

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

**ОТЧЕТ**  
**по учебной практике**  
**Тема: Разработка визуализатора алгоритма**

Студент гр. 9303	_____	Куршев Е.О.
Студент гр. 9303	_____	Муратов Р.А.
Студент гр. 9303	_____	Низовцов Р.С.
Руководитель	_____	Жангиров Т.Р.

Санкт-Петербург

2021

## ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студент Куршев Е.О. группы 9303

Студент Муратов Р.А. группы 9303

Студент Низовцов Р.С. группы 9303

Тема практики: разработка визуализатора алгоритма

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.

Алгоритм: алгоритм Куна.

Сроки прохождения практики: 01.07.2021 – 14.07.2021

Дата сдачи отчета: 06.07.2021

Дата защиты отчета: 06.07.2021

Студент гр. 9303	_____	Куршев Е.О.
Студент гр. 9303	_____	Муратов Р.А.
Студент гр. 9303	_____	Низовцов Р.С.
Руководитель	_____	Жангиров Т.Р.

## **АННОТАЦИЯ**

Цель проекта состоит в разработке и реализации визуализатора алгоритма Куна в бригаде из трёх человек на языке программирования Java. Реализация алгоритма будет происходить в три этапа (за три итерации). За каждым участником бригады закреплена своя определённая роль, которую он должен будет выполнять к сроку сдачи.

## СОДЕРЖАНИЕ

	Введение	5
1.	Требования к программе	6
1.1.	Описание модулей проекта через UML диаграммы классов	6
1.2.	Описание работы программы и алгоритма через UML диаграммы состояний	8
1.3.	Описание работы программы и алгоритма через UML диаграммы последовательности	9
1.4.	Графический прототип программы	10
2.	План разработки и распределение ролей в бригаде	12
2.1.	План разработки	12
2.2.	Распределение ролей в бригаде	12
3.	Выводы	14

## **ВВЕДЕНИЕ**

Основными задачами проекта являются реализация алгоритма Куна и получение навыков работы в бригаде.

Алгоритм Куна служит для поиска наибольшего паросочетания в двудольном графе. Этот алгоритм основан на теореме Бержа: Паросочетание является максимальным тогда и только тогда, когда не существует увеличивающих относительно него цепей. Соответственно алгоритм Куна ищет в графе наибольшую увеличивающуюся цепь в двудольном графе.

Алгоритм Куна применяется для распределения и составления соотношения между двумя группами.

## 1. ТРЕБОВАНИЯ К ПРОГРАММЕ

### 1.1. Описание модулей проекта через UML диаграммы классов

Итерация 1: При проектировании архитектуры использовалась схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер (MVC, Model-View-Controller). На рис. 1 приведена UML-диаграмма классов. Классы распределены следующим образом:

- **Model:** AlgoKuhn, Graph, GraphSnapshot, History, Command, NextStep, Reset, PrevStep, ToStart, ToFinish.
- **View:** GUI, GUI\_MenuBar, View
- **Controller:** ControllerInterface, Controller.

Класс AlgoKuhn отвечает за непосредственно сам алгоритм, классы Graph и GraphSnapshot — граф и его снимок (паттерн «Снимок»), History — хранилище «снимков» графа, классы Command, NextStep, Reset, PrevStep, ToStart, ToFinish ответственны за возможные действия алгоритма (паттерн «Команда»). Классы GUI, GUI\_MenuBar, View отвечают за графическую составляющую. Классы ControllerInterface, Controller (здесь используется паттерн «Адаптер», так как Model и View имеют различные интерфейсы) образуют «прослойку», которая отвечает за взаимодействие между моделью и GUI.

Итерация 2 (см. рис. 2): Внесены некоторые изменения в архитектуру проекта: роль «главного» класса перешла от класса View к классу Controller (см. рис. 1). Добавлен интерфейс View (старый класс View удален и также удален класс GUI\_MenuBar, так как его бизнес-логика перенесена в класс GUI), который реализует класс GUI и который должен будет реализовывать класс CLI, отвечающий за работу с терминалом. Также члены бригады решили, что паттерн «Снимок» нужно применять не к классу Graph, а к классу алгоритма AlgoKuhn.

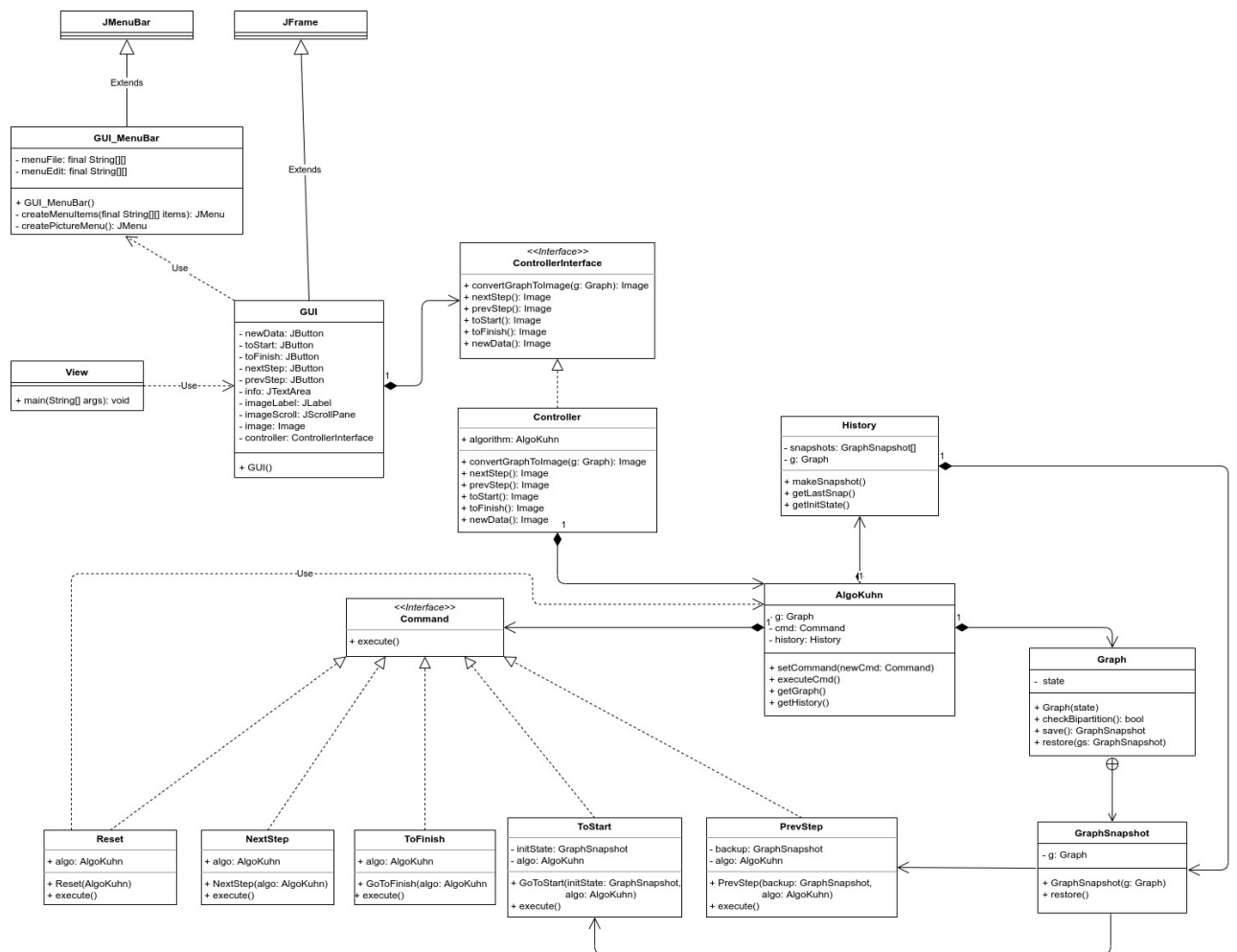


Рисунок 1 – UML-диаграмма классов (1-ая итерация)



## 8



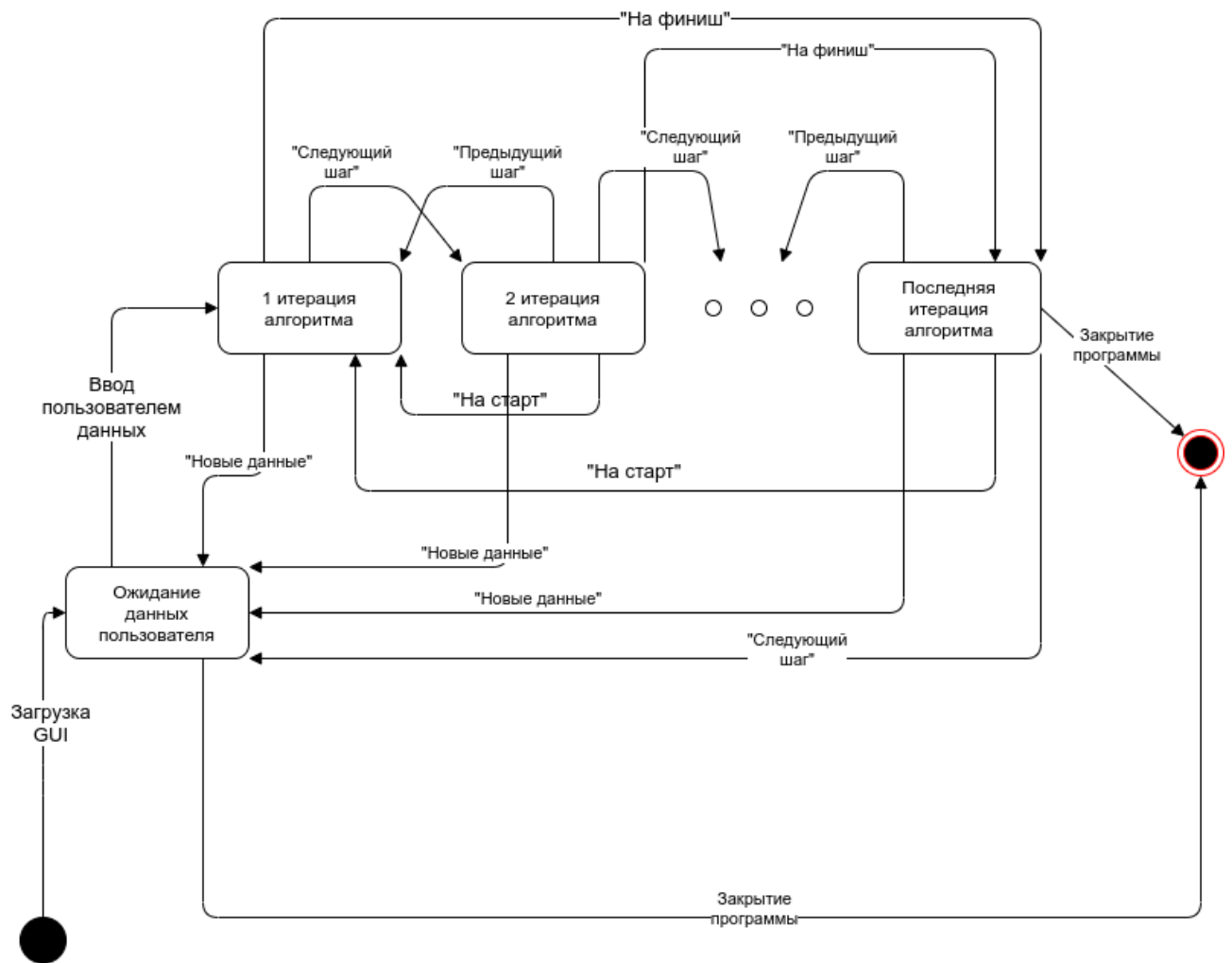


Рисунок 3 – UML-диаграмма состояний

### 1.3. Описание работы программы и алгоритма через UML диаграммы последовательности

Изначально пользователь отправляет команду создать граф. «Создатель» создаёт граф. Далее «Создатель» отправляет запрос «Проверяющей программе» с просьбой проверить граф на двудольность. В результате проверки может произойти две ситуации: граф не двудольный — тогда управление будет возвращено пользователю; граф двудольный — тогда «Проверяющая программа» посылает запрос «Выполняющей программе» с просьбой выполнить алгоритм. Дальше происходит общение между «Выполняющей программой» и «Визуализатором»: «Выполняющая программа» выполняет шаг алгоритма и просит «Визуализатор» вывести граф на экран. После вывода

графа на экран «Визуализатор» передаёт обратно управление «Выполняющей программе». Как только выполнение алгоритма закончится, «Визуализатор» возвращает управление пользователю.

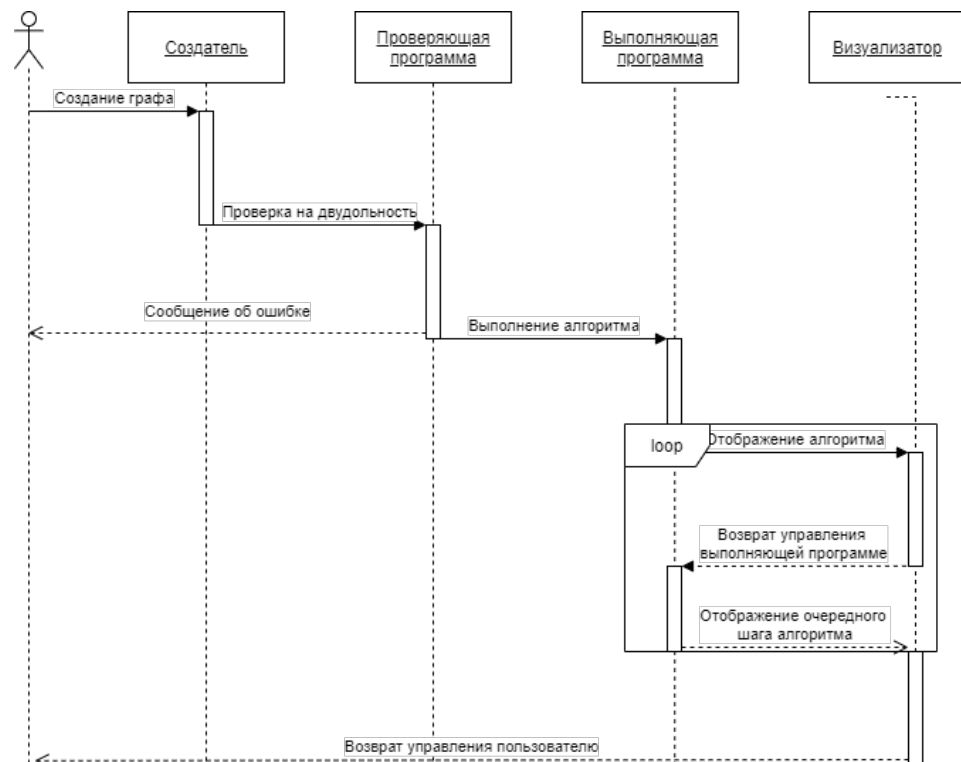


Рисунок 4 – UML-диаграмма последовательности

## 1.4 Графический прототип программы

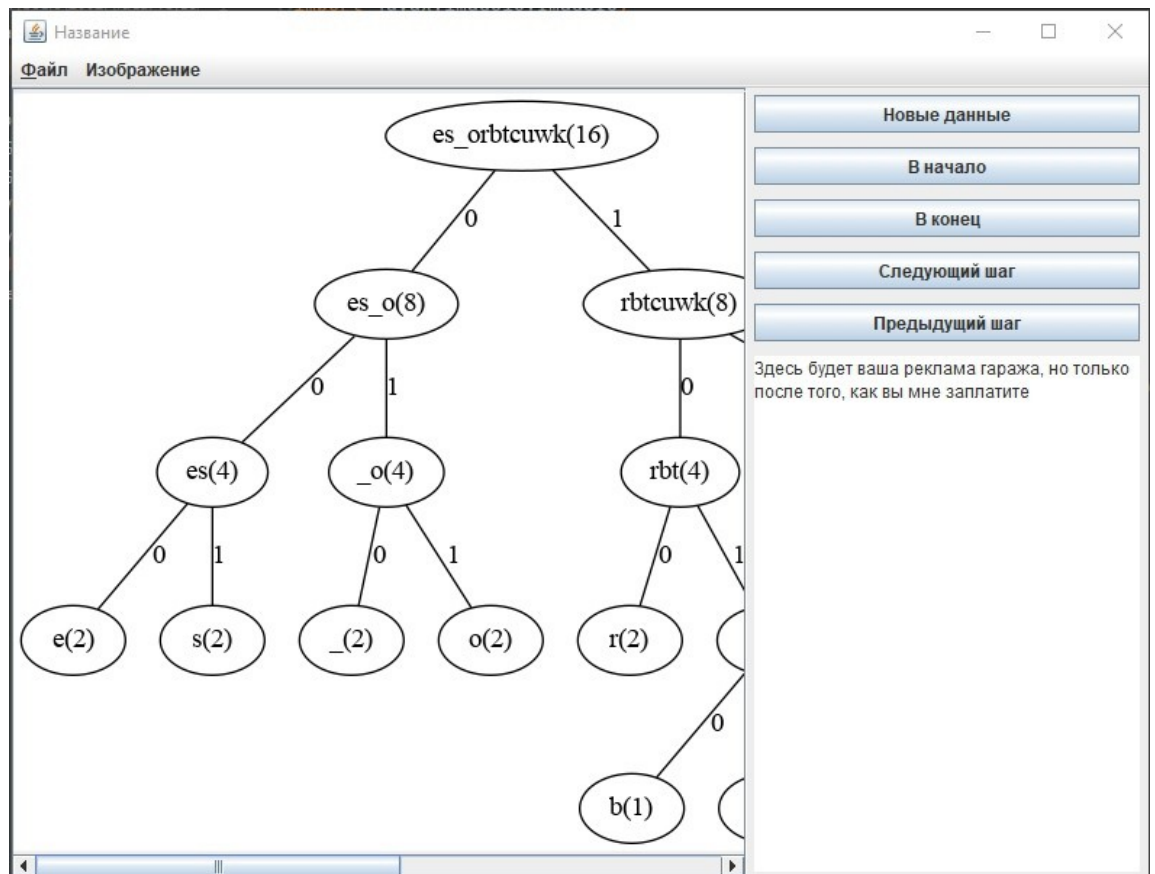


Рисунок 5 – Графический прототип визуализатора

## **2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ**

### **2.1. План разработки**

Итерация 1 (к 6.07.2021):

- 1) Создать прототип GUI.
- 2) Сделать описание архитектуры с помощью UML диаграмм.

Что было сделано:

Было выполнено то, что планировалось.

Итерация 2 (к 9.07.2021):

- 1) Реализовать частичный функционал GUI.
- 2) Реализовать алгоритм Куна.

Что было сделано:

Алгоритм Куна реализован (пока что без паттерна «Команды»), пользователь с помощью GUI выбирает файл, из которого будет прочитана информация о графе. Затем происходит проверка на корректность файла (использовался собственный формат описания графа) и проверка графа на двудольность. Если все хорошо, то алгоритм выводит в консоль результат работы алгоритма, иначе выводится причина, по которой алгоритм Куна не был запущен.

Итерация 3 (к 12.07.2021):

- 1) Реализовать полностью рабочий GUI и CLI.
- 2) Реализовать все взаимодействия с алгоритмом.

### **2.2. Распределение ролей в бригаде**

Низовцов Р.С. – реализация GUI

Куршев Е.О. – реализация алгоритма Куна + создание UML диаграмм

Муратов Р.А. – взаимодействие алгоритма и GUI + создание UML  
диаграмм

### **3. ВЫВОДЫ**

Изначально выполнение проекта было разделено между всеми участниками бригады: все выполняют важную часть в создании проекта.

В ходе первого этапа разработки были созданы UML диаграммы, которые описывают саму программу, её состояния и последовательность её выполнения. Также был создан графический прототип визуализатора.