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):

Формирование требований к программному продукту (4 страницы); Проектирование программного средства (3 страницы); Конструирование программного средства (9 страниц); 6 рисунок, 3 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 5](#_Toc111048643)

[1. Формирование требований к программному продукту и проекту 6](#_Toc111048644)

[1.1 Описание предметной области программы 6](#_Toc111048645)

[1.2 Анализ существующих решений 7](#_Toc111048646)

[1.3 Моделирование вариантов использования 8](#_Toc111048647)

[2. Проектирование программного средства 10](#_Toc111048648)

[2.1 Архитектура приложения 10](#_Toc111048649)

[2.2 Проектирование интерфейса пользователя 11](#_Toc111048650)

[3. Конструирование программного средства 13](#_Toc111048651)

[3.1 Описание классов приложения 13](#_Toc111048652)

[3.2 Моделирование взаимодействия пользователя с программой 17](#_Toc111048653)

[3.3 Верификация ПС 19](#_Toc111048654)

[Заключение 22](#_Toc111048655)

[Список использованных источников 23](#_Toc111048656)

[Приложение 1. Техническое задание 24](#_Toc111048657)

[Приложение 2. Диаграмма вариантов использования (К разделу 4.1) 28](#_Toc111048658)

[Приложение 3. Тестовые наборы для проверки работы программы с файлами (к разделу 8) 29](#_Toc111048659)

[Приложение 4. Листинг программного модуля 31](#_Toc111048660)

# **ВВЕДЕНИЕ**

При разработки приложения требуется составлять разного рода диаграммы и схемы. Таким образом, были придуманы различные средства автоматизировать процесс создания диаграмм. Были придуманы настольные 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**

## **1.1 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.

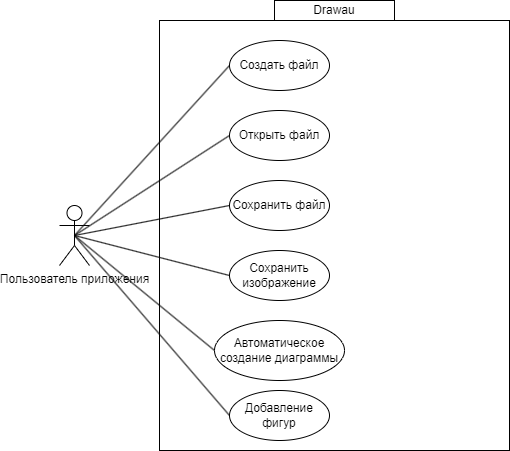


Рисунок 1 — Диаграмма вариантов использования

# **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.

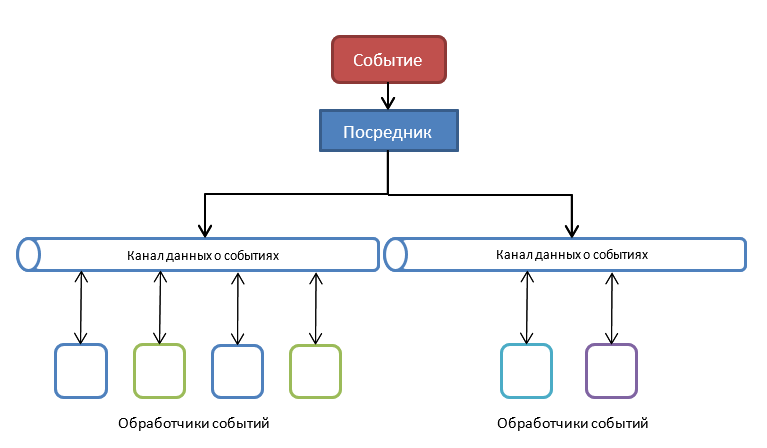


Рисунок 2 — Событийно-ориентированная структура

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.

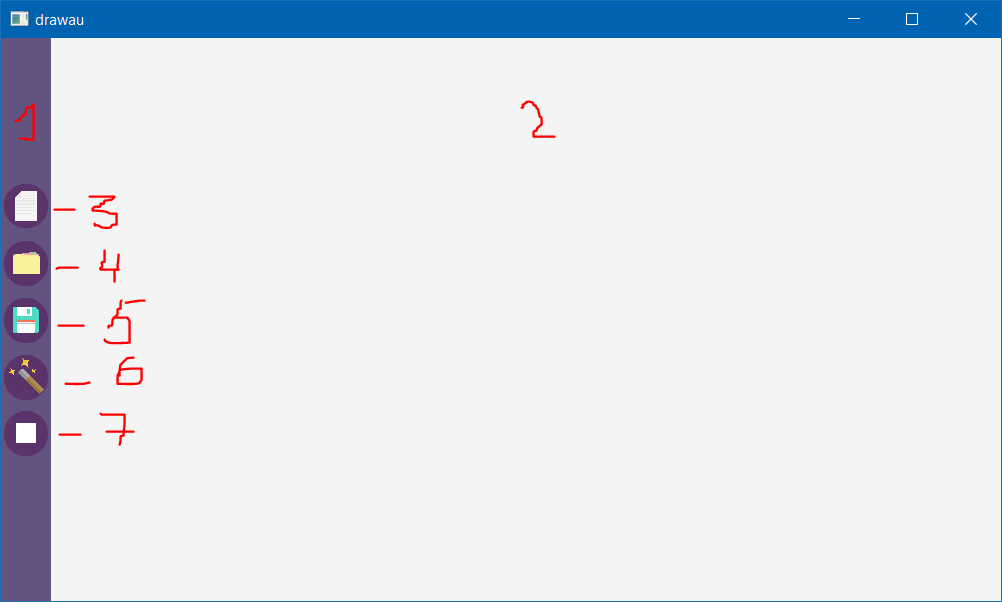


Рисунок 3 — Главное окно программы

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

На следующем рисунке показана диаграмма классов

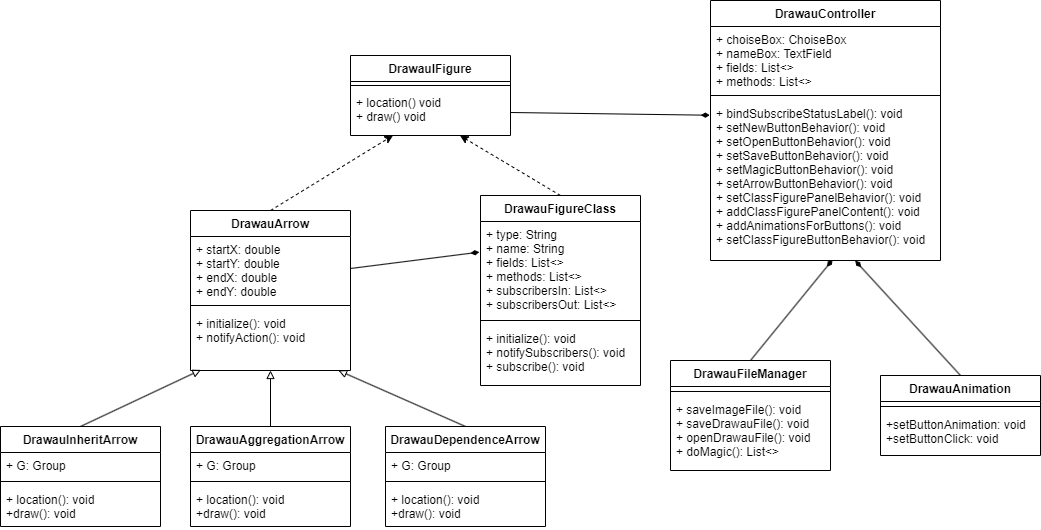


Рисунок 4 — Диаграмма классов

## **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.

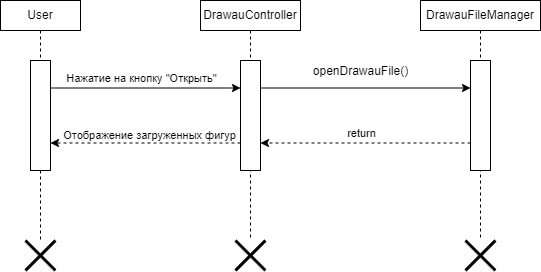


Рисунок 5 — Диаграмма открытия файла

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.

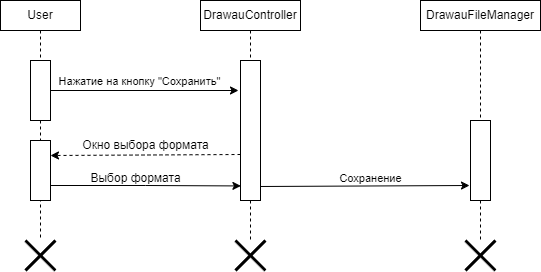


Рисунок 6 — Диаграмма сохранения файла

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]

# **Приложение 1. Техническое задание**

**1 Введение**

Техническое задание распространяется на разработку программного продукта «drawau» для облегчения создания диаграмм классов, следовательно, для некоторого облегчения разработки проекта.

**2 Основание для разработки**

Программный продукт разрабатывается в соответствии с рабочей программой по дисциплине «Технология программирования» направления подготовки «Информационные сети и технологии» Воронежского государственного технического университета от 31 августа 2021 в ходе курсового проектирования по теме «Разработка программного продукт "drawau"».

**3 Назначение разработки**

Программный продукт предназначен для облегчения разработки проектов.

**4 Требование к функциональным характеристикам программного продукта**

4.1 Функциональные требования

|  |  |
| --- | --- |
| **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 |

4.2 Требования к надежности

|  |  |
| --- | --- |
| **Идентификатор требования надежности** | **Описание** |
| N01 | Исключение ошибок при неправильном вводе пользователя |

4.3 Условие эксплуатации

|  |  |
| --- | --- |
| **Идентификатор требования производительности** | **Описание** |
| P01 | Загрузка файлов не превышает 0.3мс |

|  |  |
| --- | --- |
| **Идентификатор требования поддерживаемости** | **Описание** |
| S01 | Программа расширяема |
| S02 | Поддержка разных версий OC Windows |

4.4 Требование к составу и параметрам технических средств

Минимальные системные требования:

Процессор — Intel Pentium 2.4 ГГц

ОЗУ — 2 ГБ

4.5 Требование к информационной и программной совместимости

|  |  |
| --- | --- |
| **Идентификатор требования к интерфейсу** | **Описание** |
| I01 | Унарный дизайн элементов программы |
| I02 | Удобная интуитивно понятная навигация |

4.6 Требования к маркировке и упаковке

Не предъявляются

4.7 Требования к транспортировке и хранению

Не предъявляются

4.8 Специальные требования

Графический интерфейс состоит из двух блоков. Слева находится панель управления приложением, на ней расположены кнопки для открытия, сохранения и создания файлов, а также панель фигур, где располагаются четыре доступные фигуры. Справа расположен блог, отвечающий за рабочую область, куда добавляются все фигуры, которыми на этой области можно свободно манипулировать.

**5 Требования к программной документации**

Разрабатываемый программный продукт должен включать в себя:

Интерактивную справку по основным функциям программы;

Документ «Руководство пользователя»

**6 Технико-экономические показатели**

Трудоемкость разработки — 5 чел./ч

**7 Стадии и этапы разработки**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Согласование ТЗ | 3 день | Документ/Концепция |
| 2 | Анализ предметной области | 6 день | Документ ТЗ |
| 3 | Проектирование ПС | 2 неделя | Архитектурная спецификация |
| 4 | Кодирование и отладка | 4 неделя | Код программы |
| 5 | Тестирование | 6-7 неделя | Юнит тесты |
| 6 | Написание документации | 8 неделя | Руководство пользователя |
| 7 | Защита проекта | 9 неделя |  |

**8 Порядок контроля и приёмки**

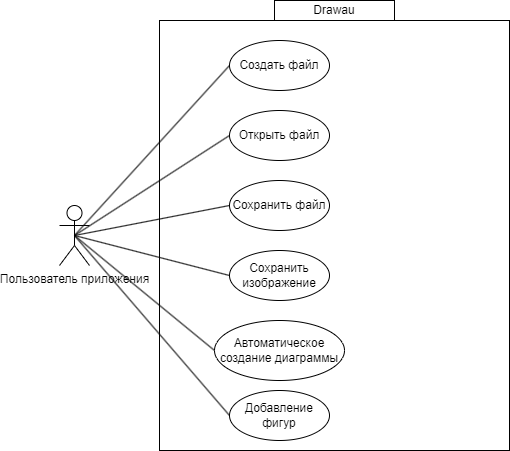
Испытание представленной модели и контроль качества ее работы провести на базе компьютерного класса. Во время испытаний проверить работу системы по тестовым сценариям (приложение 3):

• Сохранение файла в png

• Сохранение файла в drawau

• Автоматическое создание диаграммы

# **Приложение 2. Диаграмма вариантов использования (К разделу 4.1)**



# **Приложение 3. Тестовые наборы для проверки работы программы с файлами (к разделу 8)**

|  |  |  |
| --- | --- | --- |
| **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 |

# **Приложение 4. Листинг программного модуля**

Класс DrawauController

package com.example.drawau;

import javafx.beans.property.SimpleStringProperty;

import javafx.beans.property.StringProperty;

import javafx.collections.FXCollections;

import javafx.geometry.Pos;

import javafx.scene.control.\*;

import javafx.scene.layout.FlowPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Pane;

import javafx.scene.layout.VBox;

import javafx.stage.FileChooser;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.atomic.AtomicReference;

public class DrawauController {

public Button figuresButton;

public Button magicButton;

public Button saveButton;

public Button openButton;

public Button newButton;

public Button classFigure;

public Button inheritArrow;

public Button dependenceArrow;

public Button aggregationArrow;

public FlowPane figuresPanel;

public Pane workSpace;

public VBox classFigurePanel;

public HBox mainControl;

public static final double SIDE\_PANEL\_WIDTH = 40;

public ChoiceBox<String> choiceBox = new ChoiceBox<>(FXCollections.observableArrayList(

DrawauFigureClass.TYPE\_INTERFACE,

DrawauFigureClass.TYPE\_SIMPLE,

DrawauFigureClass.TYPE\_ABSTRACT));

public TextField nameBox = new TextField();

public List<DrawauFigureClass.FieldProperty> fields = new ArrayList<>();

public List<DrawauFigureClass.FieldProperty> methods = new ArrayList<>();

public Label subscribeArrowStatus;

List<DrawauFigureClass> figures = new ArrayList<>();

public static List<DrawauArrow> arrows = new ArrayList<>();

public static String ACTION = "null";

public static StringProperty subscribeStatusText = new SimpleStringProperty();

public void initialize() {

bindSubscribeStatusLabel();

setNewButtonBehavior();

setOpenButtonBehavior();

setSaveButtonBehavior();

setMagicButtonBehavior();

setArrowButtonBehavior();

setClassFigurePanelBehavior();

addClassFigurePanelContent(classFigurePanel);

addAnimationsForButtons();

setClassFigureButtonBehavior();

}

public void bindSubscribeStatusLabel() {

subscribeStatusText.bindBidirectional(subscribeArrowStatus.textProperty());

}

public void setNewButtonBehavior() {

newButton.setOnAction(action -> {

workSpace.getChildren().clear();

workSpace.getChildren().addAll(

classFigurePanel,

figuresPanel

);

});

}

public void setOpenButtonBehavior() {

openButton.setOnAction(action -> {

FileChooser fileChooser = new FileChooser();

fileChooser.getExtensionFilters().add(

new FileChooser.ExtensionFilter("drawau files (\*.drawau)", "\*.drawau"));

File file = fileChooser.showOpenDialog(null);

if (file != null)

DrawauFileManager.openDrawauFile(figures, arrows, file, workSpace);

});

}

public void setSaveButtonBehavior() {

saveButton.setOnAction(action -> {

FileChooser fileChooser = new FileChooser();

fileChooser.getExtensionFilters().addAll(

new FileChooser.ExtensionFilter("png files (\*.png)", "\*.png"),

new FileChooser.ExtensionFilter("drawau files (\*.drawau)", "\*.drawau"));

File file = fileChooser.showSaveDialog(null);

if (file != null) {

if (file.getName().endsWith(".png"))

DrawauFileManager.saveImageFile(workSpace, file);

else if (file.getName().endsWith(".drawau"))

DrawauFileManager.saveDrawauFile(figures, arrows, file);

}

});

}

public void setMagicButtonBehavior() {

magicButton.setOnAction(action -> {

try {

figures = DrawauFileManager.doMagic(workSpace, false);

} catch (IOException e) {

e.printStackTrace();

}

});

}

public void setArrowButtonBehavior() {

inheritArrow.setOnAction(action -> {

DrawauInheritArrow ar = new DrawauInheritArrow();

ar.locate(workSpace);

DrawauController.arrows.add(ar);

ACTION = DrawauFigureClass.SUBSCRIBE\_OUT;

subscribeStatusText.setValue("Отметьте начальную точку");

});

dependenceArrow.setOnAction(action -> {

DrawauDependenceArrow ar = new DrawauDependenceArrow();

ar.locate(workSpace);

DrawauController.arrows.add(ar);

ACTION = DrawauFigureClass.SUBSCRIBE\_OUT;

subscribeStatusText.setValue("Отметьте начальную точку");

});

aggregationArrow.setOnAction(action -> {

DrawauAggregationArrow ar = new DrawauAggregationArrow();

ar.locate(workSpace);

DrawauController.arrows.add(ar);

ACTION = DrawauFigureClass.SUBSCRIBE\_OUT;

subscribeStatusText.setValue("Отметьте начальную точку");

});

}

public void setClassFigurePanelBehavior() {

AtomicReference<Double> pressContainerX = new AtomicReference<>((double) 0);

AtomicReference<Double> pressContainerY = new AtomicReference<>((double) 0);

classFigurePanel.onMousePressedProperty().set(action -> {

pressContainerX.set(action.getX());

pressContainerY.set(action.getY());

});

classFigurePanel.setOnMouseDragged(action -> {

classFigurePanel.setLayoutX(action.getSceneX() - pressContainerX.get() - DrawauController.SIDE\_PANEL\_WIDTH);

classFigurePanel.setLayoutY(action.getSceneY() - pressContainerY.get());

});

}

public void addAnimationsForButtons() {

DrawauAnimation.setButtonWidthAnimation(

newButton,

openButton,

saveButton,

magicButton,

figuresButton

);

DrawauAnimation.setClickFiguresButtonAnimation(figuresButton, figuresPanel);

}

public void setClassFigureButtonBehavior() {

classFigure.setOnAction(action -> {

classFigurePanel.setVisible(true);

});

}

private void addClassFigurePanelContent(VBox classFigurePanel) {

classFigurePanel.toFront();

HBox head = new HBox();

head.setAlignment(Pos.CENTER);

head.setStyle(DrawauFigureClass.setMargin("10 0 0 0"));

nameBox.setStyle(DrawauFigureClass.setMargin("0 0 0 10"));

nameBox.setPrefSize(105, 25);

nameBox.setPromptText("Имя класса");

nameBox.setText("");

choiceBox.setPrefWidth(95);

head.getChildren().add(choiceBox);

head.getChildren().add(nameBox);

classFigurePanel.getChildren().add(head);

VBox verticalFieldBox = new VBox();

verticalFieldBox.setAlignment(Pos.TOP\_CENTER);

VBox verticalMethodBox = new VBox();

verticalMethodBox.setAlignment(Pos.TOP\_CENTER);

Button addFieldBox = new Button("Добавить поля");

Button addMethodBox = new Button("Добавить методы");

addFieldBox.setStyle(DrawauFigureClass.setMargin("5 0 5 0"));

addMethodBox.setStyle(DrawauFigureClass.setMargin("5 0 5 0"));

addFieldBox.setMinSize(150, 40);

addFieldBox.setMaxSize(150, 40);

addFieldBox.setPrefSize(150, 40);

addMethodBox.setMinSize(150, 40);

addMethodBox.setMaxSize(150, 40);

addMethodBox.setPrefSize(150, 40);

addFieldBox.setOnAction(action -> {

verticalFieldBox.getChildren().add(addFieldBox(fields));

});

addMethodBox.setOnAction(action -> {

verticalMethodBox.getChildren().add(addFieldBox(methods));

});

classFigurePanel.getChildren().add(verticalFieldBox);

classFigurePanel.getChildren().add(addFieldBox);

classFigurePanel.getChildren().add(verticalMethodBox);

classFigurePanel.getChildren().add(addMethodBox);

verticalFieldBox.getChildren().add(addFieldBox(fields));

verticalMethodBox.getChildren().add(addFieldBox(methods));

Button acceptButton = new Button("Готово");

acceptButton.setPrefSize(150, 40);

acceptButton.setMinSize(150, 40);

acceptButton.setMaxSize(150, 40);

acceptButton.setStyle(DrawauFigureClass.setMargin("5 0 5 0"));

acceptButton.setOnAction(action -> {

DrawauFigureClass f1 =

new DrawauFigureClassBuilder()

.addName(nameBox.getText())

.addType(choiceBox.getValue())

.addFields(fields)

.addMethods(methods)

.build();

f1.locate(workSpace);

f1.draw();

figures.add(f1);

classFigurePanel.setVisible(false);

classFigurePanel.getChildren().clear();

fields.clear();

methods.clear();

addClassFigurePanelContent(classFigurePanel);

});

classFigurePanel.getChildren().add(acceptButton);

}

private HBox addFieldBox(List<DrawauFigureClass.FieldProperty> fields) {

HBox fieldBox = new HBox();

fieldBox.setAlignment(Pos.CENTER);

fieldBox.setStyle(DrawauFigureClass.setMargin("10 0 0 0"));

TextField fieldName = new TextField();

TextField fieldType = new TextField();

DrawauFigureClass.FieldProperty field = new DrawauFigureClass.FieldProperty("", "");

fieldName.setStyle(DrawauFigureClass.setMargin("0 5 0 0"));

fieldType.setStyle(DrawauFigureClass.setMargin("0 0 0 5"));

fieldName.setPrefSize(100, 25);

fieldName.setMinSize(100, 25);

fieldType.setPrefSize(100, 25);

fieldType.setMinSize(100, 25);

fieldName.setPromptText("Имя");

fieldType.setPromptText("Тип");

fieldName.textProperty().addListener(change -> {

field.setName(fieldName.getText());

});

fieldType.textProperty().addListener(change -> {

field.setType(fieldType.getText());

});

fields.add(field);

fieldBox.getChildren().add(fieldName);

fieldBox.getChildren().add(fieldType);

return fieldBox;

}

}

Класс DrawaFigureClass

package com.example.drawau;

import javafx.geometry.Pos;

import javafx.scene.control.Alert;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Pane;

import javafx.scene.layout.VBox;

import javafx.scene.text.Text;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

public class DrawauFigureClass implements DrawauIFigure {

public static class FieldProperty implements Serializable {

public FieldProperty(String vname, String vtype) {

name = vname;

type = vtype;

}

private String name;

private String type;

public void setName(String value) {name = value;}

public String getName() {return name;}

public void setType(String value) {type = value;}

public String getType() {return type;}

}

public static final String TYPE\_ABSTRACT = "abstract class";

public static final String TYPE\_SIMPLE = "class";

public static final String TYPE\_INTERFACE = "interface";

public static final String SUBSCRIBE\_IN = "sub in";

public static final String SUBSCRIBE\_OUT = "sub out";

private String parent = "";

private String type;

private String name;

private List<FieldProperty> fields = new ArrayList<>();

private List<FieldProperty> methods = new ArrayList<>();

private List<DrawauArrow> subscribersIn = new ArrayList<>();

private List<DrawauArrow> subscribersOut = new ArrayList<>();

private double layoutX;

private double layoutY;

transient private VBox container = new VBox();

double pressContainerX = 0;

double pressContainerY = 0;

public DrawauFigureClass(DrawauFigureClassBuilder builder) {

container.alignmentProperty().set(Pos.TOP\_CENTER);

container.setPrefSize(builder.width, builder.height);

container.setStyle(builder.style);

container.setLayoutX(builder.layoutX);

container.setLayoutY(builder.layoutY);

addDragAction();

addClickAction();

type = builder.type;

name = builder.name;

fields = builder.fields;

methods = builder.methods;

layoutX = builder.layoutX;

layoutY = builder.layoutY;

}

public List<DrawauArrow> getSubscribersIn() { return subscribersIn; }

public List<DrawauArrow> getSubscribersOut() { return subscribersOut; }

public DrawauArrow lastSubscribeOut() {

if (subscribersOut.size() > 0)

return subscribersOut.get(subscribersOut.size()-1);

else return new DrawauInheritArrow();

}

public DrawauArrow lastSubscribeIn() {

if (subscribersIn.size() > 0)

return subscribersIn.get(subscribersIn.size()-1);

else return new DrawauInheritArrow();

}

public void setParent(String value) { parent = value; }

public String getParent() { return parent; }

public void setType(String value) { type = value; }

public String getType() { return type; }

public void setName(String value) { name = value; }

public String getName() { return name; }

public void setFieldValue(FieldProperty value) {

fields.add(value);

}

public void setMethodValue(FieldProperty value) {

methods.add(value);

}

public List<FieldProperty> getFields() { return fields; }

public List<FieldProperty> getMethods() { return methods; }

public void setSize(double hvalue, double vvalue) { container.setPrefSize(hvalue, vvalue); }

public void setStyle(String value) { container.setStyle(value); }

//public double getWidth() { return container.getWidth()>0?container.getWidth():container.getPrefWidth(); }

public double getWidth() { return container.getPrefWidth(); }

public double getHeight() { return container.getHeight()>0?container.getHeight():container.getPrefHeight(); }

public void setLayoutX(double value) {

layoutX = value;

container.setLayoutX(layoutX);

}

public void setLayoutY(double value) {

layoutY = value;

container.setLayoutY(layoutY);

}

public double getLayoutX() { return layoutX; }

public double getLayoutY() { return layoutY; }

public static String setMargin(String marginValue) {

return "-fx-padding: " + marginValue +

"; -fx-border-insets: " + marginValue +

"; -fx-background-insets: " + marginValue + ";";

}

public void initialize() {

container = new VBox();

container.alignmentProperty().set(Pos.TOP\_CENTER);

container.setPrefSize(new DrawauFigureClassBuilder().width, new DrawauFigureClassBuilder().height);

container.setStyle(new DrawauFigureClassBuilder().style);

container.setLayoutX(layoutX);

container.setLayoutY(layoutY);

addDragAction();

addClickAction();

}

private void addDragAction() {

container.onMousePressedProperty().set(action -> {

pressContainerX = action.getX();

pressContainerY = action.getY();

});

container.setOnMouseDragged(action -> {

layoutX = action.getSceneX() - pressContainerX - DrawauController.SIDE\_PANEL\_WIDTH;

container.setLayoutX(layoutX);

layoutY = action.getSceneY() - pressContainerY;

container.setLayoutY(layoutY);

notifySubscribers();

});

}

public void notifySubscribers() {

for (DrawauArrow arrow : subscribersIn)

arrow.notifyAction(arrow.getStartX(), arrow.getStartY(),

container.getLayoutX() + arrow.getEndSelfPosX(), container.getLayoutY());

for (DrawauArrow arrow : subscribersOut)

arrow.notifyAction(container.getLayoutX() + arrow.getStartSelfPosX(), container.getLayoutY(),

arrow.getEndX(), arrow.getEndY());

}

private void addClickAction() {

container.onMouseClickedProperty().set(action -> {

if (DrawauController.arrows.stream().count() != 0 && DrawauController.ACTION != "null") {

subscribe(DrawauController.arrows.get((int)DrawauController.arrows.stream().count()-1),

action.getX(), DrawauController.ACTION);

}

});

}

public void subscribe(DrawauArrow sub, double selfX, String mode) {

switch (mode) {

case SUBSCRIBE\_IN -> {

sub.setEndX(container.getLayoutX() + selfX);

sub.setEndSelfPosX(selfX);

if (container.getHeight() > 1)

sub.setEndHeightContainer(container.getHeight());

else sub.setEndHeightContainer(container.getPrefHeight());

sub.setEndY(container.getLayoutY());

subscribersIn.add(sub);

sub.draw();

DrawauController.ACTION = "null";

DrawauController.subscribeStatusText.setValue("");

}

case SUBSCRIBE\_OUT -> {

sub.setStartX(container.getLayoutX() + selfX);

sub.setStartSelfPosX(selfX);

if (container.getHeight() > 1)

sub.setStartHeightContainer(container.getHeight());

else sub.setStartHeightContainer(container.getPrefHeight());

sub.setStartY(container.getLayoutY());

subscribersOut.add(sub);

DrawauController.ACTION = SUBSCRIBE\_IN;

DrawauController.subscribeStatusText.setValue("Отметьте конечную точку.");

}

}

}

@Override

public void locate(Pane location) {

location.getChildren().add(container);

}

@Override

public void draw() {

container.getChildren().clear();

//Header-part

HBox header\_part = new HBox();

header\_part.alignmentProperty().set(Pos.CENTER);

header\_part.setStyle(setMargin("0 10 10 10"));

Text header\_part\_typeClass = new Text(type + " ");

Text header\_part\_nameClass = new Text(name);

header\_part\_nameClass.getStyleClass().add("text-style");

header\_part\_typeClass.getStyleClass().add("text-style");

header\_part.getChildren().add(header\_part\_typeClass);

header\_part.getChildren().add(header\_part\_nameClass);

container.getChildren().add(header\_part);

//Middle-part

for (FieldProperty field : fields) {

HBox middle\_part = new HBox();

middle\_part.setAlignment(Pos.BASELINE\_CENTER);

Text middle\_part\_fieldName = new Text("-" + field.getName());

Text middle\_part\_fieldType = new Text(" " + field.getType());

middle\_part\_fieldName.getStyleClass().add("text-style");

middle\_part\_fieldType.getStyleClass().add("text-style");

middle\_part.getChildren().add(middle\_part\_fieldName);

middle\_part.getChildren().add(middle\_part\_fieldType);

container.getChildren().add(middle\_part);

}

container.getChildren().get((int)container.getChildren().stream().count()-1).setStyle(setMargin("0 0 10 0"));

//Bottom-part

for (FieldProperty method : methods) {

HBox bottom\_part = new HBox();

bottom\_part.setAlignment(Pos.BASELINE\_CENTER);

Text bottom\_part\_methodName = new Text("+" + method.getName());

Text bottom\_part\_methodType = new Text(" " + method.getType());

bottom\_part\_methodName.getStyleClass().add("text-style");

bottom\_part\_methodType.getStyleClass().add("text-style");

bottom\_part.getChildren().add(bottom\_part\_methodName);

bottom\_part.getChildren().add(bottom\_part\_methodType);

container.getChildren().add(bottom\_part);

}

}

}