

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ
Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №3

Специальность ПО11

Выполнил
Е. А. Германович
студент группы ПО11

Проверил
А. А. Крощенко
ст. преп. кафедры ИИТ,
22.02.2025 г.

Брест 2025

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

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()
```

Спецификация ввода:

Введите транспорт: <1-й элемент>

Введите проживание: <2-й элемент>

Введите питание: <3-й элемент>

Введите музеи: <4-й элемент>

Введите экскурсии: <5-й элемент>

Пример:

Введите транспорт: 1
Введите проживание: 2
Введите питание: 3
Введите музеи: 1
Введите экскурсии: 2

Спецификация вывода:

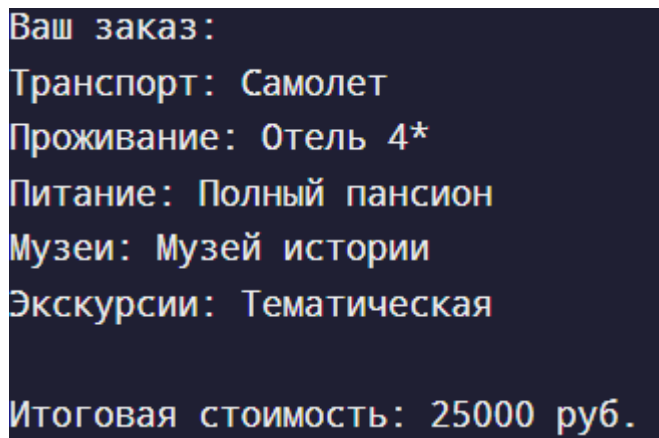
Ваш заказ:
Транспорт: <Выбранный пользователем 1-й элемент>
Проживание: <Выбранный пользователем 2-й элемент>
Питание: <Выбранный пользователем 3-й элемент>
Музеи: <Выбранный пользователем 4-й элемент>
Экскурсии: <Выбранный пользователем 5-й элемент>

Итоговая стоимость: <Сумма выбранных элементов>

Пример:

Ваш заказ:
Транспорт: Самолет
Проживание: Отель 4*
Питание: Полный пансион
Музеи: Музей истории
Экскурсии: Тематическая

Итоговая стоимость: 25000 руб.

Рисунки с результатами работы программы:

Ваш заказ:
Транспорт: Самолет
Проживание: Отель 4*
Питание: Полный пансион
Музеи: Музей истории
Экскурсии: Тематическая

Итоговая стоимость: 25000 руб.

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()
```

Спецификация ввода:

Введите действие: <Номер действия>

Пример:

Меню:

1. Добавить файл
2. Добавить директорию
3. Показать структуру файловой системы
4. Выйти

Выберите действие: 1

Введите имя файла: test

Введите размер файла (в байтах): 100

Введите расширение файла (например, txt, jpg): jpg

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: 2

Введите имя директории: home

Текущая директория: Root

Доступные поддиректории:

2. Добавить в текущую директорию

Выберите директорию или добавьте в текущую: 2

Меню:

1. Добавить файл
2. Добавить директорию
3. Показать структуру файловой системы
4. Выйти

Выберите действие: 3

Рисунки с результатами работы программы:

```

Меню:
1. Добавить файл
2. Добавить директорию
3. Показать структуру файловой системы
4. Выйти
Выберите действие: 1
Введите имя файла: test
Введите размер файла (в байтах): 100
Введите расширение файла (например, txt, jpg): jpg
0
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: 2
Введите имя директории: home
Текущая директория: Root
Доступные поддиректории:
2. Добавить в текущую директорию
Выберите директорию или добавьте в текущую: 2

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

Структура файловой системы:
📁 Root (Size: 100 bytes)
  📄 test.jpg (Size: 100 bytes, Created: 2025-03-15 06:26:39)
  📁 home (Size: 0 bytes)

```

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()
```

Спецификация ввода:

Введите действие: <Номер действия>

Пример:

Меню:

1. Добавить файл
2. Добавить директорию
3. Показать структуру файловой системы
4. Выйти

Выберите действие: 1

Введите имя файла: test

Введите размер файла (в байтах): 100

Введите расширение файла (например, txt, jpg): jpg

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: 2

Введите имя директории: home

Текущая директория: Root

Доступные поддиректории:

2. Добавить в текущую директорию

Выберите директорию или добавьте в текущую: 2

Меню:

1. Добавить файл
2. Добавить директорию
3. Показать структуру файловой системы
4. Выйти

Выберите действие: 3

Рисунки с результатами работы программы:

```
Меню:
1. Добавить файл
2. Добавить директорию
3. Показать структуру файловой системы
4. Выйти
Выберите действие: 1
Введите имя файла: test
Введите размер файла (в байтах): 100
Введите расширение файла (например, txt, jpg): jpg
0
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: 2
Введите имя директории: home
Текущая директория: Root
Доступные поддиректории:
2. Добавить в текущую директорию
Выберите директорию или добавьте в текущую: 2

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

Структура файловой системы:
📁 Root (Size: 100 bytes)
  📄 test.jpg (Size: 100 bytes, Created: 2025-03-15 06:26:39)
  📁 home (Size: 0 bytes)
```

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