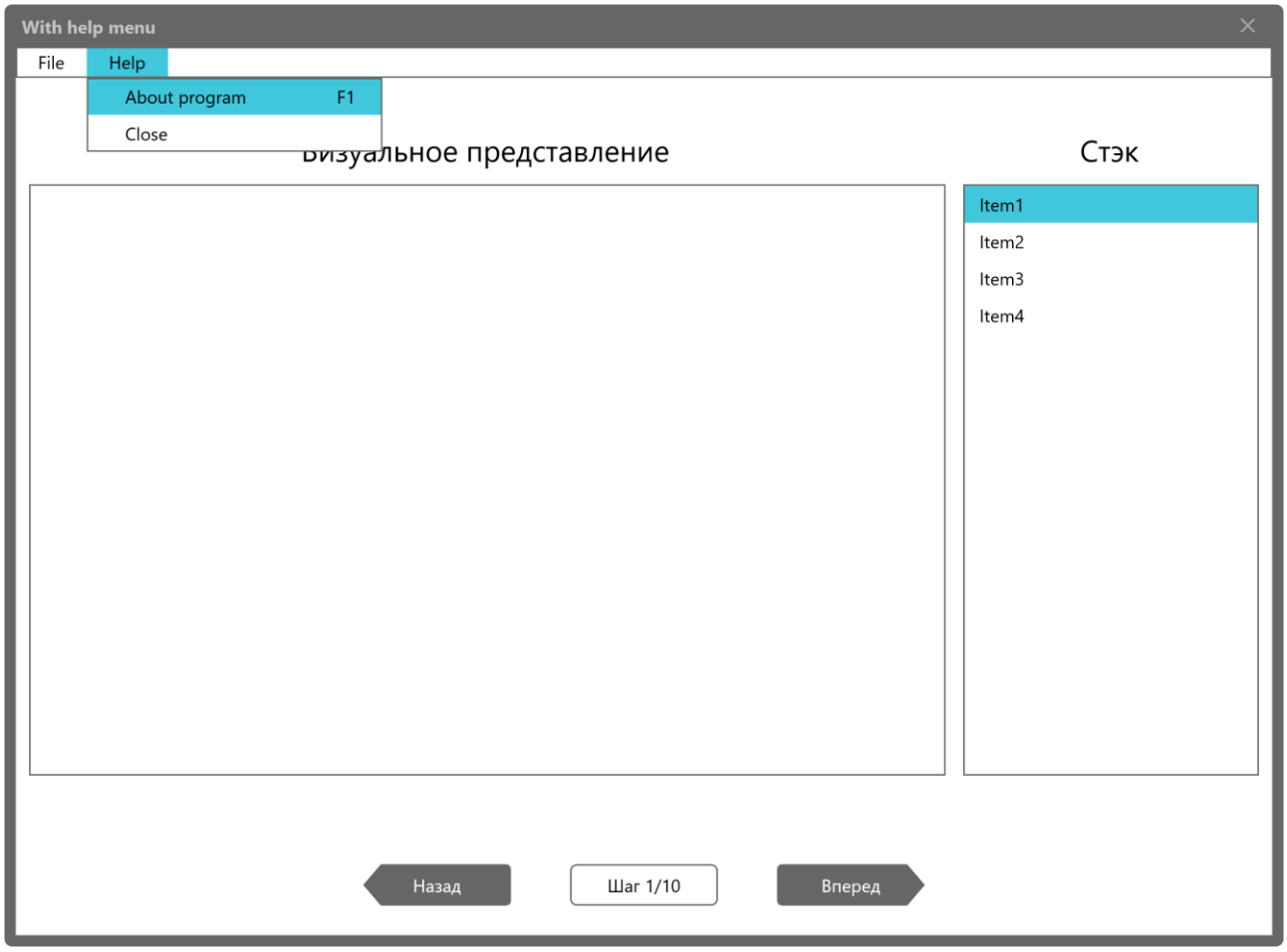

Рисунок 6 – Главное окно с меню «помощь»

1.1.3 Требования к алгоритму

Алгоритм обходит весь граф при помощи поиска в глубину и, анализирую время входа в вершину и текущий таймер определяет мосты. Алгоритм сохраняет пошаговый ход работы для визуального отображения.

1.1.4 Требования к выходным данным

Выходными данными является изображение графа с отмеченными мостами

1.2 Требования к программе после уточнения

1.2.1 Требования к входным данным

Для корректной работы алгоритма требуется:

– Имя файла, содержащего граф, корректно записанный

– Ввод графа вручную через редактор

1.2.2 Требования к визуализации

Измененный интерфейс программы представлен на рисунках 7-12.

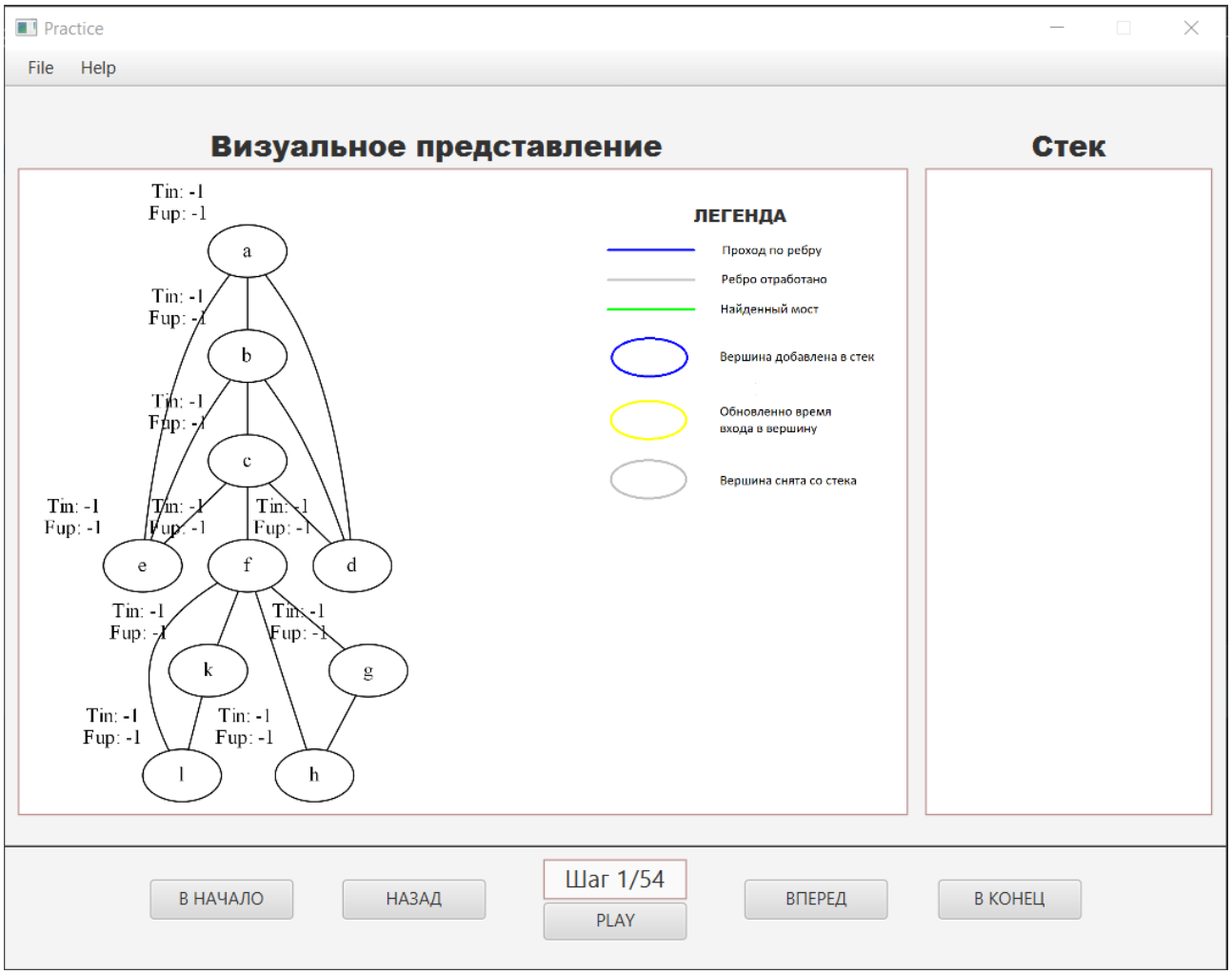

Рисунок 7 – Обновленный интерфейс и графическое отображение графа.

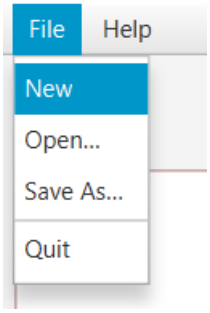

Рисунок 8 – выпадающее меню File.

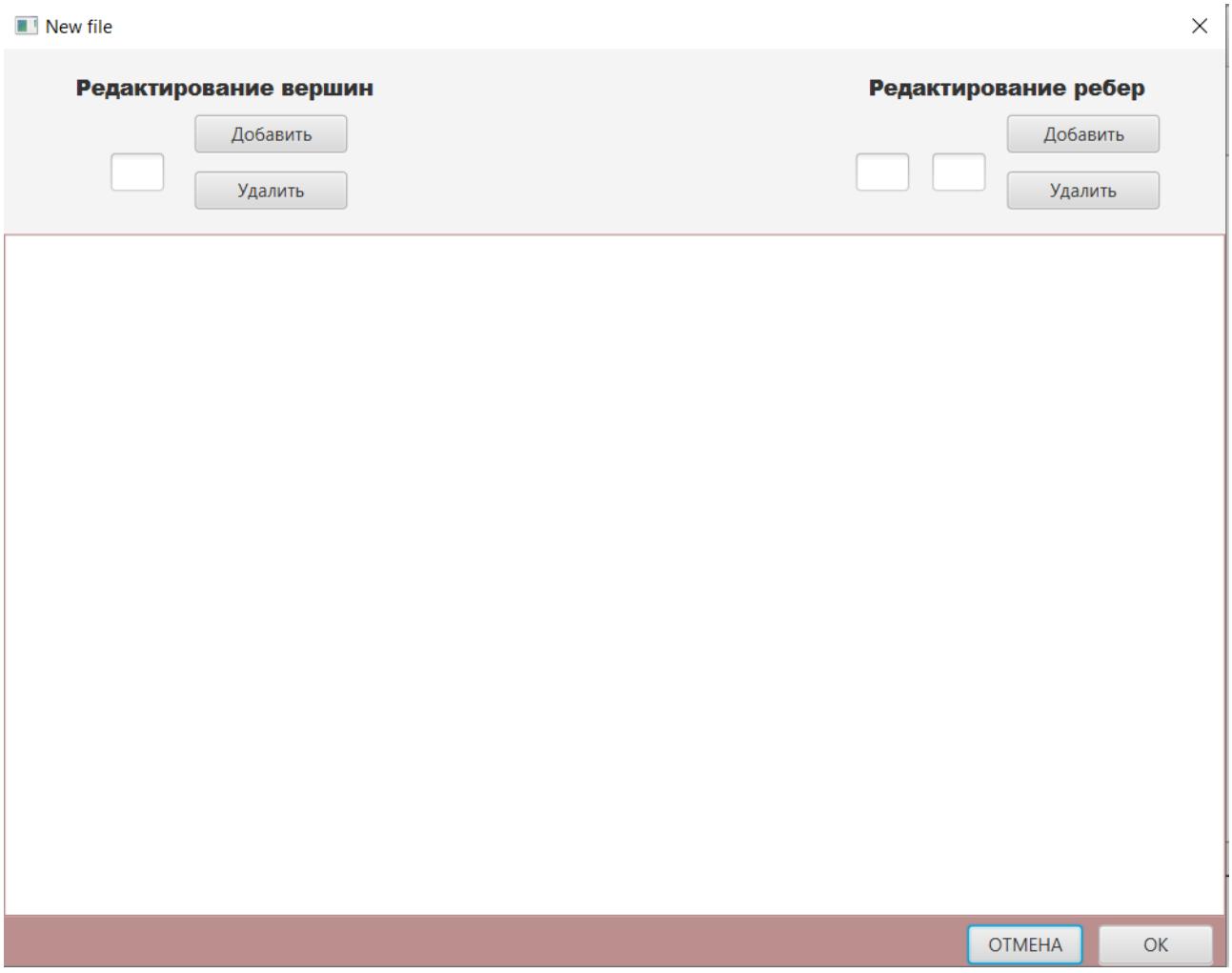

Рисунок 9 – редактор для ввода графа.

Другим способом ввода является загрузка файла с сохраненным графом. Соответствующее окно представлено на рисунке 10.

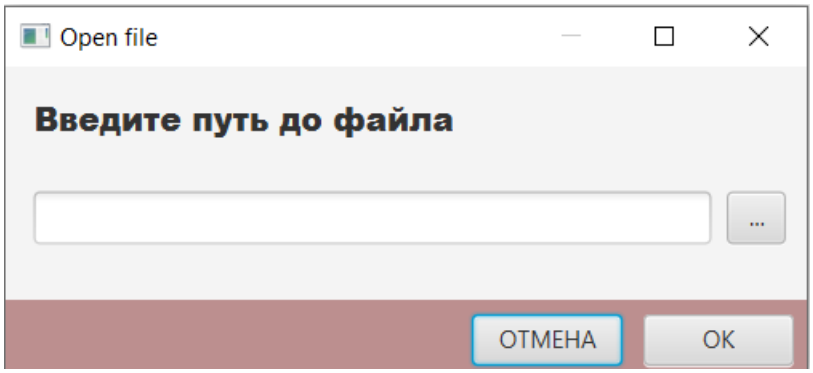

Рисунок 10 – Окно для загрузки и сохранения файла.

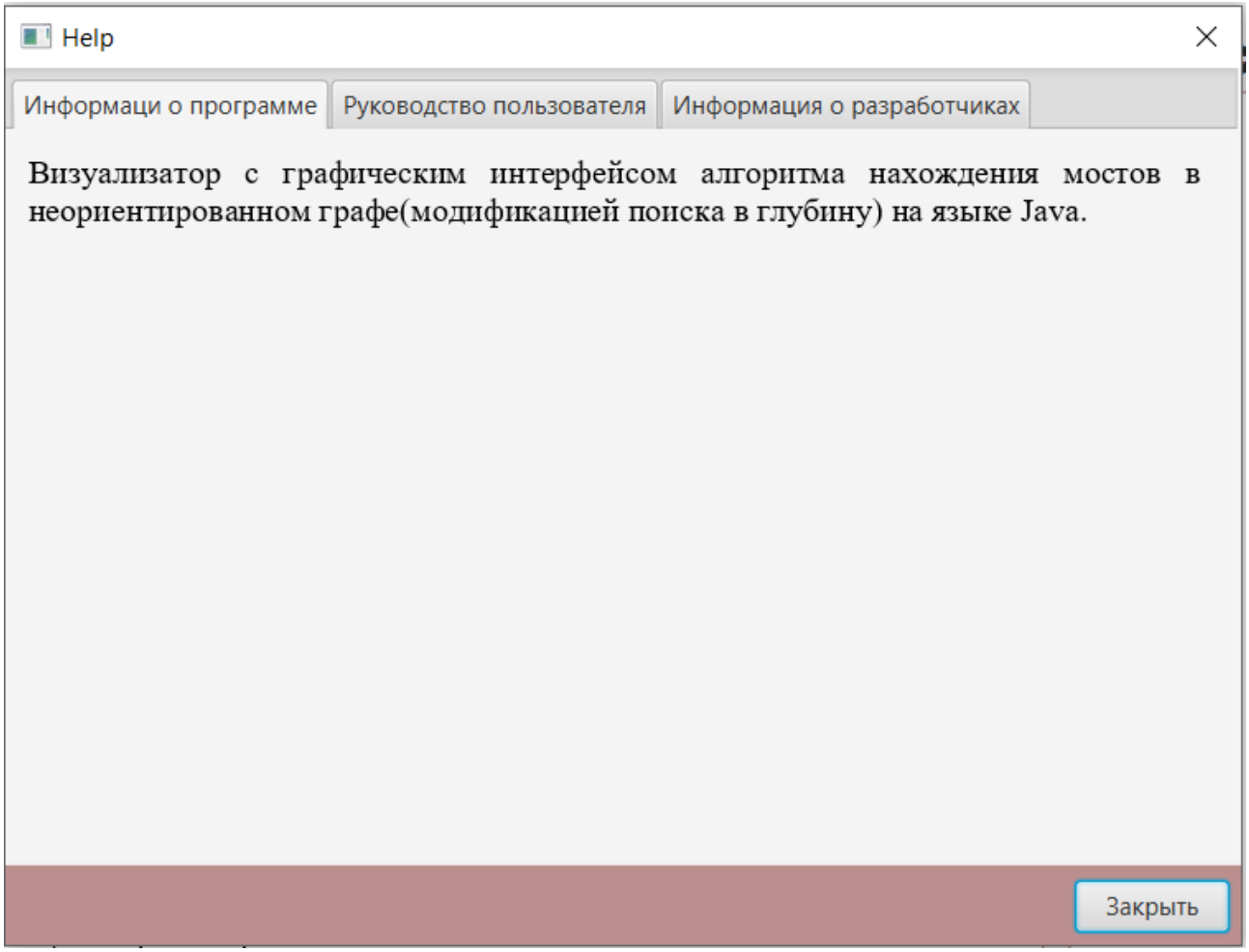

Рисунок 11 – Информация о программе.

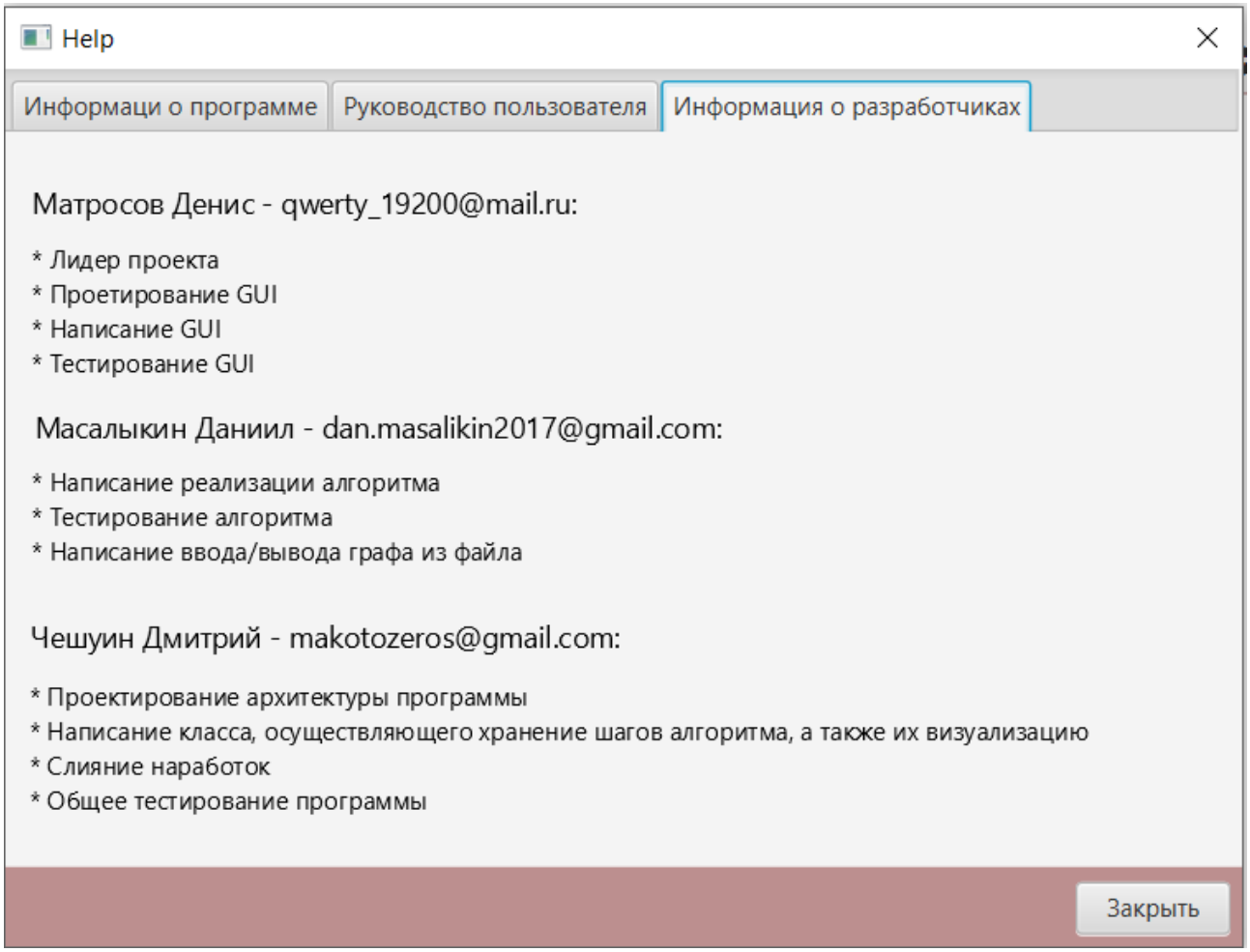

Рисунок 12 – информация о разработчиках

1.2.3 Требования к алгоритму

Алгоритм обходит весь граф при помощи поиска в глубину и, анализирую время входа в вершину и текущий таймер определяет мосты. Алгоритм сохраняет пошаговый ход работы для визуального отображения.

1.2.4 Требования к выходным данным

В качестве выходных данных используется пошаговая иллюстрация работы алгоритма(на последнем шаге гарантированно будут подсвечены зеленым все существующие мосты). Пример финального состояния показан на рисунке 8.

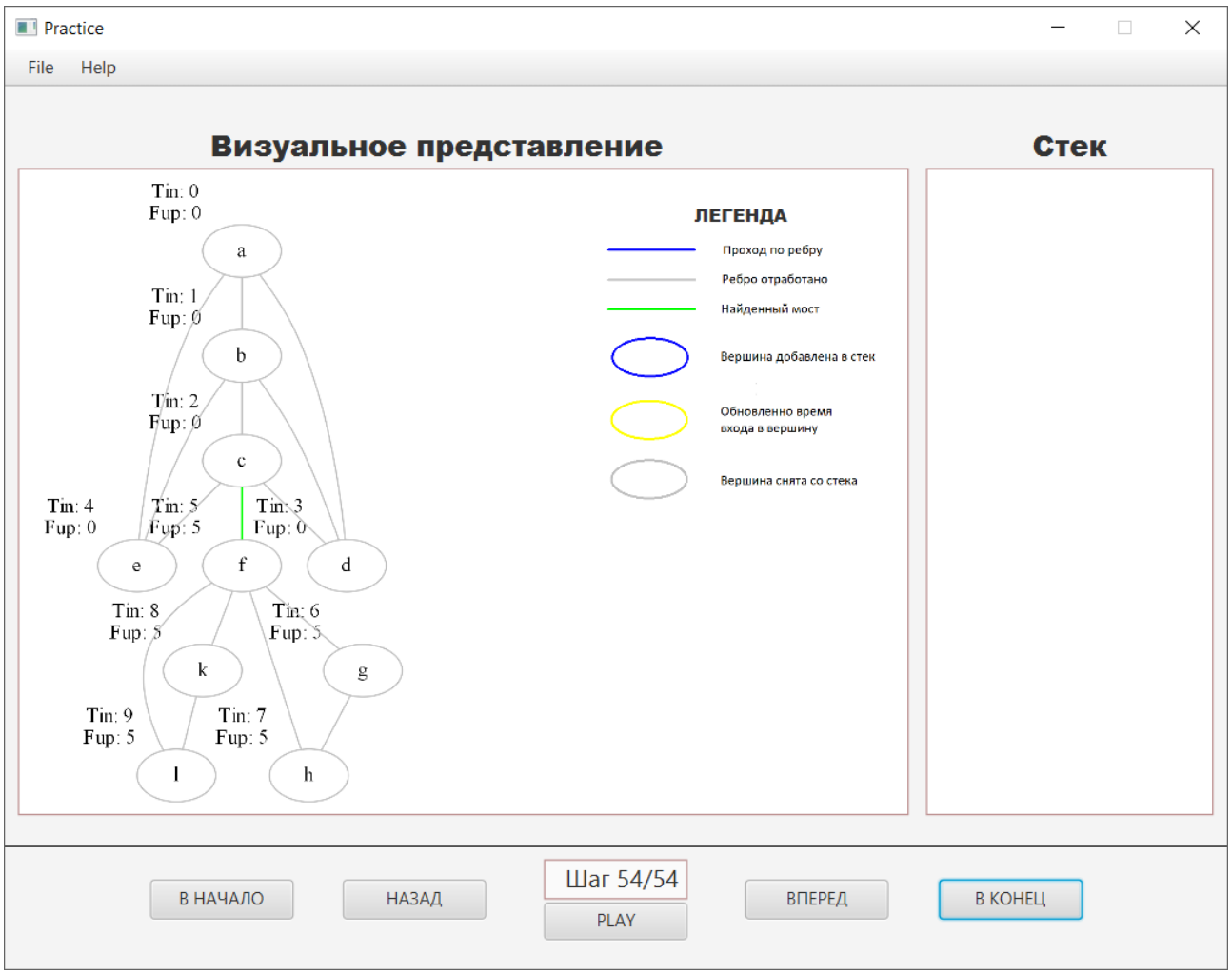

Рисунок 13 – Результат работы программы.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

- Обсудить задание, распределить роли, выбрать необходимые средства разработки и структуры данных – до 03.07
- Создать прототипа приложения – до 04.07
- Создать первую версию приложения – до 05.07
- Доработать первую версию(в том числе с учетом правок) до второй версии – до 07.07
- Создать финальную версию и начать тестирование – до 08.07
- Завершить тестирование программы – до 09.07
- Сдать проект преподавателю – до 10.07

2.2. Распределение ролей в бригаде

- Масалыкин Даниил:
  – Написание алгоритма поиска мостов
  – Реализация ввода из файла и сохранения в файл
  – написание отчета
- Матросов Денис
  – Лидер проекта
  – Написание GUI
  – Проектирование GUI
- Чешуин Дмитрий
  – Проектирование архитектуры программы
  – Написание класса, осуществляющего хранение шагов алгоритма, а также их визуализацию
  – Слияние наработок

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структуры данных

Для реализации проекта потребовалось разработать классы Node и Graph. Класс Node хранит информацию о вершине графа и ребрах, исходящих из нее. Поля класса Node:

- final private String name – имя вершины
- private final HashMap<Node, Node> edges = new HashMap<>() – ребра данной вершины.

Методы класса Node:

- Node(String name) – конструктор класса
- public boolean equals(Object o) – проверка на равенство двух объектов
- public int hashCode() – получение хэш-кода
- public String getName() – получение имени вершины
- public Set<Node> getEdges() – получение сета вершин, к которым идут ребра от данной.

Поля класса Graph:

- private final HashMap<Node, Node> node – вершины графа.

Методы класса Graph:

- public void addNode(String name) – добавление вершины
- public void deleteNode(String name) – удаление ввершины
- public boolean addEdge(String from, String to) – добавление ребра
- public void deleteEdge(String from, String to) – удаление ребра
- public Node getNode(String name) – получение вершины
- public Set<Node> getNodes() – получение вершин
- public LinkedList<String> nodesToList() – получение вершин в виде списка
- public LinkedList<Pair<String, String>> edgesToList() – получение ребер в виде списка

- public LinkedList<String> toStringList() – получение списка строк

3.2. Основные методы

Методы для реализации алгоритма находятся в классе BridgeFinder:

- public void findBridges(Graph graph) – данный метод инициализирует необходимые переменные(HashMap used, fin, fup) и запускает поиск в глубину для непросмотренных вершин.
- private void dfs(Graph.Node node, Graph.Node parent) – непосредственно поиск в глубину. Реализован с помощью рекурсии. Данный метод завершает свою работу и возвращается на предыдущий шаг рекурсии, если все смежные вершины уже просмотрены.

4. ТЕСТИРОВАНИЕ

4.1. Тестирование алгоритма

Для тестирования был запущен только алгоритм без основной программы. Результаты представлены в таблице 1

Таблица 1 – Тестирование алгоритма

Входные данные	Выходные данные
a b c d e f g h k l a b a d a e b a b c b d b e c b c d c e c f d a d b d c e a e b e c f c f g f h f k f l g f	BRIDGE f – c

g h h f h g k f k l l f l k	
b a c d a b c d	BRIDGE a – b BRIDGE c – d
a b c	«» (Ничего не было выведено т. к. нет мостов)
a b c d e f g a b a c a d a g b a b c b d b f c a c b c d c e c f d a d b d c d e d f e c e d e f e g f b	«» (Ничего не было выведено т. к. нет мостов)

f c f d f e f g g a g e g f	

4.2. Тестирование GUI и пошагового вывода

Программа была запущена с входными данными из пункта 4.1 и ожидаемый результат совпал с полученным(к примеру последний шаг 4 графа представлен на рисунке 9)

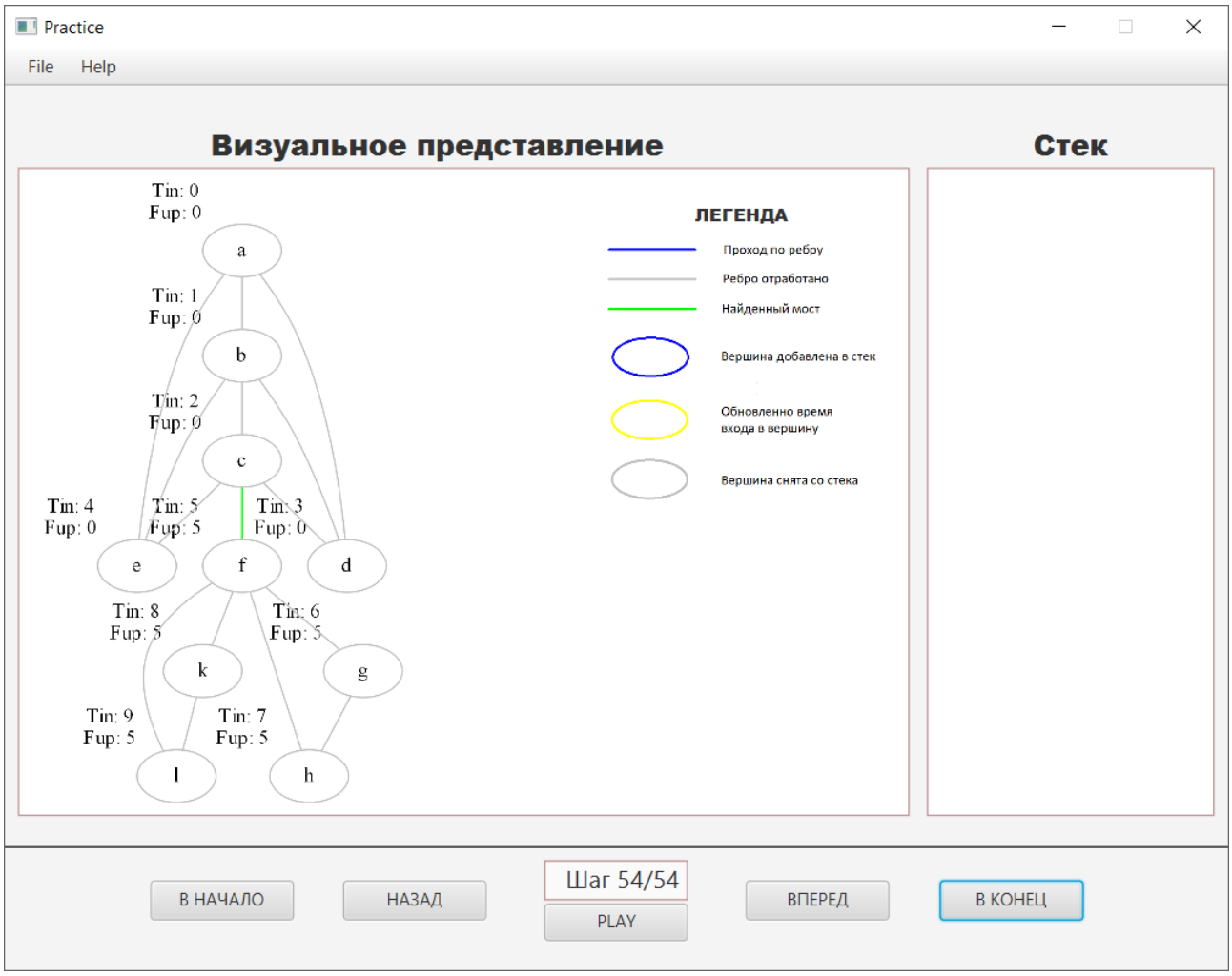

Рисунок 14 – Последний шаг графа.

Для остальных графов также верно пошагово выполнялся алгоритм.

ЗАКЛЮЧЕНИЕ

Разработка поставленной задачи была выполнена в соответствии с планом. Было спроектировано и запрограммировано приложения для поиска мостов в неориентированном невзвешенном графе. Также был разработан графический интерфейс, визуально отображающий результаты работы алгоритма и позволяющий управлять возможностями приложения. Основной алгоритм и графический интерфейс были протестированы. Поставленные задачи были выполнены полностью.

Таким образом разработка приложения была завершена успешно с полным выполнением плана и реализацией дополнительного функционала.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления

2. Официальная документация к Java: https://docs.oracle.com/en/java/javase/

3. Java. Эффективное программирование. Блох Джошуа 2014 год

4. Учебный курс по основам Java на Stepik: https://stepik.org/course/187/

5. Википедия: https://ru.wikipedia.org

6. https://ru.stackoverflow.com/

7. https://habr.com/ru/

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Application.java

```
package application;

import application.algorithm.BridgesFinder;
import application.algorithm.Graph;
import application.algorithm.GraphLoader;
import application.gui.MainWindow;
import application.stepper.Stepper;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.LinkedList;

public class Application {
    public static final Stepper stepper = new Stepper();
    public static Graph graph = new Graph();

    public static void main(String[] args) throws IOException {
        File temp = new File("temp.png");
        temp.delete();
        temp.createNewFile();

        MainWindow.execute(args);
    }

    public static void setNewGraph(Graph graph)
    {
        Application.graph = graph;
        Application.stepper.clear();

        BridgesFinder finder = new BridgesFinder();
        finder.findBridges(graph);
    }

    public static void loadGraphFromFile(String path) throws FileNotFoundException {
        Application.graph = GraphLoader.LoadGraph(path);
        Application.stepper.clear();
```

```
        BridgesFinder finder = new BridgesFinder();
        finder.findBridges(graph);
    }

    public static void saveGraphToFile(String path) throws IOException {
        GraphLoader.saveGraph( path, Application.graph);
    }
}
```

Graph.java

```
package application.algorithm;

import javafx.util.Pair;

import java.util.*;

/**
 * Represents an undirected graph that contains many unique nodes.
 * A Node can contain many edges to other nodes.
 */
public class Graph {
    /**
     * Represents an Node of graph with coordinates and set of edges.
     */
    public static class Node
    {
        final private String name;
        private final HashMap<Node, Node> edges = new HashMap<>();

        /**
         * Create a new object of this class..
         * @param name This Node's name.
         */
        Node(String name)
        {
            this.name = name;
        }

        @Override
        public boolean equals(Object o)
```

```
    {
        if (this == o)
        {
            return true;
        }

        if (o == null || getClass() != o.getClass())
        {
            return false;
        }

        Node node = (Node) o;

        return name.equals(node.name);
    }

    @Override
    public int hashCode()
    {
        return Objects.hash(name);
    }

    /**
     * Return the name of this Node.
     * @return name of this Node.
     */
    public String getName()
    {
        return name;
    }

    /**
     * Return the set of edges of this Node.
     * @return Set of edges of this Node.
     */
    public Set<Node> getEdges()
    {
        return edges.keySet();
    }
}

private final HashMap<Node, Node> nodes = new HashMap<>();
```

```
/**
 * Add new node to the graph.
 * @param name new Node's name.
 */
public void addNode(String name)
{
   Node node = new Node(name);
   nodes.putIfAbsent(node, node);
}

/**
 * Delete node from graph.
 * @param name name of Node that we want to delete
 */
public void deleteNode(String name)
{
   Node node = nodes.get(new Node(name));
   if(node != null)
   {
      for (Node toNode : node.edges.keySet())
      {
         toNode.edges.remove(node);
      }

      node.edges.clear();
      nodes.remove(node);
   }
}

/**
 * Add new edge to the graph.
 * @param from first node name.
 * @param to second node name position.
 * @return True, if edge added, and False if error has occurred.
 */
public boolean addEdge(String from, String to)
{
   if(nodes.containsKey(new Node(from)) && nodes.containsKey(new Node(to)))
   {
      Node fromNode = nodes.get(new Node(from));
      Node toNode = nodes.get(new Node(to));
```

```
        fromNode.edges.putIfAbsent(toNode, toNode);
        toNode.edges.putIfAbsent(fromNode, fromNode);

        return true;
    }

    return false;
}

/**
 * Delete edge from graph.
 * @param from first node name.
 * @param to second node name position.
 */
public void deleteEdge(String from, String to)
{
    if(nodes.containsKey(new Node(from)) && nodes.containsKey(new Node(to)))
    {
        Node fromNode = nodes.get(new Node(from));
        Node toNode = nodes.get(new Node(to));

        fromNode.edges.remove(toNode);
        toNode.edges.remove(fromNode);
    }
}

/**
 * Return Node by it's name.
 * @param name name of Node.
 * @return Node, if it is exist and null, if not exist.
 */
public Node getNode(String name)
{
    return nodes.get(new Node(name));
}

/**
 * Return set of Nodes, that graph contains.
 * @return Set of Nodes.
 */
public Set<Node> getNodes()
```

```
{
   return nodes.keySet();
}

public LinkedList<String> nodesToList()
{
   LinkedList<String> list = new LinkedList<>();

   for(Node node : getNodes())
   {
      list.add(node.name);
   }

   return list;
}

public LinkedList<Pair<String, String>> edgesToList()
{
   HashSet<Pair<String, String>> edges = new HashSet<>();

   for (Node node : getNodes())
   {
      for(Node edge : node.edges.keySet()) {
         Pair<String, String> dEdge = new Pair<>(node.name, edge.name);
         Pair<String, String> rEdge = new Pair<>(edge.name, node.name);



         if (!edges.contains(dEdge) && !edges.contains(rEdge))
         {
            edges.add(new Pair<>(node.name, edge.name));

         }
      }
   }

   return new LinkedList<>(edges);
}

public LinkedList<String> toStringList()
{
   LinkedList<String> list = new LinkedList<>();
   for(Node node : getNodes())
```

```
        {
            list.add(node.name);
        }

        HashSet<Pair<String, String>> edges = new HashSet<>();

        for (Node node : getNodes())
        {
            for(Node edge : node.edges.keySet()) {
                Pair<String, String> dEdge = new Pair<>(node.name, edge.name);
                Pair<String, String> rEdge = new Pair<>(edge.name, node.name);


                if (!edges.contains(dEdge) && !edges.contains(rEdge))
                {
                    edges.add(new Pair<>(node.name, edge.name));
                }
            }
        }

        for(Pair<String, String> edge : edges)
        {
            list.add(edge.getKey() + " -- " + edge.getValue());
        }

        return list;
    }
}
```

BridgeFinder.java

```
package application.algorithm;

import application.Application;
import application.stepper.*;

import  java.util.*;

public class BridgesFinder{
    private final HashMap<String, Boolean> used = new HashMap<>();
    private int timer;
```

```
private final HashMap<String, Integer> tin = new HashMap<>();
private final HashMap<String, Integer> fup = new HashMap<>();

private void dfs(Graph.Node node, Graph.Node parent){
    used.put(node.getName(), true);
    tin.put(node.getName(), timer);
    fup.put(node.getName(), timer);

    timer += 1;

    for (Graph.Node edge : node.getEdges()) {
        if (edge == parent) continue;
        if (used.get(edge.getName())) {
          Application.stepper.newStep(null,
                new EdgeAction(EdgeAction.Action.EDGE_STARTED, edge, node));

          int newFup = Math.min(fup.get(node.getName()), tin.get(edge.getName()));

          NodeAction fupUpdate = new NodeAction(NodeAction.Action.TIME_UPDATED,
                node, tin.get(node.getName()), newFup, tin.get(node.getName()),
                fup.get(node.getName()));

          fup.put(node.getName(), newFup);

          Application.stepper.newStep(fupUpdate,
                new EdgeAction(EdgeAction.Action.EDGE_FINISHED_COMMON, edge, node));
        }
        else {
          Application.stepper.newStep(new NodeAction(NodeAction.Action.NODE_STARTED,
                    edge, timer, timer, -1, -1),
                new EdgeAction(EdgeAction.Action.EDGE_STARTED, edge, node));

          dfs(edge, node);
          int newFup = Math.min(fup.get(node.getName()), fup.get(edge.getName()));

          NodeAction fupUpdate = new NodeAction(NodeAction.Action.TIME_UPDATED,
                node, tin.get(node.getName()), newFup, tin.get(node.getName()),
                fup.get(node.getName()));

          fup.put(node.getName(), newFup);
          if (fup.get(edge.getName()) > tin.get(node.getName())) {
```

```
                Application.stepper.newStep(fupUpdate,
                        new EdgeAction(EdgeAction.Action.EDGE_FINISHED_BRIDGE, edge, node));

                System.out.println("BRIDGE: " + edge.getName() + " -- " + node.getName());
            }
            else
            {
                Application.stepper.newStep(fupUpdate,
                        new EdgeAction(EdgeAction.Action.EDGE_FINISHED_COMMON, edge, node));
            }
        }
    }

    Application.stepper.newStep(new NodeAction(NodeAction.Action.NODE_FINISHED,
            node, tin.get(node.getName()), fup.get(node.getName()),
            tin.get(node.getName()), fup.get(node.getName())), null);
  }

  public void findBridges(Graph graph){
    timer = 0;

    for(Graph.Node node : graph.getNodes()){
      used.put(node.getName(), Boolean.FALSE);
    }

    for(Graph.Node node : graph.getNodes()){
      if(!used.get(node.getName())) {
        Application.stepper.newStep(new NodeAction(NodeAction.Action.NODE_STARTED,
                node, timer, timer, -1, -1), null);
        dfs(node, null);
      }
    }
  }
}
```

GraphLoader.java

```
package application.algorithm;

import java.io.*;
import java.util.Iterator;
```

```
import java.util.Scanner;

public class GraphLoader {

   public static Graph LoadGraph(String path) throws FileNotFoundException {
      Graph graph = new Graph();
      Scanner scanner = new Scanner(new BufferedReader(new InputStreamReader(new FileInputStream(path))));
      String currString;
      scanner.useDelimiter("\n");
      currString = scanner.next();
      String[] subStr;
      while (!currString.equals("")) {
         graph.addNode(currString);
         currString = scanner.next();
      }
      while (scanner.hasNext()) {
            String dst, src;
            currString = scanner.next();
            subStr = currString.split(" ");
            if(subStr[0].equals(""))
               break;
            src = subStr[0];
            dst = subStr[1];
            System.out.println(src + " " + dst);
            System.out.println(graph.addEdge(src, dst));
         }
         scanner.close();
      return graph;
   }

   public static void saveGraph(String path, Graph graph) throws IOException {
      FileWriter writer = new FileWriter(path, false);
      Iterator<Graph.Node> i = graph.getNodes().iterator();
      while(i.hasNext()){
         Graph.Node curr = i.next();
         writer.write(curr.getName() + "\n");
      }
      writer.write("\n");
      i = graph.getNodes().iterator();
      while(i.hasNext()){
         Graph.Node curr = i.next();
         for (Graph.Node node : curr.getEdges()) {
```

```
            writer.write(curr.getName() + " " + node.getName() + "\n");
          }
        }
        writer.close();
    }
}
```

GraphViz.java

```
package application.graphviz;
// GraphViz.java - a simple API to call dot from Java programs

/*$Id$*/
/*
 ********************************************************************************
 *                                                *
 *                (c) Copyright Laszlo Szathmary                    *
 *                                                *
 * This program is free software; you can redistribute it and/or modify it    *
 * under the terms of the GNU Lesser General Public License as published by   *
 * the Free Software Foundation; either version 2.1 of the License, or        *
 * (at your option) any later version.                                        *
 *                                                *
 * This program is distributed in the hope that it will be useful, but        *
 * WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY *
 * or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public    *
 * License for more details.                                                  *
 *                                                *
 * You should have received a copy of the GNU Lesser General Public License   *
 * along with this program; if not, write to the Free Software Foundation,    *
 * Inc., 675 Mass Ave, Cambridge, MA 02139, USA.                              *
 *                                                *
 ********************************************************************************
 */

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
```

```
import java.io.InputStreamReader;
import java.nio.file.Path;
import java.nio.file.Paths;

/**
 * <dl>
 * <dt>Purpose: GraphViz Java API
 * <dd>
 *
 * <dt>Description:
 * <dd> With this Java class you can simply call dot
 *      from your Java programs.
 * <dt>Example usage:
 * <dd>
 * <pre>
 *    GraphViz gv = new GraphViz();
 *    gv.addln(gv.start_graph());
 *    gv.addln("A -> B;");
 *    gv.addln("A -> C;");
 *    gv.addln(gv.end_graph());
 *    System.out.println(gv.getDotSource());
 *
 *    String type = "gif";
 *    String representationType="dot";
 *    File out = new File("out." + type);   // out.gif in this example
 *    gv.writeGraphToFile( gv.getGraph(gv.getDotSource(), type, representationType), out );
 * </pre>
 * </dd>
 *
 * </dl>
 *
 * @version v0.6.1, 2016/04/10 (April) -- Patch of Markus Keunecke is added.
 * The eclipse project configuration was extended with the maven nature.
 * @version v0.6, 2013/11/28 (November) -- Patch of Olivier Duplouy is added. Now you
 * can specify the representation type of your graph: dot, neato, fdp, sfdp, twopi, circo
 * @version v0.5.1, 2013/03/18 (March) -- Patch of Juan Hoyos (Mac support)
 * @version v0.5, 2012/04/24 (April) -- Patch of Abdur Rahman (OS detection + start subgraph +
 * read config file)
 * @version v0.4, 2011/02/05 (February) -- Patch of Keheliya Gallaba is added. Now you
 * can specify the type of the output file: gif, dot, fig, pdf, ps, svg, png, etc.
 * @version v0.3, 2010/11/29 (November) -- Windows support + ability to read the graph from a text file
 * @version v0.2, 2010/07/22 (July) -- bug fix
```

```
 * @version v0.1, 2003/12/04 (December) -- first release
 * @author  Laszlo Szathmary (<a href="jabba.laci@gmail.com">jabba.laci@gmail.com</a>)
 */
public class GraphViz
{
    /**
     * Detects the client's operating system.
     */
    private final static String osName = System.getProperty("os.name").replaceAll("\\s","");

    /**
     * The image size in dpi. 96 dpi is normal size. Higher values are 10% higher each.
     * Lower values 10% lower each.
     *
     * dpi patch by Peter Mueller
     */
    private final int[] dpiSizes = {46, 51, 57, 63, 70, 78, 86, 96, 106, 116, 128, 141, 155, 170, 187, 206, 226, 249};

    /**
     * Define the index in the image size array.
     */
    private int currentDpiPos = 16;

    /**
     * Increase the image size (dpi).
     */
    public void increaseDpi() {
        if ( this.currentDpiPos < (this.dpiSizes.length - 1) ) {
            ++this.currentDpiPos;
        }
    }

    /**
     * Decrease the image size (dpi).
     */
    public void decreaseDpi() {
        if (this.currentDpiPos > 0) {
            --this.currentDpiPos;
        }
    }

    public int getImageDpi() {
```

```
    return this.dpiSizes[this.currentDpiPos];
}

/**
 * The source of the graph written in dot language.
 */
private StringBuilder graph = new StringBuilder();

private String tempDir;

private String executable;

/**
 * Convenience Constructor with default OS specific pathes
 * creates a new GraphViz object that will contain a graph.
 * Windows:
 * executable = c:/Program Files (x86)/Graphviz 2.28/bin/dot.exe
 * tempDir = c:/temp
 * MacOs:
 * executable = /usr/local/bin/dot
 * tempDir = /tmp
 * Linux:
 * executable = /usr/bin/dot
 * tempDir = /tmp
 */
public GraphViz() {
    if (GraphViz.osName.contains("Windows")) {
        this.tempDir = "C:\\temp";
        this.executable = "C:\\Users\\danma\\Desktop\\Graphviz2.38\\bin\\dot.exe";
    } else if (GraphViz.osName.equals("MacOSX")) {
        this.tempDir = "/tmp";
        this.executable = "/usr/local/bin/dot";
    } else if (GraphViz.osName.equals("Linux")) {
        this.tempDir = "/tmp";
        this.executable = "/usr/bin/dot";
    }
}

/**
 * Configurable Constructor with path to executable dot and a temp dir
 *
 * @param executable absolute path to dot executable
```

```
 * @param tempDir absolute path to temp directory
 */
public GraphViz(String executable, String tempDir) {
    this.executable = executable;
    this.tempDir = tempDir;
}

/**
 * Returns the graph's source description in dot language.
 * @return Source of the graph in dot language.
 */
public String getDotSource() {
    return this.graph.toString();
}

/**
 * Adds a string to the graph's source (without newline).
 */
public void add(String line) {
    this.graph.append(line);
}

/**
 * Adds a string to the graph's source (with newline).
 */
public void addln(String line) {
    this.graph.append(line + "\n\r");
}

/**
 * Adds a newline to the graph's source.
 */
public void addln() {
    this.graph.append('\n');
}

public void clearGraph(){
    this.graph = new StringBuilder();
}

/**
 * Returns the graph as an image in binary format.
```

```
 * @param dot_source Source of the graph to be drawn.
 * @param type Type of the output image to be produced, e.g.: gif, dot, fig, pdf, ps, svg, png.
 * @param representationType Type of how you want to represent the graph:
 * <ul>
 *      <li>dot</li>
 *      <li>neato</li>
 *      <li>fdp</li>
 *      <li>sfdp</li>
 *      <li>twopi</li>
 *      <li>circo</li>
 * </ul>
 * @see http://www.graphviz.org under the Roadmap title
 * @return A byte array containing the image of the graph.
 */
public byte[] getGraph(String dot_source, String type, String representationType)
{
   File dot;
   byte[] img_stream = null;

   try {
      dot = writeDotSourceToFile(dot_source);
      if (dot != null)
      {
         img_stream = get_img_stream(dot, type, representationType);
         if (dot.delete() == false) {
            System.err.println("Warning: " + dot.getAbsolutePath() + " could not be deleted!");
         }
         return img_stream;
      }
      return null;
   } catch (java.io.IOException ioe) { return null; }
}

/**
 * Writes the graph's image in a file.
 * @param img   A byte array containing the image of the graph.
 * @param file  Name of the file to where we want to write.
 * @return Success: 1, Failure: -1
 */
public int writeGraphToFile(byte[] img, String file)
{
   File to = new File(file);
```

```
    return writeGraphToFile(img, to);
}

/**
 * Writes the graph's image in a file.
 * @param img   A byte array containing the image of the graph.
 * @param to    A File object to where we want to write.
 * @return Success: 1, Failure: -1
 */
public int writeGraphToFile(byte[] img, File to)
{
    try {
        FileOutputStream fos = new FileOutputStream(to);
        fos.write(img);
        fos.close();
    } catch (java.io.IOException ioe) { return -1; }
    return 1;
}

/**
 * It will call the external dot program, and return the image in
 * binary format.
 * @param dot Source of the graph (in dot language).
 * @param type Type of the output image to be produced, e.g.: gif, dot, fig, pdf, ps, svg, png.
 * @param representationType Type of how you want to represent the graph:
 * <ul>
 *      <li>dot</li>
 *      <li>neato</li>
 *      <li>fdp</li>
 *      <li>sfdp</li>
 *      <li>twopi</li>
 *      <li>circo</li>
 * </ul>
 * @see http://www.graphviz.org under the Roadmap title
 * @return The image of the graph in .gif format.
 */
private byte[] get_img_stream(File dot, String type, String representationType)
{
    File img;
    byte[] img_stream = null;

    try {
```

```
            img = File.createTempFile("graph_", "." + type, new File(this.tempDir));
            Runtime rt = Runtime.getRuntime();

            // patch by Mike Chenault
            // representation type with -K argument by Olivier Duplouy
            String[] args = { executable, "-T" + type, "-K" + representationType, "-Gdpi=" + dpiSizes[this.currentDpiPos],
dot.getAbsolutePath(), "-o", img.getAbsolutePath() };
            Process p = rt.exec(args);
            p.waitFor();

            FileInputStream in = new FileInputStream(img.getAbsolutePath());
            img_stream = new byte[in.available()];
            in.read(img_stream);
            // Close it if we need to
            if( in != null ) {
               in.close();
            }

            if (img.delete() == false) {
               System.err.println("Warning: " + img.getAbsolutePath() + " could not be deleted!");
            }
         }
         catch (java.io.IOException ioe) {
            System.err.println("Error:    in I/O processing of tempfile in dir " + tempDir + "\n");
            System.err.println("       or in calling external command");
            ioe.printStackTrace();
         }
         catch (java.lang.InterruptedException ie) {
            System.err.println("Error: the execution of the external program was interrupted");
            ie.printStackTrace();
         }

         return img_stream;
      }

      /**
       * Writes the source of the graph in a file, and returns the written file
       * as a File object.
       * @param str Source of the graph (in dot language).
       * @return The file (as a File object) that contains the source of the graph.
       */
      private File writeDotSourceToFile(String str) throws java.io.IOException
```

```
{
   File temp;
   try {
      temp = File.createTempFile("graph_", ".dot.tmp", new File(tempDir));
      FileWriter fout = new FileWriter(temp);
      fout.write(str);
      fout.close();
   }
   catch (Exception e) {
      System.err.println("Error: I/O error while writing the dot source to temp file!");
      return null;
   }
   return temp;
}

/**
 * Returns a string that is used to start a graph.
 * @return A string to open a graph.
 */
public String start_graph() {
   return "graph G {";
}

/**
 * Returns a string that is used to end a graph.
 * @return A string to close a graph.
 */
public String end_graph() {
   return "}";
}

/**
 * Takes the cluster or subgraph id as input parameter and returns a string
 * that is used to start a subgraph.
 * @return A string to open a subgraph.
 */
public String start_subgraph(int clusterid) {
   return "subgraph cluster_" + clusterid + " {";
}

/**
 * Returns a string that is used to end a graph.
```

```
     * @return A string to close a graph.
     */
    public String end_subgraph() {
        return "}";
    }

    /**
     * Read a DOT graph from a text file.
     *
     * @param input Input text file containing the DOT graph
     * source.
     */
    public void readSource(String input)
    {
        StringBuilder sb = new StringBuilder();

        try
        {
            FileInputStream fis = new FileInputStream(input);
            DataInputStream dis = new DataInputStream(fis);
            BufferedReader br = new BufferedReader(new InputStreamReader(dis));
            String line;
            while ((line = br.readLine()) != null) {
                sb.append(line);
            }
            dis.close();
        }
        catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
        }

        this.graph = sb;
    }

} // end of class GraphViz
```

Proba.java

```
package application.graphviz;

import java.io.File;
```

```
import java.util.LinkedList;


public class Proba
{
   /**
    * Construct a DOT graph in memory, convert it
    * to image and store the image in the file system.
    */
   public static String makeGraph(LinkedList<String> param)
   {
      GraphViz gv = new GraphViz();
      gv.addln(gv.start_graph());
      for (String Var : param)
      {
         gv.addln(Var + ";");
      }
      gv.addln(gv.end_graph());
      //System.out.println(gv.getDotSource());

      gv.increaseDpi();   // 106 dpi

      String type = "png";
      //      String type = "gif";
      //      String type = "dot";
      //      String type = "fig";    // open with xfig
      //      String type = "pdf";
      //      String type = "ps";
      //      String type = "svg";    // open with inkscape
      //      String type = "png";
      //      String type = "plain";

      String repesentationType= "dot";
      //                String repesentationType= "dot";
      //                String repesentationType= "neato";
      //                String repesentationType= "fdp";
      //                String repesentationType= "sfdp";
      //                String repesentationType= "twopi";
      //                String repesentationType= "circo";

      String path = "temp" + "."+ type;
      File out = new File(path);
```

```
        gv.writeGraphToFile( gv.getGraph(gv.getDotSource(), type, repesentationType), out );

        return path;
    }
}
```

Stepper.java

```
package application.stepper;

import application.Application;
import application.algorithm.Graph;
import application.graphviz.Proba;
import javafx.util.Pair;

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

public class Stepper {
    private final ArrayList<HashMap<String, NodeInfo>> nodeSteps = new ArrayList<>();
    private final ArrayList<HashMap<Pair<String, String>, EdgeInfo>> edgeSteps = new ArrayList<>();
    private final ArrayList<ArrayList<String>> stackSteps = new ArrayList<>();
    private int cursor = -1;

    private static class NodeInfo
    {
        private final String name;
        private final String color;
        private final String label;

        public NodeInfo(String name, String color, int tin, int fup) {
            this.name = name;
            this.color = color;
            this.label = "Tin: " + Integer.toString(tin) + '\n' + "Fup: " + Integer.toString(fup);
        }

        public String toStr()
        {
            return name + " [color=\"" + color + "\"" + ", xlabel=\"" + label + "\"" + "]";
```

```
    }
}

private static class EdgeInfo
{
    private final String from;
    private final String to;
    private final String color;

    public EdgeInfo(String from, String to, String color) {
        this.from = from;
        this.to = to;
        this.color = color;
    }

    public String toStr()
    {
        return from + " -- " + to + " [color=\"" + color + "\"]";
    }
}

public void newStep(NodeAction nodeAction, EdgeAction edgeAction)
{
    HashMap<String, NodeInfo> nodeStep = new HashMap<>(nodeSteps.get(nodeSteps.size() - 1));
    HashMap<Pair<String, String>, EdgeInfo> edgeStep = new HashMap<>(edgeSteps.get(edgeSteps.size() - 1));
    ArrayList<String> stackStep = new ArrayList<>(stackSteps.get(stackSteps.size() - 1));

    if (nodeAction != null)
    {
        switch (nodeAction.getAction())
        {
            case TIME_UPDATED:
                nodeStep.put(nodeAction.getNode().getName(),new NodeInfo(nodeAction.getNode().getName(),
                        "yellow", nodeAction.getTin(), nodeAction.getFup()));
                break;
            case NODE_STARTED:
                nodeStep.put(nodeAction.getNode().getName(),new NodeInfo(nodeAction.getNode().getName(),
                        "blue", nodeAction.getTin(), nodeAction.getFup()));

                stackStep.add(nodeAction.getNode().getName());
                break;
            case NODE_FINISHED:
```

```
                nodeStep.put(nodeAction.getNode().getName(),
                    new NodeInfo(nodeAction.getNode().getName(),
                    "grey", nodeAction.getTin(), nodeAction.getFup()));

                stackStep.remove(stackStep.size() - 1);
                break;
            }
        }

        if (edgeAction != null)
        {
            Pair<String, String> dEdge = new Pair<>(edgeAction.getParent().getName(),
edgeAction.getNode().getName());
            Pair<String, String> rEdge = new Pair<>(edgeAction.getNode().getName(),
edgeAction.getParent().getName());
            Pair<String, String> edge = null;
            if(edgeStep.containsKey(dEdge))
            {
                edge = dEdge;
            }
            else if(edgeStep.containsKey(rEdge))
            {
                edge = rEdge;
            }

            if(edge != null) {
                switch (edgeAction.getAction()) {
                    case EDGE_STARTED:
                        edgeStep.put(edge,
                            new EdgeInfo(edge.getKey(), edge.getValue(), "blue"));
                        break;
                    case EDGE_FINISHED_COMMON:
                        edgeStep.put(edge,
                            new EdgeInfo(edge.getKey(), edge.getValue(), "grey"));
                        break;
                    case EDGE_FINISHED_BRIDGE:
                        edgeStep.put(edge,
                            new EdgeInfo(edge.getKey(), edge.getValue(), "green"));
                        break;
                }
            }
        }
```

```
    nodeSteps.add(nodeStep);
    edgeSteps.add(edgeStep);
    stackSteps.add(stackStep);
}

private void activeStep(int step)
{
    if (cursor < stackSteps.size() && cursor >= 0) {
        LinkedList<String> markup = new LinkedList<>();

        for (EdgeInfo edge : edgeSteps.get(step).values())
        {
            markup.push(edge.toStr());
        }

        for (NodeInfo node : nodeSteps.get(step).values())
        {
            markup.push(node.toStr());
        }

        Proba.makeGraph(markup);
    }
}

private void fromGraph(Graph graph)
{
    HashMap<String, NodeInfo> nodeStep = new HashMap<>();
    HashMap<Pair<String, String>, EdgeInfo> edgeStep = new HashMap<>();
    ArrayList<String> stackStep = new ArrayList<>();

    for(Pair<String, String> edge : graph.edgesToList())
    {
        edgeStep.put(edge, new EdgeInfo(edge.getKey(),edge.getValue(), "black"));
    }

    for(String node : graph.nodesToList())
    {
        nodeStep.put(node, new  NodeInfo(node, "black", -1, -1));
    }

    nodeSteps.add(nodeStep);
```

```
    edgeSteps.add(edgeStep);
    stackSteps.add(stackStep);

    cursor = -1;

    nextStep();
}

public ArrayList<String> nextStep()
{
    if (cursor + 1 < stackSteps.size())
    {
        cursor += 1;
        activeStep(cursor);
        return stackSteps.get(cursor);
    }
    else
    {
        return new ArrayList<>();
    }
}

public ArrayList<String> prevStep()
{
    if (cursor > 0)
    {
        cursor -= 1;
        activeStep(cursor);
        return stackSteps.get(cursor);
    }
    else
    {
        return new ArrayList<>();
    }
}

public int getStepCount()
{
    return stackSteps.size();
}

public void clear()
```

```
    {
        nodeSteps.clear();
        edgeSteps.clear();
        stackSteps.clear();

        fromGraph(Application.graph);
    }

}
```

NodeAction.java

```
package application.stepper;

import application.algorithm.Graph;

public class NodeAction
{
    public Action getAction() {
        return action;
    }

    public Graph.Node getNode() {
        return node;
    }

    public int getTin() {
        return tin;
    }

    public int getFup() {
        return fup;
    }

    private final Action action;
    private final Graph.Node node;
    private final int tin;
    private final int fup;
    private final int oldTin;
    private final int oldFup;
```

```
public NodeAction(Action action, Graph.Node node, int tin, int fup, int oldTin, int oldFup) {
    this.action = action;
    this.node = node;
    this.tin = tin;
    this.fup = fup;
    this.oldTin = oldTin;
    this.oldFup = oldFup;
}

public NodeAction back()
{
    return new NodeAction(Action.back(action), node, oldTin, oldFup, tin, fup);
}

public enum Action
{
    NONE,
    BASE,
    TIME_UPDATED,
    NODE_STARTED,
    NODE_FINISHED;

    public static Action back(Action action)
    {
        switch (action)
        {
            case NODE_STARTED : {
                return BASE;
            }
            case NODE_FINISHED : {
                return NODE_STARTED;
            }
            case TIME_UPDATED : {
                return TIME_UPDATED;
            }
            case NONE : {
                return NONE;
            }
            default : throw new IllegalStateException("Unexpected value: " + action);
        }
    }
}
```

```
}
```

EdgeAction.java

```
package application.stepper;

import application.algorithm.Graph;

public class EdgeAction
{
   public Action getAction() {
      return action;
   }

   public Graph.Node getNode() {
      return node;
   }

   public Graph.Node getParent() {
      return parent;
   }

   private final Action action;
   private final Graph.Node node;
   private final Graph.Node parent;

   public EdgeAction(Action action, Graph.Node node, Graph.Node parent) {
      this.action = action;
      this.node = node;
      this.parent = parent;
   }

   public EdgeAction back()
   {
      return new EdgeAction(Action.back(action), node, parent);
   }
   public String getEdge() { return node.getName() + " -- " + parent.getName(); };


   public enum Action
   {
```

```
        NONE,
        BASE,
        EDGE_STARTED,
        EDGE_FINISHED_COMMON,
        EDGE_FINISHED_BRIDGE;

        public static Action back(Action action)
        {
            switch (action)
            {
                case EDGE_STARTED : {
                    return BASE;
                }
                case EDGE_FINISHED_COMMON :
                case EDGE_FINISHED_BRIDGE : {
                    return EDGE_STARTED;
                }
                case NONE : {
                    return NONE;
                }
                default : throw new IllegalStateException("Unexpected value: " + action);
            }
        }

    }
}
```

HelpController.java

```
package application.gui;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class HelpController {

    @FXML
    private Button closeButton;
```

```
    @FXML
    void Close(){
        Stage stage = (Stage)closeButton.getScene().getWindow();
        stage.close();
    }
}
```

Help.fxml

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

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Accordion?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ButtonBar?>
<?import javafx.scene.control.Tab?>
<?import javafx.scene.control.TabPane?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane prefHeight="427.0" prefWidth="550.0" xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="application.gui.HelpController">
  <children>
    <ButtonBar layoutY="387.0" prefHeight="40.0" prefWidth="600.0" style="-fx-background-color: BC8F8F;">
      <buttons>
        <Button fx:id="closeButton" maxHeight="-Infinity" mnemonicParsing="false" onAction="#Close"
prefHeight="15.0" text="Закрыть" />
      </buttons>
       <padding>
         <Insets right="6.0" />
       </padding>
    </ButtonBar>
    <Accordion layoutX="98.0" layoutY="80.0" />
    <TabPane layoutX="-2.0" prefHeight="400.0" prefWidth="600.0" tabClosingPolicy="UNAVAILABLE">
      <tabs>
       <Tab text="Информаци о программе">
         <content>
           <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="346.0" prefWidth="600.0">
                <children>
```

<Text layoutX="13.0" layoutY="27.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Визуализатор с графическим интерфейсом алгоритма нахождения мостов в неориентированном графе(модификацией поиска в глубину) на языке Java." textAlignment="JUSTIFY" wrappingWidth="573.3367691040039">

<Font name="Times New Roman" size="16.0" />

</Text>
</children>
</AnchorPane>
</content>
</Tab>
<Tab text="Руководство пользователя">
<content>
<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="358.0" prefWidth="606.0">
<children>
<Text layoutX="14.0" layoutY="27.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Тыкайте на все кнопки, авось заработает!" />
</children></AnchorPane>
</content>
</Tab>
<Tab text="Информация о разработчиках">
<content>
<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0">
<children>
<Text layoutX="15.0" layoutY="40.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Матросов Денис - qwerty_19200@mail.ru:" wrappingWidth="579.7367324829102">

<Font size="14.0" />

</Text>
<Text layoutX="15.0" layoutY="66.0" strokeType="OUTSIDE" strokeWidth="0.0" text="* Лидер проекта" wrappingWidth="241.33673858642578" />
<Text layoutX="15.0" layoutY="83.0" strokeType="OUTSIDE" strokeWidth="0.0" text="* Проетирование GUI" wrappingWidth="232.53668975830078" />
<Text layoutX="15.0" layoutY="100.0" strokeType="OUTSIDE" strokeWidth="0.0" text="* Написание GUI" wrappingWidth="235.73670196533203" />
<Text layoutX="15.0" layoutY="117.0" strokeType="OUTSIDE" strokeWidth="0.0" text="* Тестирование GUI" wrappingWidth="227.73673248291016" />
<Text layoutX="16.0" layoutY="149.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Масалыкин Даниил - dan.masalikin2017@gmail.com:">
<font>

```
                  <Font size="14.0" />
                </font>
              </Text>
              <Text layoutX="15.0" layoutY="174.0" strokeType="OUTSIDE" strokeWidth="0.0" text="*
Написание реализации алгоритма" />
              <Text layoutX="15.0" layoutY="191.0" strokeType="OUTSIDE" strokeWidth="0.0" text="*
Тестирование алгоритма" />
              <Text layoutX="15.0" layoutY="208.0" strokeType="OUTSIDE" strokeWidth="0.0" text="*
Написание ввода/вывода графа из файла" />
              <Text layoutX="14.0" layoutY="250.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Чешуин
Дмитрий - makotozeros@gmail.com:">
                <font>
                  <Font size="14.0" />
                </font>
              </Text>
              <Text layoutX="14.0" layoutY="278.0" strokeType="OUTSIDE" strokeWidth="0.0" text="*
Проектирование архитектуры программы" />
              <Text layoutX="14.0" layoutY="295.0" strokeType="OUTSIDE" strokeWidth="0.0" text="*
Написание класса, осуществляющего хранение шагов алгоритма, а также их визуализацию" />
              <Text layoutX="14.0" layoutY="312.0" strokeType="OUTSIDE" strokeWidth="0.0" text="* Слияние
наработок" />
              <Text layoutX="14.0" layoutY="329.0" strokeType="OUTSIDE" strokeWidth="0.0" text="* Общее
тестирование программы" />
            </children>
          </AnchorPane>
        </content>
      </Tab>
    </tabs>
  </TabPane>
 </children>
</AnchorPane>
```

MainWindow.java

```
package application.gui;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
```

```
public class MainWindow extends Application {

   @Override
   public void start(Stage primaryStage) throws Exception{
      Parent root = FXMLLoader.load(getClass().getResource("MainWindow.fxml"));
      primaryStage.setTitle("Practice");
      primaryStage.setResizable(false);
      primaryStage.setScene(new Scene(root, 850, 640));
      primaryStage.show();
   }
   public static void execute(String[] args) {
      launch(args);
   }
}
```

MainWindow.fxml

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ListView?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.SeparatorMenuItem?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.shape.Line?>
<?import javafx.scene.text.Font?>

<VBox prefHeight="600.0" prefWidth="850.0" xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="application.gui.MainWindowController">
  <children>
    <MenuBar VBox.vgrow="NEVER">
      <menus>
        <Menu mnemonicParsing="false" text="File">
```

```
        <items>
          <MenuItem fx:id="newButton" mnemonicParsing="false" onAction="#initNewFileWindow" text="New" />
          <MenuItem fx:id="openFileButton" mnemonicParsing="false" onAction="#initOpenFileWindow"
text="Open…" />
          <MenuItem fx:id="saveButton" mnemonicParsing="false" onAction="#initSaveFileWindow" text="Save As…"
/>
              <SeparatorMenuItem mnemonicParsing="false" />
          <MenuItem fx:id="quitButton" mnemonicParsing="false" onAction="#handleCloseButtonAction" text="Quit"
/>
        </items>
      </Menu>
      <Menu mnemonicParsing="false" text="Help">
        <items>
          <MenuItem fx:id="helpButton" mnemonicParsing="false" onAction="#initNewHelpWindow" text="About
program" />
        </items>
      </Menu>
    </menus>
  </MenuBar>
  <AnchorPane fx:id="anchorPane" maxHeight="600.0" maxWidth="850.0" minHeight="600.0" minWidth="850.0"
prefHeight="600.0" prefWidth="850.0" VBox.vgrow="NEVER">
    <children>
        <Button fx:id="nextStepButton" layoutX="515.0" layoutY="552.0" minHeight="28.0" minWidth="100.0"
mnemonicParsing="false" onAction="#nextstep" text="ВПЕРЕД" />
        <Button fx:id="prevStepButton" layoutX="235.0" layoutY="552.0" minHeight="28.0" minWidth="100.0"
mnemonicParsing="false" onAction="#prevstep" text="НАЗАД" />
        <Label layoutX="143.0" layoutY="27.0" text="Визуальное представление" textAlignment="CENTER">
          <font>
            <Font name="Arial Black" size="20.0" />
          </font>
        </Label>
        <Label layoutX="715.0" layoutY="27.0" text="Стек" textAlignment="CENTER">
          <font>
            <Font name="Arial Black" size="20.0" />
          </font>
        </Label>
        <Label fx:id="stepNumberField" layoutX="375.0" layoutY="552.0" prefHeight="28.0" prefWidth="100.0"
style="-fx-background-color: fafafa; -fx-border-color: BC8F8F;" text="   Шаг 1/10">
          <font>
            <Font size="17.0" />
          </font>
        </Label>
```

```
        <ListView fx:id="stackList" layoutX="641.0" layoutY="57.0" prefHeight="450.0" prefWidth="200.0" style="-
fx-background-color: ffffff; -fx-border-color: BC8F8F;" />
        <BorderPane layoutX="101.0" layoutY="139.0" prefHeight="200.0" prefWidth="200.0" />
        <Line endX="735.4000244140625" layoutX="115.0" layoutY="529.0" startX="-115.00000762939453" />
        <Pane fx:id="graphPane" layoutX="9.0" layoutY="57.0" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="450.0" prefWidth="620.0" style="-fx-background-color: ffffff; -fx-border-color: BC8F8F;">
          <children>
            <ImageView fx:id="GraphView" fitHeight="440.0" fitWidth="306.0" layoutX="159.0" layoutY="4.0"
pickOnBounds="true" preserveRatio="true" />
          </children></Pane>
      </children>
    </AnchorPane>
  </children>
</Vbox>
```

MainWindowController.java

```
package application.gui;

import application.Application;
import application.graphviz.Proba;

import application.stepper.EdgeAction;
import application.stepper.NodeAction;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.MenuItem;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Pane;
import javafx.stage.Modality;
import javafx.stage.Stage;
```

```
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.LinkedList;

public class MainWindowController {

    int countOfStep = 10;
    int currentStep = 1;

    @FXML
    public MenuItem newButton;

    @FXML
    public MenuItem openFileButton;

    @FXML
    public MenuItem saveButton;

    @FXML
    public MenuItem quitButton;

    @FXML
    public MenuItem helpButton;

    @FXML
    public AnchorPane anchorPane;

    @FXML
    public Button nextStepButton;

    @FXML
    public Button prevStepButton;

    @FXML
    public Label stepNumberField;

    @FXML
    public ListView<String> stackList;

    @FXML
    public Pane graphPane;
```

```
@FXML
private ImageView GraphView;

void setCountOfStep(){
    this.countOfStep = Application.stepper.getStepCount() + 1;
}

@FXML
void initialize(){
    setCountOfStep();
    currentStep = 1;
    stepNumberField.setText("   IIIar " + currentStep + "/" + (countOfStep - 1));
    try {
        printImg();
    }catch (IOException e){
        e.printStackTrace();
    }
}

void reInit(){
    stepNumberField.setText("   IIIar " + currentStep + "/" +  (countOfStep - 1));
    try {
        printImg();
    }catch (IOException e){
        e.printStackTrace();
    }
}

@FXML
void initNewFileWindow(){
    Parent root;
    try {
        root = FXMLLoader.load(getClass().getResource("NewFile.fxml"));
        Stage stage = new Stage();
        stage.setTitle("New file");
        stage.setResizable(false);
        stage.initModality(Modality.WINDOW_MODAL);
        stage.initOwner((Stage)graphPane.getScene().getWindow());
        stage.setScene(new Scene(root, 800, 600));
        stage.show();
        stage.setOnHiding( event -> {
```

```
            initialize();
        });
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}

@FXML
void initNewHelpWindow(){
    Parent root;
    try {
        root = FXMLLoader.load(getClass().getResource("Help.fxml"));
        Stage stage = new Stage();
        stage.setTitle("Help");
        stage.setResizable(false);
        stage.initModality(Modality.WINDOW_MODAL);
        stage.initOwner((Stage)graphPane.getScene().getWindow());
        stage.setScene(new Scene(root, 600, 425));
        stage.show();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}

@FXML
void initOpenFileWindow(){
    Parent root;
    try {
        root = FXMLLoader.load(getClass().getResource("OpenFile.fxml"));
        Stage stage = new Stage();
        stage.setTitle("Open file");
        stage.setResizable(true);
        stage.initModality(Modality.WINDOW_MODAL);
        stage.initOwner((Stage)graphPane.getScene().getWindow());
        stage.setScene(new Scene(root, 400, 150));
        stage.show();
        stage.setOnHiding( event -> {
            initialize();
        });
    }
```

```
    catch (IOException e) {
        e.printStackTrace();
    }
}

@FXML
void initSaveFileWindow(){
    Parent root;
    try {
        root = FXMLLoader.load(getClass().getResource("SaveFile.fxml"));
        Stage stage = new Stage();
        stage.setTitle("Open file");
        stage.setResizable(false);
        stage.initModality(Modality.WINDOW_MODAL);
        stage.initOwner((Stage)graphPane.getScene().getWindow());
        stage.setScene(new Scene(root, 400, 150));
        stage.show();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}

@FXML
void handleCloseButtonAction(){
    Stage stage = (Stage)graphPane.getScene().getWindow();
    stage.close();
}

public static String reverseString(String inputString) {
    int stringLength = inputString.length();
    String result = "";
    for (int i = 0; i < stringLength; i++) {
        result = inputString.charAt(i) + result;
    }
    return result;
}

@FXML
void nextstep(){
    if (currentStep <  (countOfStep - 1)){
        currentStep = currentStep + 1;
```

```
            ArrayList<String> stack = Application.stepper.nextStep();

            ObservableList<String> observableStack = FXCollections.observableArrayList(stack);
            stackList.setItems(observableStack);
        }
        reInit();
    }

    @FXML
    void prevstep(){
        if (currentStep > 1){
            currentStep = currentStep - 1;
            ArrayList<String> stack = Application.stepper.prevStep();

            ObservableList<String> observableStack = FXCollections.observableArrayList(stack);
            stackList.setItems(observableStack);
        }
        reInit();
    }

    public void printImg() throws IOException{
        FileInputStream inputstream;
        inputstream = new FileInputStream("temp.png");
        Image img = new Image(inputstream);
        GraphView.setImage(img);
    }
}
```

NewFileController.java

```
package application.gui;

import application.Application;
import application.algorithm.Graph;
import application.graphviz.Proba;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.Pane;
```

```
import javafx.stage.Stage;

import java.io.FileInputStream;
import java.io.IOException;

public class NewFileController {
    private final Graph graph = new Graph();

    @FXML
    private Button firstTool;

    @FXML
    private Button secondTool;

    @FXML
    private Button thirdTool;

    @FXML
    private Button cancelButton;

    @FXML
    private Button okButton;

    @FXML
    private Pane workSpace;

    @FXML
    private ImageView Image;

    @FXML
    private TextField node1;

    @FXML
    private TextField node2;

    @FXML
    private Button addButton;

    @FXML
    private Button deleteButton;

    @FXML
```

```
private TextField nodeField;

@FXML
private Button addNodeButton;

@FXML
private Button deleteNodeButton;

private static boolean isOnlyDigits(String str) {
    return str.matches("[a-za-я]");
}

@FXML
void addNode() {
    String node = nodeField.getText();
    if(isOnlyDigits(node)){
        graph.addNode(node);
    }
    try {
        printImg();
    }catch (IOException e){
        //do nothing
    }
    nodeField.clear();
}

@FXML
void deleteNodeEdge() {
    String node = nodeField.getText();
    if(isOnlyDigits(node)){
        graph.deleteNode(node);
    }
    try {
        printImg();
    }catch (IOException e){
        //do nothing
    }
    nodeField.clear();
}

@FXML
void addEdge() {
```

```
    String node = node1.getText();
    String parrent = node2.getText();
    boolean f1 = isOnlyDigits(node);
    boolean f2 = isOnlyDigits(parrent);
    if(f1 && f2){
        graph.addNode(node);
        graph.addNode(parrent);
        graph.addEdge(node, parrent);
    }
    try {
        printImg();
    }catch (IOException e){
        //do nothing
    }
    node1.clear();
    node2.clear();
}

@FXML
void deleteEdge() {
    String node = node1.getText();
    String parrent = node2.getText();
    if(isOnlyDigits(node) && isOnlyDigits(parrent)){
        graph.deleteEdge(node, parrent);

    }
    try {
        printImg();
    }catch (IOException e){
        //do nothing
    }
    node1.clear();
    node2.clear();
}

public void printImg() throws IOException {
    Proba.makeGraph(graph.toStringList());
    FileInputStream inputstream;
    inputstream = new FileInputStream("temp.png");
    javafx.scene.image.Image img = new Image(inputstream);
    Image.setImage(img);
}
```

```
    @FXML
    void Confirm(){
        Stage stage = (Stage)okButton.getScene().getWindow();

        Application.setNewGraph(graph);

        stage.close();
    }

    @FXML
    void Cancel(){
        Stage stage = (Stage)cancelButton.getScene().getWindow();
        stage.close();
    }
}
```

NewFile.fxml

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

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ButtonBar?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="600.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="application.gui.NewFileController">
  <children>
    <ButtonBar layoutY="566.0" prefHeight="40.0" prefWidth="800.0" style="-fx-background-color: BC8F8F;">
      <buttons>
        <Button fx:id="cancelButton" mnemonicParsing="false" onAction="#Cancel" text="OTMEHA" />
          <Button fx:id="okButton" mnemonicParsing="false" onAction="#Confirm" text="OK" />
      </buttons>
       <padding>
          <Insets right="8.0" />
```

```
        </padding>
      </ButtonBar>
      <Pane fx:id="workSpace" layoutY="121.0" prefHeight="446.0" prefWidth="800.0" style="-fx-background-color:
ffffff; -fx-border-color: BC8F8F;">
        <children>
          <ImageView fx:id="Image" fitHeight="412.0" fitWidth="771.0" layoutX="14.0" layoutY="20.0"
pickOnBounds="true" preserveRatio="true" />
        </children></Pane>
      <Pane layoutX="7.0" prefHeight="118.0" prefWidth="783.0">
        <children>
          <TextField fx:id="node1" layoutX="551.0" layoutY="68.0" prefHeight="25.0" prefWidth="35.0" />
          <TextField fx:id="node2" layoutX="601.0" layoutY="68.0" prefHeight="25.0" prefWidth="35.0" />
          <Button fx:id="addButton" layoutX="650.0" layoutY="43.0" mnemonicParsing="false" onAction="#addEdge"
prefWidth="100.0" text="Добавить" />
          <Button fx:id="deleteButton" layoutX="650.0" layoutY="80.0" mnemonicParsing="false"
onAction="#deleteEdge" prefWidth="100.0" text="Удалить" />
          <TextField fx:id="nodeField" layoutX="63.0" layoutY="68.0" prefWidth="35.0" />
          <Label layoutX="559.0" layoutY="14.0" text="Редактирование ребер">
            <font>
              <Font name="Arial Black" size="14.0" />
            </font>
          </Label>
          <Label layoutX="40.0" layoutY="14.0" text="Редактирование вершин">
            <font>
              <Font name="Arial Black" size="14.0" />
            </font>
          </Label>
          <Button fx:id="addNodeButton" layoutX="118.0" layoutY="43.0" mnemonicParsing="false"
onAction="#addNode" prefWidth="100.0" text="Добавить" />
          <Button fx:id="deleteNodeButton" layoutX="118.0" layoutY="80.0" mnemonicParsing="false"
onAction="#deleteNodeEdge" prefWidth="100.0" text="Удалить" />
        </children>
      </Pane>
    </children>
</AnchorPane>
```

OpenFileController.java

```
package application.gui;

import application.Application;
```

```
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.stage.FileChooser;
import javafx.stage.Stage;


import java.io.File;
import java.io.FileNotFoundException;

public class OpenFileController {

    @FXML
    private Button cancelButton;

    @FXML
    private Button confirmButton;

    @FXML
    private TextField textField;

    @FXML
    void Confirm(){
        Stage stage = (Stage)confirmButton.getScene().getWindow();
        if(!textField.getText().isBlank()) {
            try {
                Application.loadGraphFromFile(textField.getText());
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
        }
        stage.close();
    }

    @FXML
    void Cancel(){
        Stage stage = (Stage)cancelButton.getScene().getWindow();
        stage.close();
    }

    @FXML
    void ChooseFile(){
```

```
        final FileChooser fileChooser = new FileChooser();
        fileChooser.getExtensionFilters().addAll(
                new FileChooser.ExtensionFilter("Graph", "*.graph"));
        textField.clear();
        File file = fileChooser.showOpenDialog((Stage)textField.getScene().getWindow());
        if (file != null) {
           textField.setText(file.getAbsolutePath());
        }
    }
}
```

OpenFile.fxml

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

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ButtonBar?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>


<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="150.0" prefWidth="400.0" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="application.gui.OpenFileController">
  <children>
    <ButtonBar layoutY="116.0" prefHeight="40.0" prefWidth="400.0" style="-fx-background-color: BC8F8F;">
      <buttons>
        <Button fx:id="cancelButton" mnemonicParsing="false" onAction="#Cancel" text="OTMEHA" />
          <Button fx:id="confirmButton" mnemonicParsing="false" onAction="#Confirm" text="OK" />
      </buttons>
       <padding>
          <Insets right="7.0" />
       </padding>
    </ButtonBar>
    <Label layoutX="14.0" layoutY="14.0" text="Введите путь до файла">
      <font>
        <Font name="Arial Black" size="16.0" />
      </font>
```

```
    </Label>
    <TextField fx:id="textField" layoutX="14.0" layoutY="62.0" prefHeight="26.0" prefWidth="338.0" />
    <Button fx:id="choose" layoutX="360.0" layoutY="62.0" mnemonicParsing="false" onAction="#ChooseFile"
prefHeight="26.0" prefWidth="29.0" text="Button" />
  </children>
</AnchorPane>
```

SaveFileController.java

```
package application.gui;

    import application.Application;
    import javafx.fxml.FXML;
    import javafx.scene.control.Button;
    import javafx.scene.control.TextField;
    import javafx.stage.FileChooser;
    import javafx.stage.Stage;

    import java.io.File;
    import java.io.IOException;

public class SaveFileController {

  @FXML
  private Button cancelButton;

  @FXML
  private Button confirmButton;

  @FXML
  private TextField textField;

  @FXML
  void Confirm(){
    Stage stage = (Stage)confirmButton.getScene().getWindow();
    if(!textField.getText().isBlank()) {
      try {
        Application.saveGraphToFile(textField.getText());
      } catch (IOException e) {
        e.printStackTrace();
      }
```

```
        }
        stage.close();
    }

    @FXML
    void Cancel(){
        Stage stage = (Stage)cancelButton.getScene().getWindow();
        stage.close();
    }

    @FXML
    void ChooseDirectory(){
        final FileChooser fileChooser = new FileChooser();
        textField.clear();

        fileChooser.getExtensionFilters().addAll(
                new FileChooser.ExtensionFilter("Graph", "*.graph"));

        File file = fileChooser.showSaveDialog((Stage)textField.getScene().getWindow());
        if (file != null) {
            textField.setText(file.getPath());
        }
    }
}
```

SaveFile.fxml

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

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ButtonBar?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="150.0" prefWidth="400.0" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="application.gui.SaveFileController">
    <children>
```

```
      <ButtonBar layoutY="116.0" prefHeight="40.0" prefWidth="400.0" style="-fx-background-color: BC8F8F;">
        <buttons>
          <Button fx:id="cancelButton" mnemonicParsing="false" onAction="#Cancel" text="ОТМЕНА" />
          <Button fx:id="confirmButton" mnemonicParsing="false" onAction="#Confirm" text="ОК" />
        </buttons>
        <padding>
          <Insets right="7.0" />
        </padding>
      </ButtonBar>
      <Label layoutX="14.0" layoutY="22.0" text="Введите путь к файлу">
        <font>
          <Font name="Arial Black" size="16.0" />
        </font>
      </Label>
      <TextField fx:id="textField" layoutX="14.0" layoutY="62.0" prefHeight="26.0" prefWidth="334.0" />
    <Button fx:id="choose" layoutX="359.0" layoutY="62.0" mnemonicParsing="false"
onAction="#ChooseDirectory" prefHeight="26.0" prefWidth="26.0" text="Button" />
  </children>
</AnchorPane>
```

module-info.java

```
open module Practice.Cheshuin.Matrosov.Masalykin {
  requires javafx.graphics;
  requires javafx.controls;
  requires javafx.media;
  requires javafx.base;
  requires javafx.web;
  requires javafx.swing;
  requires javafx.fxml;
}
```