Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 4

з дисципліни «Теорія розробки ПЗ»

Тема: ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Варіант №3

Виконав:
студент групи ІА-13
Губенко Єгор Олександрович

Київ, 2023

Тема: ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY».

Завдання.

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми.

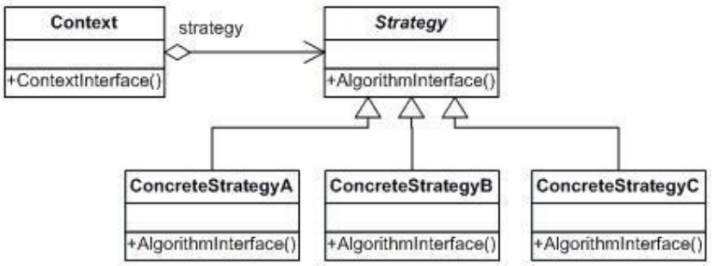
Варіант.

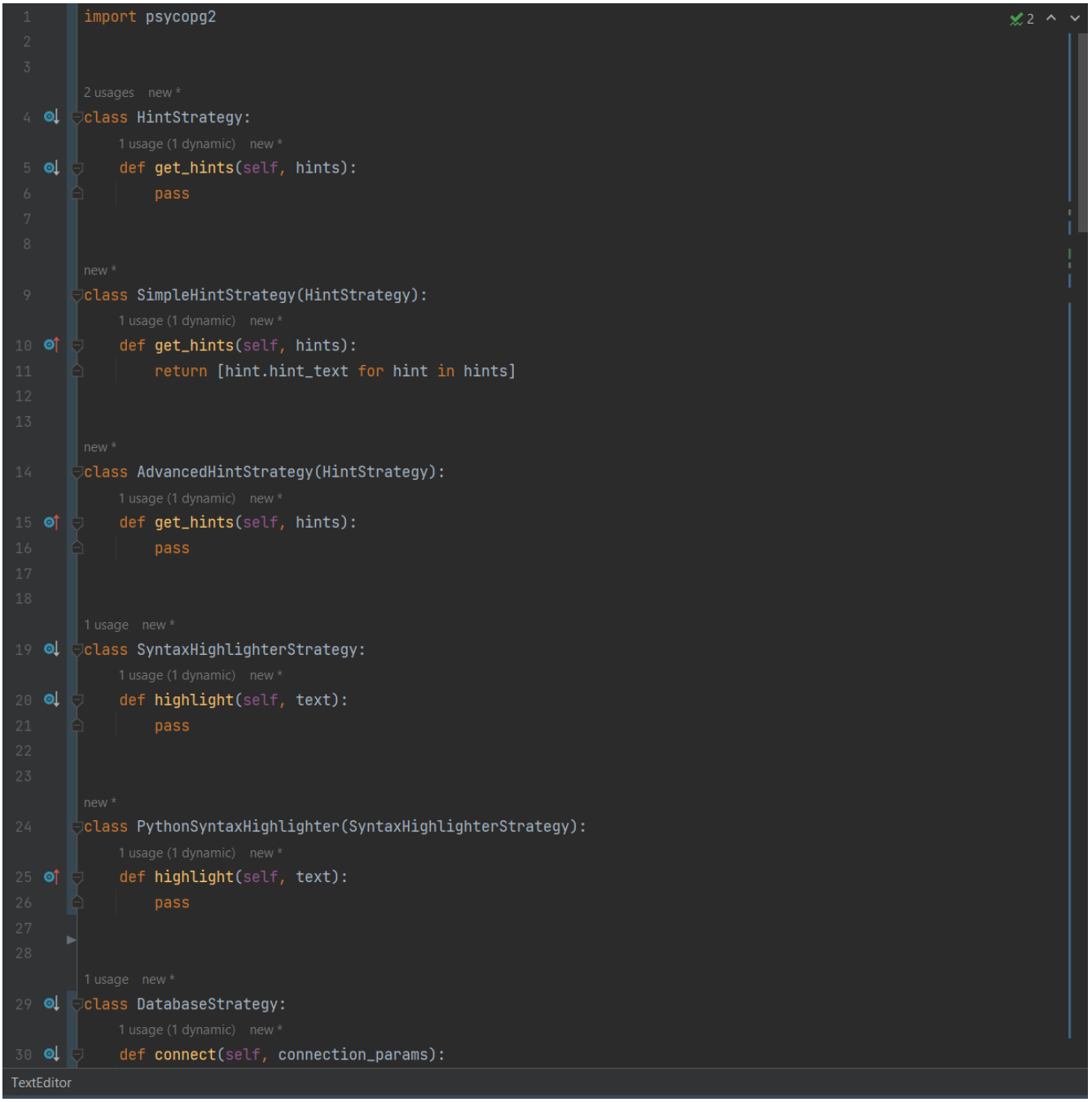
3. Текстовий редактор (strategy, command, observer, template method, flyweight, SOA)
Текстовий редактор повинен вміти розпізнавати текстові файли в будь-якій кодуванні, мати розширені функції редагування: макроси, сніппети, підказки, закладки, перехід на рядок / сторінку, підсвічування синтаксису (для однієї мови програмування або розмітки на розсуд студента).

Хід роботи

Шаблон Strategy

Шаблон «Strategy» (Стратегія) дозволяє змінювати деякий алгоритм поведінки об'єкта іншим алгоритмом, що досягає ту ж мету іншим способом.
Прикладом можуть служити алгоритми сортування: кожен алгоритм має власну реалізацію і визначений в окремому класі; вони можуть бути взаємозамінними в об'єкті, який їх використовує.




import psycopg2
2 usages new *
class HintStrategy:
1 usage (1 dynamic) new *
def get_hints(self, hints):
pass
new *
class SimpleHintStrategy(HintStrategy):
1 usage (1 dynamic) new *
def get_hints(self, hints):
return [hint.hint_text for hint in hints]
new *
class AdvancedHintStrategy(HintStrategy):
1 usage (1 dynamic) new *
def get_hints(self, hints):
pass
1 usage new *
class SyntaxHighlighterStrategy:
1 usage (1 dynamic) new *
def highlight(self, text):
pass
new *
class PythonSyntaxHighlighter(SyntaxHighlighterStrategy):
1 usage (1 dynamic) new *
def highlight(self, text):
pass
1 usage new *
class DatabaseStrategy:
1 usage (1 dynamic) new *
def connect(self, connection_params):
TextEditor

```
        pass


    1 usage (1 dynamic)  new *
    def save_to_database(self, file_content):
        pass


new *
class PostgreSQLDatabaseStrategy(DatabaseStrategy):
    new *
    def __init__(self):
        self.connection = None

    1 usage (1 dynamic)  new *
    def connect(self, connection_params):
        self.connection = psycopg2.connect(**connection_params)

    1 usage (1 dynamic)  new *
    def save_to_database(self, file_content):
        pass


1 usage  new *
class BookmarkStrategy:
    1 usage (1 dynamic)  new *
    def process_bookmarks(self, bookmarks):
        pass


new *
class DefaultBookmarkStrategy(BookmarkStrategy):
    1 usage (1 dynamic)  new *
    def process_bookmarks(self, bookmarks):
        pass


1 usage  new *
class MacroStrategy:
    1 usage (1 dynamic)  new *
    def execute_macros(self, macros):
        pass

```

TextEditor

```
62
                                                                    2
    new *
63  class DefaultMacroStrategy(MacroStrategy):
        1 usage (1 dynamic)   new *
64      def execute_macros(self, macros):
65          pass
66
67
    1 usage   new *
68  class SnippetStrategy:
        1 usage (1 dynamic)   new *
69      def apply_snippets(self, snippets):
70          pass
71
72
    new *
73  class DefaultSnippetStrategy(SnippetStrategy):
        1 usage (1 dynamic)   new *
74      def apply_snippets(self, snippets):
75          pass
76
77
    new *
78  class TextEditor:
        new *
79      def __init__(self, file_content="", encoding="utf-8"):
80          self.file_content = file_content
81          self.encoding = encoding
82          self.syntax_highlighter_strategy = None
83          self.bookmark_strategy = None
84          self.macro_strategy = None
85          self.snippet_strategy = None
86          self.hint_strategy = None
87          self.database_strategy = None
88          self.hints = []
89          self.bookmarks = []
90          self.macros = []
91          self.snippets = []
92
        new *
93      def set_syntax_highlighter_strategy(self, syntax_highlighter_strategy):
94          self.syntax_highlighter_strategy = syntax_highlighter_strategy
```

TextEditor

```
95
                                                                    2
        new *
96      def set_bookmark_strategy(self, bookmark_strategy):
97          self.bookmark_strategy = bookmark_strategy
98
        new *
99      def set_macro_strategy(self, macro_strategy):
100         self.macro_strategy = macro_strategy
101
        new *
102     def set_snippet_strategy(self, snippet_strategy):
103         self.snippet_strategy = snippet_strategy
104
        new *
105     def set_hint_strategy(self, hint_strategy):
106         self.hint_strategy = hint_strategy
107
        new *
108     def set_database_strategy(self, database_strategy):
109         self.database_strategy = database_strategy
110
        new *
111     def open_file(self, file_path, encoding):
112         self.file_content = read_file(file_path, encoding)
113         self.encoding = encoding
114
        new *
115     def save_file(self, file_path):
116         save_file(file_path, self.file_content, self.encoding)
117
        new *
118     def edit_text(self, changes):
119         pass
120
        new *
121     def execute_syntax_highlighting(self):
122         if self.syntax_highlighter_strategy:
123             self.syntax_highlighter_strategy.highlight(self.file_content)
124
        1 usage (1 dynamic)   new *
125     def process_bookmarks(self):
126         if self.bookmark_strategy:
```

TextEditor

Висновок: у даній лабораторній було реалізовано шаблон Стратегія, що дало змогу розділити функціонал на стратегії та окремі класи, що, своєю чергою, робить використання та зміну функціонала зручнішим.