**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Направление: 09.03.03 – «Прикладная информатика»

КУРСОВАЯ РАБОТА

**УСПЕВАЕМОСТЬ СТУДЕНТОВ НА ФАКУЛЬТЕТЕ**

Студент 2 курса

Группа 09-852

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.  \_\_\_\_\_\_\_\_\_\_\_\_  С.С.Саидмуродов

Научный руководитель

Доцент кафедры

технологий программирования

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.  \_\_\_\_\_\_\_\_\_\_\_\_  И.С.Балафендиева

Казань-2019

Оглавление

[1 Введение 3](#_Toc28295417)

[2 Основная часть 4](#_Toc28295418)

[2.1 Назначение системы 4](#_Toc28295419)

[2.1.1 Анализ предметной области 4](#_Toc28295420)

[2.1.2 Пользовательские требования 4](#_Toc28295421)

[2.1.3 Глоссарий 5](#_Toc28295422)

[2.2 Основные функции системы 6](#_Toc28295423)

[2.2.1 Варианты использования 6](#_Toc28295424)

[2.2.1.1 Вход в систему 6](#_Toc28295425)

[2.2.1.2 Поток преподавателя 6](#_Toc28295426)

[2.2.1.3 Студент 7](#_Toc28295427)

[2.2.1.4 Администратор 7](#_Toc28295428)

[2.2.1.5 Добавление студентов 7](#_Toc28295429)

[2.2.1.6 Внесение результатов сдачи сессии 8](#_Toc28295430)

[2.2.1.7 Получение результатов сессии 8](#_Toc28295431)

[2.2.1.8 Получение статистики 9](#_Toc28295432)

[2.3 Каталог пользователей 9](#_Toc28295433)

[2.4 Каталог требований к интерфейсу 10](#_Toc28295434)

[2.5 Выбор программного средства реализации 10](#_Toc28295435)

[2.6 Структура хранения данных и моделирования сущностей 10](#_Toc28295436)

[2.6.1 Пользователи 10](#_Toc28295437)

[2.6.2 Студенты 12](#_Toc28295438)

[2.7 Структура диалогового взаимодействия с пользователем 14](#_Toc28295439)

[2.8 Основные экранные элементы интерфейса 18](#_Toc28295440)

[2.9 Структура справочной системы 18](#_Toc28295441)

[3 Заключение 19](#_Toc28295442)

[4 Список использованных источников 20](#_Toc28295443)

# Введение

Информационная система – программно-аппаратная система, предназначенная для хранения, поиска и обработки информации. Предназначена для своевременного обеспечения надлежащих людей надлежащей информацией, то есть для удовлетворения конкретных информационных потребностей в рамках определённой предметной области, при этом результатом функционирования системы являются информационные массивы, базы данных или информационные услуги.

Целью данной курсовой работы является создание базы информационной системы для автоматизации процесса учета и анализа студентов и их успеваемости.

# Основная часть

В выбранном мной задании требуется создать информационную систему, поддерживающую режимы учета учащихся и результатов сдачи экзаменов, анализ сессии по семестрам, по факультетам, специальностям, генерации отчетов отличников и двоечников.

## Назначение системы

Система предназначена для упрощения записи результатов, учёта студентов, получения статистики результатов сдачи сессии. Система должна быть масштабируемой, то есть поддерживать добавление новых сущностей (студентов и преподавателей).

### Анализ предметной области

Область применения информационной системы – высшее учебное заведение. Ясно что программой пользуются преподаватели и студенты. Преподаватели должны иметь возможность выставлять результаты сессии и просматривать статистику. Студенты должны иметь возможность увидеть свои оценки. Для добавления (учёта новых) преподавателей и студентов целесообразно назначить управленца, мы его условимся называть администратором. Для более глубокого анализа предметной области постараюсь сформировать пользовательские требования.

### Пользовательские требования

Первым шагом в формировании требований является идентификация опорных точек зрения. Во всех методах формирования требований, основанных на использовании точек зрения, начальная идентификация является наиболее трудной задачей. Но как было сказано ранее, у нас уже есть как минимум три типа пользователей: администратор, преподаватели и студенты. Для формирования требований необходимо продумать базовый функционал необходимый для применения информационной системы. Результатом мозговой атаки является диаграмма идентификации точек зрения *(Рисунок 1*):

Рисунок 1 Диаграмма идентификации точек зрения

### Глоссарий

*Студент* – учащийся учебного заведения.

*Преподаватель* – преподаватель учебного заведения.

*Пользователь* – пользователи, зарегистрированные в системе, студенты и преподаватели.

*Администратор* – человек имеющий полномочия добавления или удаления преподавателей в систему.

*Результаты* экзаменов – оценки, полученные студентом на экзамене по 100-бальной системе.

*Отличники* – студенты, со средним баллом на экзаменах более 86 баллов.

*Хорошисты* – студенты, со средним баллом на экзаменах более 76 и менее 85 баллов.

*Середнячки* – студенты, со средним баллом на экзаменах более 56 и менее 75 баллов.

*Двоечники* – студенты, со средним баллом на экзаменах менее 55 баллов.

*Сессия* – экзамены, проходящие в конце каждого семестра.

*Логин* – имя пользователя, необходимое для входа в систему. Логин не может быть пустым, содержать пробелы и начинаться с символов не являющимися буквой.

*Пароль* – пароль пользователя, необходимый для входа в систему и защиты системы от несанкционированного доступа. Пароль не может быть пустым, не должен содержать пробел и должен содержать в себе минимум 4 цифры и 4 символа.

## Основные функции системы

В результате анализа предметной области мы получили диаграмму идентификации точек зрения, теперь же для более конкретного описания пользовательских требований к системе необходимо продумать варианты использования системы.

### Варианты использования

#### Вход в систему

Краткое описание

Данный вариант использования описывает вход пользователя в систему учета успеваемости студентов. Пользователем могут выступать студенты, преподаватели и администратор.

Основной поток

1. Система запрашивает имя пользователя и пароль.

2. Пользователь вводит имя и пароль.

3. Система подтверждает имя и пароль, после чего открывается доступ в систему.

4. В зависимости от типа вошедшего пользователя запускается его поток. (*поток Преподавателя, поток Студента, поток Администратора*).

Альтернативные потоки: Неправильное имя/пароль

Если во время выполнения основного потока обнаружится, что пользователь ввел неправильное имя и/или пароль, система выводит сообщение об ошибке. Пользователь может вернуться к началу основного потока или отказаться от входа в систему, при этом выполнение варианта использования завершается

#### Поток преподавателя

Краткое описание

Поток преподавателя выполняется при определении того, что в систему вошел преподаватель.

Основной поток

1. Система предлагает на выбор выполнение следующих потоков: *Внесение результатов сдачи сессии, Получение статистики.*

Альтернативный поток

При желании преподаватели могут выйти из данного потока нажав кнопку выхода

#### Студент

Краткое описание

Поток студента выполняется при входе в систему студента.

Основной поток

1. Система предлагает на выбор выполнение следующих потоков: *Получение результатов сессии, Получение статистики*.

Альтернативный поток

При желании студенты могут выйти из данного потока нажав кнопку выхода.

#### Администратор

Краткое описание

Поток администратора запускается при входе администратора.

Основной поток

1. Система предлагает на выбор выполнение следующих потоков: *Добавление* *студентов*, *Добавление* *преподавателей*.

Альтернативный поток

При желании администратор может выйти из данного потока нажав кнопку выхода.

#### Добавление студентов

Данный поток описывает процесс добавления студента в систему.

Основной поток

1. Система предлагает ввести следующие данные студента: фамилию, имя, отчество, академическую группу, направление обучения, факультет.

2. Система проверяет существует ли студент с введёнными данными и предлагает ввести логин и пароль студента, система проверяет подходят ли логин и пароль по заданным правилами.

3. Система создаёт запись в базе данных запись на основе введенных данных.

4. Основной поток завершается

Альтернативные потоки

Если в базе данных существует запись о данном студенте, то система выводит сообщение об ошибке и завершает поток. Если введенные пароль и логин не подходят по правилам, описанным в глоссарии, то система выводит ошибку, после чего пользователь может исправить логин или пароль.

#### Внесение результатов сдачи сессии

Данный поток описывает внесение результатов сессии

Основной поток

1. Система предлагает выбрать действие (добавить оценку, изменить оценку) и выполняет выбранные подчинённые потоки.

Добавление оценки

1. Система предлагает выбрать студента

2. Система предлагает ввести оценку студента

3. Система выполняет запись в базу данных

Изменить оценку

1. Система предлагает выбрать студента

2. Система предлагает изменить оценку студента

3. Система меняет значение в записи в базе данных

Альтернативный поток

Если оценка была добавлена раньше, то предлагает изменить её или выйти на основной поток.

Предусловие

Пользователь должен быть преподавателем.

#### Получение результатов сессии

Данный поток необходим для описания процесса получения результатов сессии

Основной поток

Система находит записи о успеваемости данного студента и выводит их на экран в виде таблицы, распределяя оценки по семестрам и предметам.

Альтернативный поток

Если сессия ещё не началась, то система уведомляет об этом и выводит результаты предыдущих сессий.

Предусловие

Текущий пользователь должен быть студентом

#### Получение статистики

В данном потоке преподаватели и студенты могут смотреть статистику по сессиям, семестрам и факультетам.

Основной поток

1. Система предлагает выбрать как предоставить данные (по группам, по факультетам)

2. На основе выбора выдает результаты сессий, средние баллы студентов.

Альтернативный поток

Если в базе данных еще нет оценок, то пользователь получит соответствующее сообщение и поток завершится.

Генерация отчетов

В данном потоке студенты и преподаватели могут получить отчеты отличников и двоечников.

## Каталог пользователей

Как мы выяснили ранее в нашей пока что абстрактной системе существует три типа пользователей: администратор, преподаватели и студенты. При этом каждый тип пользователей различается от другого по возможностям работы в нашей системе. Но, с другой стороны, типы пользователей при абстрагировании имеют различный набор персональных данных для обработки в системе. Например, нам не важны данные администратора, так как они не представляют интересы других пользователей, поэтому у администратора будет только логин и пароль. У преподавателей будем сохранять имена, фамилии (для отображения в списках преподавателей в окне администратора), логин и пароль. У студентов нам нужны больше данных: имя, фамилия, группа (для группировки по академической группе), факультет (для аналитики по факультетам), институт (для аналитики по институтам).

## Каталог требований к интерфейсу

Интерфейс должен быть удобным и не требующим много действий для выполнения операций. Для этого я счёл необходимым замены текстовых полей в окнах регистраций и ввода данных на выпадающие списки там, где это возможно. Так же для упрощения и облегчения восприятия аналитические данные будут демонстрироваться в виде диаграмм.

## Выбор программного средства реализации

Основным языком программирования системы выбрана Java, так как её полная объектно-ориентированность позволяет структурировать большое количество программного кода, что весьма удобно при создании крупных проектов. Для разработки информационной системы потребуется среда разработки (NetBeans, IntelliJ IDEA, Eclipse; Я лично предпочитаю IntelliJ IDEA от Jet Brains, так как она позволяет быстро находить ошибки в коде и имеет функции автоматического исправления многих ошибок в коде) для создания Java программ и их последующей отладкой, библиотека для работы с графическим интерфейсом JavaFX. Для предварительной отладки интерфейса буду пользоваться программной Scene Builder.

## Структура хранения данных и моделирования сущностей

Так как было обусловлено о неиспользовании в курсовой работе баз данных, то для хранения данных будем пользоваться фалами. Для разработки программы воспользуемся схемой разделения данных MVC. Далее будет описана модель программы, представлением окон послужат fxml-файлы, а контроллерами выступят классы которые будут именоваться как название фала fxml со словом Controller на конце.

### Пользователи

Как было сказано ранее в нашей системе существуют пользователи, делящиеся на три категории: администратор, студенты и преподаватели. Так как у них общие поля только логин и пароль, то мы можем хранить логин и пароль в одном файле, где каждая строка будет соответствовать одному пользователю, через пробел будем записывать логин и пароль. Но логин и пароль являются не только ключом доступа в систему, но и данными для администратора, так как он может задать новых пользователей и изменить логин и пароль для созданный ранее пользователей. С другой стороны, нам необходимо по логину паролю идентифицировать не только тип пользователя, но и конкретного пользователя. Для идентификации пользователей мы будем использовать сами логины, так как при регистрации будет проверяться уникальность логинов. Тип данных пользователей для хранения логинов, паролей и типов пользователей зададим класс User, с полями логин, пароль и тип пользователя.

Рисунок Код полей класса User

Так же в классе зададим статичный список для хранения пользователей, что бы при большом количестве пользователей и нескольких попытках нам не пришлось считывать файл с диска, так как это может снизить производительность. Поэтому опишем метод initArrray(), который будет срабатывать при инициализации системы. Данный метод будет считывать файл с логинами, паролями и типами пользователей, будет создавать объекты User и добавлять их в статичный список. Данные о пользователях будут храниться в файле users.txt.

Рисунок Метод initArray() для инициализации статичного поля userArrayList

Так как для хранения в оперативной памяти объекта класса User используется конструктор, то для создания нового пользователя зададим метод addUser(), добавляющий нового пользователя в статичный список и записать его логин, пароль и тип в файл.

Рисунок Метод addUser для добавления нового пользователя

Таким образом мы получим структуру для хранения данных пользователей для входа в систему в виде файла.

Во время входа в систему срабатывает метод userType(), в который передаются логин и пароль как параметры, а он возвращает строку в которую записан тип пользователя, соответственно мы можем проверить является ли эта строка “admin”, “student” или “professor”, для случая ввода не существующих в системе логинов и паролей метод userType() будет возвращать строку “unknown”.

Рисунок Метод для проверки типа пользователя

### Студенты

Кроме логина пароля в нашей информационной системе пользователи имеют тип, и у каждого типа пользователей свой тип данный, кроме администратора, так как мы обусловились считать, что данные администратора не являются необходимыми. Для дальнейшей обработки данных студентов, у студентов должны быть поля: группа, факультет, институт, имя и фамилия. Так как название факультетов и институтов часто состоят из нескольких слов, то для удобства будем хранить их в отдельных файлах, и сделаем классы для работы с ними.



Рисунок Поля класса Student

В классе оставим поле login для привязки к классу User, и сделаем поле studentID для идентификации студента, так как login может изменятся. При изменении логина в User будем менять его и для объекта Student. 

Рисунок Блок кода для нахождения уникального ID

Для того что бы studentID не повторялся и не возрастал бесконечно зададим его вычисление в конструкторе, пройдемся по studentArraylist, в который записываются студенты из файла, и найдём первое целое число по возрастанию, которое не встречается среди studentID существующих студентов. Так же в классе пропишем геттеры и сеттеры.

Классы Group, Faculty и Institute имеют схожий функционал, хранят в своих файлах названия и идентификаторы, с одной лишь разницей в форматировании записи между именами и идентификаторами ставятся двойные проценты «%%», так как имена могут содержать пробелы; в файле students.txt записываются только идентификаторы, а при считывании объекты воссоздаются по данным из файла, а в объекты Student передаются ссылки на объекты Group, Faculty и Institute. Таким образом мы избегаем многоразовой записи данных в файлы, экономим оперативную память, и создаём возможность лёгкого переименования институтов, факультетов и институтов.

## Структура диалогового взаимодействия с пользователем

Рисунок Окно входа в систему

При запуске программы происходит открытие окна «Вход в систему», в котором выполняется поток *Вход в систему.*

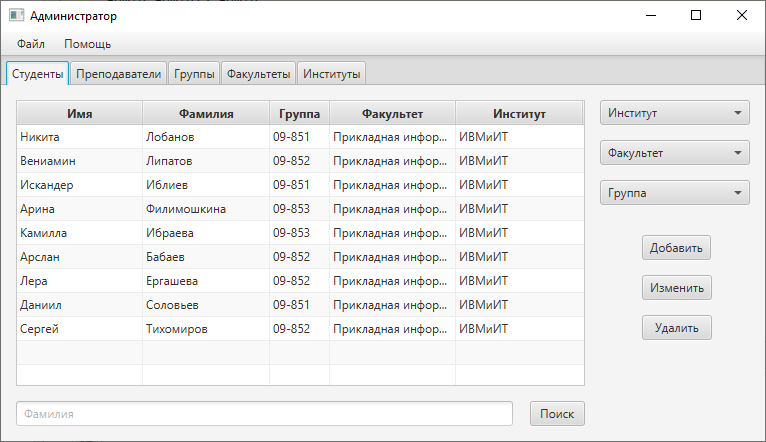
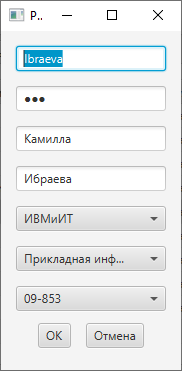
Сначала опишу поток администратора. Если после ввода логина и пароля типом пользователя вернулось “admin”, то открывается сцена администратора, при чем не происходит закрытия окон. В своём окне администратор может просматривать студентов, преподавателей, группы факультеты и институты. При желание может удалять, добавлять и изменять данные студентов и преподавателей. Информация о студентах выводится в виде таблицы со столбцами «Имя», «Фамилия», «Группа», «Факультет» и «Институт». Для упрощения навигации по таблице, в окне присутствуют выпадающие списки, для задания фильтров по группам, факультетам и институтам. Так же в окне есть поиск. Для работы с данными студентов присутствуют функциональные кнопки «Добавить», «Изменить» и «Удалить», которые соответствуют добавлению нового студента, изменению данных существующего, и удалению студентов из системы. Так же в окне администратора есть меню с вкладками Файл и Помощь, в которых администратор может выбрать действия, например выйти, получить статистику, получить информацию о ПО.

Рисунок Рабочее окно администратора системы

Рисунок Редактор студентов

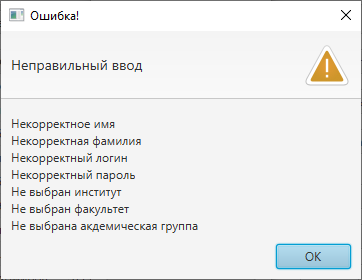
При добавлении новых студентов или изменении данных существующих, выводится окно Редактор студентов. При изменении данных оно автоматически заполняется данными, при добавлении оно появляется пустым. Для ускорения процесса добавления и изменения в окне поля группы, направления и института задаются с помощью выпадающего списка. При нажатии на кнопку Отмена происходит закрытие окна без изменения данных (без создания нового студента в системе). При нажатии ОК сначала данные проверяются на соответствие с требованиями, и потом только происходит добавление или изменение. При несоответствии выходит окно с перечнем ошибок.

Рисунок Окно с ошибками при регистрации нового студента

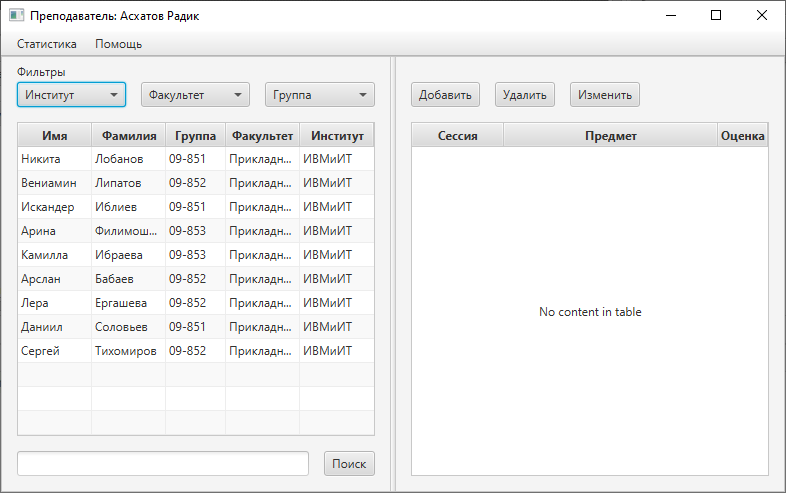
Администратор добавляет студентов и преподавателей, а оценки оформляют преподаватели. Окно преподавателя состоит из двух таблиц, в одном ыводится список студентов, в другом оценки выбранного студента. Преподаватели могут ставить оценки, изменять их и удалять ошибочные оценки. Так же преподаватели могут получить статистику. При попытке удаления или изменения оценки без её выбора выходит ошибка, то же самое произойдет при попытке добавления, изменения и удаления оценки без выбора студента. Для более учкоренной работы с таблицей студентов можно фильтровать по институтам, факультетам и группам, есть поиск по фамилии.

Рисунок Окно преподавателя

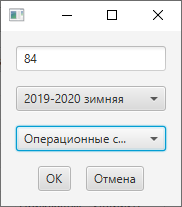
При добавлении и редактировании оценок появляется окно редактора оценок. Для ускоренной записи оценок преподавателю нужно лишь набрать оценку и выбрать сессию и предмет. Оценка целое число, не может быть отрицательной, меньше 100. При вводе неправильной оценки выводится окно ошибки.

Рисунок Окно редактора оценок

Так же преподаватели могут просмотреть статистику оценок.

В меню помощь всех окон системы есть пункт Выход, при нажатии на неё пользователь попадает в главное окно входа.

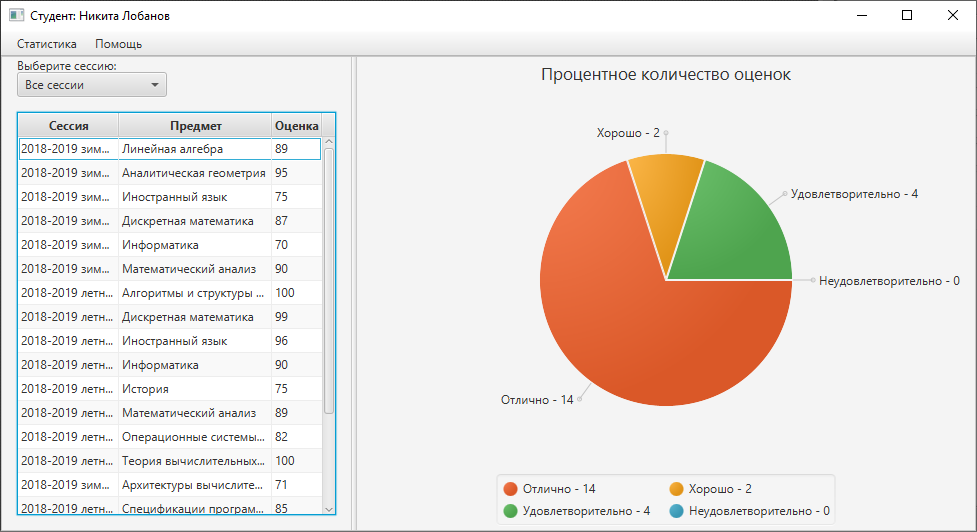
У студентов есть свои окна со своей компоновкой. На таблице выводятся оценки в виде сессия – предмет – оценка, есть возможность фильтровать оценки по сессиям. Справа от таблицы расположена аналитическая диаграмма. При применении фильтрации данные обновляются на диаграмме и таблице. Студент также может просмотреть статистику оценок других студентов по группам, при чем статистика анонимная. Статистика ведется по группам, факультетам и институтам.

Рисунок Окно студента

## Основные экранные элементы интерфейса

Для ввода текста использовались TextField, таблицы для вывода массивов данных, диаграммы (PieChart) для графического представления, и ComboBox для организации выбора.

## Структура справочной системы

На всех окнах в меню Помощь присутствует пункт Инструкция, при нажатии на которую открывается окно с инструкцией по использованию ПО. При чём инструкция для каждого пользователя своя.

# Заключение

В ходе выполнения курсовой работы мной был получен опыт разработки информационной системы. Стало ясно что неважно насколько крутой будет информационная модель, необходимо её правильно преподать пользователю. Например, если бы в нашей информационной системе не использовались диаграммы, то было бы сложнее уловить информацию. То же самое касается других элементов интерфейса.

# Список использованных источников

1. Java® Platform, Standard Edition & Java Development Kit, Version 13 API Specification [Электронный ресурс] // docs.oracle.com: официальный сайт документации Java. URL: <https://docs.oracle.com/en/java/javase/13/docs/api/index.html> (дата обращения: 10.12.2019)
2. API documentation for JavaFX 13 [Электронный ресурс] // openjfx.io: официальный сайт проекта JavaFX. URL: <https://openjfx.io/javadoc/13/> (дата обращения: 5.12.2019)
3. Раздел Java сайта Metanit [Электронный ресурс] // metanit.com: Сайт о программировании. URL: <https://metanit.com/java/> (дата обращения: 9.12.2019)