МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

Факультет информационных технологий и компьютерной безопасности

(факультет)

Кафедра Систем управления и информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине Технологии программирования

тема Разработка программного средства «Drawau»

**Расчетно-пояснительная записка**

Разработал студент Я.С. Каминский

Подпись, дата Инициалы, фамилия

Руководитель О.В. Минакова

Подпись, дата Инициалы, фамилия

Члены комиссии

Подпись, дата Инициалы, фамилия

Подпись, дата Инициалы, фамилия

Нормоконтролер

Подпись, дата Инициалы, фамилия

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

Кафедра Систем управления и информационных технологий в строительстве

ЗАДАНИЕ

на курсовой проект

по дисциплине Технологии программирования

Тема работы Разработка программного средства «Drawau»

Студент группы бИС-191 Каминский Ян Сергеевич

Фамилия, имя, отчество

Технические условия процессор Ryzen5 2500U 2,0 ГГц, операционная система Windows 10, ОЗУ 8 ГБ

Содержание и объем проекта (графические работы, расчеты и прочее):

теоретические аспекты современного положения разработки информационных систем на основе web-технологий (5 страниц); проектирование информационной системы (14 страниц); реализация информационной системы (6 страниц); 26 рисунок, 7 таблиц, 3 приложения.

Сроки выполнения этапов теоретические аспекты современного положения разработки информационных систем на основе web-технологий (08.02 – 13.03); проектирование информационной системы (15.03 – 17.04); реализация информационной системы (19.04 – 22.05); оформление расчётно-пояснительной записки (24.05 – 29.05)

Срок защиты курсового проекта 29.05.2022

Руководитель О.В. Минакова

Подпись, дата Инициалы, фамилия

Задание принял студент 02.02.2022 Я.С. Каминский

Подпись, дата Инициалы, фамилия

**Замечания руководителя**

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ**

При разработки приложения требуется составлять разного рода диаграммы и схемы. Таким образом, были придуманы различные средства автоматизировать процесс создания диаграмм. Были придуманы стационарные приложения и онлайн сервисы для осуществления этих целей. Они позволяют без сложностей и достаточно быстро составить самые разные диаграммы.

Было решено разработать программный продукт, который позволит не только создавать диаграммы, но и сохранять их как в качестве просматриваемого изображения, так и в качестве специального файла, для того чтобы сохранять прогресс и иметь возможность вернуться к разработке. Одной из особенности приложения будет возможность создать диаграмму автоматически, на основе файлов-классов.

Актуальность моего приложения заключается в создании эффективной программы, пользователи которой смогут запросто разобраться в интерфейсе и доступных функциях.

Главной задачей моей курсовой работы заключается в правильности составленных алгоритмов и функций компонентов. Потребуется создать полноценную правильно-функционирующую программу и изучить некоторые процедуры и функции.

1. **Формирование требований к программному продукту и проекту**
   1. **Описание предметной области программы**

Диаграмма — графическое представление данных линейными отрезками или геометрическими фигурами, позволяющее быстро оценить соотношение нескольких величин. Представляет собой геометрическое символьное изображение информации с применением различных приёмов техники визуализации.

Диаграмма классов — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Приложение для построения диаграммы классов представляет собой комплекс инструментов, позволяющий изобразить связь между классами в рамках UML. Все сущности реального мира, с которыми собирается работать программист, должны быть представлены объектами классов в программе. При этом у каждого класса должно быть только одно назначение и уникально осмысленное имя, которое будет связано с этой целью.

Рассмотрим элементы диаграммы классов:

Класс — элемент диаграммы, обозначающий множество объектов, обладающих одинаковой внутренней структурой, поведением и отношениями с объектами других классов. Изображается класс на диаграмме в виде прямоугольника, разделённого на три секции: имя класса, список полей, список методов.

Отношение зависимости — отношение зависимости используют, чтобы показать, что изменение одного класса требует изменение другого класса.Выглядит как пунктирная стрелка.

Отношение наследования — отношение наследования используется, чтобы показать, что один класс является родителем (базовым классом или суперклассом) для другого класса (потомка, производного класса). Выглядит как обычная стрелка с плоским концом.

Отношение агрегации — отношение агрегации между двумя классами показывает, что один из них включает в себя другой класс в качестве составной части. Выглядит как стрелка с ромбом на конце.

Отношение композиции — Отношение композиции является частным случаем отношения агрегации. Однако у него есть одно отличие – классы-части, которые он соединяет с классом-целым, не могут существовать обособленно.

Отношение ассоциации — Отношение ассоциации используют, чтобы показать, что между классами (например, между двумя классами) существует некоторая связь. Обычно с помощью него на диаграмме классов показывают, что один класс пользуется функционалом другого класса. Выглядит как стрелка с заострённым концом.

**1.2 Анализ существующих решений**

Так как разрабатываемое приложение предназначено для составления диаграмм классов, будем рассматривать соответствующие аналоги.

|  |  |  |  |
| --- | --- | --- | --- |
| Проблема | Сегмент рынка | Drawio | Make flowchart |
| Сохранение в качестве изображения | U | + | + |
| Сохранение в качестве специального файла | U | + | + |
| Автоматическое создание диаграмм | U | ‒ | ‒ |

Исходя из проведенного анализа можно сделать следующие выводы относительно требований к разрабатываемому продукту:

1. Разрабатываемое приложение должно иметь возможность сохранить диаграмму как изображение.

2. Разрабатываемое приложение должно иметь возможность сохранить диаграмму как специальный файл.

3. Разрабатываемое приложение должно иметь возможность автоматически создать диаграмму из файлов-классов.

**1.3 Моделирование вариантов использования**

Функциональные требования к программному средству «drawau»:

|  |  |
| --- | --- |
| **Идентификатор функционального требования** | **Описание** |
| FR01 | Добавление фигуры класса |
| FR02 | Добавление стрелки-наследования |
| FR03 | Добавление стрелки-агрегации |
| FR04 | Добавление стрелки-зависимости |
| FR05 | Сохранение работы |
| FR06 | Открытие работы |
| FR07 | Автоматическое создание диаграммы |

На рисунке 3 представлена диаграмма вариантов использования программы.

**2. Проектирование программного средства**

В качестве среды разработки программного средства выбрана свободная интегрированная среда разработки Intellij IDEA. Для разработки данного ПО используется модуль JavaFX.

**2.1 Архитектура приложения**

Поскольку пользовательский интерфейс будет зависеть от архитектуры приложения, с неё и начнём.

Разрабатываемое приложение имеет событийно-ориентированную архитектуру, рисунок 4.

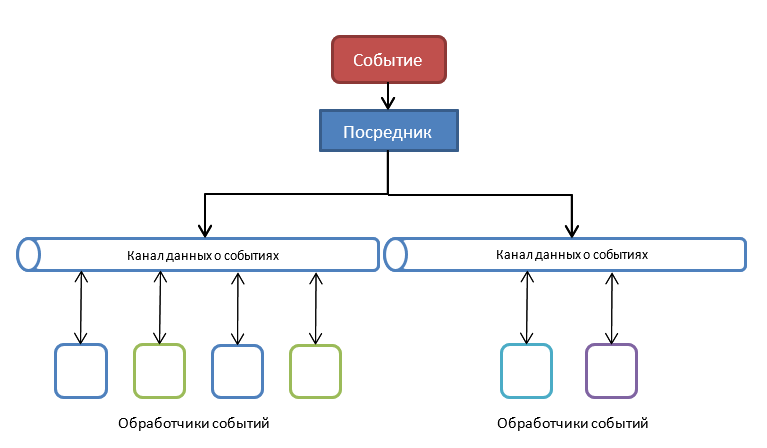


Рисунок 4 — Событийно-ориентированная структура

Архитектура, управляемая событиями — является шаблоном архитектуры программного обеспечения, позволяющим создание, определение, потребление и реакцию на события.

Событие можно определить, как «существенное изменение состояния». Например, когда покупатель приобретает автомобиль, состояние автомобиля изменяется с «продаваемого» на «проданный». Системная архитектура продавца автомобилей может рассматривать это изменение состояния как событие, создаваемое, публикуемое, определяемое и потребляемое различными приложениями в составе архитектуры.

**2.2 Проектирование интерфейса пользователя**

С точки зрения программного обеспечения интерфейс состоит из двух компонентов: набора процессов ввода-вывода и процесса диалога (регулируемого обмена информацией между пользователем и компьютером). Обмен информацией осуществляется в виде передачи сообщений и управляющего сигнала.

Для создания пользовательского интерфейса было решено использовать JavaFX. JavaFX — платформа на основе Java для приложений с насыщенным графическим интерфейсом. Может использоваться в качестве создания настольных приложений, работающих непосредственно в операционных системах, и для интернет-приложений, работающих в браузерах, и для приложений на мобильных устройствах. JavaFX упрощает создание графического интерфейса за счет графической оболочки.

Исходя из сформулированных требований к программному инструменту, главное окно должно содержать рабочую область и боковую панель с следующими кнопками:

• Создать файл

• Открыть файл («.png» или «.drawau»)

• Сохранить файл

• Создать диаграмму автоматически

• Список фигур:

• Фигура класса

• Фигура отношения наследования

• Фигура отношения агрегации

• Фигура отношения зависимости

Сбоку слева в качестве боковой панели расположен VBox, на котором расположены кнопки, осуществляющие основной функционал программы.

Сбоку справа в качестве рабочей области расположена панель Pane, на которой будут располагаться фигуры класса и стрелки отношений.

В качестве фигур класса также был использован VBox.

В качестве фигур отношения был использован Group, в котором в качестве линии использовалось Line, а в качестве наконечника стрелки ‒ Polygon.

В качестве панели для выбора фигуры была использована панель FlowPane.

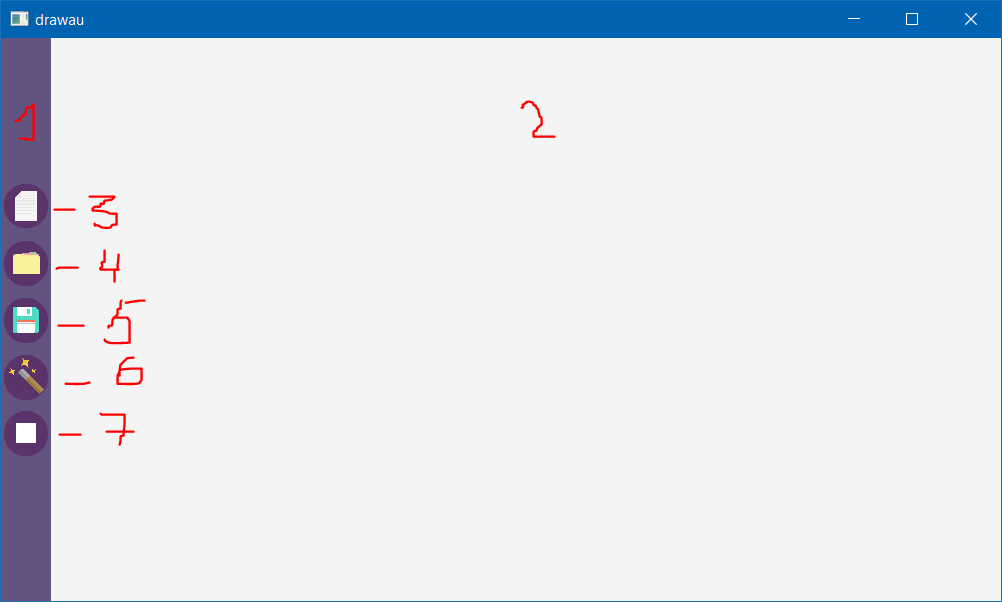


Рисунок 5 — Главное окно программы

Главное окно делиться на следующие области:

1. Боковая панель

2. Рабочая область

3. Создать файл

4. Открыть файл

5. Сохранить файл

6. Магия(Автоматическое создание диаграммы)

7. Панель фигур

**3. Конструирование программного средства**

**3.1 Описание классов приложения**

Диаграммы классов используются при моделировании ПС наиболее часто. Они являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру.

Диаграмма классов не отображает динамическое поведение объектов, изображенных на ней классов. На диаграммах классов показываются классы, интерфейсы и отношения между ними. Диаграмма классов определяет типы классов приложения и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

В ходе проектирования программы были разработаны следующие классы:

* DrawauController — класс приложения, содержащий элементы управления
* DrawauIFigure — интерфейс, обязующий описание методов draw() и location()
* DrawauArrow — абстрактный класс, реализующий интерфейс DrawauIFigure. В этом классе описаны основные свойства элементов отношения.
* DrawauInheritArrow — класс представляющий собой элемент отношения-наследования
* DrawauDependenceArrow — класс представляющий собой элемент отношения-зависимости
* DrawauAggregationArrow — класс представляющий собой элемент отношения-агрегации
* DrawauAnimation — класс отвечающий за анимации элементов управления
* DrawauFigureClass — класс, реализующий интерфейс DrawauIFigure, отвечает за элемент фигуры класса
* DrawauFigureClassBuilder — класс, отвечающий за построение класса DrafauFigureClass
* DrawauFileManager — класс, отвечающий за работу с файлами: сохраниение, загрузку.

Описание атрибутов и методов класса DrawauController:

* Initialize() — функция, которая запускается при старте приложения
* bindSubscripeStatusLabel() — функция, которая привязывает поведение стрелок к элементу Label, позволяя отслеживать статус добавляемой стрелки
* setNewButtonBehavior() — функция, которая описывает действие кнопки создания файла
* setOpenButtonBehavior() — функция, которая описывает поведение кнопки открытия файла
* setSaveButtonBahavior() — функция, которая описывает поведение кнопки сохранения файла
* setMagicButtonBehavior() — функция, которая описывает поведение кнопки автоматического создания диаграммы
* setArrowButtonBehavior() — функция, которая описывает поведение кнопок добавления элементов отношения
* setClassFigurePanelBehavior() — функция, которая описывает поведение панели добавления элемента фигуры класса
* addAnimationForButtons() — функция, которая описывает добавление анимации элементам управдения
* setClassFigureButtonBehavior() — функция, которая описывает добавление элемента фигуры класса
* addClassFigurePanelContent(VBox) — функция, которая описывает содержание элемента панели класса фигуры
* addFieldBox() — функция, которая описывает определенный элемент панели класса фигур

Описание атрибутов и методов класса DrawauAnimation:

* setButtonWidthAnimation(Button…) — функция, которая описывает добавление анимации кнопкам
* setClickFiguresButtonAnimation(Button, FlowPane) — функция, которая описывает анимацию клика по кнопки фигуры

Описание атрибутов и методов класса DrawauIFigure:

* draw() — функция, которая отвечает за рисование фигуры
* location(Pane) — функция, которая отвечает за расположение фигуры на рабочей области

Описание атрибутов и методов класса DrawauArrow:

* initialize() — инициализирует основные свойства класса
* notifyAction() — функция, которая рисует новое положение стрелки

Описание атрибутов и методов класса DrawauFigureClass:

* initialize() — инициализирует основные свойства класса
* addDragAction() — добавляет фигуре способность перемещаться с помощью мыши
* addClickAction() — описывает событие нажатия на фигуру
* subscribe() — подписывает на себя стрелки отношений
* notifySubscribers() — оповещает подписчиков об изменение своего местоположения

Описание атрибутов и методов класса DrawauFileManager:

* saveImageFile(Pane, File) — сохраняет диаграмму классов в качестве изображения
* saveDrawauFile() — сохраняет диаграмму классов в качестве специального файла
* openDrawauFile() — открывает специальный файл и преобразовывает данные оттуда в диаграмму классов
* doMagic() — открывает папку с файлами-классами и автоматически создает диаграмму классов

**3.2 Моделирование взаимодействия пользователя с программой**

Вспоминая описанные выше требования программа должна выполнять следующие функции:

• Открытие файла

• Сохранения файла

• Автоматическое создание файла

• Добавление фигур на рабочую область

С точки зрения пользователя все эти функции будут выполняться по нажатию кнопок.

На рисунке показа диаграмма процессов сообщений между экземплярами классов, отвечающих за открытие файла. После того как пользователь нажмет на кнопку «открыть», в контроллере запускается инициализированная функция, где создается объект класса FileChooser, вызывается окно выбора файла, после выбора файла вызывается метод openDrawauFile() класса DrawauFileManager, возвращающий лист фигур. После на рабочей области появляются все фигуры и связи между ними.

После того как пользователь нажмет на кнопку «сохранить», в контроллере запускается инициализированная функция, где создается объект класса, позволяющий выбрать директорию сохранения и имя файла. Затем вызывается метод saveImageFile() или saveDrawauFile() в зависимости от выбранного формата файла при сохранении.

После того как пользователь нажмет на кнопку «магия», в контроллере запускается инициализированный метод, вызывающий метод doMagic() класса DrawauFileManager, возвращающий лист элементов класса и связей между ними.

После того как пользователь нажмет на кнопку «фигуры», в контроллере запускается инициализированный метод, который делает панель фигур видимой.

После того как пользователь нажмет на кнопку создания фигуры класса, в контроллере вызовется инициализированный метод, который сделает панель создания фигур класса видимой, где после подтверждения создания фигуры создается объект DrawauFigureClass с помощью DrawauFigureClassBuilder и вызываются методы location() и draw().

После того как пользователь выберет стрелку для создания, в контроллере вызовется инициализированный метод, который поменяет Label статус на «отметка начальной точки», создастся объект соответствующей стрелки, вызовется метод location(), затем после выбора начальной точки(при нажатии на определенную фигуру класса) вызовется метод нажатия соответствующей фигуры, где инициализируются определенные поля стрелки и изменят статус Label на «отметка конечной точки», после выбора конечной точки статус Label станет пустым, а у стрелки вызовется метод draw().

**3.3 Верификация ПС**

После того как код программы был написан, необходимо произвести функциональное тестирование программного продукта «drawau».

Test Case – это тестовый артефакт, суть которого заключается в выполнении некоторого количества действий и/или условий, необходимых для проверки определенной функциональности разрабатываемой программной системы. Структура данного артефакта заключается в «троице»: Выполняемое действие (Action) – Ожидаемый результат (Expected result) – Фактический результат (Test result).

|  |  |  |
| --- | --- | --- |
| **Название:** | Тест сохранения диаграммы в png | |
| **Функция:** | Сохранение диаграммы | |
| **Действие** | **Ожидаемый результат** | **Результат теста:**  **• пройден**  **• провален**  **• заблокирован** |
| **Предусловие** |  |  |
| Разместить фигуры на область | Доступно сохранение диаграммы | пройден |
| **Шаги теста:** |  |  |
| Нажать на кнопку сохранения | Выводит окно с выбором директории | пройден |
| Выбрать формат сохранения «.png» | Добавляет в конец названия файла «.png» | пройден |
| Нажать кнопку готово | Сохраняет файл в формате «.png», который доступен к просмотру | пройден |
| **Постусловие:** |  |  |
| Открыть изображение средствами просмотра изображения | Изображение без ошибок просматривается | пройден |

|  |  |  |
| --- | --- | --- |
| **Название:** | Тест сохранения диаграммы в drawau | |
| **Функция:** | Сохранение диаграммы | |
| **Действие** | **Ожидаемый результат** | **Результат теста:**  **• пройден**  **• провален**  **• заблокирован** |
| **Предусловие** |  |  |
| Разместить фигуры на область | Доступно сохранение диаграммы | пройден |
| **Шаги теста:** |  |  |
| Нажать на кнопку сохранения | Выводит окно с выбором директории | пройден |
| Выбрать формат сохранения «.drawau» | Добавляет в конец названия файла «.drawau» | пройден |
| Нажать кнопку готово | Сохраняет файл в формате «.drawau» | пройден |
| **Постусловие:** |  |  |
| Открыть файл в этой программе | В рабочей области разместились ожидаемые диаграммы | пройден |

|  |  |  |
| --- | --- | --- |
| **Название:** | Тест автоматического создания диаграммы | |
| **Функция:** | Создание диаграммы из папки | |
| **Действие** | **Ожидаемый результат** | **Результат теста:**  **• пройден**  **• провален**  **• заблокирован** |
| **Предусловие** |  |  |
| Запустить приложение | Доступна функция «магия» | пройден |
| **Шаги теста:** |  |  |
| Нажать на кнопку «магия» | Выводит окно с выбором директории | пройден |
| Нажать кнопку готово | В рабочей области разместились ожидаемые элементы и связи между ними | пройден |
| **Постусловие:** |  |  |
| Сохранить диаграмму | Диаграмма сохранена, открывается и просматривается | пройден |

**Заключение**

В ходе курсового проекта была реализована программа, позволяющая создавать диаграммы классов, как в ручную, так и автоматически. Интерфейс программы был разработан интуитивно понятным и не сложным, все функции программа выполняет без сложностей.

Данная работа закрепила знания, полученные в процессе изучения дисциплины «Технология программирования», а также повысила навыки самостоятельно разработки приложений.

**Список использованных источников**

1. Дронов, В. JavaScript в Web-дизайне / В. Дронов. - М.: СПб: БХВ, 2001. - 880 c.

2. Монкур Освой самостоятельно JavaScript за 24 часа / Монкур, Майкл. - М.: Вильямс, 2001. - 320 c.

3. Мугал, Халид А. Java. Руководство по подготовке к сдаче сертификационного экзамена CX-310-035 (+ CD-ROM) / Мугал, Халид А., Расмуссен, Рольф В.. - М.: КУДИЦ-Образ, 2006. - 688 c.

4. Негрино JavaScript для начинающих / Негрино, Том. - М.: НТ Пресс, 2007. - 544 c.

5. Ноултон, П. Java 2 в подлиннике / П. Ноултон, Г. Шилдт. - М.: СПб: BHV, 2001. - 400 c.

6. Хабр – Сообщество IT-специалистов [Электронный ресурс] / https://habr.com/ru/post/554388/ – Моделирование данных: зачем нужно и как реализовать [1.05.2022]

7. pro-prof – Блог программиста [Электронный ресурс] / https://proprof.com/archives/3212 – Диаграммы классов UML [1.05.2022]

8. OSP – Гид по технологиям [Электронный ресурс] / https://www.osp.ru/os/2019/02/13054964 – Верификация и валидация ПО: технологии и инструменты [1.05.2022]