МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

автоматизованих систем обробки інформації та управління

**Пояснювальна записка**

до курсової роботи

з дисципліни

“Об’єктно-орієнтовне програмування 2 ”

на тему

"Редактор тексту та растрових зображень"

|  |  |  |
| --- | --- | --- |
| Керівник : Подрубайло О. О. |  | Виконав : Нечай Дмитро Олександрович |
| Допущений до захисту |  | студент гр. ІП-41,ФІОТ |
| І\_\_\_І \_\_\_\_\_\_\_\_ 201\_ \_\_\_\_\_\_  підпис |  | 2 курс  № IП-4117 |
| Захистив з оцінкою  \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  оцінка підпис  І\_\_\_І \_\_\_\_\_\_\_\_\_ 201\_ |  | Підпис: |

**Зміст**

[1. Огляд існуючих рішень 3](#_Toc450996492)

[2. Проектування архітектури системи 6](#_Toc450996493)

[3. Реалізація програмного комплексу 18](#_Toc450996494)

[4. Опис застосування системи 37](#_Toc450996495)

[Висновок 44](#_Toc450996496)

[Список літератури 45](#_Toc450996497)

[Додатки 46](#_Toc450996498)

[Код програми 46](#_Toc450996499)

# Огляд існуючих рішень

Текстовий редактор — комп'ютерна програма-застосунок, призначена для створення й зміни текстових файлів (вставки, видалення та копіювання тексту, заміни змісту, сортування рядків), а також їх перегляду на моніторі, виводу на друк, пошуку фрагментів тексту тощо.

Деякі текстові редактори забезпечують також розширену функціональність: підсвічування синтаксису, сортування рядків, шаблони, конвертацію кодування символів тощо. Така функціональність часто характерна для редакторів коду, призначених для написання сирцевого коду комп'ютерних програм.

Інші текстові редактори мають розширені функції форматування тексту, впровадження в нього графіків, формул, таблиць та об'єктів. Такі редактори часто називають текстовими процесорами й призначені вони для створення різного роду документів — від особистих листів до офіційних паперів. Класичні приклади — Microsoft Word, Libre Office Writer та WPS Office Writer.

Ще один клас програм цієї групи — текстові середовища — по суті, повноцінні робочі середовища, в яких можна вирішувати найрізноманітніші завдання: за допомогою надбудов вони дозволяють писати й читати листи, веб-канали, працювати в вікі й Вебі, вести щоденник, керувати списками адрес і завдань.

При проектуванні програмного забезпечення потрібно розуміти основні принципи проектування архітектури системи. Розробник повинен мати основні базові навички, такі як:

* Розуміння принципів ООП;
* Знанням патернів проектування;
* Знання технологій ООД, ООА;

Однією з перших проблем, які постають перед розробником є правильне складання вимог, які потребує замовник, та правильна декомпозиція системи. Розробник повинен зрозуміти, який функціонал повинен бути розроблений, та спроектувати прототип кінцевого продукту.

Давайте розглянемо основні завдання проектувальника при розробці редактора векторних зображень:

1. Визначення призначення продукту.

Текстовий редактор з підтримкою растрових зображень призначений для створення, редагування та збереження документів, що складаються з тексту (представленого послідовністю символів) та зображень.

1. Опис функціоналу системи.

При розробці текстового редактора з підтримкою растрових зображень потрібно чітко описати всі вимоги до продукту. Перевірку задоволення вимог потрібно перевіряти через певні проміжки часу, до того моменту як не буде задоволено всі вимоги.

1. Проектування функціональної моделі.

Після складання вимог до редактора потрібно спроектувати функціональну модель продукту. Вона випливає з вимог, виявлених на минулому кроці.

1. Визначення структур даних

На цьому кроці потрібно виділити певні абстракції в редакторі векторних зображень, а також які структури даних будуть застосовуватися.

1. Розробка продукту

На цьому найдовшому етапі відбувається розробка продукта, враховуючи вимоги та структури даних, які були виявлені та вибрані на попередніх кроках. Розробник повинен, застосовуючи принципи ООП та патернів проектування, розробити систему, яка буде покривати всі виявлені вимоги.

Текстові редактори проектуються по принципу «що бачиш, те й отримуєш» (WYSIWYG). Користувачу надається інтуїтивно зрозумілий інтерфейс для малювання графічних зображень. Системи з графічним інтерфейсом (GUI) найзручніше проектувати за патерном Model-View-Controller(MVC). Тобто, складається з трьох компонентів:

1. Моделі, в обов’язки якої входить зберігання даних і забезпечення інтерфейсу до них.
2. Вигляду, який відповідальних за представлення цих даних користувачеві.
3. Контролера, який керує компонентами системи, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни Моделі.

Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

MVC поділяє цю частину системи на три самостійні частини: введення даних, компонент обробки даних і виведення інформації. Модель, як вже було відмічено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент Модель не залежить від процесу введення або виведення даних. Компонент виводу Вигляд може мати декілька взаємопов'язаних областей, наприклад, різні панелі, такі як панель інструментів, меню, панель для малювання У функції Контролера входить моніторинг за подіями, що виникають в результаті дій користувача (зміна положення курсора миші, натиснення кнопки або малювання фігури).

Наведемо приклади деяких текстових редакторів:

1. **Notepad** – це текстовий редактор, розрблений корпорацією Microsoft, що входить до складу будь-якого ПК під управлінням ОС Windows. Підтримує такі функції:
   * Створення документу.
   * Редагування існуючих документів.
   * Збереження документу.
   * Вставка тексту.
   * Видалення тексту.
   * Виділення.
   * Повернення на попередню та наступну дії.
2. **Vim** – текстовий редактор, розроблений відкритою спільнотою розробників. Один з найросповсюдженіших тестопих редакторів в UNIX та UNIX-like система. Підтримує всі можливості, що надає Notepad та додає можливість розроблювати свої плагіни.
3. **Microsoft Word** –текстовий процесор, розрблений корпорацією Microsoft. Є повноцінним текстовим процесором, а тому надає можливості форматування тексту та роботу з зображеннями (їх вставка та редагування).

Висновок:

У цьому розділі було розглянуто існуючі рішення у проектуванні редактора векторної графіки та визначено основні етапи проектування обраної системи. На основі розглянутих питань було визначено основну функціональність редактора векторної графіки, який проектується:

* Можливість створення документу.
* Можливість редагування існуючого документу.
* Можливість збереження документу.
* Можливість вводити текст.
* Можливість зміювати шрифт тексту та робити його курсивним або напівжирним.
* Можливість додавати зображення.
* Можливість змінювати розмір зображення.
* Можливість видаляти елементи.
* Можливість виділення послідовності елементів.
* Можливість виконати операції : вставка, вирізання та копіювання послідовності елементів.
* Можливість повернення до попереднього та наступного стану системи.

# Проектування архітектури системи

Виконаємо проектування архітектури системи «Редактор тексту та растрових зображень», врахувавши принципи проектування системи, які описані в розділі «Огляд існуючих рішень».

Спершу визначимо призначення системи.

Редактор тексту та растрових зображень повинен надавати користувачу можливість редагування тексту, його шрифту та розміру. У ролі допоміжної функції було вибрано можливість додавати зобрадення та змінювати їх розмір.

Давайте виконаємо функціональну декомпозицію системи.

Для проведення декомпозиції системи було обрано метод визначення прецедентів функціональної моделі:

Таблиця 2.1 – Прецеденти функціональної моделі

|  |  |  |
| --- | --- | --- |
| № | Прецедент | Опис |
| 1 | Завантаження документу | Система повинна надавати користувачу можливість завантажувати текстові документи. |
| 2 | Збереження документу | Система повинна надавати користувачу можливість зберігати отриманий документ. |
| 3 | Відображення документу | Система повинна відображати документ, поточну каретку та виділення. |
| 4 | Редагування документу | Користувач має можливість додавати текст, змінювати шрифт, додавати зображення, змінювати їх розмір, видаляти будь-який елемент чи їх сукупність. |
| 5 | Ведення історії змін | Система зберігає історію всіх дій користувача під час сесії тп надає можливість перейти до попереднього або наступного стану. |

Наступним кроком є декомпозиція прецедентів.

Таблиця 2.2 – Декомпозовані прецеденти

|  |  |  |
| --- | --- | --- |
| № | Прецедент | Опис |
| **1** | **Завантаження документу** | |
| 1.1 | Перегляд файлової системи | Користувач має можливість вибрати будь-який файл на свій розсуд. |
| 1.1.1 | Конвертація текстових даних до об’єктів, що використовуються в системі | Конвертація текстових даних до об’єктів, що використовуються в системі. |
| 1.1.1.1 | Завантаження документу в редактор | Документ був коректно доданий в редактор. |
| **2** | **Завантаження документу** | |
| 2.1 | Перегляд файлової системи | Користувач має можливість вибрати будь-який файл на свій розсуд. |
| 2.1.1 | Конвертація об’єктів, що використовуються в системі до текстових даних | Конвертація об’єктів, що використовуються в системі, до текстових даних. |
| 2.1.1.1 | Завантаження документу в редактор | Документ був коректно збережений. |
| **3** | **Відображення документу** | |
| 3.1 | Відображення документу | Документ відображається на панелі редатора. |
| 3.2 | Відображення каретки | На панелі редактора відображається каретка в її потосному положенні. |
| 3.3 | Відображення області виділення | На панелі редактора відображається область виділення. |
| **4** | Редагування документу | |
| 4.1 | Додавання тексту | Користувач має можливість додати текст. |
| 4.2 | Зміна стилю тексту | Користувач має можливість змінити стиль тексту. |
| 4.3 | Додавання зображень | Користувач має можливість додати зображення. |
| 4.5 | Зміна розміру зображень | Користувач має можливість змінити розмір зображення. |
| 4.6 | Виділення елементів | Користувач має можливість виділити послідовність елементів. |
| 4.7 | Видалення елементів | Користувач має можливість видалити послідовність елементів. |
| 4.8 | Копіювання елементів | Користувач має можливість копіювати послідовність елементів. |
| 4.9 | Вставка елементів | Користувач має можливість вставити послідовність елементів. |
| **5** | **Ведення історії змін** | |
| 5.1 | Збереження змін | Система зберігає всі зміни, що відбулися впродовж сесії |
| 5.2 | Перехід до попереднього стану | Користувач має можливість перейти до попереднього стану натиснувши кнопку Undo або натиснувши комбінацію клавіш Ctrl+Z |
| 5.3 | Перехід до наступного стану | Користувач має можливість перейти до наступного стану натиснувши кнопку Redo або натиснувши комбінацію клавіш Ctrl+Y |

Створивши функціональну модель, можна провести декомпозицію системи на підсистеми та визначити зв’язки між ними. Всього в редакторі векторних зображень можна виділити 5 підсистем:

* Конвертації даних.
* Представлення документу в пам’яті.
* Відображення документу.
* Внесення змін.
* Створення елементів.

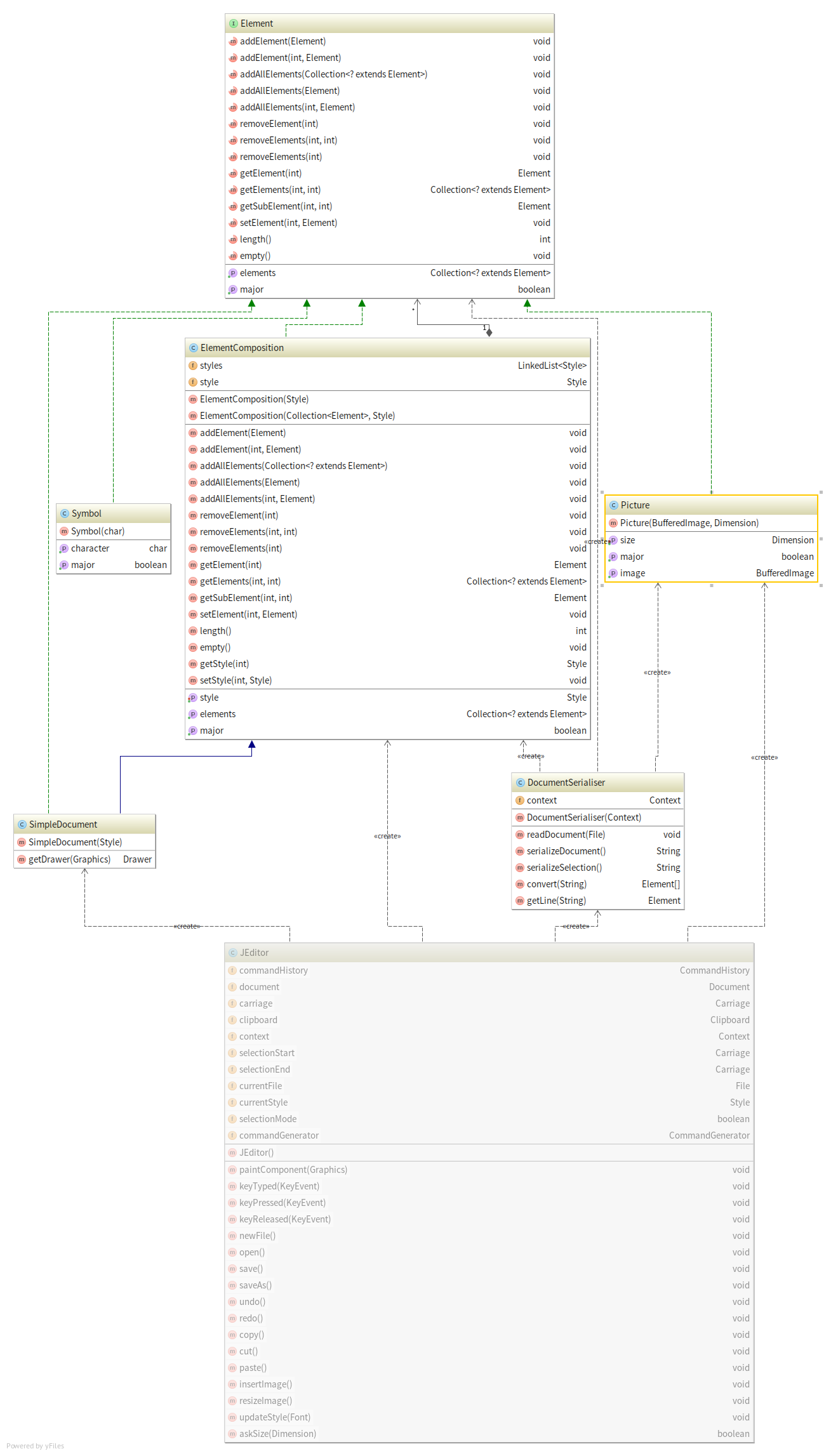
Детально розглянемо кожну підсистему.

1. Підсистема «Конвертації даних»

Редактор текстових документів повинен вміти конвертувати текстові дані в об’єкти, які використовуються для побудови структури елемента у пам’яті. Також редактор повинен надавати можливість ексорту та імпорту даних в буфер обміну даними. Обов’язковою складовою цієї підсистеми є можливість завантажувати та зберігати текстові файли.

Збереження та завантаження файлів може викликатися 3 шляхами : через пункт меню, кнопку на панелі інтсрументів або гарячу клавішу. Після переходу в режим завантаження або загрузеи користувачу пропонується вибрати файл з яким будуть проводитись маніпуляції. Далі дані серіалізуються або десеріалізуються та переміщаються у відповідне місце : оперативна пам’ять для завантаженого документу та файл в файловій системі для збереженого документу.

Копіювання в та з буферу обміну може викликатися 2 шляхами : кнопка на панелі інструментів або гаряча клавіша. При копіюванні даних в буфер обміну, виділена область передається в підсистему конвертації, де перетворюється в звичайний текстовий рядок, а потім поміщається в буфер. При копіюванні з буферу, підсистема конвертування інтерпритує дані, що лежать в буфері, та за дапомогою підсистеми створення елементів приводить зчитані дані до виду, який можна використовувати при побудові віртуального представлення документу.



Малюнок 2.1 – Підсистема «Конвертації даних»

1. Підсистема «Представлення документу в пам’яті»

Редактор документу повинен зберігати віртуальне представлення документу у пам’яті у зручному для редагування вигляді. За основу під час проектування було взято DOM модель. Документ – це композитний елемент, що включає в себе інші композитні елементи або примітивні елементи.

Було вирішено, що документ буде складатися з сукупності рядків, які є представлені у вигляді об’єктів, що є сукупностями примітивних елементів. Примітивні елементи бувають двох типів : символ та зображення. Також кожен примітивний елемент повинен мати свій стиль.

Оскільки зберігати екземпляр кожного примітивного класа разом з екземпляром його стилю є ресурсозатратно, то було вирішено відділити стиль елементу від його представлення у пам’яті та передати стиль на управління композиту елементів.

1. Підсистема «Відображення документу»

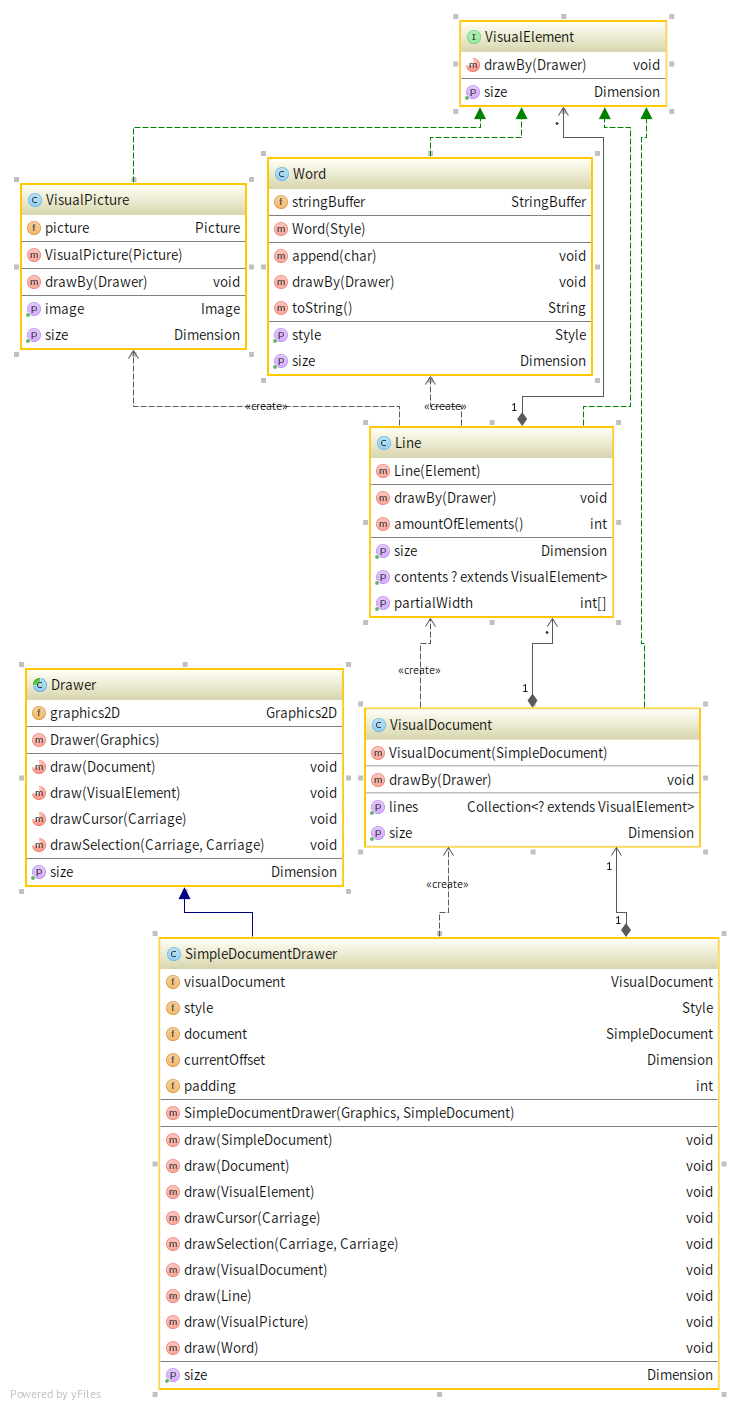
Формат в якому зберігається віртуальне представлення документу не дає змоги вивести документ на екран, адже він розроблений для зберігання даних.

Для представлення графічних даних було вирішено розробити аналогічну структуру. Існує документ, який складається з рядків а ті в свою чергу включають в себе лексеми та зображення.

Оскільки типів документів може бути декілька і кожен з них повинен по-різному відмальовуватися на форму, було вирішено що сутнічть, яка буду відповідати за відображення конкретного елементу буде створювати сам елемент. При створені ця сукупність перетворю документ у формат, зручний для відображення і по черзі проходить по всім елементам відображуючи їх на екран.



Малюнок 2.2 – Підсистема «Представлення документу в пам’яті»



Малюнок 2.3 – Підсистема «Відображення документу»

1. Підсистема «Внесення змін»

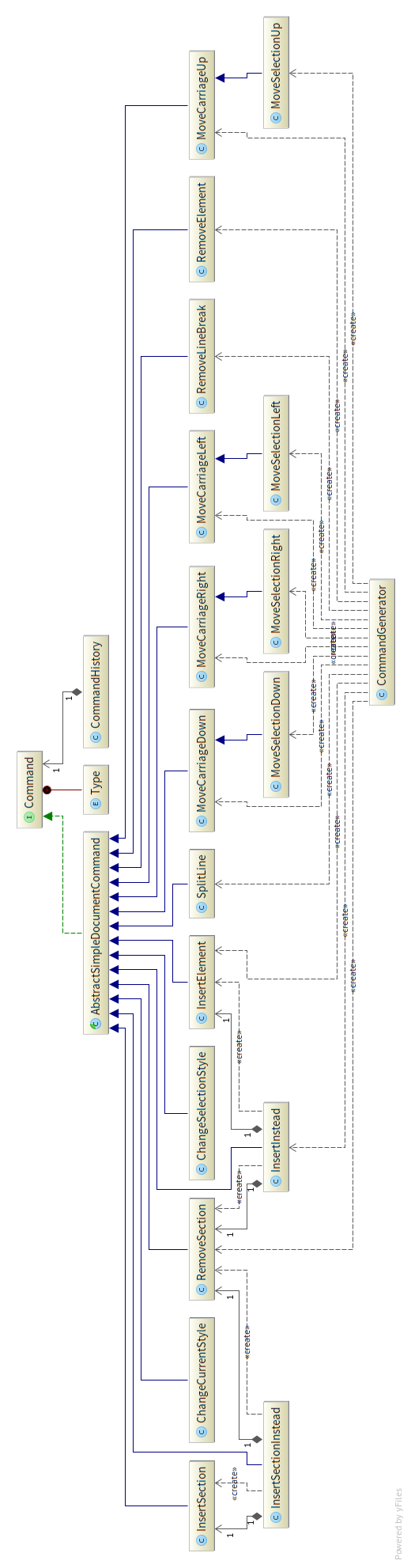
Система надає користувачу можливість відмінити останню виконану дію та перейти до наступної дії. Користувач може виконати ці дії натиснувши відповідно кнопки Undo/Redo на панелі інструментів або натиснувши комбінацію клавіш CTRL + Z/ CTRL + SHIFT + Z відповдно.

Реалізувати цей механізм можна двома способами, за допомогою паттерну Memento та паттерну Command. Проаналізувавши переваги та недоліки обох шаблонів було вирішено взяти за основу Command, адже в документі буде відбуватись багато маленьких за розміром змін, а тому використання Memento призвело б до використання занадто великої кількості оперативної пам'яті.

Кожна команда створюється як реакція на певну дію, тому було створенно клас, який інкапсулює в собі процес визначення необхідної дії та повертає необхідну команду.

Так як мета цієї підсистеми забезпечити простий для користувача перехід на попередній стан, то кожна комманда повинна вміти виконуватися та повертати документ до початкового стану.

Оскільки ми повертаємося завжди на попередню дію , то для збереження комманд було вирішено використовувати 2 стеки. Так як повертатися завжди лише на одну дію назад є незручно, було введено механізм типів команд та їх значущості. При переході до іншого стану виконується перехід до перошої значної зміни того ж типу.

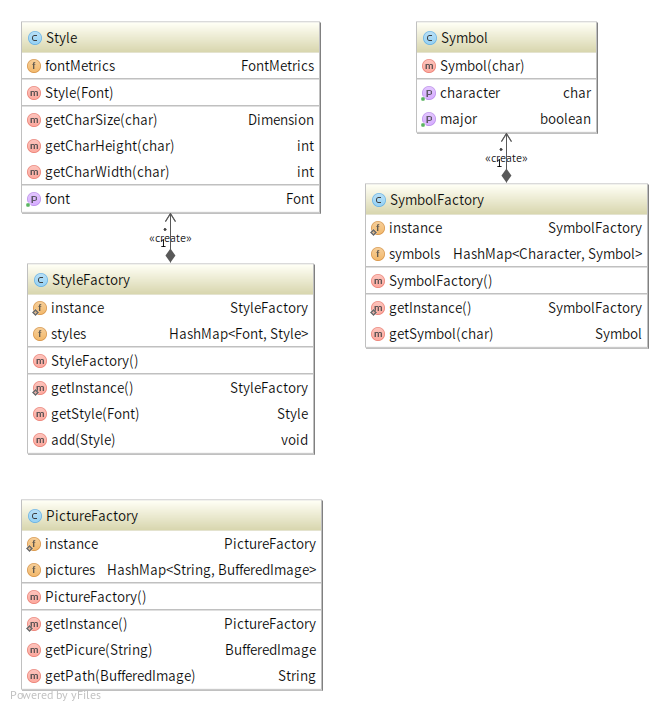


Малюнок 2.4 – Підсистема «Внесення змін»

1. Підсистема «Створення елементів»

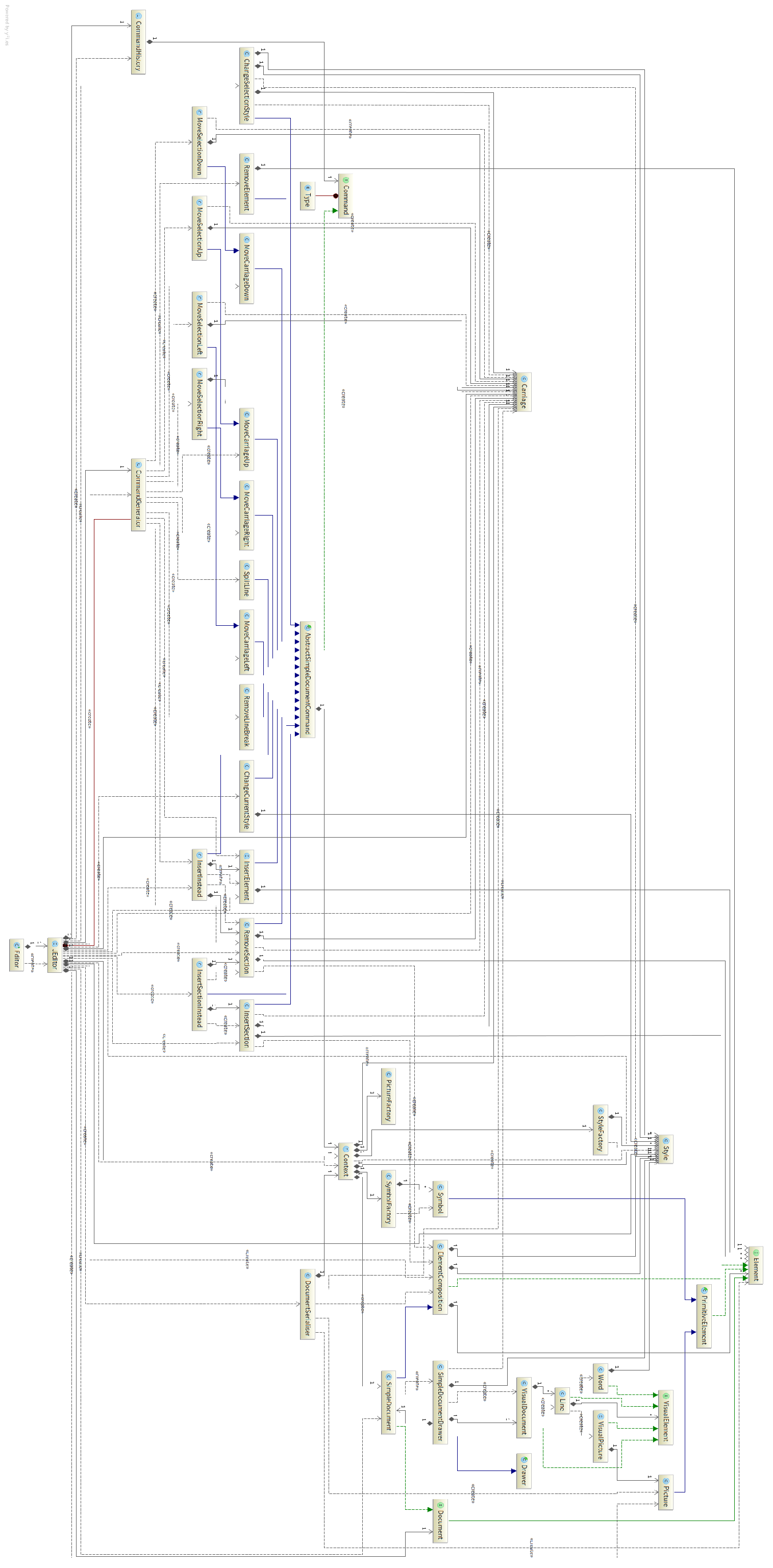
Оскільки документ може досягати доволі великих розмірів а певні його складові обов’язково будуть повторюватися багато разів, як от символ або стиль, та документ містить в собі елементи, що займають значну область пам’яті і дублюівння який є дуже неоптимальним, як от зображення, було вирішено створити «Пул» для зображень, символів та стилів.

В результаті кожне зображення, символ та стиль зберігаються лише один раз. Для кожного «Пулу» було створено свій клас, який інкапсулює в собі процес перевірки на існування певного об’єкту. Таким чином було знешкоджено лишні витрати пам’яті та прозорий інтерфейс для інших підсистем, вони просто відправляють запит на елемент, що буде відповідати заданим параметрам.



Малюнок 2.5 – Підсистема «Створення елементів»

Після розгляду всіх підсистем, можна побудувати загальну діаграму класів спроектованого редактора векторної графіки. На загальній діаграмі класів зображені всі класи системи та зв’язки між ними.



Малюнок 2.6 – Загальна діаграма класів

Висновок

Розроблена архітектура системи складається з таких підсистем:

* Конвертації даних.
* Представлення документу в пам’яті.
* Відображення документу.
* Внесення змін.
* Створення елементів.

При проектуванні були використані можливості ООП та патерни проектування. Були застосовані такі патерни:

1. Composite

Дозволив cтворити DOM-подібну структуру, що надає можливість доволі просто додавати нові типи композитних елементів.

1. Visitor

Цей патерн дозволяє виконати спільну дію над багатьма однотипними, але різними елементами. В даному випадку – відображення композитного документа.

1. Command

Дозволило задати команди для виконання Undo/Redo інкапсулюючи від користувача реалізацію.

1. Abstract Factory

Дозволив інкапсолювати процес створення необхідної команди.

1. Factory Method

Надав можливість додавати нові типи документів та класів, що відповідають за їх відображення без зміни коду форми редактора.

1. Flyweight

Зменшив кількість пам’яті необхідної для для роботи редактора.

1. Singleton

Гарантує що в межах нашої програми буде лише один «Пул» для кожного типу об’єктів.

# Реалізація програмного комплексу

В цьому розділі приведений детальний опис класів системи та розглянуто їхнє функціональне призначення.

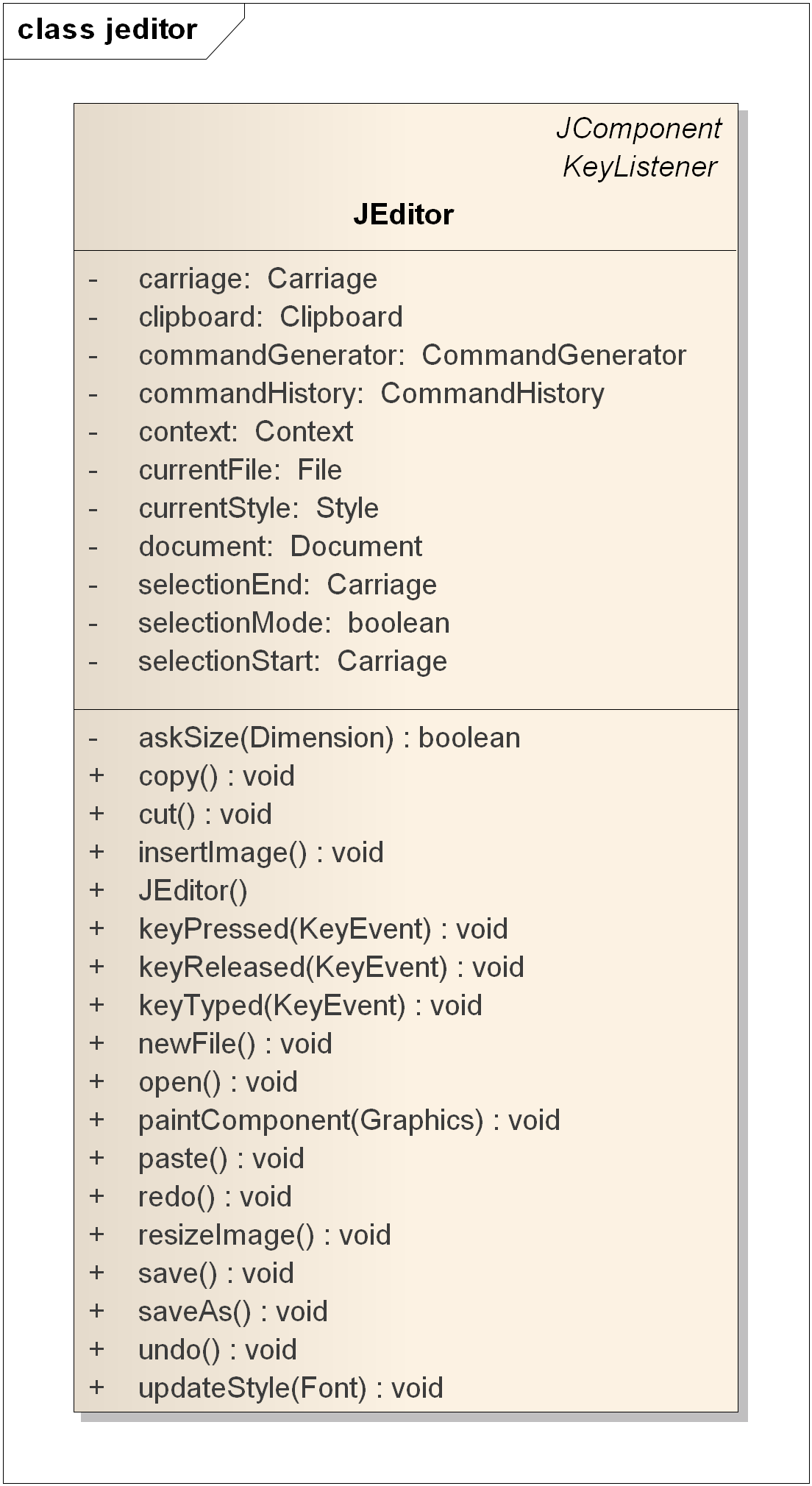
1. Опис графічного інтерфейсу
   1. Editor



Малюнок 3.1 – Клас Editor

Editor призначений для відображення шрафічного інтерфейсу користувача. Складається з меню, панелі інструментів та компоненти відображення документу. Взаємодіє лише з классом JEdit, якому подає сигнали про виконання певих дій.

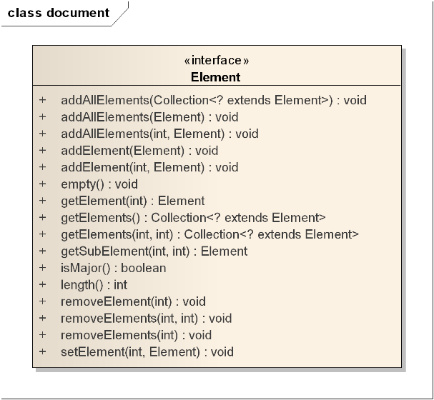
* 1. JEditor



Малюнок 3.2 – Клас JEditor

JEditor слугує для відображення документу та внесення в нього змін. Взаємодіє з класами: CommandHistory, для збереження історії змін, CommandGenerator, створює команди, Document, в якому зберігається віртуальне представлення документа, Drawer, для відображення документа, DocumentSerialiser, для ковертації даних.

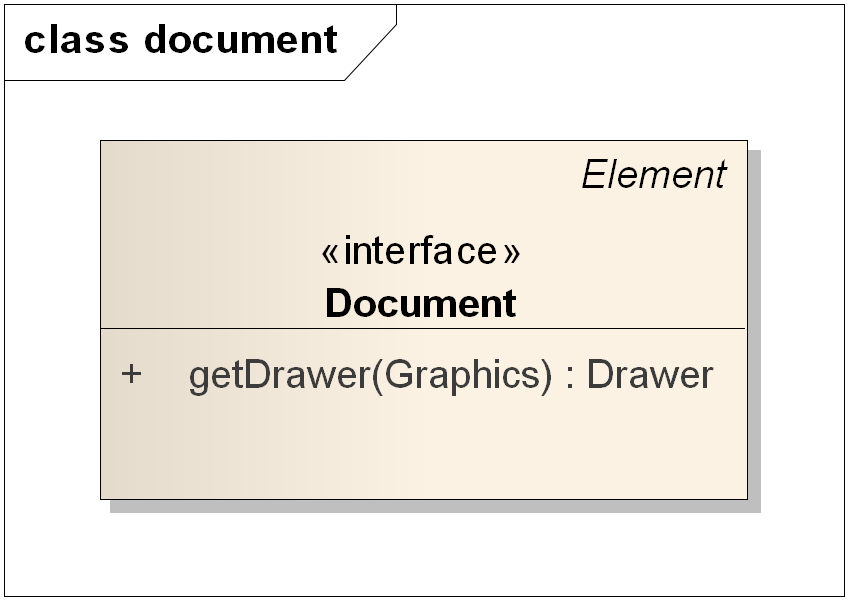
1. Опис підсистеми «Представлення документу в пам’яті»
   1. Element



Малюнок 3.3 – Інтерфейс Element

Інтерфейс для представлення елементу структури документа. Реалізуючи цей інтерфейс можна створювати нові елементи нового типу для документу

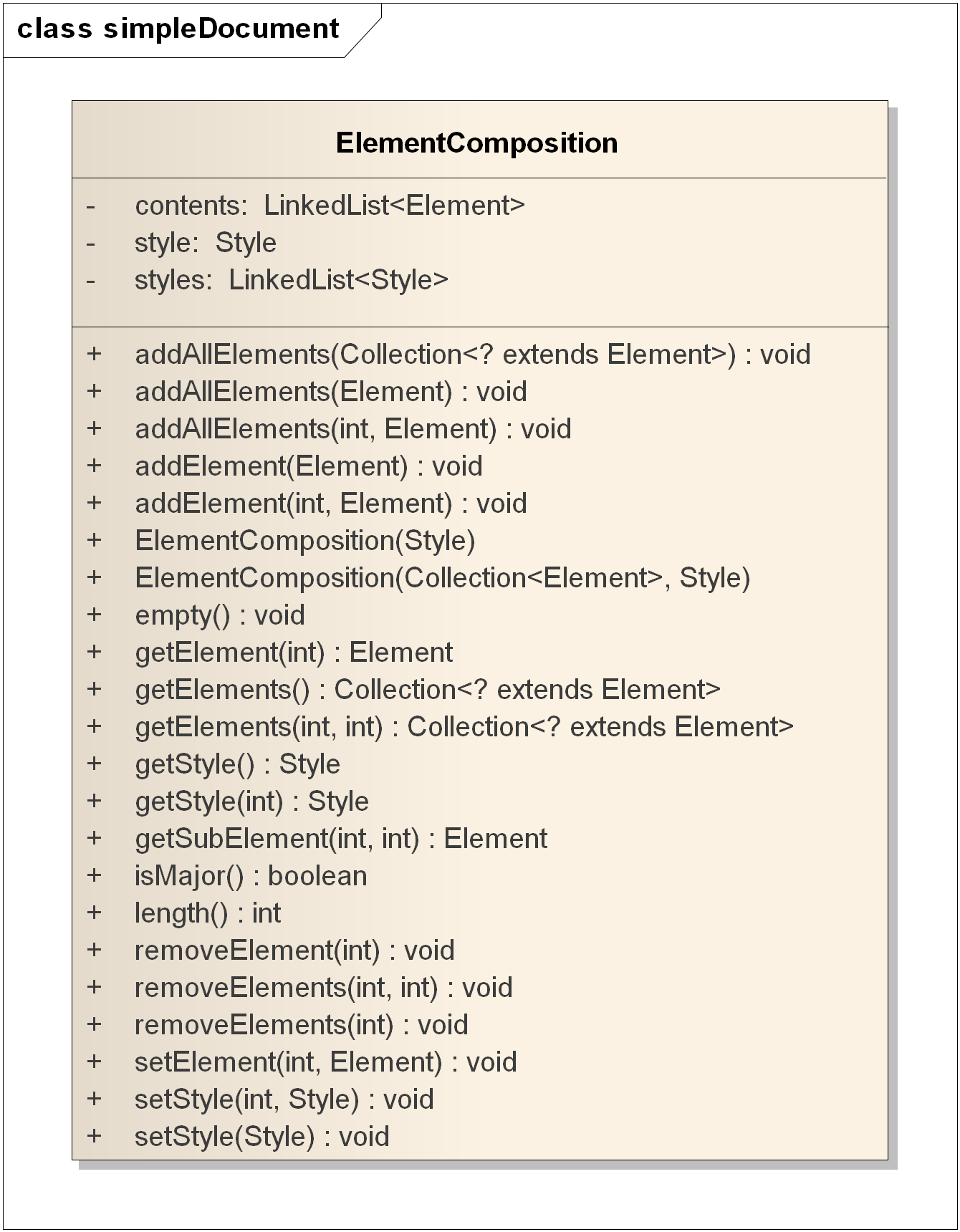
* 1. Document



Малюнок 3.4 – Інтерфейс Document

Інтерфейс для представлення DOM-like тестового документу. Реалізуючи цей інтерфейс можна створювати документи нових типів.

* 1. ElementComposition



Малюнок 3.5 – Абстрактний клас ElementComposition

Представляє собою композит елементів. Містить в собі спимок стилів для кожного елеиента. Розширюючи цей інтерфейс можна створювати нові типи композитних елементів.

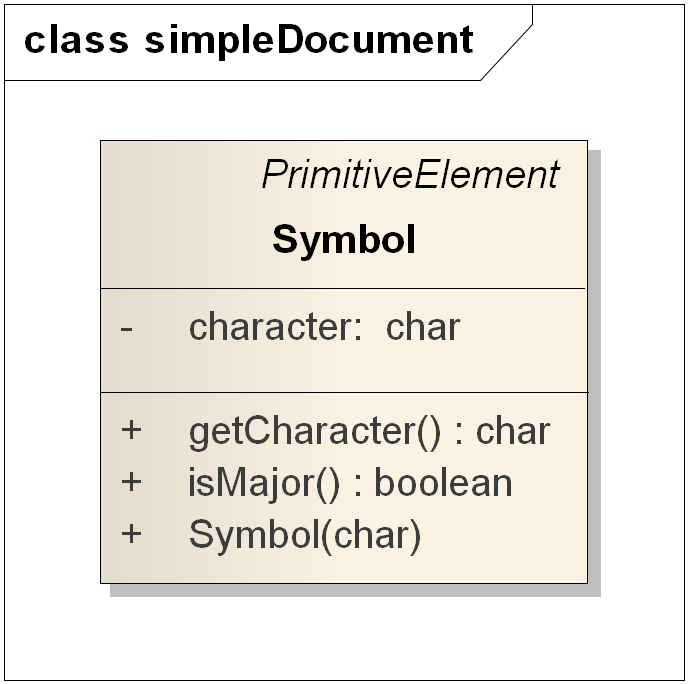
* 1. PrimitiveElement



Малюнок 3.6 – Абстрактний клас PrimitiveElement

Абстрактний клас для створення нових примітивних елементів.

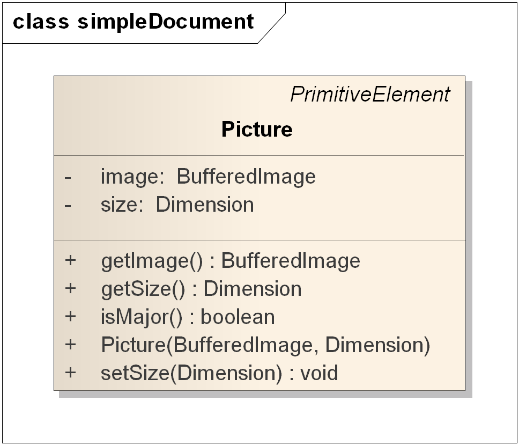
* 1. **Symbol**



Малюнок 3.7 – Клас Symbol

**Клас, що бредставлює собою символ в структурі документа. Містить значення символа який представляє.**

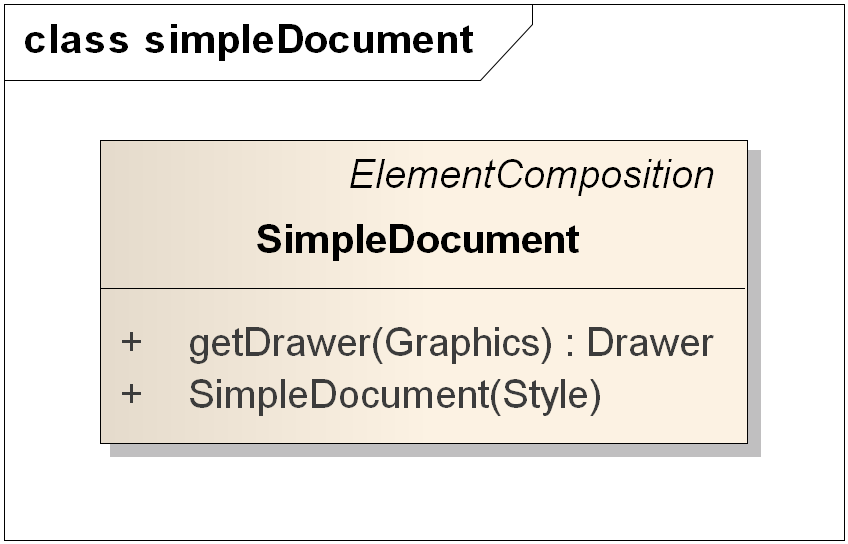
* 1. **Picture**



Малюнок 3.8 – Клас Picture

**Клас, що бредставлює собою зображення в структурі документа. Містить зображення яке представляє та розмір.**

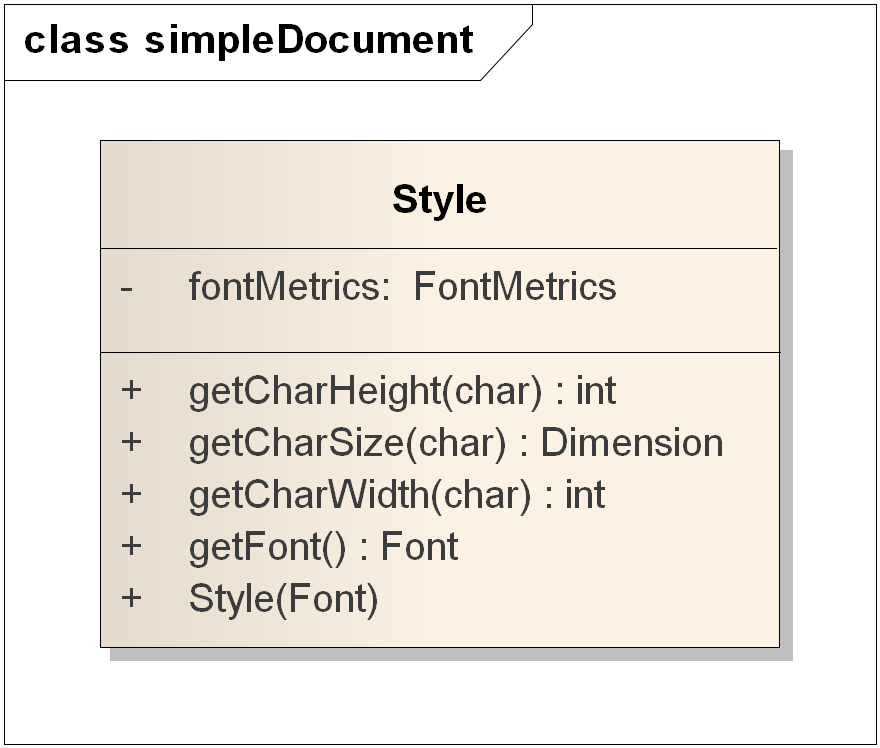
* 1. **SimpleDocument**



Малюнок 3.9 – Клас SimpleDocument

**Клас, що представляє віртуальне відображення тестового документу, складається з екземплярів класу ElementComposition. Створює векземпляр класу SimpleDocumentDrawer.**

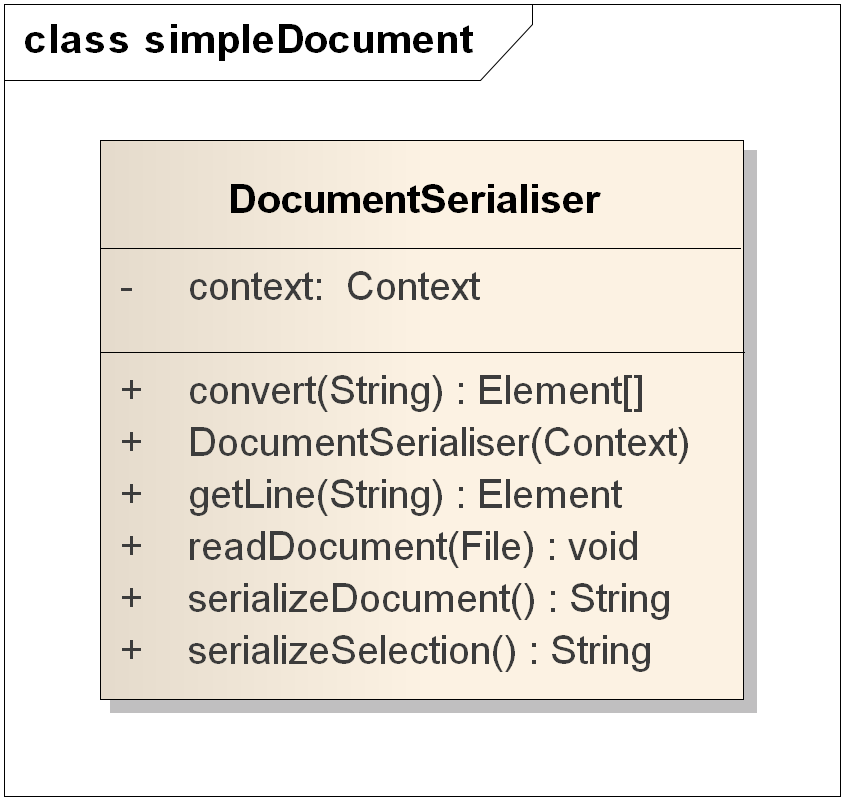
* 1. **Style**



Малюнок 3.10 – Клас Style

**Клас, що представляє стиль симфола, шрифт, розмір та тип (напівжирний, курсив).**

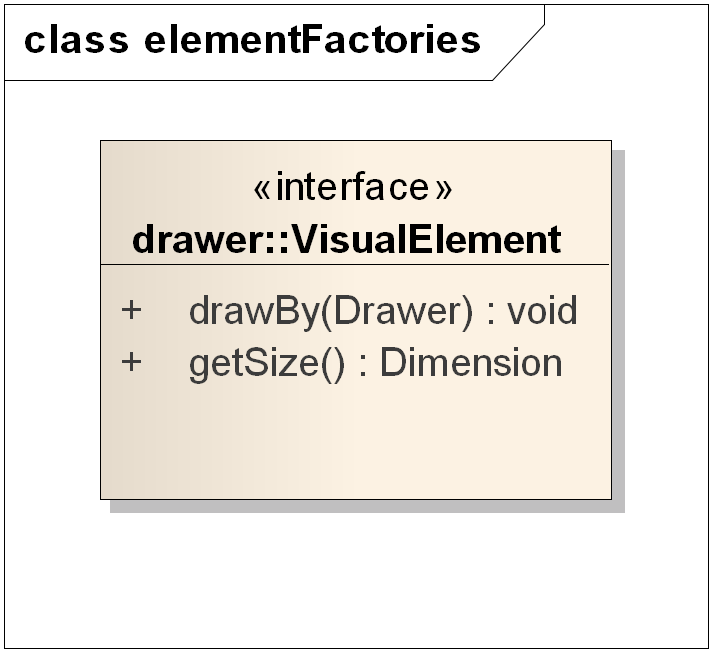
1. Опис підсистеми «Конвертації даних»
   1. DocumentSerialiser



Малюнок 3.11 – Клас DocumentSerialiser

Клас, що служить для серіалізації та десеріалізації документа. Взаємодія з файловою системою, буфером обміну та класом Context.

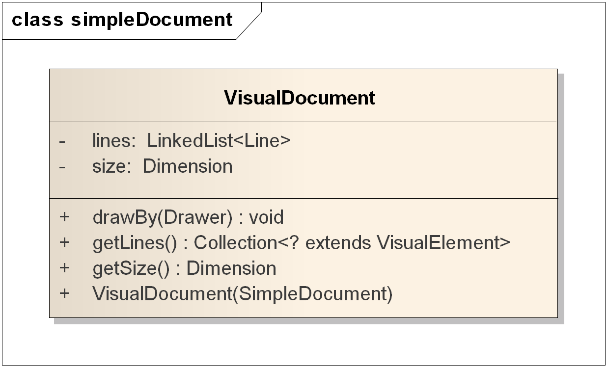
1. Опис підсистеми «Відображення документу»
   1. VisualElement



Малюнок 3.12 – Клас VisualElement

Інтерфейс для опису елементу графічного представлення документу. Реалізуючи цей інтерфейс можна створювати нові типи елементів, що будуть відображені на екрані.

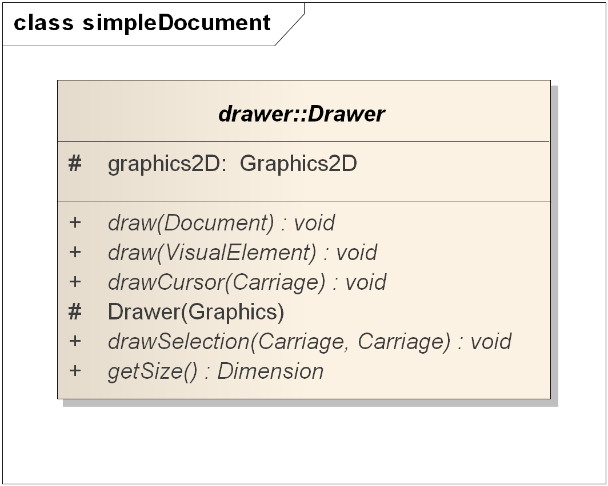
* 1. VisualDocument



Малюнок 3.13 – Клас VisualDocument

Інтерфейс для опису графічного представлення документу. Реалізуючи цей інтерфейс можна свторювати нові типи графічного представлення документу.

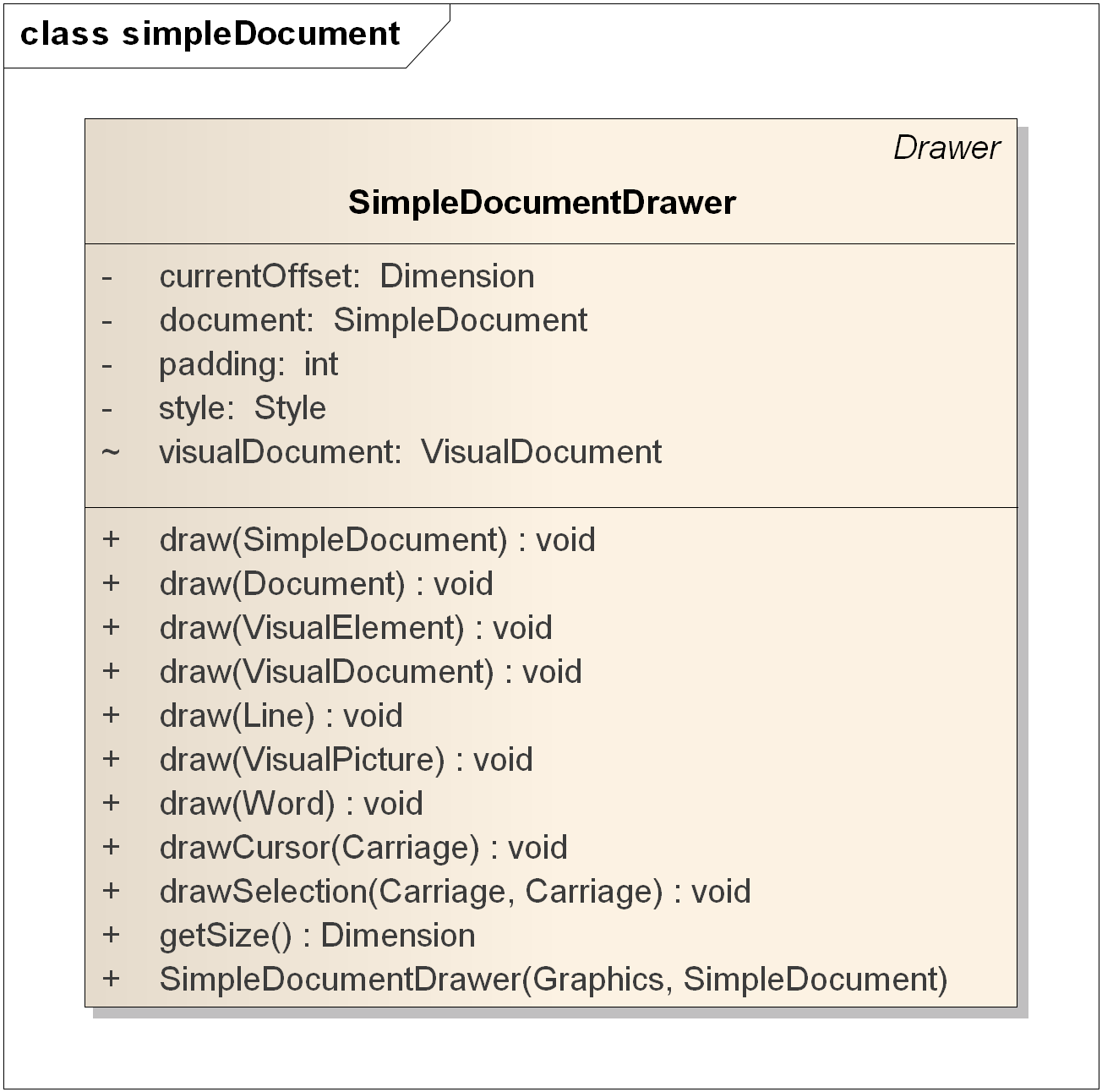
* 1. Drawer



Малюнок 3.14 – Інтерфейс Drawer

Інтерфейс для опису класу вілповідального за відображення документу на екрані. Реалізуючи цей інтерфейс можна створювати нові класи, що будуть відображати документи іншого типу, або будут різнитися в алгоритмі відображення.

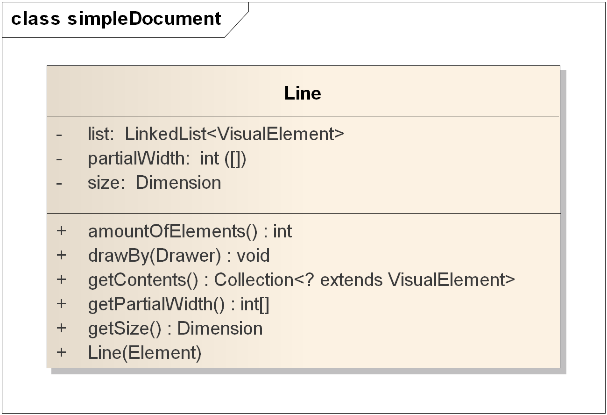
* 1. SimpleDocumentDrawer



Малюнок 3.15 – Клас SimpleDocumentDrawer

Клас, що служить для відображення документа типу SimpleDocument. Ствоює VisualDocument на основі SimpleDocument та відображає його.

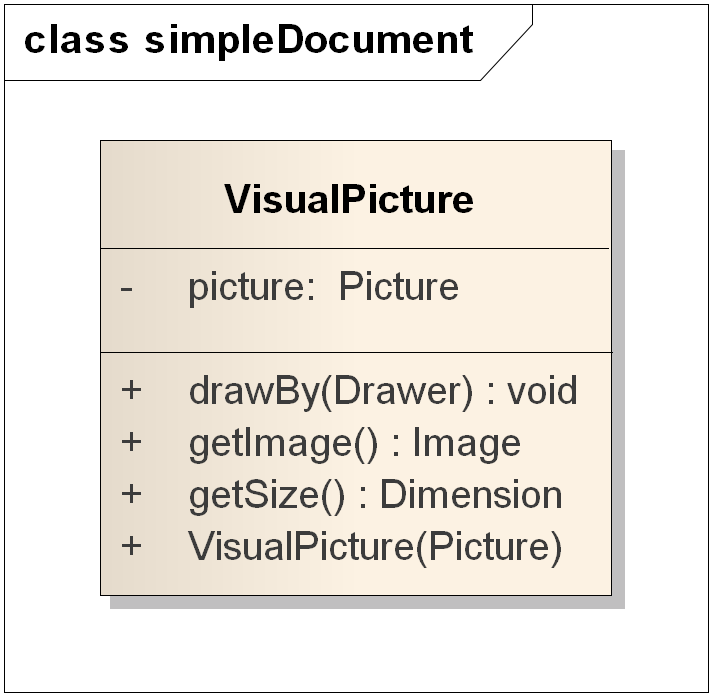
* 1. Line



Малюнок 3.16 – Клас Line

Клас, що служить для відображення сукупності примітивних елементів графічного представлення документу. Складається з Word та VisualPicture.

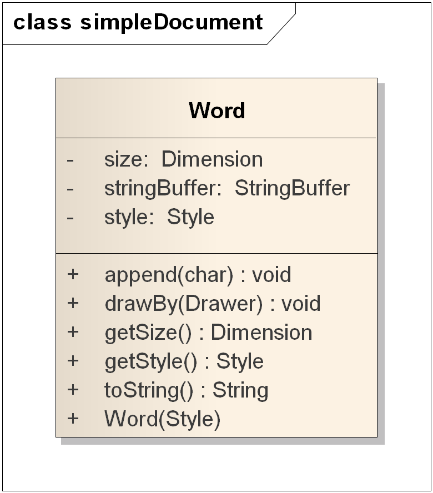
* 1. VisualPicture



Малюнок 3.17 – Клас VisualPicture

Клас, що служить для представлення зображення в структурі VisualDocument.

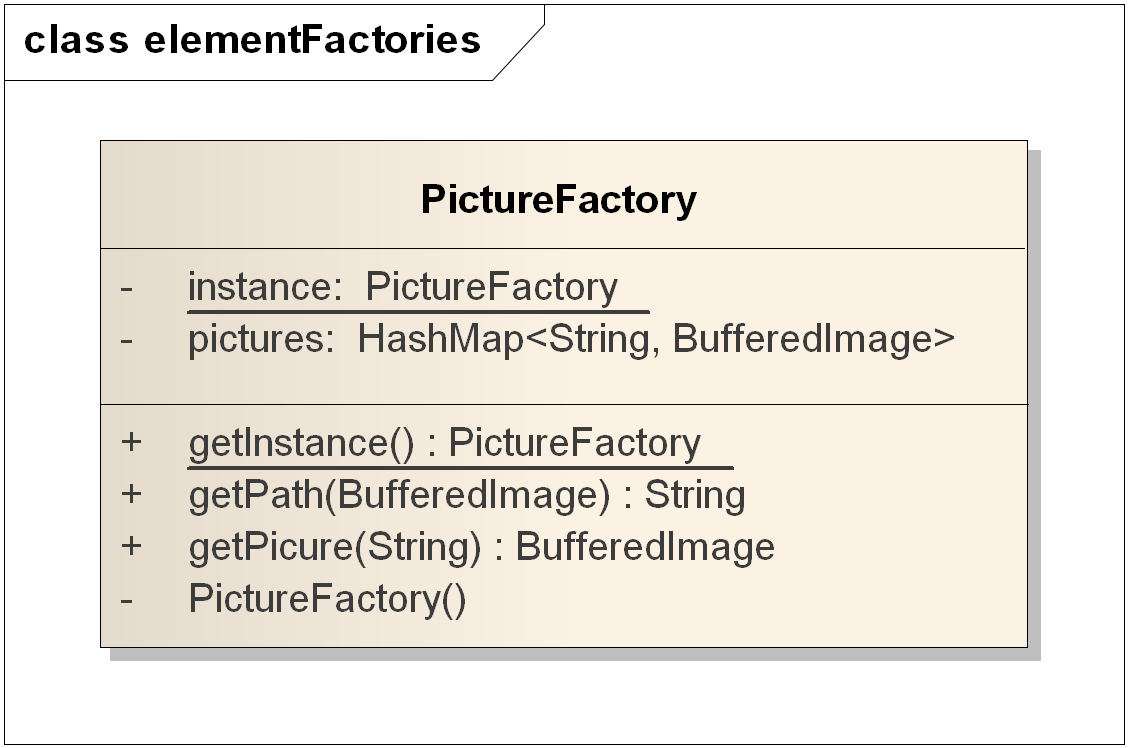
* 1. Word.



Малюнок 3.18 – Клас Word

Клас, що служить для представлення лексеми в структурі VisualDocument. Включає стиль, який Drawer використовую для відображення.

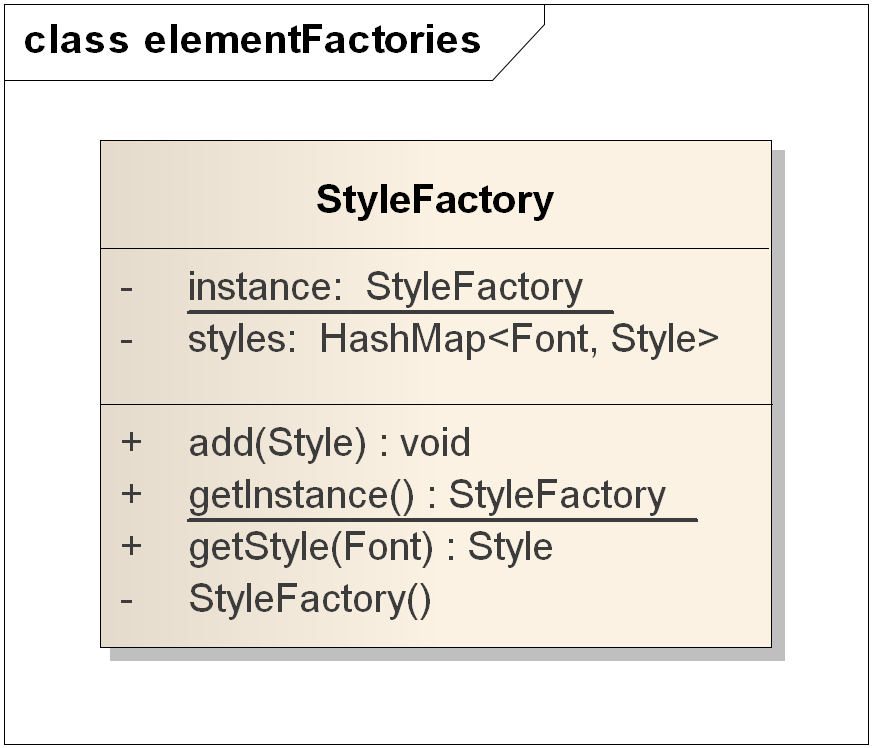
1. Опис підсистеми «Створення елементів»
   1. PictureFactory



Малюнок 3.19 – Клас PictureFactory

Клас, що служить для зчитування зображень с файлової системи.

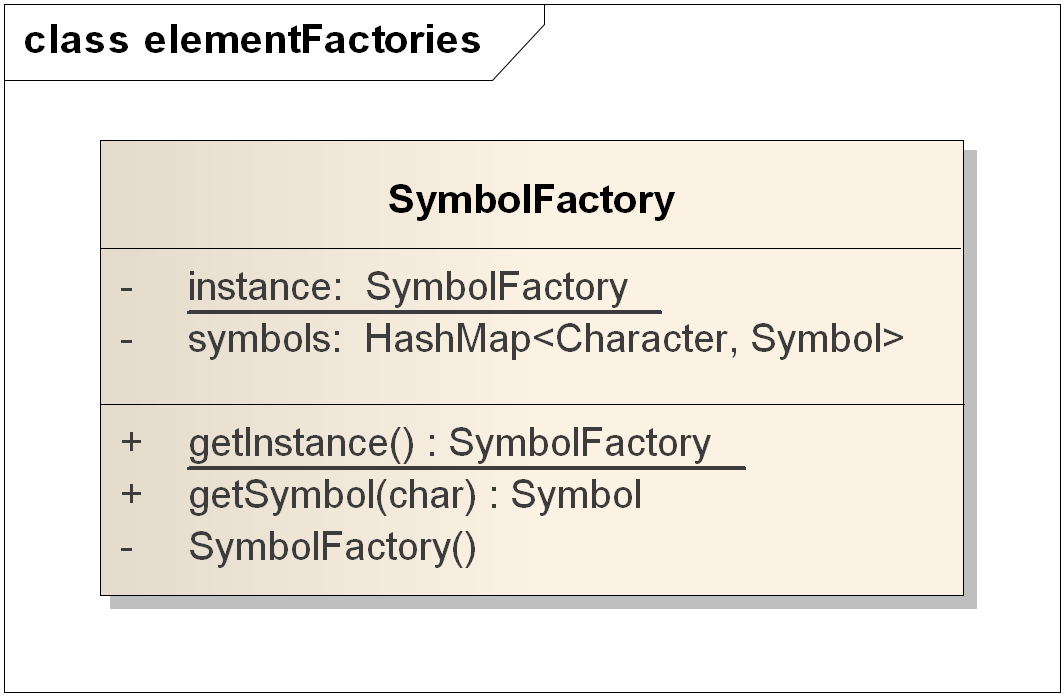
* 1. **StyleFactory**



Малюнок 3.20 – Клас StyleFactory

Клас, що служить для створення стилів.

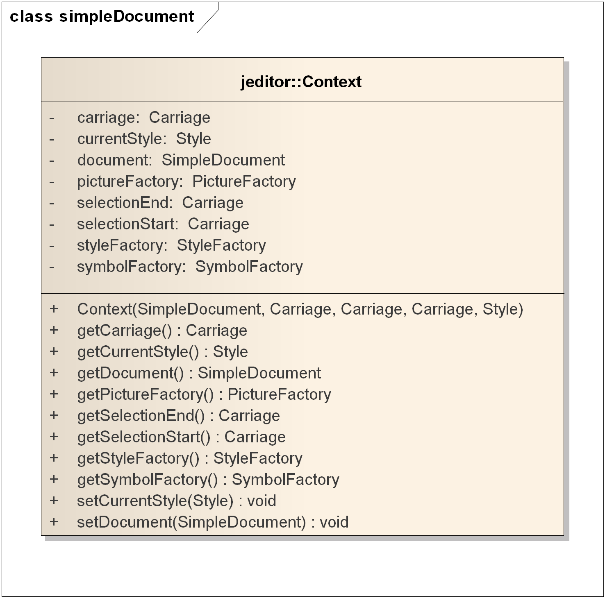
* 1. SymbolFactory



Малюнок 3.21 – Клас SymbolFactory

Клас, що служить для створення символів.

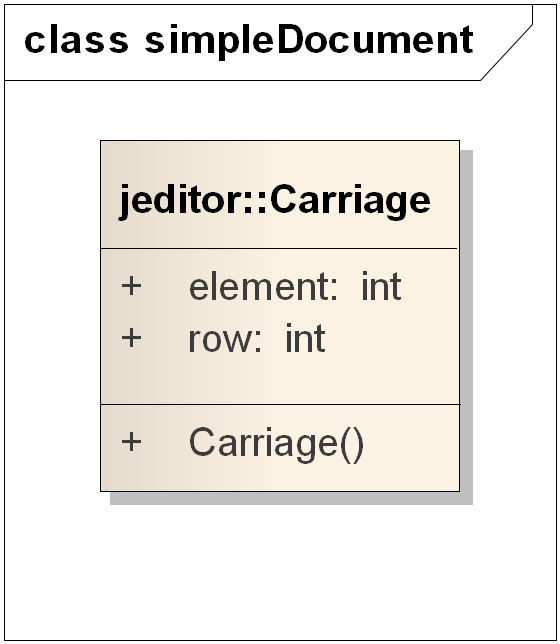
1. Опис підсистеми «Внесення змін»
   1. Context



Малюнок 3.22 – Клас Context

Клас, що представляє контекст, в якому виконуються зміни (інформація про документ, позиція каретка, інформація про виділену область).

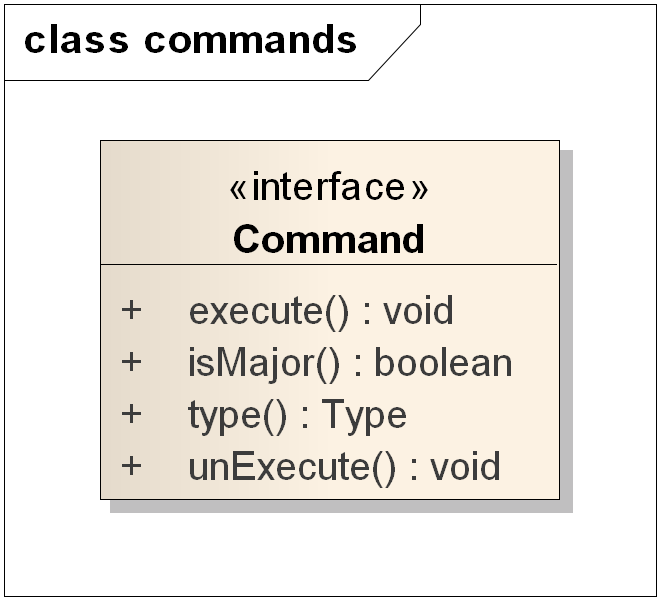
* 1. Carriage



Малюнок 3.23 – Клас Carriage

Клас, що представляє положення каретки в документі.

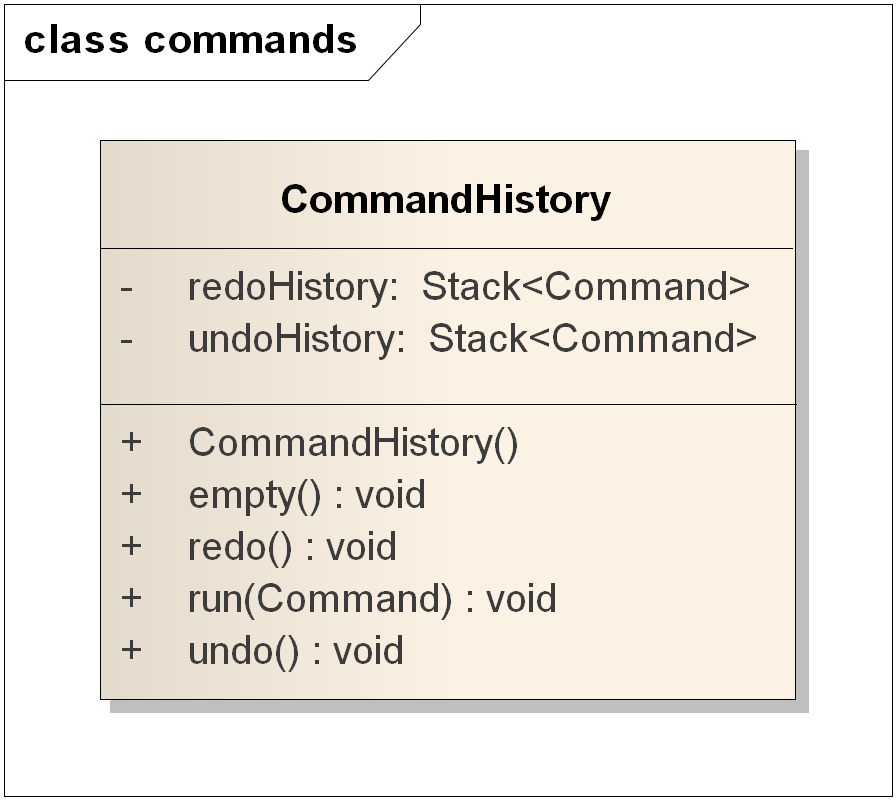
* 1. Command



Малюнок 3.24 – Інтерфейс Command

Інтерфейс для опису команди. Реалізуючи цей інтерфейс можна створювати нові команди по зміні документу.

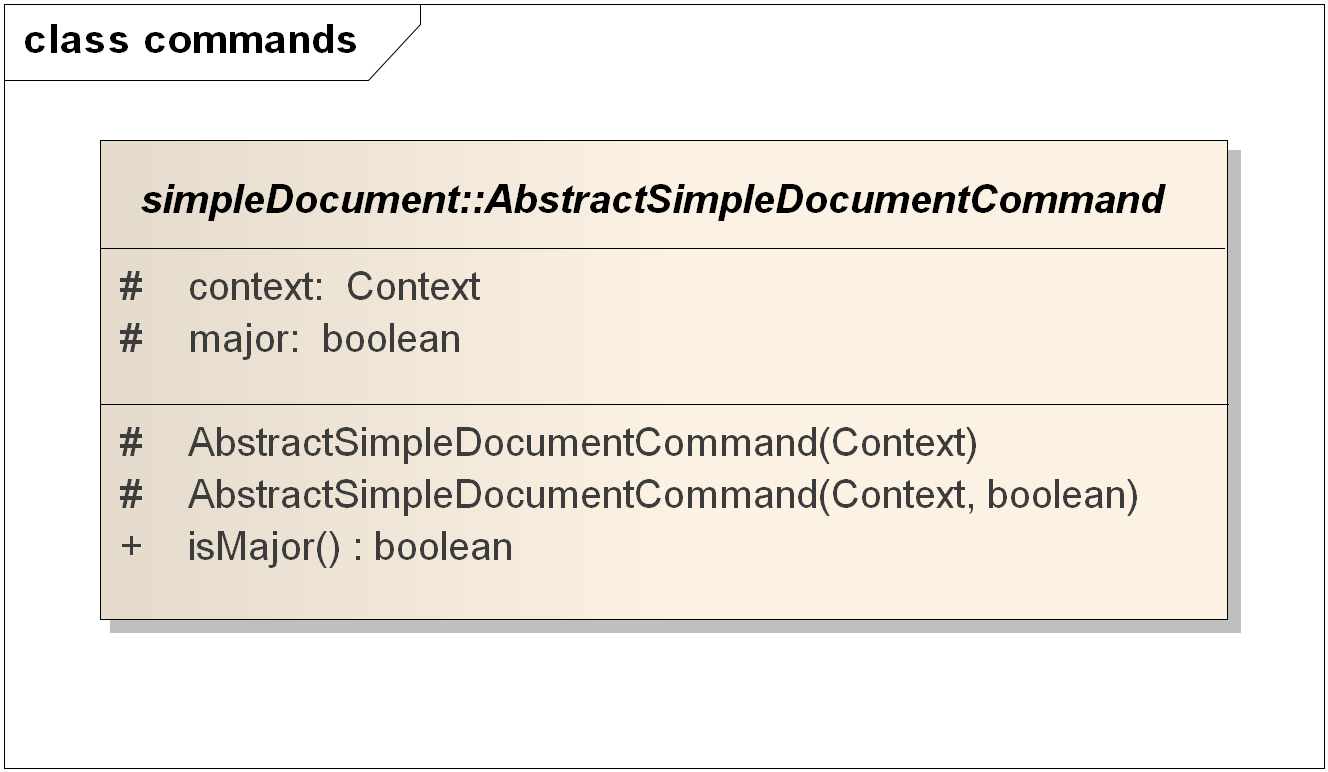
* 1. CommandHistory



Малюнок 3.25 – Клас CommandHistory

Клас, що служить для збереження історії змін. Представляє собою 2 стека команд.

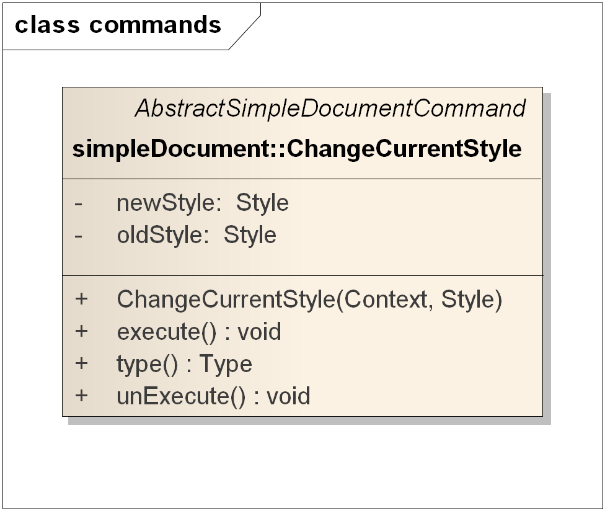
* 1. AbstractSimpleDocumentCommand



Малюнок 3.26 – Абстрактий клас AbstractSimpleDocumentCommand

Абстрактий клас на основі якого створюються нові команди для управління SimleDocumentю.

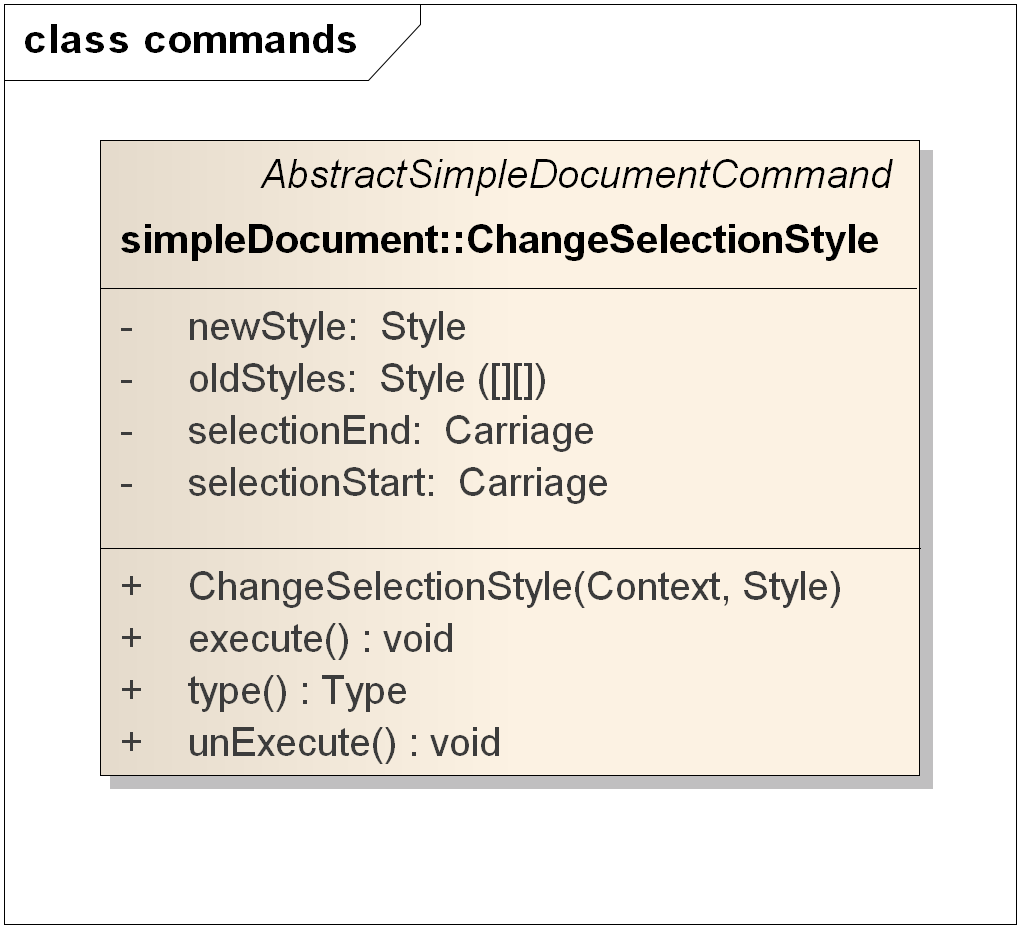
* 1. ChangeCurrentStyle



Малюнок 3.27 – Клас ChangeCurrentStyle

Команда, яка відповідає за зміну поточного стилю.

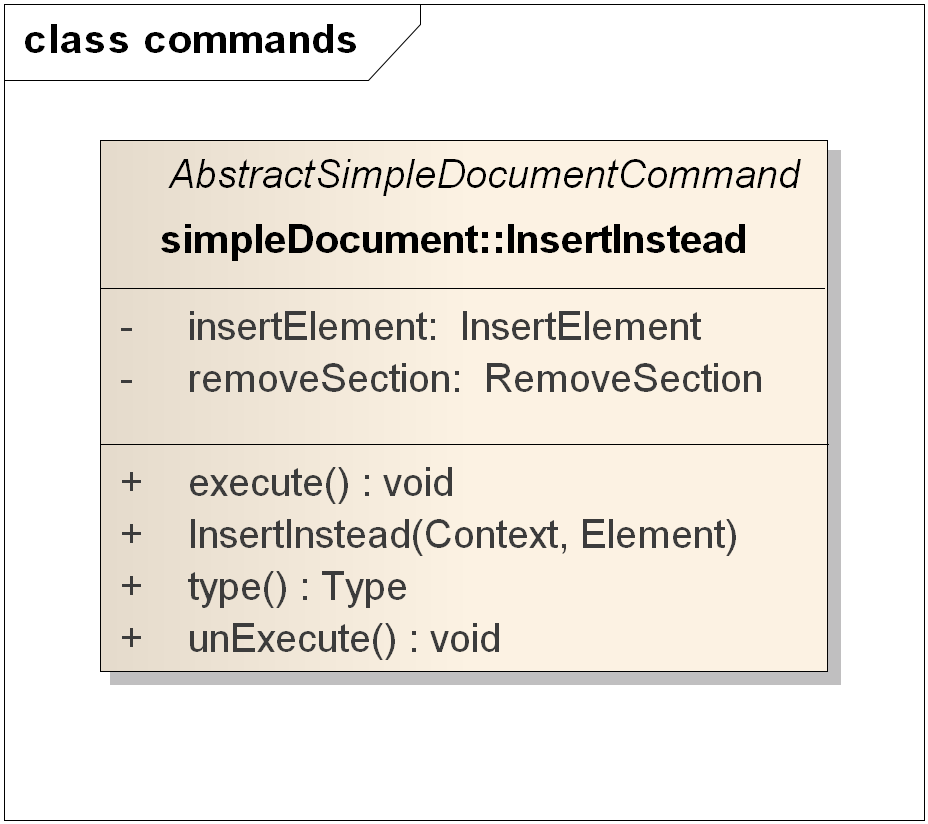
* 1. ChangeSelectionStyle



Малюнок 3.28 – Клас ChangeSelectionStyle

Команда, яка відповідає за зміну поточного стилю виділення.

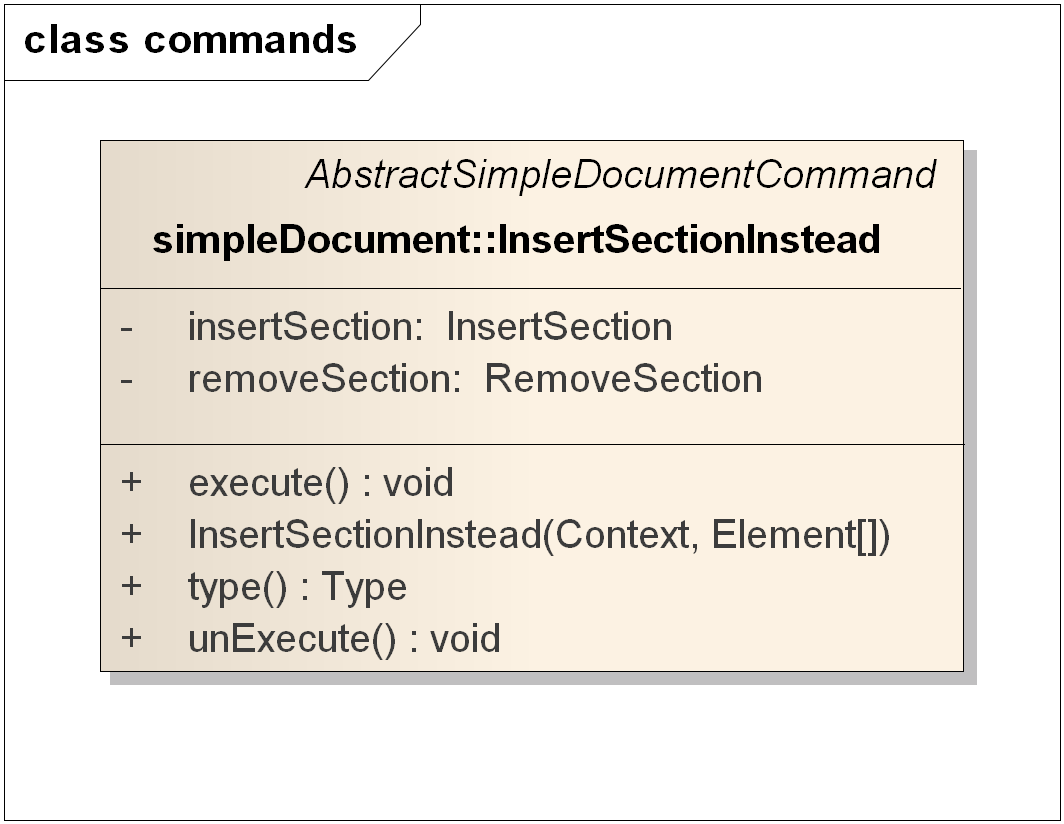
* 1. InsertInstead



Малюнок 3.29 – Клас InsertInstead

Команда, яка відповідає за вставку елемента замість виділеної області.

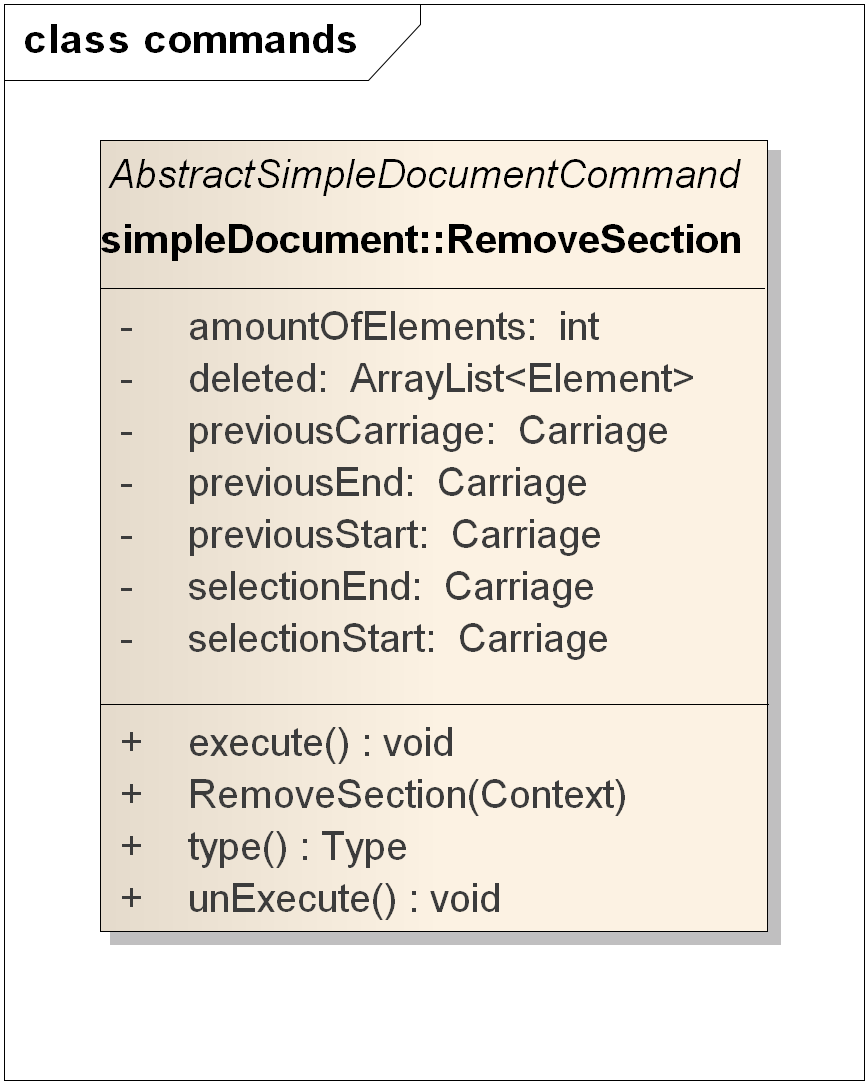
* 1. InsertSectionInstead



Малюнок 3.30 – Клас InsertSectionInstead

Команда, яка відповідає за вставку групи елементів замість виділеної області.

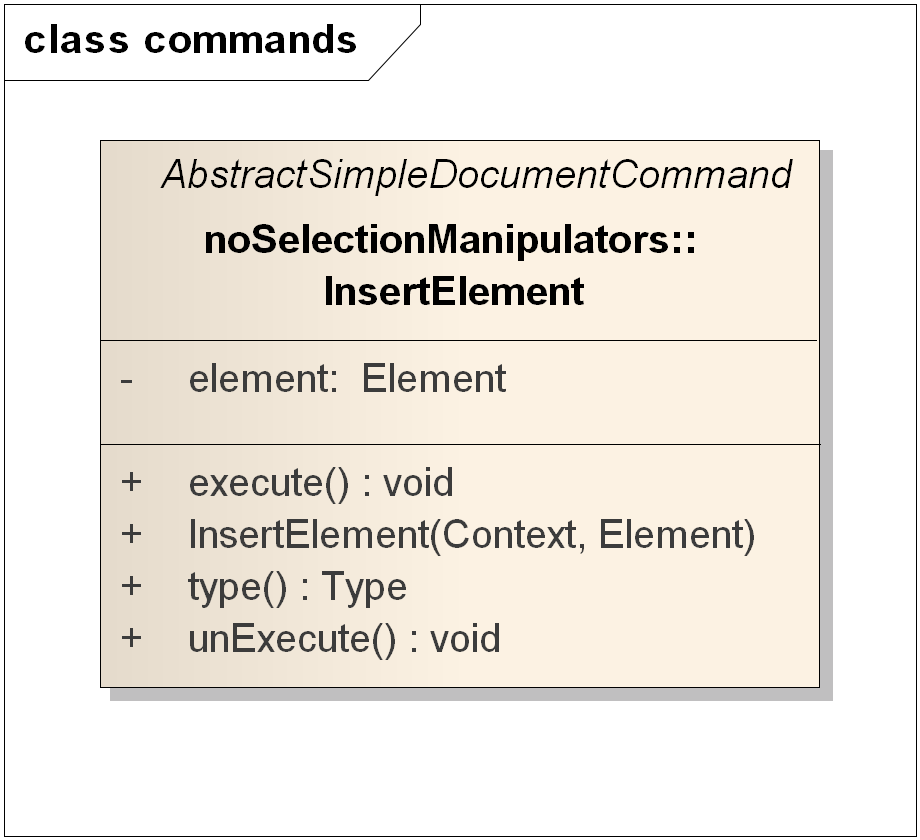
* 1. RemoveSection



Малюнок 3.31 – Клас RemoveSection

Команда, яка відповідає за видалення виділеної області.

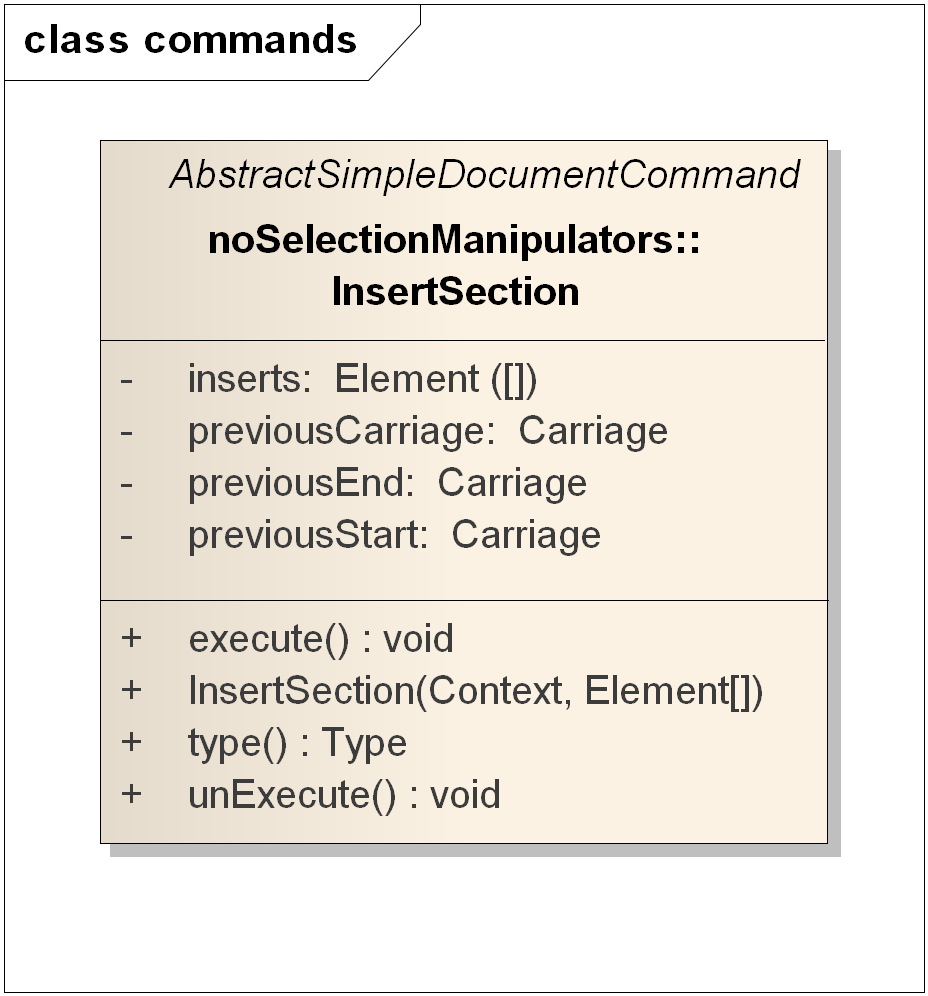
* 1. InsertElement



Малюнок 3.32 – Клас InsertElement

Команда, яка відповідає за вставку елемента.

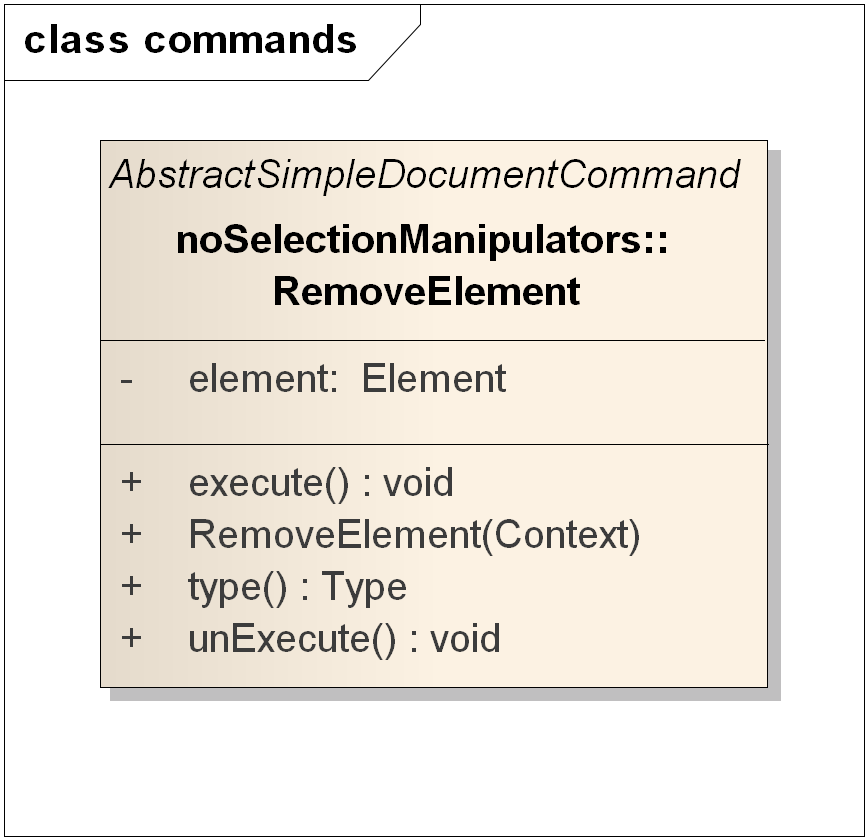
* 1. InsertSection



Малюнок 3.33 – Клас InsertSection

Команда, яка відповідає за вставку групи.

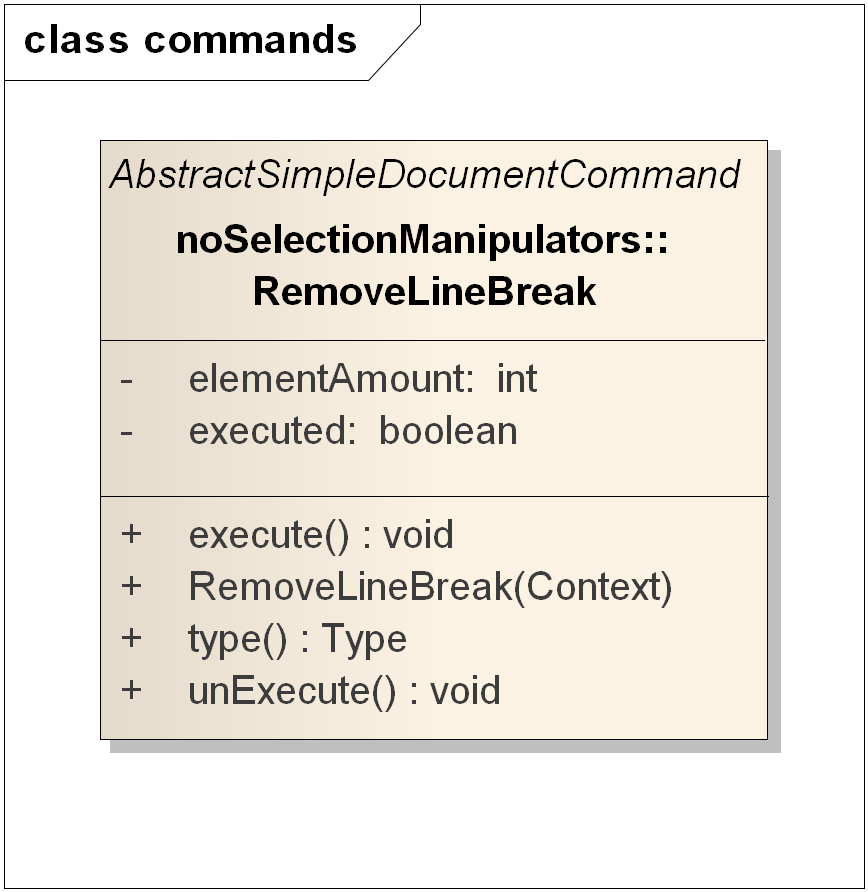
* 1. RemoveElement



Малюнок 3.34 – Клас RemoveElement

Команда, яка відповідає за видалення елемента.

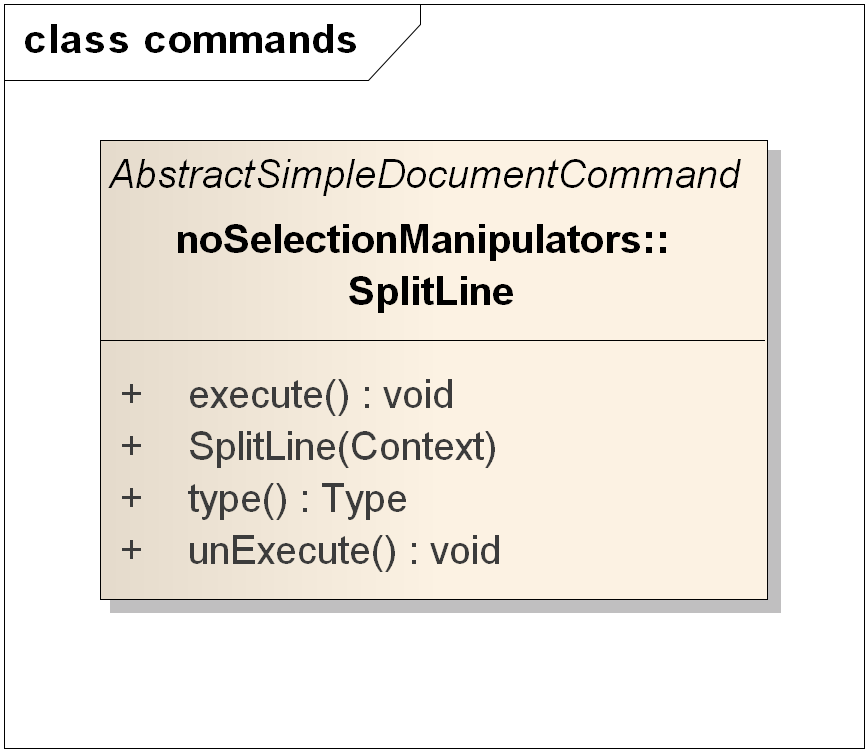
* 1. RemoveLineBreak



Малюнок 3.35 – Клас RemoveLineBreak

Команда, яка відповідає за видалення переносу рядка.

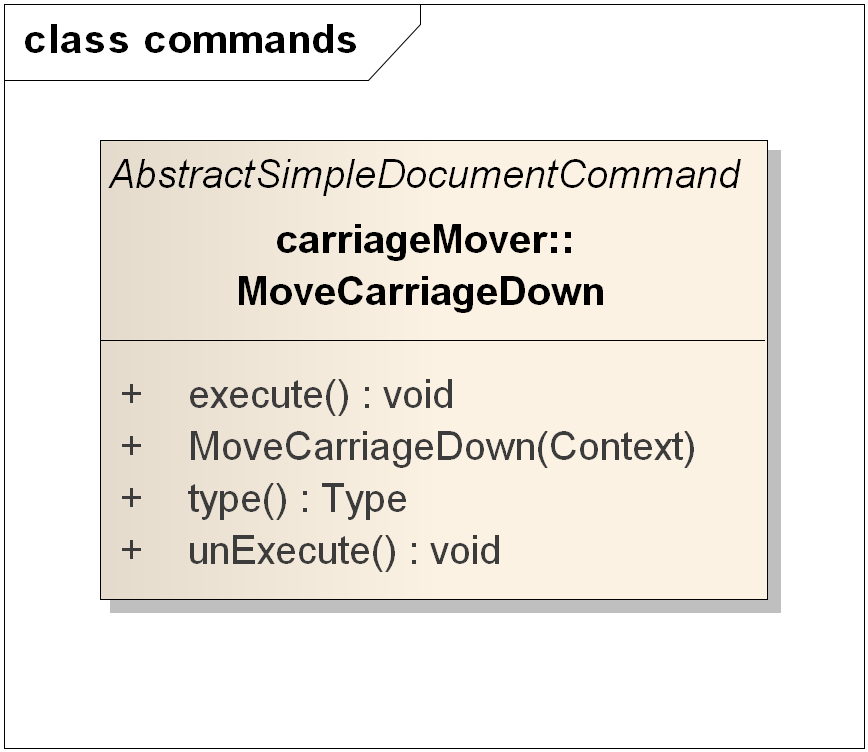
* 1. SplitLine



Малюнок 3.36 – Клас SplitLine

Команда, яка відповідає за перенос рядка.

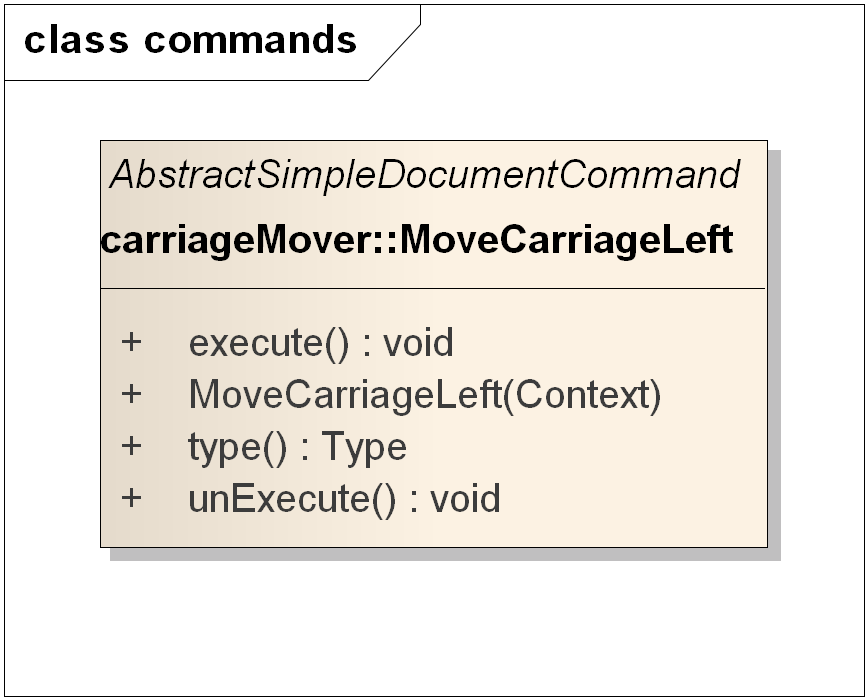
* 1. MoveCarriageDown



Малюнок 3.37 – Клас MoveCarriageDown

Команда, яка відповідає за переміщення каретки вниз.

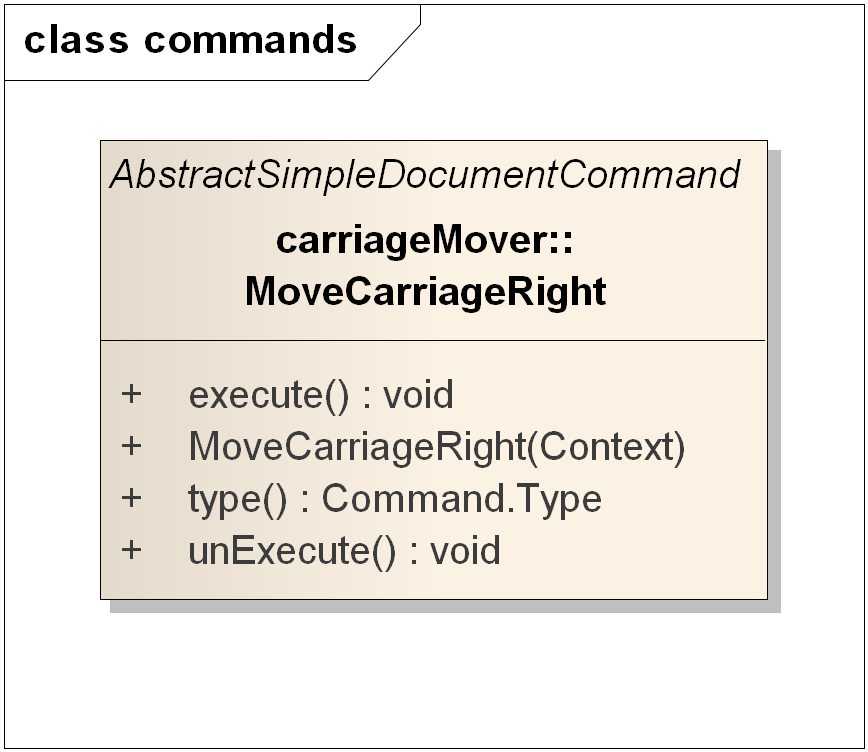
* 1. MoveCarriageLeft



Малюнок 3.38 – Клас MoveCarriageLeft

Команда, яка відповідає за переміщення каретки вліво.

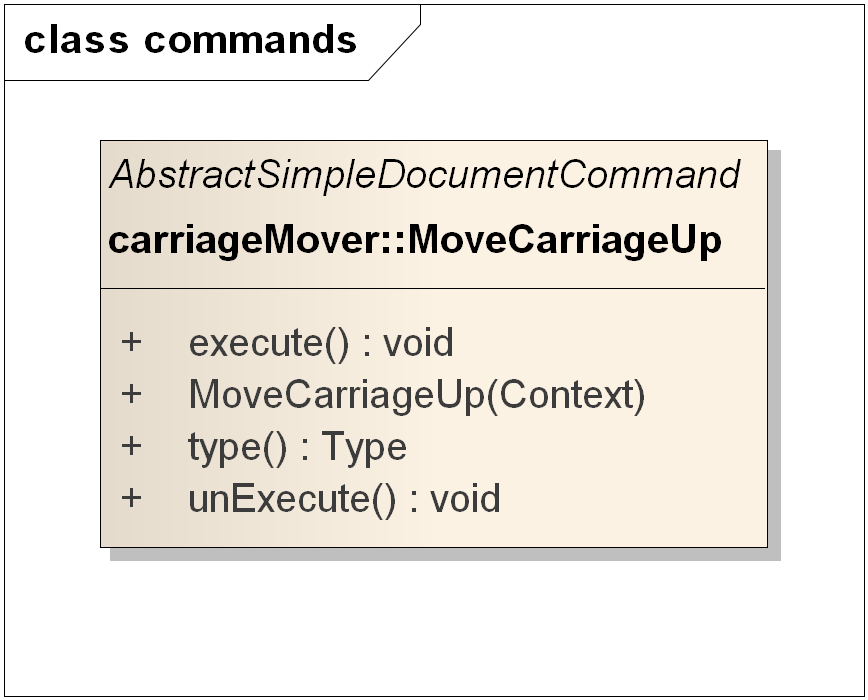
* 1. MoveCarriageRight



Малюнок 3.39 – Клас MoveCarriageRight

Команда, яка відповідає за переміщення каретки вправо.

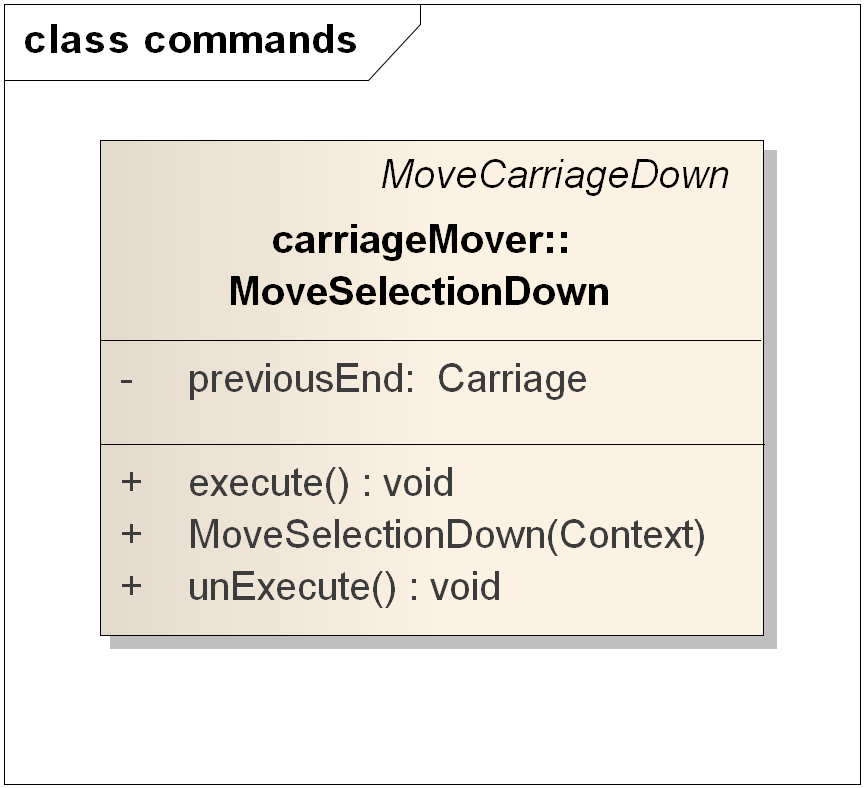
* 1. MoveCarriageUp



Малюнок 3.40 – Клас MoveCarriageUp

Команда, яка відповідає за переміщення каретки вверх.

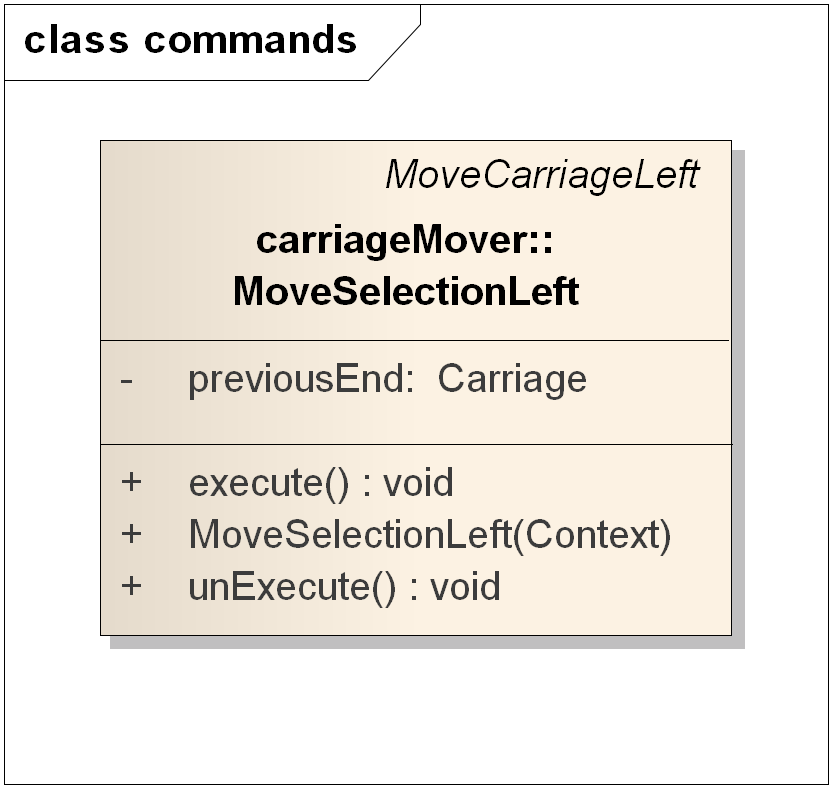
* 1. MoveSelectionDown



Малюнок 3.41 – Клас MoveSelectionDown

Команда, яка відповідає за переміщення виділення вниз .

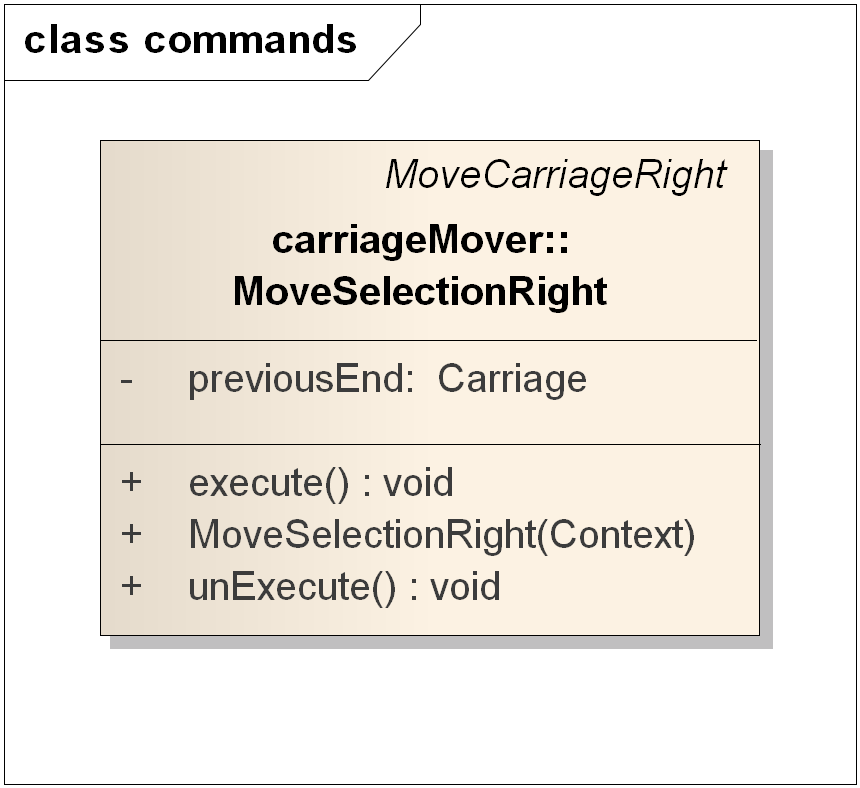
* 1. MoveSelectionLeft



Малюнок 3.42 – Клас MoveSelectionLeft

Команда, яка відповідає за переміщення виділення вліво.

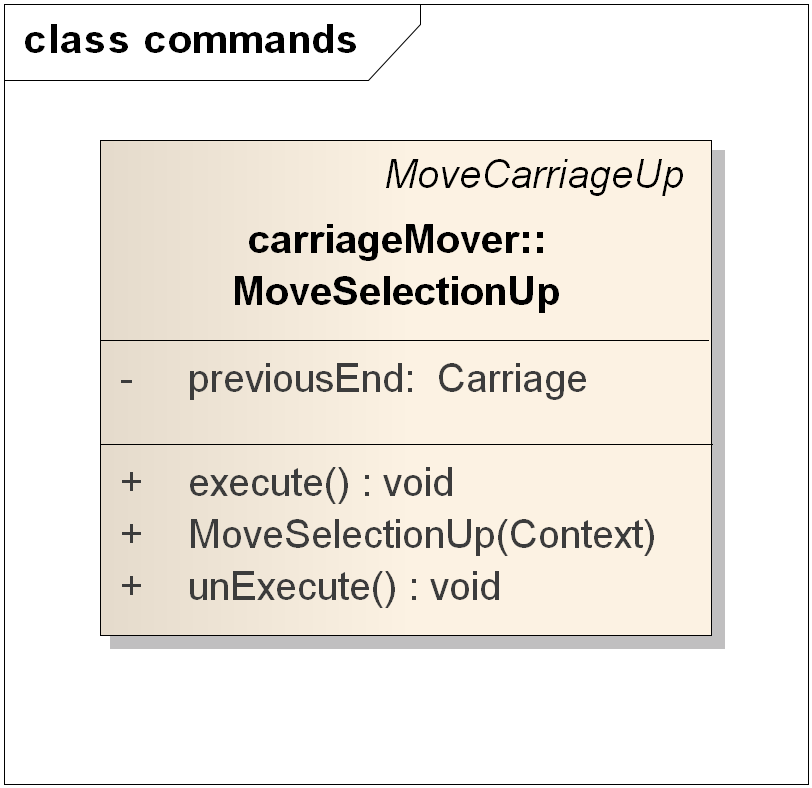
* 1. MoveSelectionRight



Малюнок 3.43 – Клас MoveSelectionLeft

Команда, яка відповідає за переміщення виділення вправо.

* 1. MoveSelectionUp



Малюнок 3.44 – Клас MoveSelectionUp

Команда, яка відповідає за переміщення виділення вверх.

Висновок

У процесі розробки текстового редактора відповідно до вимог виявлених у п.1 та рішень прийнятих у п.3 було реалізовано такі підсистеми :

* Графічний інтерфейс користувача.
* Конвертації даних.
* Представлення документу в пам’яті.
* Відображення документу.
* Внесення змін.
* Створення елементів.

Кожна підсистема була розроблена відповідно до рішень прийнятих в п.2..

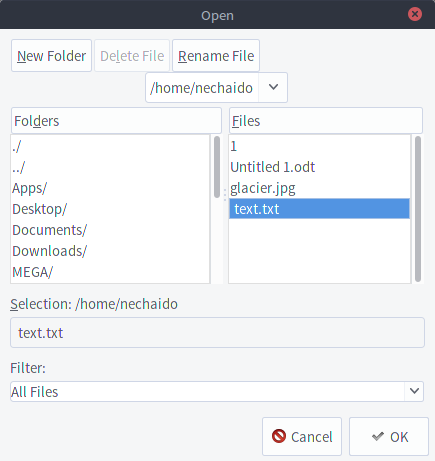
Під час реалізації цієї системи неоднократно з’являлись питання, про те як зробити систему розширюваною. Рішення цих проблем було знайдено за допомогою паттернів.

Результуюча система є повністю працездатною та задовольняє всі вимоги, сформовані у п.1.

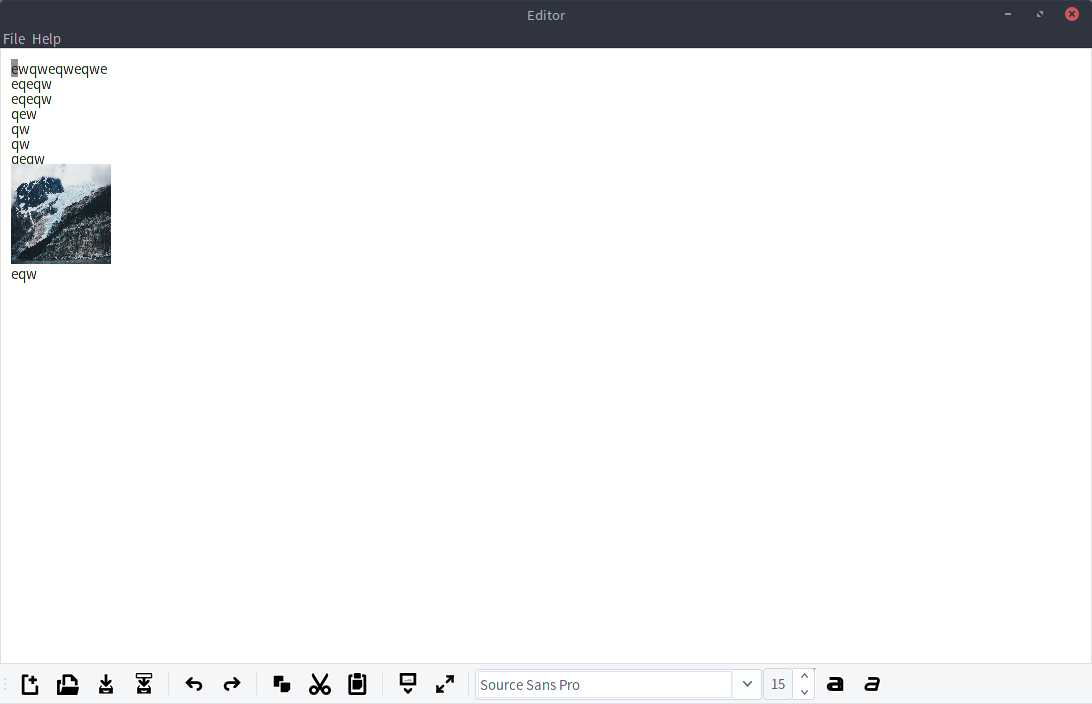
# Опис застосування системи

Текстовий редактор застосовується для створенння, редагування та текстових документів. Особливістю даного редактора є те, що він підтримує роботу з зображеннями.

Після запуску програми користувач має можливість відкрити або створити файл.

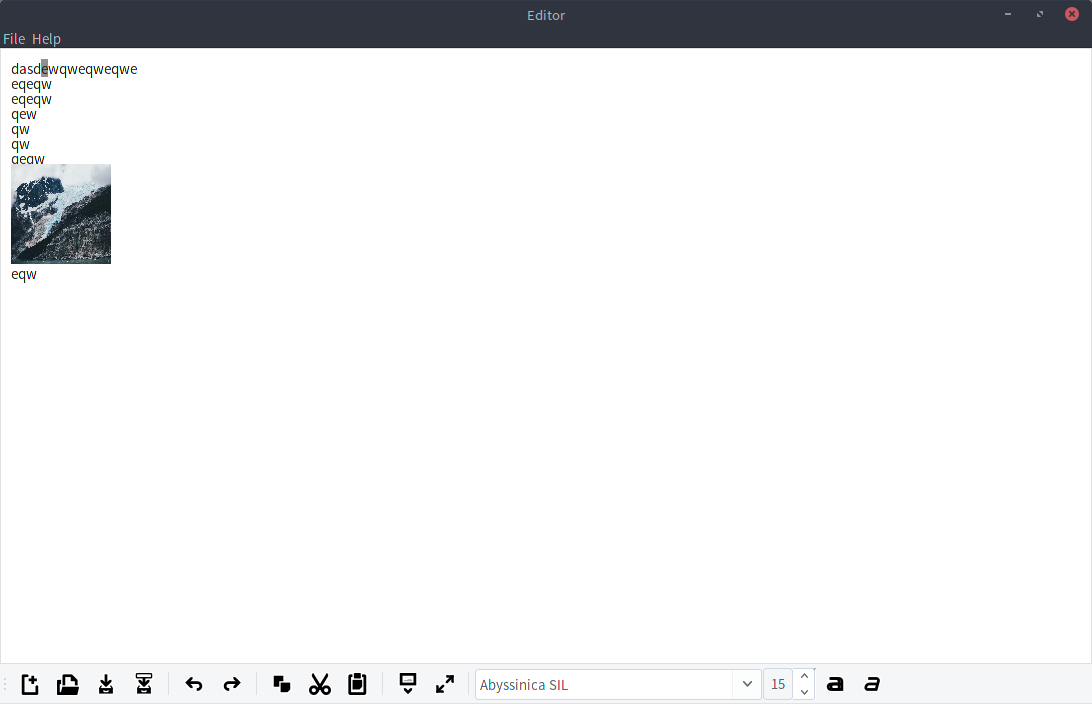
****

Малюнок 4.1 – відкритя документа

****

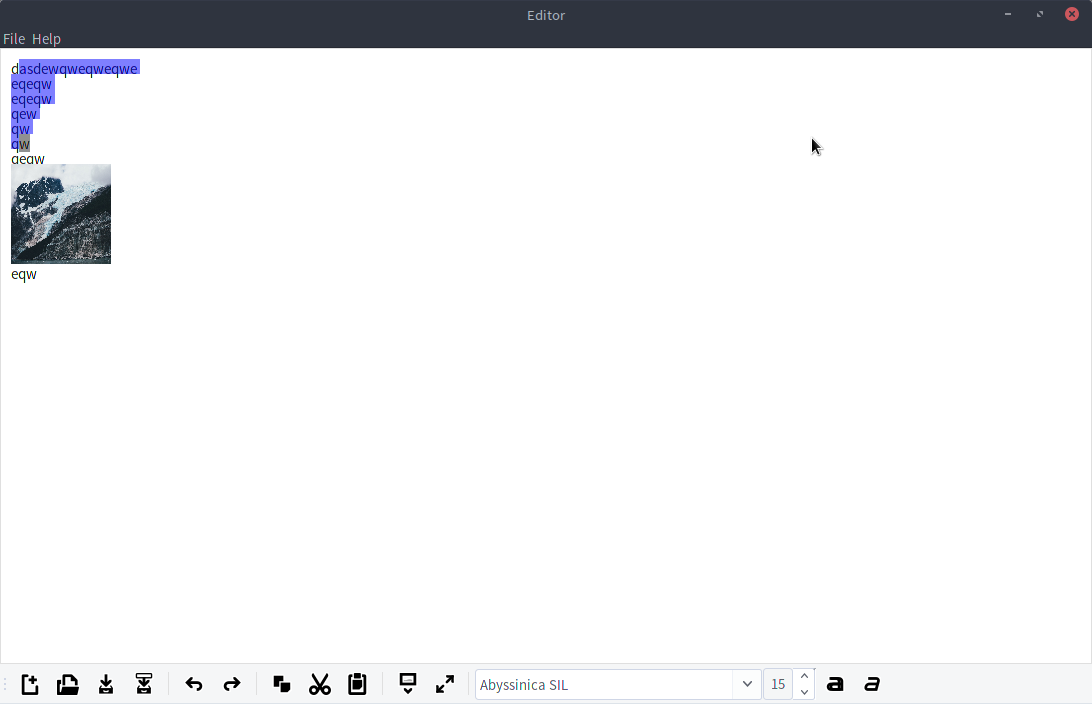
Малюнок 4.2 – відкритий документ

У відкритому файлі користувач може вводити текст.



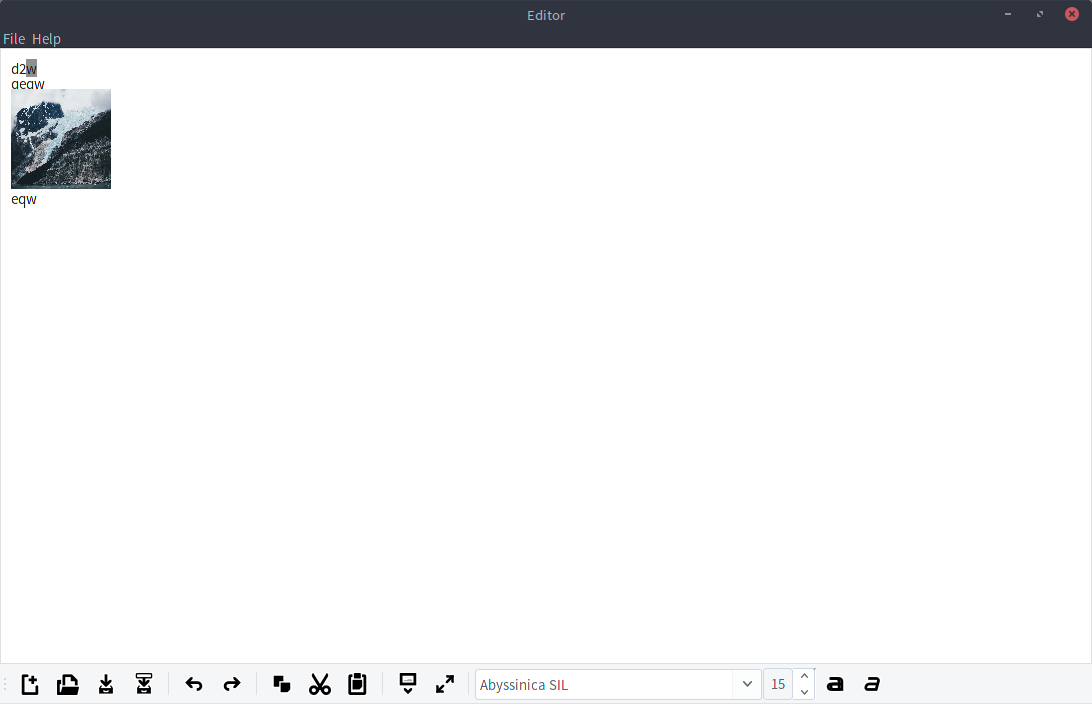
Малюнок 4.3 – користувач ввів текст

Система надає можливість виділяти текст, для цього потрібно зажати SHIFT та перемістити каретку за допомогою стрілок на клавіатурі.



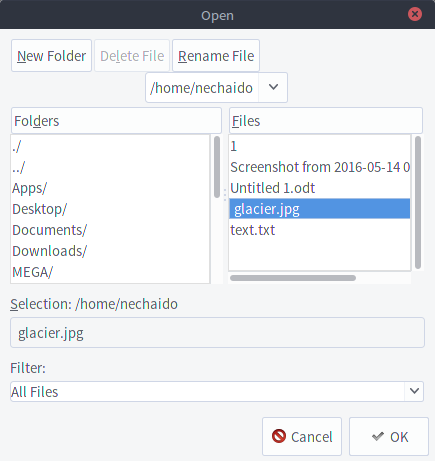
Малюнок 4.4 – користувач виділив блок

Якщо вставити едемент при активно виділені – воно видалиться і на його місце стане введений елемент.

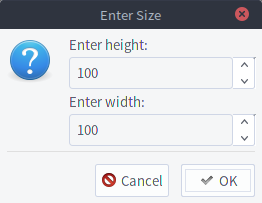


Малюнок 4.5 – користувач вставив символ замість виділення

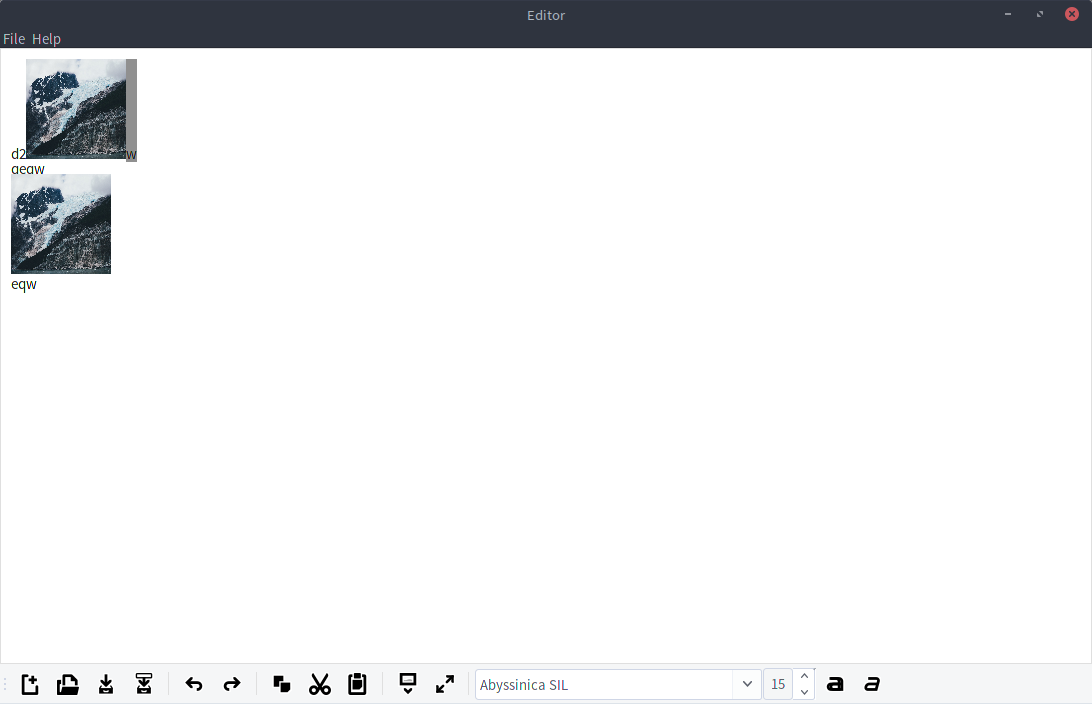
Система надає можливість вставити зображення, для цього необхідно натиснути на кнопку вставки зображення, вибрати файл та розмір.



Малюнок 4.6 – користувач вибирає зображення для додавання

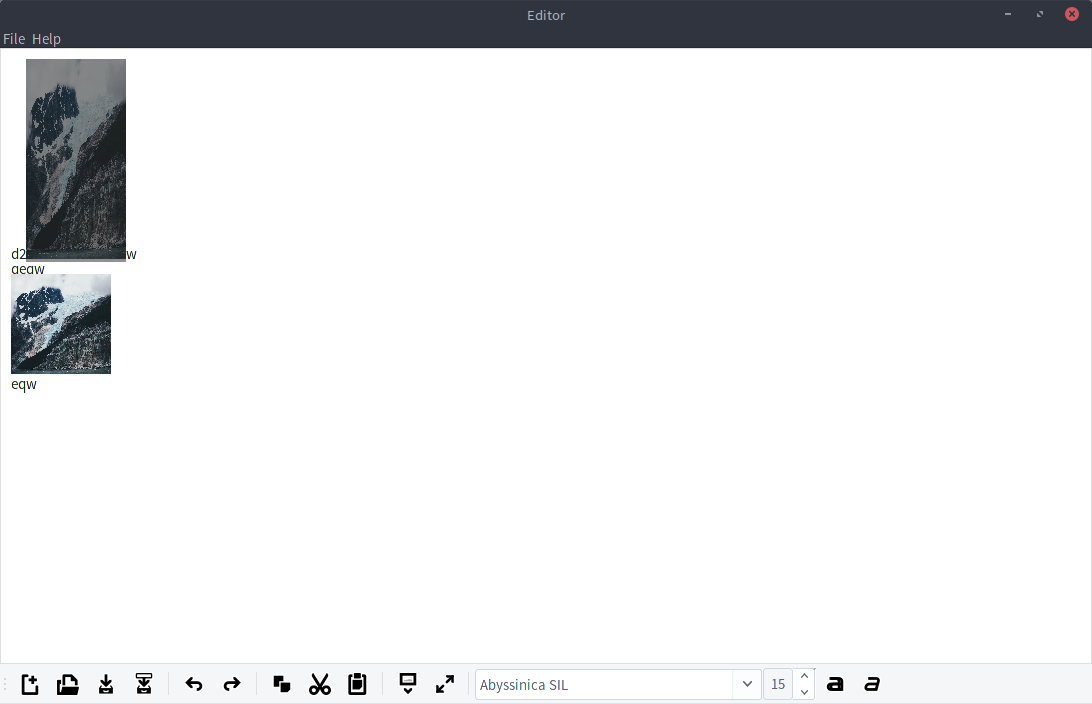


Малюнок 4.7 – користувач розмір зображення



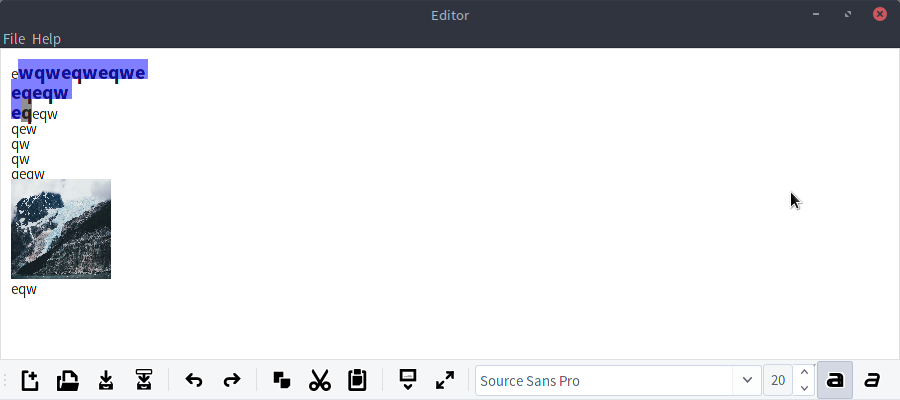
Малюнок 4.8 – додане зображення

Користувач може змінити розмір зображення, для цього є кнопка «Змінити розмір»



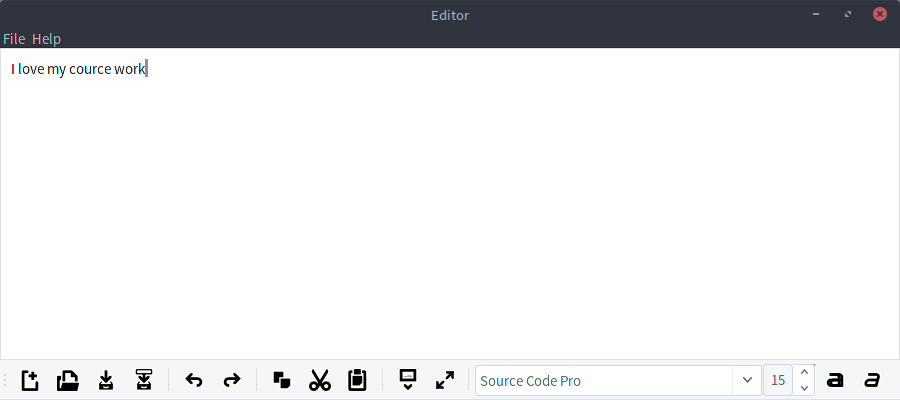
Малюнок 4.9 – користувач змінив розмір зображення

Система надає можливість змінити стиль тексту, для цього – виділити текст та вибрати стиль на панелі інструментів.

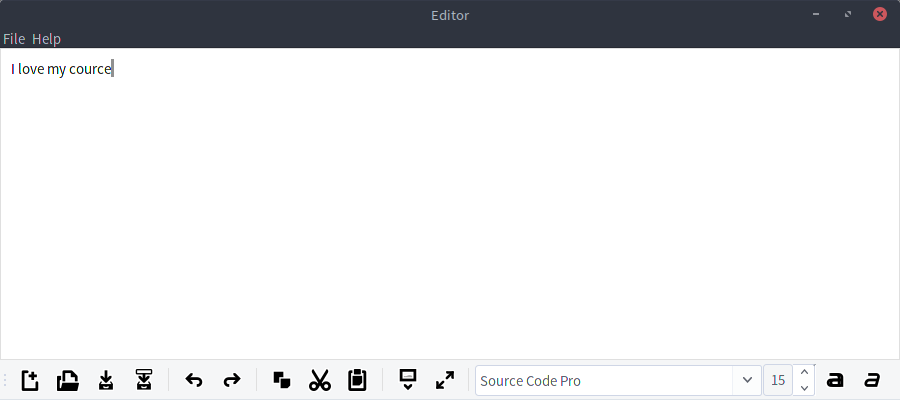


Малюнок 4.10 – користувач виділив область та змінив розмір шрифта

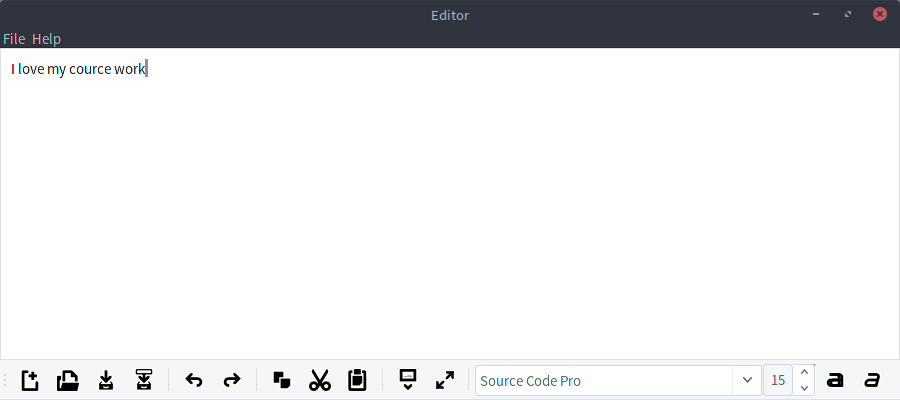
У користувача є можливість повернутись до попереднього\наступного стану, для цього потрібно внести якісь зміни та скористатись відповідною кнопкою.



Малюнок 4.11 – користувач ввів текст



Малюнок 4.11 – користувач використав функцію undo



Малюнок 4.11 – користувач використав функцію redo

Висновок

Було спроектовано та розроблено систему, яка повністю покриває запланований функціонал розділу 1:

* Можливість створення документу.
* Можливість редагування існуючого документу.
* Можливість збереження документу.
* Можливість вводити текст.
* Можливість зміювати шрифт тексту та робити його курсивним або напівжирним.
* Можливість додавати зображення.
* Можливість змінювати розмір зображення.
* Можливість видаляти елементи.
* Можливість виділення послідовності елементів.
* Можливість виконати операції : вставка, вирізання та копіювання послідовності елементів.
* Можливість повернення до попереднього та наступного стану системи.

# Висновок

Отже, був розроблений текстовий редактор з підтримкою растрових зображень та історії змін. Створений редактор підтримує створення, завантаження, редагування та зберігання текстових документів.

* Користувачу доступно: Можливість створення документу.
* Можливість редагування існуючого документу.
* Можливість збереження документу.
* Можливість вводити текст.
* Можливість зміювати шрифт тексту та робити його курсивним або напівжирним.
* Можливість додавати зображення.
* Можливість змінювати розмір зображення.
* Можливість видаляти елементи.
* Можливість виділення послідовності елементів.
* Можливість виконати операції : вставка, вирізання та копіювання послідовності елементів.
* Можливість повернення до попереднього та наступного стану системи.

При проектуванні були використані можливості ООП та патерни проектування. Були застосовані такі патерни:

1. Composite
2. Visitor
3. Command
4. Abstract Factory
5. Factory Method
6. Flyweight
7. Singleton

В результаті розробки текстового редактора була розроблена система, що повністю задовольняє вимогам сформованим в п.1. В процесі розробки та проектування системи було використанно знання ООП на практиці та покращено знання того як працюють текстові редактори.

# Список літератури

1. Сухин Н. А.: Поурочные разработки по информатике. – М.: ВАКО, 2013. – 304 с. – ISBN 978-5-408-01270-1
2. Кейс "Проектирование графического редактора" : - Software Design Популярно о проектировании программ - 4.01.2011 - Режим доступу: http://askofen.blogspot.com/2011/01/blog-post.html. – Назва з екрана
3. Фаулер, Мартин. «Архитектура корпоративных программных приложений» .: Пер. с англ. - М.: Издательский дом "Вильямс", 2006. - 544 с.: ил. - Парал. тит. англ.
4. Йенер М., Фидом А. «Паттерны проектирования» .: Пер. с англ. - М.: Издательский дом "Вильямс", 2010. - 421 с.
5. Мак-Колм Смит Джейсон. «Элементарные шаблоны проектирования» .: Пер. с англ. - М.: Издательский дом "Вильямс", 2012. - 221 с.
6. Макдональд Дж.С. «Схемы основных паттернов проектирования» М.: Высшая школа, 2007.–340 с.
7. .Мельниченко В.В., Легейда В.В. Настоящий самоучитель компьютерной графики. - М.: Век +, 2005. - 560с.
8. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. П75 Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста») ISBN 5-272-00355-1
9. Тидвелл Дж. Разработка пользовательских интерфейсов.— Питер, 2007. Jenifer Tidwell. Designing Interfaces, 2nd Edition. Patterns for Effective Interaction Design. — O’Reilly, 2010.
10. Шилдт Г. «Java 8. Полное руководство. 9 – е издание». Пер. с англ. – М.: Издательский дом «Вильямс», 2016. – 1376с.

# Додатки

## Код програми

package com.nechaido.editor;

import com.nechaido.editor.jeditor.JEditor;

import javax.swing.\*;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.InputEvent;

import java.net.URL;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public class Editor extends JFrame{

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

new Editor();

}

});

}

private Editor(){

super();

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (ClassNotFoundException | InstantiationException | IllegalAccessException | UnsupportedLookAndFeelException e) {

e.printStackTrace();

}

createGUI();

setVisible(true);

}

private void createGUI() {

setTitle("Editor");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

editor = new JEditor();

editor.setFocusable(true);

editorScrollPane = new JScrollPane(editor);

getContentPane().add(editorScrollPane, BorderLayout.CENTER);

createMenuBar();

createToolbar();

getContentPane().add(toolBar, BorderLayout.PAGE\_END);

pack();

setSize(900, 400);

setMinimumSize(new Dimension(900, 400));

editor.addKeyListener(editor);

createActions();

}

private JEditor editor;

private JScrollPane editorScrollPane;

private JMenuBar menuBar;

private JMenu file;

private JMenuItem newFile;

private JMenuItem openFile;

private JMenuItem saveFile;

private JMenuItem saveFileAs;

private JMenuItem exit;

private JMenu help;

private JMenuItem about;

private JToolBar toolBar;

private JButton newFileButton;

private JButton openFileButton;

private JButton saveFileButton;

private JButton saveFileAsButton;

private JButton undoButton;

private JButton redoButton;

private JButton copyButton;

private JButton cutButton;

private JButton pasteButton;

private JButton insertImageButton;

private JButton resizeImageButton;

private JComboBox fontBox;

private JSpinner fontSize;

private JToggleButton boldText;

private JToggleButton italicText;

private void createMenuBar(){

menuBar = new JMenuBar();

file = new JMenu("File");

newFile = new JMenuItem("New File");

file.add(newFile);

openFile = new JMenuItem("Open File");

file.add(openFile);

saveFile = new JMenuItem("Save File");

file.add(saveFile);

saveFileAs = new JMenuItem("Save File As");

file.add(saveFileAs);

file.add(new JSeparator());

exit = new JMenuItem("Exit");

file.add(exit);

help = new JMenu("Help");

about = new JMenuItem("About");

help.add(about);

menuBar.add(file);

menuBar.add(help);

setJMenuBar(menuBar);

}

private void createToolbar() {

toolBar = new JToolBar();

newFileButton = makeButton("document-new", "New File");

openFileButton = makeButton("document-open", "Open File");

saveFileButton = makeButton("document-save", "Save File");

saveFileAsButton = makeButton("document-save-as", "Save File As");

toolBar.add(newFileButton);

toolBar.add(openFileButton);

toolBar.add(saveFileButton);

toolBar.add(saveFileAsButton);

toolBar.addSeparator();

undoButton = makeButton("edit-undo", "Undo");

redoButton = makeButton("edit-redo", "Redo");

toolBar.add(undoButton);

toolBar.add(redoButton);

toolBar.addSeparator();

copyButton = makeButton("edit-copy", "Copy");

cutButton = makeButton("edit-cut", "Cut");

pasteButton = makeButton("edit-paste", "Paste");

toolBar.add(copyButton);

toolBar.add(cutButton);

toolBar.add(pasteButton);

toolBar.addSeparator();

insertImageButton = makeButton("image-insert", "Insert Image");

resizeImageButton = makeButton("image-resize", "Resize Image");

toolBar.add(insertImageButton);

toolBar.add(resizeImageButton);

toolBar.addSeparator();

fontBox = new JComboBox();

fillFonts();

fontSize = new JSpinner(new SpinnerNumberModel());

Dimension d = fontBox.getSize();

d.width = 150;

fontBox.setSize(d);

fontSize.setMinimumSize(d);

fontSize.setMaximumSize(d);

d = fontSize.getSize();

d.width = 50;

fontSize.setSize(d);

fontSize.setMinimumSize(d);

fontSize.setValue(15);

boldText = makeToggleButton("format-text-bold", "Bold");

italicText = makeToggleButton("format-text-italic", "Italic");

toolBar.add(fontBox);

toolBar.add(fontSize);

toolBar.add(boldText);

toolBar.add(italicText);

}

private JButton makeButton(String imageName, String altText) {

JButton button = new JButton();

String imgLocation = "resources/icons/" + imageName + ".png";

URL imageURL = Editor.class.getResource(imgLocation);

button.setToolTipText(altText);

if (imageURL != null) {

button.setIcon(new ImageIcon(imageURL, altText));

} else {

button.setText(altText);

System.out.println("Resource not found: " + imgLocation);

}

button.setFocusable(false);

return button;

}

private JToggleButton makeToggleButton(String imageName, String altText) {

JToggleButton button = new JToggleButton();

String imgLocation = "resources/icons/" + imageName + ".png";

URL imageURL = Editor.class.getResource(imgLocation);

button.setToolTipText(altText);

if (imageURL != null) {

button.setIcon(new ImageIcon(imageURL, altText));

} else {

button.setText(altText);

System.out.println("Resource not found: " + imgLocation);

}

button.setFocusable(false);

return button;

}

private void fillFonts(){

GraphicsEnvironment e = GraphicsEnvironment.getLocalGraphicsEnvironment();

Font[] fonts = e.getAllFonts();

for (Font f : fonts) {

fontBox.addItem(f.getFontName());

}

fontBox.setSelectedItem("Source Sans Pro");

}

private void createActions(){

about.addActionListener(actionEvent -> JOptionPane.showMessageDialog(null, "Developed by Dmytro Nechai\nnechaido@gmail.com", "About", JOptionPane.INFORMATION\_MESSAGE));

newFile.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_N, java.awt.event.InputEvent.CTRL\_MASK));

openFile.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_O, java.awt.event.InputEvent.CTRL\_MASK));

saveFile.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_S, java.awt.event.InputEvent.CTRL\_MASK));

saveFileAs.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_S, java.awt.event.InputEvent.CTRL\_MASK | InputEvent.SHIFT\_MASK));

newFile.addActionListener(actionEvent -> editor.newFile());

newFileButton.addActionListener(actionEvent -> editor.newFile());

openFile.addActionListener(actionEvent -> editor.open());

openFileButton.addActionListener(actionEvent -> editor.open());

saveFile.addActionListener(actionEvent -> editor.save());

saveFileButton.addActionListener(actionEvent -> editor.save());

saveFileAs.addActionListener(actionEvent -> editor.saveAs());

saveFileAsButton.addActionListener(actionEvent -> editor.saveAs());

undoButton.addActionListener(actionEvent -> editor.undo());

redoButton.addActionListener(actionEvent -> editor.redo());

copyButton.addActionListener(actionEvent -> editor.copy());

cutButton.addActionListener(actionEvent -> editor.cut());

pasteButton.addActionListener(actionEvent -> editor.paste());

insertImageButton.addActionListener(actionEvent -> editor.insertImage());

resizeImageButton.addActionListener(actionEvent -> editor.resizeImage());

fontBox.addActionListener(actionEvent -> {

editor.updateStyle(getStyle());

editor.grabFocus();

});

fontSize.addChangeListener(changeEvent -> {

editor.updateStyle(getStyle());

editor.grabFocus();

});

boldText.addActionListener(actionEvent -> {

editor.updateStyle(getStyle());

editor.grabFocus();

});

italicText.addActionListener(actionEvent -> {

editor.updateStyle(getStyle());

editor.grabFocus();

});

}

private Font getStyle() {

int type = Font.PLAIN;

if (boldText.isSelected()){

type += Font.BOLD;

}

if (italicText.isSelected()){

type += Font.ITALIC;

}

return new Font(fontBox.getSelectedItem().toString(), type, (Integer) fontSize.getValue());

}

}

package com.nechaido.editor.jeditor;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.commands.CommandHistory;

import com.nechaido.editor.jeditor.commands.simpleDocument.\*;

import com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators.InsertSection;

import com.nechaido.editor.jeditor.commands.simpleDocument.carriageMover.\*;

import com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators.InsertElement;

import com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators.RemoveElement;

import com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators.RemoveLineBreak;

import com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators.SplitLine;

import com.nechaido.editor.jeditor.document.Document;

import com.nechaido.editor.jeditor.document.simpleDocument.\*;

import com.nechaido.editor.jeditor.drawer.Drawer;

import javax.swing.\*;

import javax.xml.soap.Text;

import java.awt.\*;

import java.awt.datatransfer.Clipboard;

import java.awt.datatransfer.DataFlavor;

import java.awt.datatransfer.StringSelection;

import java.awt.datatransfer.UnsupportedFlavorException;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public class JEditor extends JComponent implements KeyListener {

private CommandHistory commandHistory;

private Document document;

private Carriage carriage;

private Clipboard clipboard;

private Context context;

private Carriage selectionStart;

private Carriage selectionEnd;

private File currentFile;

private Style currentStyle;

private boolean selectionMode;

private CommandGenerator commandGenerator;

public JEditor() {

commandHistory = new CommandHistory();

commandGenerator = new CommandGenerator();

clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();

currentStyle = new Style(new Font("Source Sans Pro", Font.PLAIN, 15));

setEnabled(false);

}

@Override

public void paintComponent(Graphics graphics) {

if (isEnabled()){

super.paintComponent(graphics);

((Graphics2D) graphics).setRenderingHint(RenderingHints.KEY\_TEXT\_ANTIALIASING,

RenderingHints.VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB);

super.paintComponent(graphics);

Drawer drawer = document.getDrawer(graphics);

drawer.draw(document);

drawer.drawCursor(carriage);

if (selectionMode){

drawer.drawSelection(selectionStart, selectionEnd);

}

setPreferredSize(drawer.getSize());

}

}

@Override

public void keyTyped(KeyEvent keyEvent) {

}

@Override

public void keyPressed(KeyEvent keyEvent) {

if (keyEvent.isControlDown()){

if (keyEvent.getKeyCode() == KeyEvent.VK\_Z && keyEvent.isShiftDown()){

if (!isEnabled()){

return;

}

redo();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_Z) {

if (!isEnabled()){

return;

}

undo();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_S && keyEvent.isShiftDown()){

if (!isEnabled()){

return;

}

saveAs();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_S) {

if (!isEnabled()){

return;

}

save();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_N) {

newFile();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_O) {

open();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_C) {

if (!isEnabled()){

return;

}

copy();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_X) {

if (!isEnabled()){

return;

}

cut();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_V) {

if (!isEnabled()){

return;

}

paste();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_I && keyEvent.isShiftDown()){

if (!isEnabled()){

return;

}

insertImage();

} else if (keyEvent.getKeyCode() == KeyEvent.VK\_R){

if (!isEnabled()){

return;

}

resizeImage();

}

} else {

if (!isEnabled()){

return;

}

commandHistory.run(commandGenerator.getCommand(keyEvent));

revalidate();

repaint();

}

}

@Override

public void keyReleased(KeyEvent keyEvent) {

}

public void newFile() {

JFileChooser fileChooser = new JFileChooser();

fileChooser.setCurrentDirectory(new File(System.getProperty("user.home")));

int result = fileChooser.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

currentFile = fileChooser.getSelectedFile();

document.addElement(new ElementComposition(currentStyle));

document = new SimpleDocument(context.getCurrentStyle());

commandHistory.empty();

carriage = new Carriage();

selectionStart = new Carriage();

selectionEnd = new Carriage();

context = new Context((SimpleDocument) document, carriage, selectionStart, selectionEnd, currentStyle);

currentStyle = context.getCurrentStyle();

setEnabled(true);

}

}

public void open() {

JFileChooser fileChooser = new JFileChooser();

fileChooser.setCurrentDirectory(new File(System.getProperty("user.home")));

int result = fileChooser.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

document = new SimpleDocument(currentStyle);

currentFile = fileChooser.getSelectedFile();

commandHistory.empty();

carriage = new Carriage();

selectionStart = new Carriage();

selectionEnd = new Carriage();

context = new Context((SimpleDocument) document, carriage, selectionStart, selectionEnd, currentStyle);

currentStyle = context.getCurrentStyle();

new DocumentSerialiser(context).readDocument(currentFile);

setEnabled(true);

}

}

public void save() {

if (!isEnabled()){

return;

}

try {

BufferedWriter bw = new BufferedWriter(new FileWriter(currentFile));

bw.write(new DocumentSerialiser(context).serializeDocument());

bw.close();

commandHistory.empty();

} catch (IOException e) {

e.printStackTrace();

}

}

public void saveAs() {

if (!isEnabled()){

return;

}

JFileChooser fileChooser = new JFileChooser();

fileChooser.setCurrentDirectory(new File(System.getProperty("user.home")));

int result = fileChooser.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

currentFile = fileChooser.getSelectedFile();

save();

}

}

public void undo() {

if (!isEnabled()){

return;

}

commandHistory.undo();

revalidate();

repaint();

}

public void redo() {

if (!isEnabled()){

return;

}

commandHistory.redo();

revalidate();

repaint();

}

public void copy() {

if (!isEnabled()){

return;

}

if (selectionMode){

StringSelection selection = new StringSelection(new DocumentSerialiser(context).serializeSelection());

clipboard.setContents(selection, selection);

}

}

public void cut() {

if (!isEnabled()){

return;

}

if (selectionMode){

StringSelection selection = new StringSelection(new DocumentSerialiser(context).serializeSelection());

clipboard.setContents(selection, selection);

commandHistory.run(new RemoveSection(context));

revalidate();

repaint();

}

}

public void paste() {

if (!isEnabled()){

return;

}

try {

String clip = clipboard.getData(DataFlavor.stringFlavor).toString();

if (selectionMode){

commandHistory.run(new InsertSectionInstead(context, new DocumentSerialiser(context).convert(clip)));

} else {

commandHistory.run(new InsertSection(context, new DocumentSerialiser(context).convert(clip)));

}

revalidate();

repaint();

} catch (UnsupportedFlavorException | IOException e) {

}

}

public void insertImage() {

if (!isEnabled()){

return;

}

JFileChooser fileChooser = new JFileChooser();

fileChooser.setCurrentDirectory(new File(System.getProperty("user.home")));

int result = fileChooser.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

Dimension size = new Dimension();

if (askSize(size)){

if (selectionMode) {

commandHistory.run(new InsertInstead(context,

new Picture(context.getPictureFactory().getPicure(fileChooser.getSelectedFile().getAbsolutePath()), size)));

} else {

commandHistory.run(new InsertElement(context,

new Picture(context.getPictureFactory().getPicure(fileChooser.getSelectedFile().getAbsolutePath()), size)));

}

revalidate();

repaint();

}

}

}

public void resizeImage() {

if (!isEnabled()){

return;

}

if (!selectionMode && document.getElement(carriage.row).getElement(carriage.element) instanceof Picture){

Dimension size = new Dimension();

if (askSize(size)){

((Picture) document.getElement(carriage.row).getElement(carriage.element)).setSize(size);

revalidate();

repaint();

}

}

}

public void updateStyle(Font style) {

if (!isEnabled()){

return;

}

if (selectionMode){

commandHistory.run(new ChangeSelectionStyle(context, context.getStyleFactory().getStyle(style)));

} else {

commandHistory.run(new ChangeCurrentStyle(context, context.getStyleFactory().getStyle(style)));

}

revalidate();

repaint();

}

private boolean askSize(Dimension size) {

JSpinner heightInput = new JSpinner(new SpinnerNumberModel());

JSpinner widthInput = new JSpinner(new SpinnerNumberModel());

heightInput.setValue(100);

widthInput.setValue(100);

Object[] message = {

"Enter height:", heightInput,

"Enter width:", widthInput

};

int option = JOptionPane.showConfirmDialog(null, message, "Enter Size", JOptionPane.OK\_CANCEL\_OPTION);

size.height = (int) heightInput.getValue();

size.width = (int) widthInput.getValue();

return option == JOptionPane.OK\_OPTION;

}

private class CommandGenerator {

private Command getCommand(KeyEvent keyEvent){

int key = keyEvent.getKeyCode();

Command result = null;

if (key == KeyEvent.VK\_KP\_RIGHT || key == KeyEvent.VK\_RIGHT) {

if (keyEvent.isShiftDown()){

result = new MoveSelectionRight(context);

} else {

result = new MoveCarriageRight(context);

}

} else if (key == KeyEvent.VK\_KP\_LEFT || key == KeyEvent.VK\_LEFT) {

if (keyEvent.isShiftDown()){

result = new MoveSelectionLeft(context);

} else {

result = new MoveCarriageLeft(context);

}

} else if (key == KeyEvent.VK\_KP\_DOWN || key == KeyEvent.VK\_DOWN) {

if (keyEvent.isShiftDown()){

result = new MoveSelectionDown(context);

} else {

result = new MoveCarriageDown(context);

}

} else if (key == KeyEvent.VK\_KP\_UP || key == KeyEvent.VK\_UP) {

if (keyEvent.isShiftDown()){

result = new MoveSelectionUp(context);

} else {

result = new MoveCarriageUp(context);

}

} else if (key == KeyEvent.VK\_BACK\_SPACE) {

if (selectionMode){

result = new RemoveSection(context);

} else if (carriage.element == 0){

result = new RemoveLineBreak(context);

} else {

result = new RemoveElement(context);

}

} else if (key == KeyEvent.VK\_ENTER) {

result = new SplitLine(context);

} else if (Character.isAlphabetic(key) || Character.isDigit(key) || Character.isSpaceChar(key)) {

if (selectionMode){

result = new InsertInstead(context, context.getSymbolFactory().getSymbol(keyEvent.getKeyChar()));

} else {

result = new InsertElement(context, context.getSymbolFactory().getSymbol(keyEvent.getKeyChar()));

}

}

if (!selectionMode){

selectionStart.row = carriage.row;

selectionStart.element = carriage.element;

selectionEnd.row = carriage.row;

selectionEnd.element = carriage.element;

}

selectionMode = keyEvent.isShiftDown();

return result;

}

}

}

package com.nechaido.editor.jeditor;

import com.nechaido.editor.jeditor.elementFactories.PictureFactory;

import com.nechaido.editor.jeditor.elementFactories.StyleFactory;

import com.nechaido.editor.jeditor.elementFactories.SymbolFactory;

import com.nechaido.editor.jeditor.document.simpleDocument.SimpleDocument;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

import java.awt.\*;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class Context {

private SimpleDocument document;

private StyleFactory styleFactory;

private SymbolFactory symbolFactory;

private PictureFactory pictureFactory;

private Carriage carriage;

private Carriage selectionStart;

private Carriage selectionEnd;

private Style currentStyle;

public Context(SimpleDocument document, Carriage carriage, Carriage selectionStart, Carriage selectionEnd, Style style) {

this.document = document;

this.carriage = carriage;

this.selectionStart = selectionStart;

this.selectionEnd = selectionEnd;

styleFactory = StyleFactory.getInstance();

symbolFactory = SymbolFactory.getInstance();

pictureFactory = PictureFactory.getInstance();

styleFactory.add(new Style(style.getFont()));

currentStyle = styleFactory.getStyle(style.getFont());

}

public SimpleDocument getDocument() {

return document;

}

public void setDocument(SimpleDocument document) {

this.document = document;

}

public SymbolFactory getSymbolFactory() {

return symbolFactory;

}

public PictureFactory getPictureFactory() {

return pictureFactory;

}

public Carriage getCarriage() {

return carriage;

}

public Carriage getSelectionStart() {

return selectionStart;

}

public Carriage getSelectionEnd() {

return selectionEnd;

}

public Style getCurrentStyle() {

return currentStyle;

}

public void setCurrentStyle(Style style) {

currentStyle = style;

}

public StyleFactory getStyleFactory() {

return styleFactory;

}

}

package com.nechaido.editor.jeditor;

/\*\*

\* Created by nechaido on 5/12/16.

\*/

public class Carriage {

public int row;

public int element;

public Carriage() {

row = 0;

element = 0;

}

}

package com.nechaido.editor.jeditor.elementFactories;

import javax.imageio.ImageIO;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import java.util.Objects;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class PictureFactory {

private static PictureFactory instance;

private HashMap<String, BufferedImage> pictures;

private PictureFactory() {

pictures = new HashMap<>();

}

public static PictureFactory getInstance() {

if (instance == null) {

instance = new PictureFactory();

}

return instance;

}

public BufferedImage getPicure(String path) {

if (!pictures.containsKey(path)) {

try {

BufferedImage bufferedImage = ImageIO.read(new File(path));

pictures.put(path, bufferedImage);

} catch (IOException e) {

return null;

}

}

return pictures.get(path);

}

public String getPath(BufferedImage image) {

for (Map.Entry<String, BufferedImage> entry : pictures.entrySet()){

if (Objects.equals(image, entry.getValue())){

return entry.getKey();

}

}

return null;

}

}

package com.nechaido.editor.jeditor.elementFactories;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

import java.awt.\*;

import java.util.HashMap;

/\*\*

\* Created by Nechai Dmytro nechaido@gmail.com on 5/14/16.

\*/

public class StyleFactory {

private static StyleFactory instance;

private HashMap<Font, Style> styles;

private StyleFactory() {

styles = new HashMap<>();

}

public static StyleFactory getInstance() {

if (instance == null) {

instance = new StyleFactory();

}

return instance;

}

public Style getStyle(Font font) {

if (!styles.containsKey(font)) {

styles.put(font, new Style(font));

}

return styles.get(font);

}

public void add(Style style) {

styles.put(style.getFont(), style);

}

}

package com.nechaido.editor.jeditor.elementFactories;

import com.nechaido.editor.jeditor.document.simpleDocument.Symbol;

import java.util.HashMap;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class SymbolFactory {

private static SymbolFactory instance;

private HashMap<Character, Symbol> symbols;

private SymbolFactory() {

symbols = new HashMap<>();

}

public static SymbolFactory getInstance() {

if (instance == null) {

instance = new SymbolFactory();

}

return instance;

}

public Symbol getSymbol(char character) {

if (!symbols.containsKey(character)) {

symbols.put(character, new Symbol(character));

}

return symbols.get(character);

}

}

package com.nechaido.editor.jeditor.drawer;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

import java.awt.\*;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public interface VisualElement {

Dimension getSize();

void drawBy(Drawer drawer);

}

package com.nechaido.editor.jeditor.drawer;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.document.Document;

import java.awt.\*;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public abstract class Drawer {

protected Graphics2D graphics2D;

protected Drawer(Graphics graphics) {

graphics2D = (Graphics2D) graphics;

graphics2D.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

graphics2D.setRenderingHint(RenderingHints.KEY\_TEXT\_ANTIALIASING, RenderingHints.VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB);

graphics2D.setRenderingHint(RenderingHints.KEY\_TEXT\_LCD\_CONTRAST, 100);

graphics.setColor(new Color(34, 34, 34));

}

public abstract void draw(Document document);

public abstract void draw(VisualElement element);

public abstract void drawCursor(Carriage carriage);

public abstract void drawSelection(Carriage selectionStart, Carriage selectionEnd);

public abstract Dimension getSize();

}

package com.nechaido.editor.jeditor.drawer.simpleDocument;

import com.nechaido.editor.jeditor.document.simpleDocument.SimpleDocument;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

import com.nechaido.editor.jeditor.drawer.Drawer;

import com.nechaido.editor.jeditor.drawer.VisualElement;

import java.awt.\*;

import java.util.Collection;

import java.util.LinkedList;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class VisualDocument implements VisualElement {

private Dimension size;

private LinkedList<Line> lines;

public VisualDocument(SimpleDocument simpleDocument) {

lines = new LinkedList<>();

size = new Dimension(0, 0);

for (int i = 0; i < simpleDocument.length(); i++) {

Line newLine = new Line(simpleDocument.getElement(i));

lines.add(newLine);

if (size.width < newLine.getSize().width) {

size.width = newLine.getSize().width;

}

size.height += newLine.getSize().height;

}

}

public Collection<? extends VisualElement> getLines() {

return lines;

}

@Override

public Dimension getSize() {

return size;

}

@Override

public void drawBy(Drawer drawer) {

((SimpleDocumentDrawer) drawer).draw(this);

}

}

package com.nechaido.editor.jeditor.drawer.simpleDocument;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.document.Document;

import com.nechaido.editor.jeditor.document.simpleDocument.SimpleDocument;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

import com.nechaido.editor.jeditor.drawer.Drawer;

import com.nechaido.editor.jeditor.drawer.VisualElement;

import java.awt.\*;

import java.util.ArrayList;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public class SimpleDocumentDrawer extends Drawer {

VisualDocument visualDocument;

private Style style;

private SimpleDocument document;

private Dimension currentOffset;

private int padding;

public SimpleDocumentDrawer(Graphics graphics, SimpleDocument document) {

super(graphics);

this.document = document;

style = document.getStyle();

padding = 10;

currentOffset = new Dimension(0, 0);

}

public void draw(SimpleDocument simpleDocument) {

visualDocument = new VisualDocument(simpleDocument);

visualDocument.drawBy(this);

}

@Override

public void draw(Document document) {

if (document instanceof SimpleDocument) {

draw((SimpleDocument) document);

}

}

@Override

public void draw(VisualElement element) {

}

@Override

public void drawCursor(Carriage carriage) {

int row = carriage.row;

int element = carriage.element;

ArrayList<? extends VisualElement> lines = new ArrayList<>(visualDocument.getLines());

int y = padding;

int x = padding;

int width = style.getCharWidth(' ');

int height = style.getFont().getSize();

for (int i = 0; i <= row; i++) {

y += lines.get(i).getSize().height;

}

if (((Line) lines.get(row)).amountOfElements() > 0) {

int[] partialWidth = ((Line) lines.get(row)).getPartialWidth();

int elementCount = partialWidth.length - 1;

x += partialWidth[element];

if (element != elementCount) {

width = partialWidth[element + 1] - partialWidth[element];

}

height = ((Line) lines.get(row)).getSize().height;

}

Color oldColor = graphics2D.getColor();

Color newColor = new Color(oldColor.getRed(), oldColor.getGreen(), oldColor.getGreen(), 128);

graphics2D.setColor(newColor);

graphics2D.fillRect(x, y - height, width, height + 3);

graphics2D.setColor(oldColor);

}

@Override

public void drawSelection(Carriage selectionStartExternal, Carriage selectionEndExternal) {

ArrayList<VisualElement> lines = new ArrayList<>(visualDocument.getLines());

int y = padding;

int x = padding;

Carriage selectionStart = new Carriage();

Carriage selectionEnd = new Carriage();

if (selectionEndExternal.row > selectionStartExternal.row){

selectionEnd = selectionEndExternal;

selectionStart = selectionStartExternal;

} else if (selectionEndExternal.row < selectionStartExternal.row){

selectionEnd = selectionStartExternal;

selectionStart = selectionEndExternal;

} else if (selectionEndExternal.element > selectionStartExternal.element){

selectionEnd = selectionEndExternal;

selectionStart = selectionStartExternal;

} else {

selectionEnd = selectionStartExternal;

selectionStart = selectionEndExternal;

}

for (int i = 0; i < selectionStart.row; i++) {

y += lines.get(i).getSize().height;

}

for (int i = selectionStart.row; i <= selectionEnd.row; i++) {

x = padding;

int start = ((i == selectionStart.row) ? selectionStart.element : 0);

int[] partialWidth = ((Line) lines.get(i)).getPartialWidth();

int end = ((i == selectionEnd.row) ? selectionEnd.element : partialWidth.length);

int width = 0;

if (end == partialWidth.length){

width += style.getCharWidth(' ');

if (end != 0){

end--;

}

}

width += partialWidth[end] - partialWidth[start];

x += partialWidth[start];

int height = lines.get(i).getSize().height;

Color oldColor = graphics2D.getColor();

Color newColor = new Color(0, 0, 255, 128);

graphics2D.setColor(newColor);

graphics2D.fillRect(x, y, width, height);

graphics2D.setColor(oldColor);

y += height;

}

}

@Override

public Dimension getSize() {

return new Dimension(visualDocument.getSize().width + 2 \* padding, visualDocument.getSize().height + 2 \* padding);

}

public void draw(VisualDocument visualDocument) {

currentOffset.height = padding;

currentOffset.width = padding;

for (VisualElement element : visualDocument.getLines()) {

element.drawBy(this);

}

}

public void draw(Line line) {

currentOffset.height += line.getSize().height;

for (VisualElement element : line.getContents()) {

element.drawBy(this);

}

currentOffset.width = padding;

}

public void draw(VisualPicture visualPicture) {

graphics2D.drawImage(visualPicture.getImage(),

currentOffset.width, currentOffset.height - visualPicture.getSize().height,

visualPicture.getSize().width, visualPicture.getSize().height, null);

currentOffset.width += visualPicture.getSize().width;

}

public void draw(Word word) {

graphics2D.setFont(word.getStyle().getFont());

graphics2D.drawString(word.toString(), currentOffset.width, currentOffset.height);

currentOffset.width += word.getSize().width;

}

}

package com.nechaido.editor.jeditor.drawer.simpleDocument;

import com.nechaido.editor.jeditor.document.Element;

import com.nechaido.editor.jeditor.document.simpleDocument.ElementComposition;

import com.nechaido.editor.jeditor.document.simpleDocument.Picture;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

import com.nechaido.editor.jeditor.document.simpleDocument.Symbol;

import com.nechaido.editor.jeditor.drawer.Drawer;

import com.nechaido.editor.jeditor.drawer.VisualElement;

import java.awt.\*;

import java.util.Collection;

import java.util.LinkedList;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class Line implements VisualElement {

private Dimension size;

private LinkedList<VisualElement> list;

// private Style style;

private int[] partialWidth;

public Line(Element element) {

size = new Dimension();

list = new LinkedList<>();

size.height = ((ElementComposition)element).getStyle().getFont().getSize();

int i = 0;

partialWidth = new int[element.length() + 1];

int currentIndex = 0;

partialWidth[currentIndex++] = 0;

while (i < element.length()) {

Element currentElement = element.getElement(i);

if (currentElement instanceof Picture) {

Picture picture = (Picture) currentElement;

list.add(new VisualPicture(picture));

if (picture.getSize().getHeight() > size.getHeight()) {

size.height = (int) picture.getSize().getHeight();

}

int picWidth = picture.getSize().width;

size.width += picWidth;

partialWidth[currentIndex++] = partialWidth[currentIndex - 2] + picWidth;

++i;

} else if (currentElement instanceof Symbol) {

Style curStyle = ((ElementComposition) element).getStyle(i);

Word currentWord = new Word(curStyle);

currentWord.getSize().height = curStyle.getFont().getSize();

if (curStyle.getFont().getSize() > size.height){

size.height = curStyle.getFont().getSize();

}

while (currentElement instanceof Symbol && curStyle == ((ElementComposition) element).getStyle(i)) {

char ch = ((Symbol) currentElement).getCharacter();

currentWord.append(ch);

int chaWidth = curStyle.getCharWidth(ch);

size.width += chaWidth;

currentWord.getSize().width += chaWidth;

partialWidth[currentIndex++] = partialWidth[currentIndex - 2] + chaWidth;

if (i == element.length() - 1) {

++i;

break;

}

currentElement = element.getElement(++i);

}

list.add(currentWord);

} else {

++i;

}

}

}

@Override

public Dimension getSize() {

return size;

}

@Override

public void drawBy(Drawer drawer) {

((SimpleDocumentDrawer) drawer).draw(this);

}

public Collection<? extends VisualElement> getContents() {

return list;

}

public int[] getPartialWidth() {

return partialWidth;

}

public int amountOfElements() {

return list.size();

}

}

package com.nechaido.editor.jeditor.drawer.simpleDocument;

import com.nechaido.editor.jeditor.document.simpleDocument.Picture;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

import com.nechaido.editor.jeditor.drawer.Drawer;

import com.nechaido.editor.jeditor.drawer.VisualElement;

import java.awt.\*;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class VisualPicture implements VisualElement {

private Picture picture;

public VisualPicture(Picture picture) {

this.picture = picture;

}

public Image getImage() {

return picture.getImage();

}

@Override

public Dimension getSize() {

return picture.getSize();

}

@Override

public void drawBy(Drawer drawer) {

((SimpleDocumentDrawer) drawer).draw(this);

}

}

package com.nechaido.editor.jeditor.drawer.simpleDocument;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

import com.nechaido.editor.jeditor.drawer.Drawer;

import com.nechaido.editor.jeditor.drawer.VisualElement;

import java.awt.\*;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class Word implements VisualElement {

private StringBuffer stringBuffer;

private Style style;

private Dimension size;

public Word(Style style) {

stringBuffer = new StringBuffer();

this.style = style;

size = new Dimension();

}

public void append(char character) {

stringBuffer.append(character);

}

@Override

public Dimension getSize() {

return size;

}

public Style getStyle() {

return style;

}

@Override

public void drawBy(Drawer drawer) {

((SimpleDocumentDrawer) drawer).draw(this);

}

@Override

public String toString() {

return stringBuffer.toString();

}

}

package com.nechaido.editor.jeditor.document;

import java.util.Collection;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public interface Element {

boolean isMajor();

void addElement(Element element);

void addElement(int index, Element element);

void addAllElements(Collection<? extends Element> childElements);

void addAllElements(Element element);

void addAllElements(int index, Element element);

void removeElement(int index);

void removeElements(int from, int to);

void removeElements(int from);

Element getElement(int index);

Collection<? extends Element> getElements();

Collection<? extends Element> getElements(int from, int to);

Element getSubElement(int from, int to);

void setElement(int index, Element element);

int length();

void empty();

}

package com.nechaido.editor.jeditor.document;

import com.nechaido.editor.jeditor.drawer.Drawer;

import java.awt.\*;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public interface Document extends Element {

Drawer getDrawer(Graphics graphics);

}

package com.nechaido.editor.jeditor.document.simpleDocument;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.document.Document;

import com.nechaido.editor.jeditor.document.Element;

import java.awt.\*;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\* Created by Nechai Dmytro nechaido@gmail.com on 5/13/16.

\*/

public class DocumentSerialiser {

private Context context;

public DocumentSerialiser(Context context){

this.context = context;

}

public void readDocument(File file){

try {

BufferedReader br = new BufferedReader(new FileReader(file));

String s;

Document document = context.getDocument();

document.empty();

while ((s = br.readLine()) != null){

document.addElement(getLine(s));

}

if (document.length() == 0){

document.addElement(new ElementComposition(context.getCurrentStyle()));

}

} catch (java.io.IOException e) {

e.printStackTrace();

}

}

public String serializeDocument(){

StringBuffer result = new StringBuffer();

for (int i = 0; i < context.getDocument().length(); i++) {

int start = 0;

int end = context.getDocument().getElement(i).length();

for (int j = start; j < end; j++){

Element element = context.getDocument().getElement(i).getElement(j);

if (element instanceof Symbol){

result.append(((Symbol) element).getCharacter());

} else if (element instanceof Picture){

if (((Picture) element).getImage() != null){

String path = context.getPictureFactory().getPath(((Picture) element).getImage());

int width = ((Picture) element).getSize().width;

int height = ((Picture) element).getSize().height;

result.append("<img src=\"" + path + "\" width=\"" + width + "\" height=\"" + height + "\">");

}

}

}

result.append('\n');

}

return result.toString();

}

public String serializeSelection(){

StringBuffer result = new StringBuffer();

Carriage selectionStart = new Carriage();

Carriage selectionEnd = new Carriage();

if (context.getSelectionEnd().row > context.getSelectionStart().row){

selectionEnd = context.getSelectionEnd();

selectionStart = context.getSelectionStart();

} else if (context.getSelectionEnd().row < context.getSelectionStart().row){

selectionEnd = context.getSelectionStart();

selectionStart = context.getSelectionEnd();

} else if (context.getSelectionEnd().element > context.getSelectionStart().element){

selectionEnd = context.getSelectionEnd();

selectionStart = context.getSelectionStart();

} else {

selectionEnd = context.getSelectionStart();

selectionStart = context.getSelectionEnd();

}

for (int i = selectionStart.row; i <= selectionEnd.row; i++) {

int start = ((i == selectionStart.row) ? selectionStart.element : 0);

int end = ((i == selectionEnd.row) ? selectionEnd.element : context.getDocument().getElement(i).length());

for (int j = start; j < end; j++){

Element element = context.getDocument().getElement(i).getElement(j);

if (element instanceof Symbol){

result.append(((Symbol) element).getCharacter());

} else if (element instanceof Picture){

if (((Picture) element).getImage() != null){

String path = context.getPictureFactory().getPath(((Picture) element).getImage());

int width = ((Picture) element).getSize().width;

int height = ((Picture) element).getSize().height;

result.append("<img src=\"" + path + "\" width=\"" + width + "\" height=\"" + height + "\">");

}

}

}

if (end == context.getDocument().getElement(i).length()){

result.append('\n');

}

}

return result.toString();

}

public Element[] convert(String input){

String[] lines = input.split("\n");

Element[] result = new Element[lines.length];

for (int i = 0; i < lines.length; i++) {

result[i] = getLine(lines[i]);

}

return result;

}

public Element getLine(String input){

ElementComposition row = new ElementComposition(context.getCurrentStyle());

for (int i = 0; i < input.length(); i++) {

if (input.charAt(i) != '<'){

row.addElement(context.getSymbolFactory().getSymbol(input.charAt(i)));

} else {

int j = 0;

for (j = i; j < input.length() && input.charAt(j) != '>'; j++) {

}

if (j == input.length()){

row.addElement(context.getSymbolFactory().getSymbol(input.charAt(i)));

} else {

String sub = input.substring(i + 1, j);

Pattern pattern = Pattern.compile("img\\s+src=\"([\\w./]\*)\"\\s+width=\"(\\d+)\"\\s+height=\"(\\d+)\"");

Matcher matcher = pattern.matcher(sub);

String path = null;

int width = -1;

int height = -1;

if (matcher.find()){

path = matcher.group(1);

width = Integer.parseInt(matcher.group(2));

height = width = Integer.parseInt(matcher.group(3));

}

if (path != null && width != -1){

row.addElement(new Picture(context.getPictureFactory().getPicure(path), new Dimension(width, height)));

i = j;

} else {

row.addElement(context.getSymbolFactory().getSymbol(input.charAt(i)));

}

}

}

}

return row;

}

}

package com.nechaido.editor.jeditor.document.simpleDocument;

import com.nechaido.editor.jeditor.document.Element;

import java.util.Collection;

import java.util.LinkedList;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class ElementComposition implements Element {

private LinkedList<Element> contents;

private LinkedList<Style> styles;

private Style style;

public ElementComposition(Style style) {

contents = new LinkedList<>();

styles = new LinkedList<>();

this.style = style;

}

public ElementComposition(Collection<Element> elements, Style style) {

contents = new LinkedList<>(elements);

styles = new LinkedList<>();

this.style = style;

for (int i = 0; i < elements.size(); i++) {

styles.add(style);

}

}

@Override

public boolean isMajor() {

return true;

}

@Override

public void addElement(Element element) {

contents.add(element);

if (styles.size() > 0){

styles.add(styles.get(styles.size() - 1));

} else {

styles.add(style);

}

}

@Override

public void addElement(int index, Element element) {

contents.add(index, element);

if (index > 0){

styles.add(index, styles.get(index - 1));

} else {

styles.add(style);

}

}

@Override

public void addAllElements(Collection<? extends Element> childElements) {

contents.addAll(childElements);

for (int i = 0; i < childElements.size(); i++) {

if (styles.size() > 0){

styles.add(styles.get(styles.size() - 1));

} else {

styles.add(style);

}

}

}

@Override

public void addAllElements(Element element) {

Collection< ? extends Element> elements = element.getElements();

contents.addAll(elements);

for (int i = 0; i < elements.size(); i++) {

if (styles.size() > 0){

styles.add(styles.get(styles.size() - 1));

} else {

styles.add(style);

}

}

}

@Override

public void addAllElements(int index, Element element) {

Collection< ? extends Element> elements = element.getElements();

contents.addAll(index, elements);

for (int i = 0; i < elements.size(); i++) {

if (index > 0){

styles.add(index, styles.get(index - 1));

} else {

styles.add(style);

}

}

}

@Override

public void removeElement(int index) {

contents.remove(index);

styles.remove(index);

}

@Override

public void removeElements(int from, int to) {

contents.subList(from, to).clear();

styles.subList(from, to).clear();

}

@Override

public void removeElements(int from) {

contents.subList(from, contents.size()).clear();

styles.subList(from, styles.size()).clear();

}

@Override

public Element getElement(int index) {

return contents.get(index);

}

@Override

public Collection<? extends Element> getElements() {

return contents;

}

@Override

public Collection<? extends Element> getElements(int from, int to) {

return contents.subList(from, to + 1);

}

@Override

public Element getSubElement(int from, int to) {

return new ElementComposition(contents.subList(from, to + 1), style);

}

@Override

public void setElement(int index, Element element) {

contents.set(index, element);

}

@Override

public int length() {

return contents.size();

}

@Override

public void empty() {

contents.clear();

}

public Style getStyle() {

return style;

}

public Style getStyle(int i) {

if (styles.size() == 0){

return style;

}

return styles.get(i);

}

public void setStyle(int i, Style style) {

if (styles.size() == 0){

this.style = style;

}

styles.set(i, style);

}

public void setStyle(Style style) {

style = style;

}

}

package com.nechaido.editor.jeditor.document.simpleDocument;

import java.awt.\*;

import java.awt.image.BufferedImage;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public class Picture extends PrimitiveElement {

private BufferedImage image;

private Dimension size;

public Picture(BufferedImage image, Dimension size) {

this.image = image;

this.size = new Dimension(size);

}

@Override

public boolean isMajor() {

return true;

}

public Dimension getSize() {

return size;

}

public void setSize(Dimension size) {

this.size = new Dimension(size);

}

public BufferedImage getImage() {

return image;

}

}

package com.nechaido.editor.jeditor.document.simpleDocument;

import com.nechaido.editor.jeditor.document.Element;

import java.util.Collection;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public abstract class PrimitiveElement implements Element {

@Override

public void addElement(Element element) {

}

@Override

public void addElement(int index, Element element) {

}

@Override

public void addAllElements(Collection<? extends Element> childElements) {

}

@Override

public void addAllElements(Element element) {

}

@Override

public void addAllElements(int index, Element element) {

}

@Override

public void removeElement(int index) {

}

@Override

public void removeElements(int from, int to) {

}

@Override

public void removeElements(int from) {

}

@Override

public Element getElement(int index) {

return null;

}

@Override

public Collection<? extends Element> getElements() {

return null;

}

@Override

public Collection<? extends Element> getElements(int from, int to) {

return null;

}

@Override

public Element getSubElement(int from, int to) {

return null;

}

@Override

public void setElement(int index, Element element) { }

@Override

public int length() {

return 0;

}

@Override

public void empty() { }

}

package com.nechaido.editor.jeditor.document.simpleDocument;

import com.nechaido.editor.jeditor.document.Document;

import com.nechaido.editor.jeditor.drawer.Drawer;

import com.nechaido.editor.jeditor.drawer.simpleDocument.SimpleDocumentDrawer;

import java.awt.\*;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public class SimpleDocument extends ElementComposition implements Document {

public SimpleDocument(Style style) {

super(style);

}

@Override

public Drawer getDrawer(Graphics graphics) {

return new SimpleDocumentDrawer(graphics, this);

}

}

package com.nechaido.editor.jeditor.document.simpleDocument;

import java.awt.\*;

/\*\*

\* Created by nechaido on 5/8/16.

\*/

public class Style {

private FontMetrics fontMetrics;

public Style(Font font) {

Canvas canvas = new Canvas();

fontMetrics = canvas.getFontMetrics(font);

}

public Font getFont() {

return fontMetrics.getFont();

}

public Dimension getCharSize(char c) {

return new Dimension(fontMetrics.charWidth(c), fontMetrics.getHeight());

}

public int getCharHeight(char c) {

return fontMetrics.getHeight();

}

public int getCharWidth(char c) {

return fontMetrics.charWidth(c);

}

}

package com.nechaido.editor.jeditor.document.simpleDocument;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public class Symbol extends PrimitiveElement {

private char character;

public Symbol(char character) {

this.character = character;

}

public char getCharacter() {

return character;

}

@Override

public boolean isMajor() {

return Character.isSpaceChar(character);

}

}

package com.nechaido.editor.jeditor.commands;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public interface Command {

enum Type{INSERT, REMOVE, BREAKLINE, CONCATLINE, STYLECHANGE, MOVECARRIEGE};

void execute();

void unExecute();

boolean isMajor();

Type type();

}

package com.nechaido.editor.jeditor.commands;

import java.util.Stack;

/\*\*

\* Created by nechaido on 5/7/16.

\*/

public class CommandHistory {

private Stack<Command> undoHistory;

private Stack<Command> redoHistory;

public CommandHistory() {

undoHistory = new Stack<>();

redoHistory = new Stack<>();

}

public void undo() {

if (undoHistory.isEmpty()){

return;

}

Command command;

Command.Type type = undoHistory.peek().type();

while (!undoHistory.isEmpty() && type == undoHistory.peek().type()) {

command = undoHistory.pop();

redoHistory.push(command);

command.unExecute();

if (command.isMajor()) {

break;

}

}

}

public void redo() {

if (redoHistory.isEmpty()){

return;

}

Command command;

Command.Type type = redoHistory.peek().type();

while (!redoHistory.isEmpty() && type == redoHistory.peek().type()) {

command = redoHistory.pop();

undoHistory.push(command);

command.execute();

if (command.isMajor()) {

break;

}

}

}

public void run(Command command) {

if (command == null){

return;

}

if (!redoHistory.isEmpty()) {

redoHistory = new Stack<>();

}

undoHistory.push(command);

command.execute();

}

public void empty() {

undoHistory = new Stack<>();

redoHistory = new Stack<>();

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.document.Element;

import com.nechaido.editor.jeditor.document.simpleDocument.ElementComposition;

import java.util.ArrayList;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class RemoveSection extends AbstractSimpleDocumentCommand {

private Carriage previousCarriage;

private Carriage previousEnd;

private Carriage previousStart;

private Carriage selectionStart;

private Carriage selectionEnd;

private ArrayList<Element> deleted;

private int amountOfElements;

public RemoveSection(Context context) {

super(context, true);

previousCarriage = new Carriage();

previousEnd = new Carriage();

previousStart = new Carriage();

previousCarriage.row = context.getCarriage().row;

previousCarriage.element = context.getCarriage().element;

previousEnd.row = context.getSelectionEnd().row;

previousEnd.element = context.getSelectionEnd().element;

previousStart.row = context.getSelectionStart().row;

previousStart.element = context.getSelectionStart().element;

selectionStart = new Carriage();

selectionEnd = new Carriage();

if (context.getSelectionEnd().row > context.getSelectionStart().row){

selectionEnd = context.getSelectionEnd();

selectionStart = context.getSelectionStart();

} else if (context.getSelectionEnd().row < context.getSelectionStart().row){

selectionEnd = context.getSelectionStart();

selectionStart = context.getSelectionEnd();

} else if (context.getSelectionEnd().element > context.getSelectionStart().element){

selectionEnd = context.getSelectionEnd();

selectionStart = context.getSelectionStart();

} else {

selectionEnd = context.getSelectionStart();

selectionStart = context.getSelectionEnd();

}

deleted = new ArrayList<>(selectionEnd.row - selectionStart.row);

for (int i = selectionStart.row; i <= selectionEnd.row; i++) {

Element currentRow = context.getDocument().getElement(i);

int start = ((i == selectionStart.row) ? selectionStart.element : 0);

int end = ((i == selectionEnd.row) ? selectionEnd.element : currentRow.length());

end--;

if (end < 0){

deleted.add(new ElementComposition(context.getCurrentStyle()));

} else {

deleted.add(currentRow.getSubElement(start, end));

}

}

amountOfElements = 0;

}

@Override

public void execute() {

if (selectionStart.row == selectionEnd.row){

context.getDocument().getElement(selectionStart.row).removeElements(

selectionStart.element, selectionEnd.element);

} else {

context.getDocument().getElement(selectionStart.row).removeElements(selectionStart.element);

for (int i = selectionStart.row + 1; i < selectionEnd.row; i++) {

context.getDocument().removeElement(selectionStart.row + 1);

}

amountOfElements = context.getDocument().getElement(selectionStart.row).length();

for (int i = 0; i < selectionEnd.element; i++) {

context.getDocument().getElement(selectionStart.row + 1).removeElement(0);

}

context.getDocument().getElement(selectionStart.row).addAllElements(

context.getDocument().getElement(selectionStart.row + 1));

context.getDocument().removeElement(selectionStart.row + 1);

}

context.getCarriage().row = selectionStart.row;

context.getCarriage().element = selectionStart.element;

context.getSelectionEnd().row = selectionStart.row;

context.getSelectionEnd().element = selectionStart.element;

context.getSelectionStart().row = selectionStart.row;

context.getSelectionStart().element = selectionStart.element;

}

@Override

public void unExecute() {

if (previousStart.row == previousEnd.row) {

context.getDocument().getElement(selectionStart.row).addAllElements(selectionStart.element, deleted.get(0));

} else {

Element newRow = context.getDocument().getElement(selectionStart.row).getSubElement(amountOfElements, context.getDocument().getElement(selectionStart.row).length() - 1);

context.getDocument().getElement(selectionStart.row).removeElements(amountOfElements);

context.getDocument().getElement(selectionStart.row).addAllElements(deleted.get(0));

context.getDocument().addElement(selectionStart.row + 1 , newRow);

for (int i = 1; i < deleted.size() - 1; i++) {

context.getDocument().addElement(selectionStart.row + i, deleted.get(i));

}

context.getDocument().getElement(selectionStart.row + deleted.size() - 1).addAllElements(0, deleted.get(deleted.size() - 1));

}

context.getCarriage().row = previousCarriage.row;

context.getCarriage().element = previousCarriage.element;

context.getSelectionStart().row = previousStart.row;

context.getSelectionStart().element = previousStart.element;

context.getSelectionEnd().row = previousEnd.row;

context.getSelectionEnd().element = previousEnd.element;

}

@Override

public Type type() {

return Type.REMOVE;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators.InsertSection;

import com.nechaido.editor.jeditor.document.Element;

import java.util.ArrayList;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class InsertSectionInstead extends AbstractSimpleDocumentCommand {

private RemoveSection removeSection;

private InsertSection insertSection;

public InsertSectionInstead(Context context, Element[] inserts) {

super(context, true);

removeSection = new RemoveSection(context);

insertSection = new InsertSection(context, inserts);

}

@Override

public void execute() {

removeSection.execute();

insertSection.execute();

}

@Override

public void unExecute() {

insertSection.unExecute();

removeSection.unExecute();

}

@Override

public Type type() {

return Type.INSERT;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators.InsertElement;

import com.nechaido.editor.jeditor.document.Element;

/\*\*

\* Created by Nechai Dmytro nechaido@gmail.com on 5/13/16.

\*/

public class InsertInstead extends AbstractSimpleDocumentCommand {

private InsertElement insertElement;

private RemoveSection removeSection;

public InsertInstead(Context context, Element element) {

super(context, true);

removeSection = new RemoveSection(context);

insertElement = new InsertElement(context, element);

}

@Override

public void execute() {

removeSection.execute();

insertElement.execute();

}

@Override

public void unExecute() {

insertElement.unExecute();

removeSection.unExecute();

}

@Override

public Type type() {

return Type.INSERT;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.document.simpleDocument.ElementComposition;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

/\*\*

\* Created by Nechai Dmytro nechaido@gmail.com on 5/14/16.

\*/

public class ChangeSelectionStyle extends AbstractSimpleDocumentCommand {

private Style[][] oldStyles;

private Style newStyle;

private Carriage selectionStart;

private Carriage selectionEnd;

public ChangeSelectionStyle(Context context, Style style) {

super(context, true);

newStyle = style;

int size = context.getSelectionEnd().row - context.getSelectionStart().row + 1;

selectionStart = new Carriage();

selectionEnd = new Carriage();

if (context.getSelectionEnd().row > context.getSelectionStart().row){

selectionEnd = context.getSelectionEnd();

selectionStart = context.getSelectionStart();

} else if (context.getSelectionEnd().row < context.getSelectionStart().row){

selectionEnd = context.getSelectionStart();

selectionStart = context.getSelectionEnd();

} else if (context.getSelectionEnd().element > context.getSelectionStart().element){

selectionEnd = context.getSelectionEnd();

selectionStart = context.getSelectionStart();

} else {

selectionEnd = context.getSelectionStart();

selectionStart = context.getSelectionEnd();

}

oldStyles = new Style[size][];

for (int i = context.getSelectionStart().row; i <= context.getSelectionEnd().row ; i++) {

ElementComposition currentRow = (ElementComposition) context.getDocument().getElement(i);

int start = ((i == selectionStart.row) ? selectionStart.element : 0);

int end = ((i == selectionEnd.row) ? selectionEnd.element : currentRow.length() - 1);

int length = end - start + 1;

oldStyles[i] = new Style[length];

for (int j = 0; j < length; j++) {

oldStyles[i][j] = currentRow.getStyle(start+j);

}

}

}

@Override

public void execute() {

for (int i = context.getSelectionStart().row; i <= context.getSelectionEnd().row ; i++) {

ElementComposition currentRow = (ElementComposition) context.getDocument().getElement(i);

int start = ((i == selectionStart.row) ? selectionStart.element : 0);

int end = ((i == selectionEnd.row) ? selectionEnd.element : currentRow.length() - 1);

int length = end - start + 1;

for (int j = 0; j < length; j++) {

currentRow.setStyle(start+j, newStyle);

}

}

}

@Override

public void unExecute() {

for (int i = context.getSelectionStart().row; i <= context.getSelectionEnd().row ; i++) {

ElementComposition currentRow = (ElementComposition) context.getDocument().getElement(i);

int start = ((i == selectionStart.row) ? selectionStart.element : 0);

int end = ((i == selectionEnd.row) ? selectionEnd.element : currentRow.length() - 1);

int length = end - start + 1;

for (int j = 0; j < length; j++) {

currentRow.setStyle(start+j, oldStyles[i][j]);

}

}

}

@Override

public Type type() {

return null;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.document.simpleDocument.Style;

/\*\*

\* Created by Nechai Dmytro nechaido@gmail.com on 5/14/16.

\*/

public class ChangeCurrentStyle extends AbstractSimpleDocumentCommand {

private Style oldStyle;

private Style newStyle;

public ChangeCurrentStyle(Context context, Style style) {

super(context, false);

newStyle = style;

oldStyle = context.getCurrentStyle();

}

@Override

public void execute() {

context.setCurrentStyle(newStyle);

context.getDocument().setStyle(newStyle);

}

@Override

public void unExecute() {

context.setCurrentStyle(oldStyle);

context.getDocument().setStyle(oldStyle);

}

@Override

public Type type() {

return Type.STYLECHANGE;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public abstract class AbstractSimpleDocumentCommand implements Command {

protected Context context;

protected boolean major;

protected AbstractSimpleDocumentCommand(Context context) {

this.context = context;

}

protected AbstractSimpleDocumentCommand(Context context, boolean major) {

this.context = context;

this.major = major;

}

@Override

public boolean isMajor() {

return major;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

import com.nechaido.editor.jeditor.document.Element;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class InsertElement extends AbstractSimpleDocumentCommand {

private Element element;

public InsertElement(Context context, Element element) {

super(context, element.isMajor());

this.element = element;

}

@Override

public void execute() {

context.getDocument().getElement(context.getCarriage().row).addElement(

context.getCarriage().element, element);

context.getCarriage().element++;

}

@Override

public void unExecute() {

context.getCarriage().element--;

context.getDocument().getElement(context.getCarriage().row).removeElement(

context.getCarriage().element);

}

@Override

public Type type() {

return Type.INSERT;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

import com.nechaido.editor.jeditor.document.Element;

import com.nechaido.editor.jeditor.document.simpleDocument.ElementComposition;

import java.util.ArrayList;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class InsertSection extends AbstractSimpleDocumentCommand {

private Carriage previousCarriage;

private Carriage previousEnd;

private Carriage previousStart;

private Element[] inserts;

public InsertSection(Context context, Element[] inserts) {

super(context, true);

previousCarriage = new Carriage();

previousEnd = new Carriage();

previousStart = new Carriage();

previousCarriage.row = context.getCarriage().row;

previousCarriage.element = context.getCarriage().element;

previousEnd.row = context.getSelectionEnd().row;

previousEnd.element = context.getSelectionEnd().element;

previousStart.row = context.getSelectionStart().row;

previousStart.element = context.getSelectionStart().element;

this.inserts = inserts;

}

@Override

public void execute() {

if (inserts.length == 1){

context.getDocument().getElement(context.getCarriage().row).addAllElements(context.getCarriage().element, inserts[0]);

context.getCarriage().element += inserts[0].length();

} else {

Element newRow = context.getDocument().getElement(previousCarriage.row).getSubElement(

previousCarriage.element, context.getDocument().getElement(previousCarriage.row).length() - 1);

context.getDocument().getElement(previousCarriage.row).removeElements(previousCarriage.element);

for (int i = 0; i < inserts.length - 1; i++) {

context.getDocument().getElement(previousCarriage.row + i).addAllElements(inserts[i]);

context.getDocument().addElement(previousCarriage.row + i + 1, new ElementComposition(context.getCurrentStyle()));

}

context.getDocument().getElement(previousCarriage.row + inserts.length - 1).addAllElements(inserts[inserts.length - 1]);

context.getDocument().getElement(previousCarriage.row + inserts.length - 1).addAllElements(newRow);

context.getCarriage().row = previousCarriage.row + inserts.length - 1;

context.getCarriage().element = inserts[inserts.length - 1].length() - 1;

}

}

@Override

public void unExecute() {

if (inserts.length == 1){

context.getDocument().getElement(context.getCarriage().row).removeElements(

previousCarriage.element, previousCarriage.element + inserts[0].length());

} else {

context.getDocument().getElement(previousCarriage.row).removeElements(previousCarriage.element);

for (int i = 1; i < inserts.length - 1; i++) {

context.getDocument().removeElement(previousCarriage.row + 1);

}

for (int i = 0; i < inserts[inserts.length - 1].length(); i++) {

context.getDocument().getElement(previousCarriage.row + 1).removeElement(0);

}

context.getDocument().getElement(previousCarriage.row).addAllElements(

context.getDocument().getElement(previousCarriage.row + 1));

context.getDocument().removeElement(previousCarriage.row + 1);

}

context.getCarriage().row = previousCarriage.row;

context.getCarriage().element = previousCarriage.element;

context.getSelectionStart().row = previousStart.row;

context.getSelectionStart().element = previousStart.element;

context.getSelectionEnd().row = previousEnd.row;

context.getSelectionEnd().element = previousEnd.element;

}

@Override

public Type type() {

return Type.INSERT;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

import com.nechaido.editor.jeditor.document.Element;

/\*\*

\* Created by nechaido on 5/9/16.

\*/

public class RemoveElement extends AbstractSimpleDocumentCommand {

private Element element;

public RemoveElement(Context context) {

super(context);

}

@Override

public void execute() {

context.getCarriage().element--;

element = context.getDocument().getElement(context.getCarriage().row).getElement(

context.getCarriage().element);

context.getDocument().getElement(

context.getCarriage().row).removeElement(context.getCarriage().element);

major = element.isMajor();

}

@Override

public void unExecute() {

context.getDocument().getElement(context.getCarriage().row).addElement(context.getCarriage().element, element);

context.getCarriage().element++;

}

@Override

public Type type() {

return Type.REMOVE;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

import com.nechaido.editor.jeditor.document.Element;

import com.nechaido.editor.jeditor.document.simpleDocument.ElementComposition;

import java.util.Collection;

/\*\*

\* Created by nechaido on 5/12/16.

\*/

public class RemoveLineBreak extends AbstractSimpleDocumentCommand {

private int elementAmount;

private boolean executed;

public RemoveLineBreak(Context context) {

super(context, true);

executed = false;

}

@Override

public void execute() {

if (context.getCarriage().row == 0) {

return;

}

elementAmount = context.getDocument().getElement(context.getCarriage().row - 1).length();

Element secondPart = context.getDocument().getElement(context.getCarriage().row);

context.getDocument().getElement(context.getCarriage().row - 1).addAllElements(secondPart);

context.getDocument().removeElement(context.getCarriage().row);

context.getCarriage().row--;

context.getCarriage().element = elementAmount;

executed = true;

}

@Override

public void unExecute() {

if (!executed) {

return;

}

Element firstRow = context.getDocument().getElement(context.getCarriage().row);

Element newRow = firstRow.getSubElement(elementAmount, firstRow.length() - 1);

context.getDocument().addElement(context.getCarriage().row + 1, newRow);

firstRow.removeElements(elementAmount);

context.getCarriage().row++;

context.getCarriage().element = 0;

}

@Override

public Type type() {

return null;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.noSelectionManipulators;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

import com.nechaido.editor.jeditor.document.Element;

/\*\*

\* Created by nechaido on 5/12/16.

\*/

public class SplitLine extends AbstractSimpleDocumentCommand {

public SplitLine(Context context) {

super(context, true);

}

@Override

public void execute() {

Element currentRow = context.getDocument().getElement(context.getCarriage().row);

Element element = currentRow.getSubElement(context.getCarriage().element,

currentRow.length() - 1);

currentRow.removeElements(context.getCarriage().element, currentRow.length());

context.getDocument().addElement(context.getCarriage().row + 1, element);

context.getCarriage().row++;

context.getCarriage().element = 0;

}

@Override

public void unExecute() {

Element secondPart = context.getDocument().getElement(context.getCarriage().row);

context.getCarriage().row--;

context.getCarriage().element = context.getDocument().getElement(context.getCarriage().row).length();

context.getDocument().getElement(context.getCarriage().row).addAllElements(secondPart);

context.getDocument().removeElement(context.getCarriage().row + 1);

}

@Override

public Type type() {

return Type.BREAKLINE;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.carriageMover;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

/\*\*

\* Created by nechaido on 5/12/16.

\*/

public class MoveCarriageDown extends AbstractSimpleDocumentCommand{

public MoveCarriageDown(Context context) {

super(context, false);

}

@Override

public void execute() {

if (context.getCarriage().row < context.getDocument().length() - 1) {

context.getCarriage().row++;

int lastSymbol = context.getDocument().getElement(context.getCarriage().row).length();

if (context.getCarriage().element > lastSymbol) {

context.getCarriage().element = lastSymbol;

}

}

}

@Override

public void unExecute() {

if (context.getCarriage().row > 0) {

context.getCarriage().row--;

}

int lastSymbol = context.getDocument().getElement(context.getCarriage().row).length();

if (context.getCarriage().element > lastSymbol) {

context.getCarriage().element = lastSymbol;

}

}

@Override

public Type type() {

return Type.MOVECARRIEGE;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.carriageMover;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

/\*\*

\* Created by nechaido on 5/12/16.

\*/

public class MoveCarriageLeft extends AbstractSimpleDocumentCommand {

public MoveCarriageLeft(Context context){

super(context, false);

}

@Override

public void execute() {

if (context.getCarriage().element > 0) {

--context.getCarriage().element;

} else if (context.getCarriage().row > 0) {

--context.getCarriage().row;

context.getCarriage().element =

context.getDocument().getElement(context.getCarriage().row).length();

}

}

@Override

public void unExecute() {

if (context.getCarriage().element < context.getDocument().getElement(context.getCarriage().row).length()) {

context.getCarriage().element++;

} else if (context.getCarriage().row < context.getDocument().length() - 1) {

context.getCarriage().element = 0;

context.getCarriage().row++;

}

}

@Override

public Type type() {

return Type.MOVECARRIEGE;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.carriageMover;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

/\*\*

\* Created by nechaido on 5/12/16.

\*/

public class MoveCarriageRight extends AbstractSimpleDocumentCommand {

public MoveCarriageRight(Context context){

super(context);

}

@Override

public void execute() {

if (context.getCarriage().element < context.getDocument().getElement(context.getCarriage().row).length()) {

context.getCarriage().element++;

} else if (context.getCarriage().row < context.getDocument().length() - 1) {

context.getCarriage().element = 0;

context.getCarriage().row++;

}

}

@Override

public void unExecute() {

if (context.getCarriage().element > 0) {

--context.getCarriage().element;

} else if (context.getCarriage().row > 0) {

--context.getCarriage().row;

context.getCarriage().element =

context.getDocument().getElement(context.getCarriage().row).length();

}

}

@Override

public Command.Type type() {

return Command.Type.MOVECARRIEGE;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.carriageMover;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

/\*\*

\* Created by nechaido on 5/12/16.

\*/

public class MoveCarriageUp extends AbstractSimpleDocumentCommand {

public MoveCarriageUp(Context context){

super(context, false);

}

@Override

public void execute() {

if (context.getCarriage().row > 0) {

context.getCarriage().row--;

}

int lastSymbol = context.getDocument().getElement(context.getCarriage().row).length();

if (context.getCarriage().element > lastSymbol) {

context.getCarriage().element = lastSymbol;

}

}

@Override

public void unExecute() {

if (context.getCarriage().row < context.getDocument().length() - 1) {

context.getCarriage().row++;

int lastSymbol = context.getDocument().getElement(context.getCarriage().row).length();

if (context.getCarriage().element > lastSymbol) {

context.getCarriage().element = lastSymbol;

}

}

}

@Override

public Type type() {

return Type.MOVECARRIEGE;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.carriageMover;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.Context;

import com.nechaido.editor.jeditor.commands.Command;

import com.nechaido.editor.jeditor.commands.simpleDocument.AbstractSimpleDocumentCommand;

/\*\*

\* Created by Nechai Dmytro nechaido@gmail.com on 5/12/16.

\*/

public class MoveSelectionDown extends MoveCarriageDown {

private Carriage previousEnd;

public MoveSelectionDown(Context context) {

super(context);

previousEnd = new Carriage();

previousEnd.row = context.getSelectionEnd().row;

previousEnd.element = context.getSelectionEnd().element;

}

@Override

public void execute() {

super.execute();

context.getSelectionEnd().row = context.getCarriage().row;

context.getSelectionEnd().element = context.getCarriage().element;

}

@Override

public void unExecute() {

super.unExecute();

context.getSelectionEnd().row = previousEnd.row;

context.getSelectionEnd().element = previousEnd.element;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.carriageMover;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.Context;

/\*\*

\* Created by Nechai Dmytro nechaido@gmail.com on 5/12/16.

\*/

public class MoveSelectionLeft extends MoveCarriageLeft {

private Carriage previousEnd;

public MoveSelectionLeft(Context context) {

super(context);

previousEnd = new Carriage();

previousEnd.row = context.getSelectionEnd().row;

previousEnd.element = context.getSelectionEnd().element;

}

@Override

public void execute() {

super.execute();

context.getSelectionEnd().row = context.getCarriage().row;

context.getSelectionEnd().element = context.getCarriage().element;

}

@Override

public void unExecute() {

super.unExecute();

context.getSelectionEnd().row = previousEnd.row;

context.getSelectionEnd().element = previousEnd.element;

}

}

package com.nechaido.editor.jeditor.commands.simpleDocument.carriageMover;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.Context;

/\*\*

\* Created by Nechai Dmytro nechaido@gmail.com on 5/12/16.

\*/

public class MoveSelectionRight extends MoveCarriageRight {

private Carriage previousEnd;

public MoveSelectionRight(Context context) {

super(context);

previousEnd = new Carriage();

previousEnd.row = context.getSelectionEnd().row;

previousEnd.element = context.getSelectionEnd().element;

}

@Override

public void execute() {

super.execute();

context.getSelectionEnd().row = context.getCarriage().row;

context.getSelectionEnd().element = context.getCarriage().element;

}

@Override

public void unExecute() {

super.unExecute();

context.getSelectionEnd().row = previousEnd.row;

context.getSelectionEnd().element = previousEnd.element;

}

// @Override

// public void execute() {

// super.execute();

// if (context.getSelectionStart().row > context.getCarriage().row){

// context.getSelectionEnd().row = context.getSelectionStart().row ;

// context.getSelectionEnd().element = context.getSelectionStart().element;

// context.getSelectionStart().row = context.getCarriage().row;

// context.getSelectionStart().element = context.getCarriage().element;

// } else if (context.getSelectionStart().row == context.getCarriage().row){

// if (context.getSelectionStart().element > context.getCarriage().element){

// context.getSelectionStart().element = context.getCarriage().element;

// context.getSelectionEnd().row = previousStart.row;

// context.getSelectionEnd().element = previousStart.element;

// } else {

// context.getSelectionEnd().row = context.getCarriage().row;

// context.getSelectionEnd().element = context.getCarriage().element;

// }

// } else {

// context.getSelectionEnd().row = context.getCarriage().row ;

// context.getSelectionEnd().element = context.getCarriage().element;

// }

// }

// @Override

// public void unExecute() {

// super.unExecute();

// context.getSelectionStart().row = previousStart.row;

// context.getSelectionStart().element = previousStart.element;

// context.getSelectionEnd().row = previousEnd.row;

// context.getSelectionEnd().element = previousEnd.element;

// }

}

package com.nechaido.editor.jeditor.commands.simpleDocument.carriageMover;

import com.nechaido.editor.jeditor.Carriage;

import com.nechaido.editor.jeditor.Context;

/\*\*

\* Created by Nechai Dmytro nechaido@gmail.com on 5/12/16.

\*/

public class MoveSelectionUp extends MoveCarriageUp{

private Carriage previousEnd;

public MoveSelectionUp(Context context) {

super(context);

previousEnd = new Carriage();

previousEnd.row = context.getSelectionEnd().row;

previousEnd.element = context.getSelectionEnd().element;

}

@Override

public void execute() {

super.execute();

context.getSelectionEnd().row = context.getCarriage().row;

context.getSelectionEnd().element = context.getCarriage().element;

}

@Override

public void unExecute() {

super.unExecute();

context.getSelectionEnd().row = previousEnd.row;

context.getSelectionEnd().element = previousEnd.element;

}

}