Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3

По дисциплине «Современные платформы программирования»

Выполнил:
Студент 3 курса
Группы ПО-11
Микулич М. И.
Проверил:
Козик И. Д.

Брест 2025

Цель работы: приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Python.

Вариант 14

Первая группа заданий (порождающий паттерн)

Проект «Туристическое бюро». Реализовать возможность выбора программы тура (проезд, проживание, питание, посещение музеев, выставок, экскурсии и т.д.). Должна формироваться итоговая стоимость заказа.

Код программы:

```
class TourPackage:
    def __init__(self):
        self.selected_options = {}
        self.total_cost = 0

    def add_options(self, category: str, options: list, cost: int):
        self.selected_options[category] = options
        self.total_cost += cost

    def __str__(self):
        parts = ["Ваш заказ:"]
        for category, options in self.selected_options.items():
            parts.append(f"{category}: {', '.join(options)}")
        parts.append(f"\nИтоговая стоимость: {self.total_cost} руб.")
        return '\n'.join(parts)

class TourPackageBuilder:
    def __init__(self):
        self.tour_package = TourPackage()

    def add_category_options(self, category: str, options: list, cost: int):
        self.tour_package.add_options(category, options, cost)

    def get_result(self) -> TourPackage:
        return self.tour_package

def select_option(category_name: str, options: dict, is_multiple: bool = False):
    print(f"\nВыберите опции для {category_name}:")
    for idx, (option, price) in enumerate(options.items(), 1):
        print(f"{idx}. {option} - {price} руб.")

    while True:
        choice = input("Введите номера через запятую, если несколько: " if is_multiple else "Введите номер: ")
        try:
            selected_indices = list(map(int, choice.split(','))) if is_multiple else [int(choice)]

            if not all(1 <= idx <= len(options) for idx in selected_indices):
                raise ValueError

            selected = []
            total = 0
            for idx in selected_indices:
                option = list(options.keys())[idx-1]
                price = list(options.values())[idx-1]
                selected.append(option)
                total += price
            return selected, total

        except (ValueError, IndexError):
            print("Ошибка. Введите правильные номера.")

def main():
    print("Добро пожаловать в Туристическое бюро!\n")

    tour_options = {
        "Транспорт": {
            "Самолет": 15000,
            "Поезд": 8000,
            "Автобус": 5000,
            "Не включать": 0
        },
        "Проживание": {
            "Отель 3*": 3000,
            "Отель 4*": 5000,
```

```
            "Отель 5*": 8000,
            "Не включать": 0
        },
        "Питание": {
            "Без питания": 0,
            "Завтрак": 1500,
            "Полный пансион": 3000
        },
        "Музеи": {
            "Музей истории": 500,
            "Художественная галерея": 700,
            "Научный музей": 600
        },
        "Экскурсии": {
            "Обзорная": 1000,
            "Тематическая": 1500,
            "Водная": 2000
        }
    }

    multiple_choice = ['Музеи', 'Экскурсии']
    builder = TourPackageBuilder()

    for category, options in tour_options.items():
        is_multiple = category in multiple_choice
        selected_opts, cost = select_option(category, options, is_multiple)
        builder.add_category_options(category, selected_opts, cost)

    tour_package = builder.get_result()
    print(tour_package)

if __name__ == "__main__":
    main()
```

Результат работы:

```
Добро пожаловать в Туристическое бюро!


Выберите опции для Транспорт:
1. Самолет - 15000 руб.
2. Поезд - 8000 руб.
3. Автобус - 5000 руб.
4. Не включать - 0 руб.
Введите номер: 3

Выберите опции для Проживание:
1. Отель 3* - 3000 руб.
2. Отель 4* - 5000 руб.
3. Отель 5* - 8000 руб.
4. Не включать - 0 руб.
Введите номер: 2
Выберите опции для Питание:
1. Без питания - 0 руб.
2. Завтрак - 1500 руб.
3. Полный пансион - 3000 руб.
Введите номер: 1

Выберите опции для Музеи:
1. Музей истории - 500 руб.
2. Художественная галерея - 700 руб.
3. Научный музей - 600 руб.
Введите номера через запятую, если несколько: 3

Выберите опции для Экскурсии:
1. Обзорная - 1000 руб.
2. Тематическая - 1500 руб.
3. Водная - 2000 руб.
Введите номера через запятую, если несколько: 3
```

```
Ваш заказ:
Транспорт: Автобус
Проживание: Отель 4*
Питание: Без питания
Музеи: Научный музей
Экскурсии: Водная
```

Вторая группа заданий (структурный паттерн):

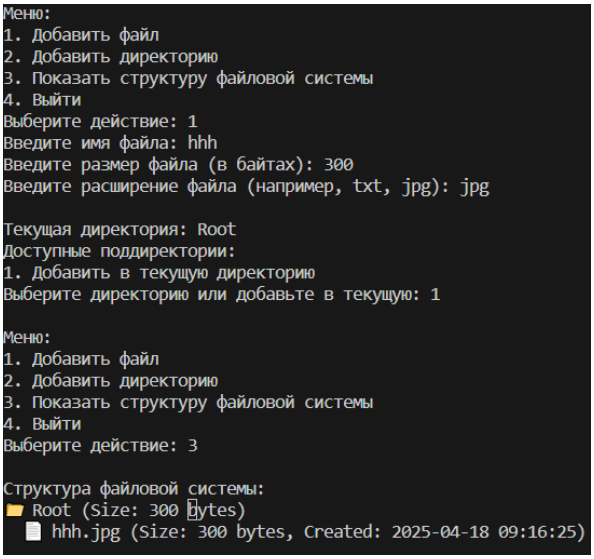
Проект «Файловая система». Реализуйте модель работы файловой системы. Должна поддерживаться иерархичность ФС на уровне директорий и отдельных файлов. Файлы могут иметь все основные присущие им атрибуты (размер, расширение, дата создания и т.д.).

Код программы:

```
from abc import ABC, abstractmethod
from datetime import datetime

class FileSystemComponent(ABC):
    @abstractmethod
    def get_size(self) -> int:
        pass

    @abstractmethod
    def display(self, indent: str = "") -> str:
        pass

class File(FileSystemComponent):
    def __init__(self, name: str, size: int, extension: str, created: datetime):
        self.name = name
        self.size = size
        self.extension = extension
        self.created = created

    def get_size(self) -> int:
        return self.size

    def display(self, indent: str = "") -> str:
        return f"{indent} {self.name}.{self.extension} (Size: {self.size} bytes, Created: {self.created.strftime('%Y-
%m-%d %H:%M:%S')})"

class Directory(FileSystemComponent):
    def __init__(self, name: str):
        self.name = name
        self.children = []

    def add(self, component: FileSystemComponent):
        self.children.append(component)

    def remove(self, component: FileSystemComponent):
        self.children.remove(component)

    def get_size(self) -> int:
        return sum(child.get_size() for child in self.children)

    def display(self, indent: str = "") -> str:
        result = f"{indent} {self.name} (Size: {self.get_size()} bytes)\n"
        for child in self.children:
            result += child.display(indent + "  ") + "\n"
        return result.rstrip()

def create_file():
    name = input("Введите имя файла: ")
    size = int(input("Введите размер файла (в байтах): "))
    extension = input("Введите расширение файла (например, txt, jpg): ")
    created = datetime.now()
    return File(name, size, extension, created)

def create_directory():
    name = input("Введите имя директории: ")
    return Directory(name)

def main():
    root = Directory("Root")
    while True:
        print("\nМеню:")
        print("1. Добавить файл")
        print("2. Добавить директорию")
        print("3. Показать структуру файловой системы")
        print("4. Выйти")
        choice = input("Выберите действие: ")

        if choice == "1":
            file = create_file()
            current = root
            while True:
                print(f"\nТекущая директория: {current.name}")
                print("Доступные поддиректории:")
                for i, child in enumerate(current.children):
                    if isinstance(child, Directory):
                        print(f"{i + 1}. {child.name}")
                print(f"{len(current.children) + 1}. Добавить в текущую директорию")
                dir_choice = input("Выберите директорию или добавьте в текущую: ")
                if dir_choice.isdigit() and 1 <= int(dir_choice) <= len(current.children):
```

```
                    current = current.children[int(dir_choice) - 1]
                else:
                    current.add(file)
                    break
        elif choice == "2":
            directory = create_directory()
            current = root
            while True:
                print(f"\nТекущая директория: {current.name}")
                print("Доступные поддиректории:")
                for i, child in enumerate(current.children):
                    if isinstance(child, Directory):
                        print(f"{i + 1}. {child.name}")
                print(f"{len(current.children) + 1}. Добавить в текущую директорию")
                dir_choice = input("Выберите директорию или добавьте в текущую: ")
                if dir_choice.isdigit() and 1 <= int(dir_choice) <= len(current.children):
                    current = current.children[int(dir_choice) - 1]
                else:
                    current.add(directory)
                    break
        elif choice == "3":
            print("\nСтруктура файловой системы:")
            print(root.display())
        elif choice == "4":
            break
        else:
            print("Неверный выбор. Попробуйте снова.")

if __name__ == "__main__":
    main()
```

Результат работы:



Третья группа заданий (поведенческий паттерн)

Реализовать вывод ФС из 2-й группы заданий. Вывод файлов/директорий должен осуществляться в случайном порядке. Вывести основные атрибуты каждого файла/директории.

Код программы:

```
from abc import ABC, abstractmethod
from datetime import datetime
import random

class FileSystemComponent(ABC):
    @abstractmethod
    def get_size(self) -> int:
        pass

    @abstractmethod
    def display(self, indent: str = "") -> str:
        pass

class File(FileSystemComponent):
    def __init__(self, name: str, size: int, extension: str, created: datetime):
```

```
        self.name = name
        self.size = size
        self.extension = extension
        self.created = created

    def get_size(self) -> int:
        return self.size

    def display(self, indent: str = "") -> str:
        return f"{indent}📄 {self.name}.{self.extension} (Size: {self.size} bytes, Created: {self.created.strftime('%Y-
%m-%d %H:%M:%S')})"

class Directory(FileSystemComponent):
    def __init__(self, name: str):
        self.name = name
        self.children = []

    def add(self, component: FileSystemComponent):
        self.children.append(component)

    def remove(self, component: FileSystemComponent):
        self.children.remove(component)

    def get_size(self) -> int:
        return sum(child.get_size() for child in self.children)

    def display(self, indent: str = "") -> str:
        result = f"{indent}📁 {self.name} (Size: {self.get_size()} bytes)\n"
        for child in self.children:
            result += child.display(indent + "  ") + "\n"
        return result.rstrip()

class DisplayStrategy(ABC):
    @abstractmethod
    def display_components(self, components: list[FileSystemComponent], indent: str = "") -> str:
        pass

class RandomOrderDisplay(DisplayStrategy):
    def display_components(self, components: list[FileSystemComponent], indent: str = "") -> str:
        random.shuffle(components)
        result = ""
        for component in components:
            result += component.display(indent) + "\n"
        return result.rstrip()

class DirectoryWithStrategy(Directory):
    def __init__(self, name: str, display_strategy: DisplayStrategy):
        super().__init__(name)
        self.display_strategy = display_strategy

    def display(self, indent: str = "") -> str:
        result = f"{indent}📁 {self.name} (Size: {self.get_size()} bytes)\n"
        result += self.display_strategy.display_components(self.children, indent + "  ")
        return result

def create_file():
    name = input("Введите имя файла: ")
    size = int(input("Введите размер файла (в байтах): "))
    extension = input("Введите расширение файла (например, txt, jpg): ")
    created = datetime.now()
    return File(name, size, extension, created)

def create_directory():
    name = input("Введите имя директории: ")
    return DirectoryWithStrategy(name, RandomOrderDisplay())

def main():
    root = DirectoryWithStrategy("Root", RandomOrderDisplay())
    while True:
        print("\nМеню:")
        print("1. Добавить файл")
        print("2. Добавить директорию")
        print("3. Показать структуру файловой системы")
        print("4. Выйти")
        choice = input("Выберите действие: ")

        if choice == "1":
            file = create_file()
            current = root
            while True:
                print(f"\nТекущая директория: {current.name}")
                print("Доступные поддиректории:")
```

```
                for i, child in enumerate(current.children):
                    if isinstance(child, DirectoryWithStrategy):
                        print(f"{i + 1}.  {child.name}")
                print(f"{len(current.children) + 1}. Добавить в текущую директорию")
                dir_choice = input("Выберите директорию или добавьте в текущую: ")
                if dir_choice.isdigit() and 1 <= int(dir_choice) <= len(current.children):
                    current = current.children[int(dir_choice) - 1]
                else:
                    current.add(file)
                    break
            elif choice == "2":
                directory = create_directory()
                current = root
                while True:
                    print(f"\nТекущая директория: {current.name}")
                    print("Доступные поддиректории:")
                    for i, child in enumerate(current.children):
                        if isinstance(child, DirectoryWithStrategy):
                            print(f"{i + 1}.  {child.name}")
                    print(f"{len(current.children) + 1}. Добавить в текущую директорию")
                    dir_choice = input("Выберите директорию или добавьте в текущую: ")
                    if dir_choice.isdigit() and 1 <= int(dir_choice) <= len(current.children):
                        current = current.children[int(dir_choice) - 1]
                    else:
                        current.add(directory)
                        break
            elif choice == "3":
                print("\nСтруктура файловой системы:")
                print(root.display())
            elif choice == "4":
                break
            else:
                print("Неверный выбор. Попробуйте снова.")

if __name__ == "__main__":
    main()
```

Результат работы:

```
Меню:
1. Добавить файл
2. Добавить директорию
3. Показать структуру файловой системы
4. Выйти
Выберите действие: 2
Введите имя директории: main

Текущая директория: Root
Доступные поддиректории:
1. Добавить в текущую директорию
Выберите директорию или добавьте в текущую: 1

Меню:
1. Добавить файл
2. Добавить директорию
3. Показать структуру файловой системы
4. Выйти
Выберите действие: 3

Структура файловой системы:
 Root (Size: 0 bytes)
   main (Size: 0 bytes)
```

Вывод: приобрел навыки применения паттернов проектирования при решении практических задач с использованием языка Python.