[Лабораторная работа №5. Паттерн «Команда» 1](#_Toc451203982)

[Обязательные задания 1](#_Toc451203983)

[Задание 1 – Редактор – 100 баллов 1](#_Toc451203984)

[Бонус в 200 баллов за покрытие классов тестами 5](#_Toc451203985)

# Лабораторная работа №5. Паттерн «Команда»

На оценку «удовлетворительно» необходимо набрать 80 баллов.

На оценку «хорошо» необходимо набрать 200 баллов.

На оценку «отлично» необходимо набрать 280 баллов.

**Дополнительные задания принимаются только после выполнения обязательных заданий.**

## Обязательные задания

### Задание 1 – Редактор – 100 баллов

Разработайте консольное приложение, позволяющее пользователю создать документ, содержащий блоки текста и изображений, и сохранить его в HTML-формате.

Команды редактирования поступают со стандартного потока ввода.

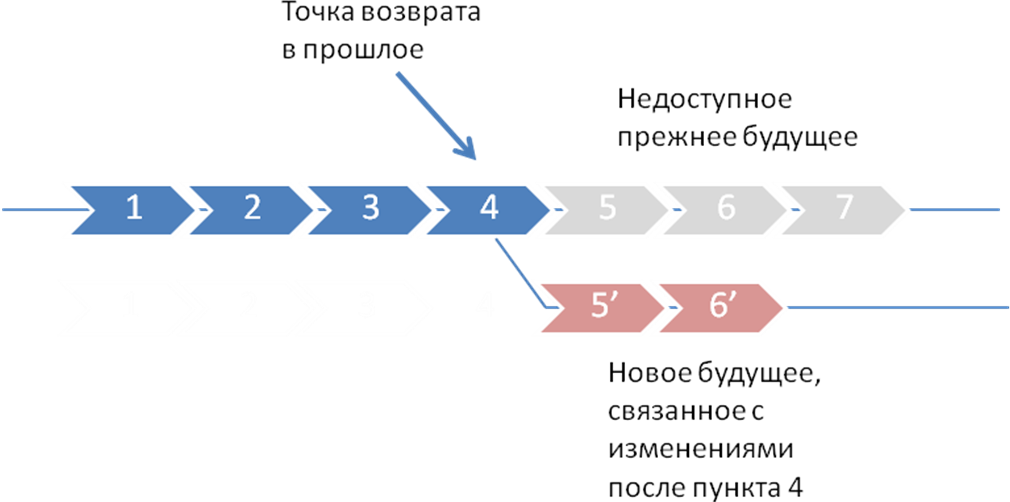
Перечень поддерживаемых команд:

* **InsertParagraph <позиция>|end <текст параграфа>**
  + Вставляет в указанную позицию документа параграф с указанным текстом. В качестве позиции вставки можно указать либо порядковый номер блока, либо **end** для вставки параграфа в конец.
  + В случае, если номер позиции превышает количество элементов в документе, пользователю выдается сообщение об ошибке, а команда игнорируется.
* **InsertImage <позиция>|end <ширина> <высота> <путь к файлу изображения>**
  + Вставляет в указанную позицию документа изображение, находящееся по указанному пути. При вставке указывается ширина и высота, что позволяет разместить изображение с указанным масштабом.
  + При вставке файл изображения должен копироваться в подкаталог images каталога, в котором содержится рабочая версия документа. Имя для изображения должно быть сгенерировано автоматически, а расширение остаться оригинальным.
    - Такой подход позволяет обеспечить целостность изображений после их вставки в документ, даже если исходный файл изображения будет удален или заменен другим.
  + Допустимые размеры изображения находятся в диапазоне от 1 до 10000 пикселей включительно по каждой из координатных осей.
  + Если номер позиции превышает количество элементов в документе, пользователю выдается сообщение об ошибке, а команда игнорируется
* **SetTitle <заголовок документа>**
  + Изменяет заголовок документа
* **List**
  + Выводит название и список элементов документа в стандартный поток вывода. Пример вывода
    - Title: Это заголовок документа
    - 0. Paragraph: Это самый первый параграф
    - 1. Image: 400 300 images/img1.png
    - 2. Paragraph: Это параграф, идущий следом за изображением.
* **ReplaceText <позиция> <текст параграфа>**
  + Заменяет текст в параграфе, находящемся в указанной позиции документа. Если в данной позиции не находится параграф, выдается сообщение об ошибке, а команда игнорируется.
  + Если номер позиции больше или равен количеству элементов в документе, пользователю выдается сообщение об ошибке, а команда игнорируется.
* **ResizeImage <позиция> <ширина> <высота>**
  + Изменяет размер изображения, находящегося в указанной позиции документа. Если в данной позиции не находится изображение, выдается сообщение об ошибке
* **DeleteItem <позиция>**
  + Удаляет элемент документа, находящийся в указанной позиции. Если указана недопустимая позиция документа, команда игнорируется, а пользователю выводится сообщение об ошибке.
* **Help**
  + выводит справку о доступных командах редактирования и их аргументах
* **Undo**
  + Отменяет действие ранее введенной команды редактирования, возвращая документ в предыдущее состояния.
  + Если операция отмены недоступна, выводится сообщение об ошибке.
* **Redo**
  + Выполняет ранее отмененную команду редактирования, возвращая документ в состояние, отменяе действие команды Undo.
  + Если операция Redo недоступна, выводится сообщение об ошибке.
* **Save <путь>**
  + Сохраняет текущее состояние документа в файл формата html с изображениями. Файлы изображений должны сохраняться в подкаталоге images внутри каталога с html-файлом. В самом html-файле должны задаваться относительные пути к изображениям.

Требования к работе механизма отмены/повтора команд редактирования.

Отмененные действия могут быть восстановлены при помощи команды Redo. Доступные для операции Redo действия удаляются, если после отмены пользователь выполнил операции изменяющие историю команд.

Следующий рисунок рассматривает ситуацию, при которой после выполнения 7 операций редактирования пользователь отменил последние 3 операции (5, 6 и 7), а затем выполнил новые операции 5’ и 6’, которые сделали невозможным возврат в состояние 5, 6 и 7. В результате доступными для отмены (и повторного выполнения) остались команды: 1, 2, 3, 4, 5’, 6’.



Глубина истории команд ограничена 10 командами. При ее заполнении самые старые команды удаляются.

Удаление команд, с которыми связано добавление или удаление ресурсов, таких как файлы, должно приводить к физическому удалению данных ресурсов. Например, если действие №5 выполняло вставку изображения cat.png, действие №6 – удаление изображения dog.png, а команда 5’ - изменение размеров изображения dog.png, то:

* Отмена команды 6 должна приводить к восстановлению изображения dog.png, а точнее, снятии отметки о его удалении.
* Отмена команды 5 должна приводить к пометке изображения cat.png для удаления. Удалять файл изображения нельзя, чтобы можно было восстановить его при помощи redo.
* Перед выполнением команды 5’ необходимо удалить команды 7, 6 и 5 (отмененные команды должны удаляться в обратном порядке, а выполненные – в прямой последовательности). При этом удаление отмененной команды 6 (удаление изображения) не должно производить никаких действий над файлом (т.к. он был восстановлен), а удаление отмененной команды 5 (вставка изображения cat.png) должно удалять файл помеченный на удаление cat.png из рабочего каталога, поскольку удаляется ветка истории команд, в котором этот файл создавался.

Можно сформулировать следующие правила реализации команд управления ресурсами:

* Добавление ресурса
  + Подготовительные операции: ресурс должен быть скопирован в документ для последующего управления
  + Отмена должна приводить к пометке ресурса для удаления
    - Удаление отмененной команды должно физически удалять ресурс
  + Повтор должен приводить к снятию пометки ресурса на удаление
    - Удаление исполненной команды не должно затрагивать ресурс
* Команда удаления ресурса
  + Выполнение команды должно приводить к пометке ресурса на удаление, а не физическому удалению ресурса, чтобы можно было отменить это действие
  + Отмена должна снимать пометку на удаление
    - Удаление отмененного действия не должно выполнять никаких манипуляций ресурсом
  + Повтор отмененного действия должен помечать ресурс на удаление
    - Удаление выполненного действия должно физически удалять ресурс, т.к. возможности отменить действие уже не существует.
* Команда модификации ресурса
  + Проще всего реализовать через создание модифицированной копии ресурса и переключения ссылки на модифицированную копию

Работа с документом, а также отмена и повтор операций должны осуществляться с использованием интерфейса IDocument:

/\*

Интерфес документа

\*/

class IDocument

{

public:

// Вставляет параграф текста в указанную позицию (сдвигая последующие элементы)

// Если параметр position не указан, вставка происходит в конец документа

virtual shared\_ptr<IParagraph> InsertParagraph(const string& text,

optional<size\_t> position = none) = 0;

// Вставляет изображение в указанную позицию (сдвигая последующие элементы)

// Параметр path задает путь к вставляемому изображению

// При вставке изображение должно копироваться в подкаталог images

// под автоматически сгенерированным именем

virtual shared\_ptr<IImage> InsertImage(const Path& path, int width, int height,

optional<size\_t> position = none) = 0;

// Возвращает количество элементов в документе

virtual size\_t GetItemsCount()const = 0;

// Доступ к элементам изображения

virtual CConstDocumentItem GetItem(size\_t index)const = 0;

virtual CDocumentItem GetItem(size\_t index) = 0;

// Удаляет элемент из документа

virtual void DeleteItem(size\_t index) = 0;

// Возвращает заголовок документа

virtual string GetTitle()const = 0;

// Изменяет заголовок документа

virtual void SetTitle(const string & title) = 0;

// Сообщает о доступности операции Undo

virtual bool CanUndo()const = 0;

// Отменяет команду редактирования

virtual void Undo() = 0;

// Сообщает о доступности операции Redo

virtual bool CanRedo()const = 0;

// Выполняет отмененную команду редактирования

virtual void Redo() = 0;

// Сохраняет документ в формате html. Изображения сохраняются в подкаталог images.

// Пути к изображениям указываются относительно пути к сохраняемому HTML файлу

virtual void Save(const Path& path)const = 0;

virtual ~IDocument() = default;

};

Операции, выполняющие модификацию состояния документа должны записываться в историю команд внутри документа.

Манипулирование изображениями и параграфами осуществляется при помощи интерфейсов IImage и IParagraph соответственно (эти операции также должны заноситься в историю команд).

/\* Параграф текста\*/

class IParagraph

{

public:

virtual string GetText()const = 0;

virtual void SetText(const string& text) = 0;

virtual ~IParagraph() = default;

};

/\*

Изображение

\*/

class IImage

{

public:

// Возвращает путь относительно каталога документа

virtual Path GetPath()const = 0;

// Ширина изображения в пикселях

virtual int GetWidth()const = 0;

// Высота изображения в пикселях

virtual int GetHeight()const = 0;

// Изменяет размер изображения

virtual void Resize(int width, int height) = 0;

virtual ~IImage() = default;

};

Операции модификации параграфов и изображений также должны записываться в общую историю команд.

Доступ к элементам документа осуществляется при помощи классов ConstDocumentItem и DocumentItem:

/\*

Неизменяемый элемент документа

\*/

class CConstDocumentItem

{

public:

// Возвращает указатель на константное изображение, либо nullptr,

// если элемент не является изображением

shared\_ptr<const IImage> GetImage()const;

// Возвращает указатель на константный параграф, либо nullptr, если элемент не является параграфом

shared\_ptr<const IParagraph> GetParagraph()const;

virtual ~CConstDocumentItem() = default;

};

/\*

Элемент документа. Позволяет получить доступ к изображению или параграфу

\*/

class CDocumentItem : public CConstDocumentItem

{

public:

// Возвращает указатель на изображение, либо nullptr, если элемент не является изображением

shared\_ptr<IImage> GetImage();

// Возвращает указатель на параграф, либо nullptr, если элемент не является параграфом

shared\_ptr<IParagraph> GetParagraph();

};

#### Бонус в 200 баллов за покрытие классов тестами

Бонус начисляется за разработку приложения в TDD/BDD стиле и покрытие модульными тестами всех классов приложения, включая классы, ведущие обмен