Министерство образования и науки Российской Федерации
Санкт-Петербургский Политехнический Университет Петра Великого
—
Институт компьютерных наук и кибербезопасности

ЛАБОРАТОРНАЯ РАБОТА №3

«Разработка текстового редактора»

по дисциплине «Объектно-ориентированное программирование»

Санкт-Петербург

1 ЦЕЛЬ РАБОТЫ

Целью работы – ознакомление с основными шаблоны проектирования программного обеспечения, получение навыка разработки приложений на базе архитектурного шаблона проектирования – Model View Controller.

2 ЗАДАЧИ

В рамках лабораторной работы необходимо разработать VIM-подобный текстовый редактор на языке программирования Python 3, удовлетворяющий следующим требованиям:

1. Программа должна быть разработана с использованием архитектурного шаблона проектирования MVC.
2. На этапе проектирования программы должна быть разработана UML-диаграмма классов.
3. Программа должна иметь TUI – Text User Interface (текстовый пользовательский интерфейс). Для разработки TUI необходимо использовать встроенный Python-модуль curses с применением шаблона проектирования «адаптер» для интеграции модуля в основную логикой программы.
    - интерфейс должен поддерживать отображение курсора и строки состояния (в качестве примера интерфейса следует рассмотреть оригинальную программу VIM);
    - строка состояния должна отображать: текущий режим работы редактора (редактирование, ввод команды, окно помощи, поиск и прочие), имя файла, номер текущей строки и общее количество строк.
4. Программа должна работать с 1-байтовой кодировкой текста.
5. Для работы программы с текстом необходимо использовать Python-обёртку над собственным классом MyString, разработанную в рамках выполнения первой лабораторной работы. В случае необходимости расширения либо корректировки поведения класса-обёртки MyString необходимо использовать шаблон проектирования «декоратор».

6. Программа должна иметь следующие режимы работы:
   A. Режим навигации и редактирования – основной режим работы, из этого режима осуществляется переход в «Режим ввода» и «Режим поиска». При старте программа должна начинать свою работу в данном режиме.
   B. Режим ввода команд. Список поддерживаемых команд представлен в приложение 1:
      — активация режима ввода команд - команда <:>;
      — выход из режима ввода команд в основной режим работы – команда <ESC>.
   C. Режим ввода. Ввод текста на строке с курсором.
      — активация режима ввода текста перед курсором – с помощью команд из раздела «Ввод текста» приложения 1;
      — активация режима ввода текста после курсора - команда <o>;
      — очистить текущую строку и начать ввод текста с начала строки - команда <S>;
      — выход из режима ввода в основной режим работы – команда <ESC>.
   D. Режим поиска. Поиск от курсора до конца или начала документа:
      — активация режима поиска до конца документа - команда </>;
      — активация режима поиска до начала документа - команда <?>;
      — выход из режима поиска в основной режим работы – команда <ESC>.
7. Проект должен содержать Unit-тесты для базового функционала программы, реализованный с помощью фреймворка pytest.

3 ЭТАПЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Процесс выполнения лабораторной работы включает в себя следующие этапы:

1. Первый этап (*занятие №1*):

— обзор предметной области: текстовый редактор VIM, TUI-интерфейсы, библиотека curses;

— подготовка каркаса проекта;

— разработка адаптера для библиотеки curses.

2. Второй этап (*занятие №2*):

— проектирование архитектуры приложения на базе шаблона MVC с использованием UML-диаграммы классов;

— разработка каркаса приложения с использованием шаблона MVC;

— разработка базовой логики хранения текста с использованием класса-обёртки MyString (Model);

— разработка временных компонентов заглушек для подсистем обработки команд и представления результатов работы (Controller, View).

3. Третий этап (*занятие №3*):

— разработка базовой логики работы с текстом (основные методы Model);

— разработка Unit-тестов для базового функционала обработки текста.

4. Четвёртый этап (*занятие №4*):

— разработка подсистемы обработки команд пользователя в различных режимах (Controller);

— разработка подсистемы представления результатов работы программы (View).

4 ДОПОЛНИТЕЛЬНЫЕ ЗАДАНИЯ

4.1 Шаблон проектирования «команда»

В рамках выполнения дополнительного задания необходимо:

1. Реализовать команду «дополнительный набор команд 1» из приложения 1, выполняющую нумерацию строк текста.
2. Использовать для реализации шаблон проектирования «команда» (использование шаблона также должно быть адаптировано для команд из базового набора).

В отчёте необходимо привести:

1. Теоретическую справку об использованном шаблоне проектирования.
2. Актуализированную UML-диаграмму классов.

4.2 Шаблон проектирования «декоратор»

В рамках выполнения дополнительного задания необходимо:

1. Реализовать команды «дополнительный набор команд 2» из приложения 1. Язык программирования для реализации механизма подсветки синтаксиса может быть выбран по усмотрению студента.
2. Использовать для реализации шаблон проектирования «декоратор».

В отчёте необходимо привести:

1. Теоретическую справку об использованном шаблоне проектирования.
2. Актуализированную UML-диаграмму классов.

4.3 Паттерн проектирования «одиночка»

В рамках выполнения дополнительного задания необходимо:

1. Добавить функционал хранения истории команд с использованием шаблона проектирования «одиночка».

2. Реализовать команды «дополнительный набор команд 3» из приложения 1.

В отчёте необходимо привести:

1. Теоретическую справку об использованном шаблоне проектирования.
2. Актуализированную UML-диаграмму классов.

4.4 MVC-фреймворк разработки приложений с TUI-интерфейсом

В рамках выполнения дополнительного задания необходимо:

1. Спроектировать фреймворк, реализующий основные механизмы взаимодействия между компонентами архитектурного паттерна «MVC», не зависящий от деталей реализации функционала текстового редактора.
2. Фреймворк должен предоставлять интерфейс создания приложения (регистрация controller, view, model) на базе шаблона проектирования «строитель».
3. Функционал текстового редактора должен быть реализован с использованием спроектированного фреймворка.
4. Реализовать демонстрационную TUI-программу отличную от текстового редактора для демонстрации применения спроектированного фреймворка.

В отчёте необходимо привести:

1. Теоретическую справку об использованном шаблоне проектирования «строитель».
2. Актуализированную UML-диаграмму классов.

5 ТРЕБОВАНИЯ К ОТЧЕТУ

Отчет должен включать следующие пункты:

1. Цель работы.
2. Задачи с указанием выполненных дополнительных заданий.
3. Ход работы – краткое описание этапов выполнения работы:

a. описание спроектированной архитектуры приложение с приведением UML-диаграммы классов;
b. описание особенностей реализации программы;
c. описание разработанных Unit-тестов для базового функционала программы;
d. примеры тестирования программы с помощью Unit-тестов;
e. примеры ручного тестирования программы;
f. примеры ручного тестирования программы на текстовых файлах большого объёма (более 512 КБ).

4. Выводы.

Приложение 1 – Список команд

	Базовый набор команд
	Дополнительный набор команд 1
	Дополнительный набор команд 2
	Дополнительный набор команд 3

Команда	**Действие**
Навигация и редактирование	
RIGHT	Перемещение курсора вправо на 1 позицию
LEFT	Перемещение курсора вправо на 1 позицию
UP	Перемещение курсора вверх на 1 строку (положение курсора при этом должно сохраняться, либо перемещаться в конец строки в случае, если строка короче)
DOWN	Перемещение курсора вниз на 1 строку (положение курсора при этом должно сохраняться, либо перемещаться в конец строки в случае, если строка короче)
^ (или 0, нуль)	Перемещение курсора в начало строки
$	Перемещение курсора в конец строки
w	Перемещение курсора в конец слова справа от курсора (если курсор стоит в конце слова, то перемещаем курсор на конец следующего слова)
b	Перемещение курсора в начало слова слева от курсора (если курсор стоит в начале слова, то перемещаем курсор на начало предыдущего слова)
gg	Перейти в начало файла
G	Перейти в конец файла
*N*G	Перейти на строку с номером *N*. Например 10G – переход на 10-ю строку
PG_UP	Перейти на экран вверх
PG_DOWN	Перейти на экран вниз
x	Удалить символ после курсора
diw	Удалить слово под курсором, включая пробел справа.
dd	Вырезать текущую строку
yy	Копировать текущую строку
yw	Копировать слово под курсором

p	Вставить после курсора
u	Отмена последней команды
U	Отмена всех изменений в текущей строке
Поиск	
/*text*<CR>	Поиск строки *text* от курсора до конца файла. Если строка найдена – переместить курсор в начало строки
?*text*<CR>	Поиск строки *text* от курсора до начала файла. Если строка найдена – переместить курсор в начало строки.
n	Повторить поиск
N	Повторить поиск в обратном направлении
Ввод текста	
i	Ввод текста перед курсором
I	Перейти в начало строки и начать ввод текста
A	Перейти в конец строки и начать ввод текста
S	Удалить содержимое строки и начать ввод текста
r	Заменить один символ под курсором
Команды для Режима команд	
o *filename*	Открыть файл filename
x	Записать в текущий файл и выйти
w	Записать в текущий файл
w *filename*	Записать в filename
q	Выйти. Если файл был изменён, то выход возможен только через q!
q!	Выйти без сохранения
wq!	Записать в текущий файл и выйти
e!	Отменить все изменения
number	Переход на строку number
set num	Включить/Отключить нумерацию строк
sy	Включить подсветку синтаксиса
h	Вывести справку по командам