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

С. С. Жватель

студент группы ПО11

Проверил

А. А. Крощенко

ст. преп. кафедры ИИТ,

12.04.2025 г.

Брест 2025

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

**Общее задание**

• Прочитать задания, взятые из каждой группы, соответствующей одному из

трех основных типов паттернов;

• Определить паттерн проектирования, который может использоваться при

реализации задания. Пояснить свой выбор;

• Реализовать фрагмент программной системы, используя выбранный паттерн.

Реализовать все необходимые дополнительные классы.

**Задание 1. Первая группа заданий (порождающий паттерн)**

**Проект «Бургер-закусочная». Реализовать возможность формирования заказа из определенных позиций (тип бургера (веганский, куриный и т.д.)), напиток (холодный – пепси, кока-кола и т.д.; горячий – кофе, чай и т.д.), тип упаковки – с собой, на месте. Должна формироваться итоговая стоимость заказа.**

Выполнение:

**Код программы:**

"""Module for managing burger orders using the Builder pattern."""  
  
  
class Burger:  
 """Class representing a burger order with type, drink, and packaging."""  
  
 def \_\_init\_\_(self):  
 self.burger\_type = None  
 self.drink = None  
 self.packaging = None  
  
 def \_\_str\_\_(self):  
 """Return string representation of the burger order."""  
 return f"Burger: {self.burger\_type}, Drink: {self.drink}, Packaging: {self.packaging}"  
  
 def cost(self):  
 """Calculate total cost of the burger order."""  
 cost\_map = {  
 "vegan": 10,  
 "chicken": 12,  
 "pepsi": 3,  
 "cola": 3,  
 "coffee": 4,  
 "tea": 4,  
 "takeaway": 1,  
 "eat\_in": 0,  
 }  
 burger\_cost = cost\_map.get(self.burger\_type, 0)  
 drink\_cost = cost\_map.get(self.drink, 0)  
 packaging\_cost = cost\_map.get(self.packaging, 0)  
 return burger\_cost + drink\_cost + packaging\_cost  
  
  
class BurgerBuilder:  
 """Builder class for constructing a Burger object step-by-step."""  
  
 def \_\_init\_\_(self):  
 self.burger = Burger()  
  
 def set\_burger\_type(self, burger\_type):  
 """Set the burger type."""  
 self.burger.burger\_type = burger\_type  
 return self  
  
 def set\_drink(self, drink):  
 """Set the drink type."""  
 self.burger.drink = drink  
 return self  
  
 def set\_packaging(self, packaging):  
 """Set the packaging type."""  
 self.burger.packaging = packaging  
 return self  
  
 def build(self):  
 """Return the constructed Burger object."""  
 return self.burger  
  
  
def create\_order():  
 """Prompt user to create a burger order interactively."""  
 builder = BurgerBuilder()  
  
 print("Available burgers: vegan, chicken")  
 while True:  
 burger\_type = input("Choose burger type: ").lower()  
 if burger\_type in ["vegan", "chicken"]:  
 builder.set\_burger\_type(burger\_type)  
 break  
 print("Invalid choice. Try again.")  
  
 print("Available drinks: pepsi, cola, coffee, tea")  
 while True:  
 drink = input("Choose drink: ").lower()  
 if drink in ["pepsi", "cola", "coffee", "tea"]:  
 builder.set\_drink(drink)  
 break  
 print("Invalid choice. Try again.")  
  
 print("Available packaging: takeaway, eat\_in")  
 while True:  
 packaging = input("Choose packaging: ").lower()  
 if packaging in ["takeaway", "eat\_in"]:  
 builder.set\_packaging(packaging)  
 break  
 print("Invalid choice. Try again.")  
  
 return builder.build()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 order = create\_order()  
 print(order)  
 print(f"Total cost: ${order.cost()}")

**Спецификация ввода:**

 Тип бургера (vegan, chicken).

 Напиток (pepsi, cola, coffee, tea).

 Упаковка (takeaway, eat\_in).

**Пример:**

Available burgers: vegan, chicken

Choose burger type: vegan

Available drinks: pepsi, cola, coffee, tea

Choose drink: coffee

Available packaging: takeaway, eat\_in

Choose packaging: takeaway

**Спецификация вывода:**

 Описание заказа: тип бургера, напиток, упаковка.

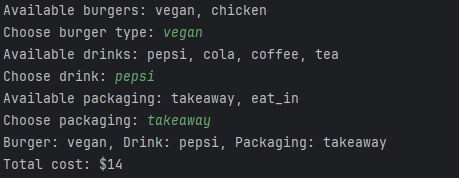
 Итоговая стоимость заказа.

**Пример:**

Burger: vegan, Drink: coffee, Packaging: takeaway

Total cost: $15

**Рисунки с результатами работы программы:**

****

**Задание 2. Вторая группа заданий (структурный паттерн)**

**Проект «Часы». В проекте должен быть реализован класс, который дает возможность пользоваться часами со стрелками так же, как и цифровыми часами. В классе «Часы со стрелками» хранятся повороты стрелок.**

Выполнение:

**Код программы:**

"""Module for adapting analog clock to digital clock using the Adapter pattern."""  
  
from abc import ABC, abstractmethod  
  
  
class DigitalClock(ABC):  
 """Abstract base class for digital clocks."""  
  
 # pylint: disable=too-few-public-methods  
  
 @abstractmethod  
 def get\_time(self):  
 """Return the time in digital format (hh:mm)."""  
  
  
class AnalogClock:  
 """Class representing an analog clock with hour and minute hand angles."""  
  
 # pylint: disable=too-few-public-methods  
  
 def \_\_init\_\_(self, hour\_angle, minute\_angle):  
 self.hour\_angle = hour\_angle  
 self.minute\_angle = minute\_angle  
  
 def get\_angles(self):  
 """Return the hour and minute hand angles."""  
 return self.hour\_angle, self.minute\_angle  
  
  
class AnalogToDigitalAdapter(DigitalClock):  
 """Adapter to convert analog clock to digital clock interface."""  
  
 # pylint: disable=too-few-public-methods  
  
 def \_\_init\_\_(self, analog):  
 self.analog\_clock = analog  
  
 def get\_time(self):  
 """Convert analog angles to digital time format (hh:mm)."""  
 hour\_angle, minute\_angle = self.analog\_clock.get\_angles()  
 hours = int((hour\_angle % 360) / 30)  
 minutes = int((minute\_angle % 360) / 6)  
 return f"{hours:02d}:{minutes:02d}"  
  
  
def create\_clock():  
 """Prompt user to input angles for an analog clock."""  
 while True:  
 try:  
 hour\_angle = float(input("Enter hour hand angle (0-360): "))  
 if 0 <= hour\_angle <= 360:  
 break  
 print("Angle must be between 0 and 360.")  
 except ValueError:  
 print("Invalid input. Enter a number.")  
  
 while True:  
 try:  
 minute\_angle = float(input("Enter minute hand angle (0-360): "))  
 if 0 <= minute\_angle <= 360:  
 break  
 print("Angle must be between 0 and 360.")  
 except ValueError:  
 print("Invalid input. Enter a number.")  
  
 return AnalogClock(hour\_angle, minute\_angle)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 clock = create\_clock()  
 adapter = AnalogToDigitalAdapter(clock)  
 print(f"Time: {adapter.get\_time()}")

**Спецификация ввода:**

 Угол поворота часовой стрелки (0–360 градусов).

 Угол поворота минутной стрелки (0–360 градусов).

**Пример:**

Enter hour hand angle (0-360): 90

Enter minute hand angle (0-360): 180

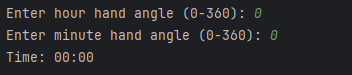
**Спецификация вывода:**

Время в цифровом формате (hh:mm).

**Пример:**

Time: 03:30

**Рисунки с результатами работы программы:**

****

**Задание 3. Третья группа заданий (поведенческий паттерн)**

**Шифрование текстового файла. Реализовать класс-шифровщик текстового файла с поддержкой различных алгоритмов шифрования. Возможные варианты шифрования: удаление всех гласных букв из текста, изменение букв текста на буквы, получаемые фиксированным сдвигом из алфавита (например, шифром буквы а будет являться буква д для сдвига 4 и т.д.), применение операции исключающее или с заданным ключом.**

Выполнение:

**Код программы:**

"""Module for encrypting text files using the Strategy pattern."""  
  
import os  
from abc import ABC, abstractmethod  
  
  
class EncryptionStrategy(ABC):  
 """Abstract base class for encryption strategies."""  
  
 # pylint: disable=too-few-public-methods  
  
 @abstractmethod  
 def encrypt(self, text):  
 """Encrypt the input text."""  
  
  
class RemoveVowelsStrategy(EncryptionStrategy):  
 """Strategy to encrypt text by removing vowels."""  
  
 # pylint: disable=too-few-public-methods  
  
 def encrypt(self, text):  
 """Remove all vowels from the input text."""  
 vowels = set("aeiouAEIOU")  
 return "".join(char for char in text if char not in vowels)  
  
  
class CaesarCipherStrategy(EncryptionStrategy):  
 """Strategy to encrypt text using Caesar cipher with a shift."""  
  
 # pylint: disable=too-few-public-methods  
  
 def \_\_init\_\_(self, shift=4):  
 self.shift = shift  
  
 def encrypt(self, text):  
 """Apply Caesar cipher with specified shift to the input text."""  
 result = ""  
 for char in text:  
 if char.isalpha():  
 ascii\_offset = ord("A") if char.isupper() else ord("a")  
 result += chr((ord(char) - ascii\_offset + self.shift) % 26 + ascii\_offset)  
 else:  
 result += char  
 return result  
  
  
class XORCipherStrategy(EncryptionStrategy):  
 """Strategy to encrypt text using XOR with a key."""  
  
 # pylint: disable=too-few-public-methods  
  
 def \_\_init\_\_(self, key="secret"):  
 self.key = key  
  
 def encrypt(self, text):  
 """Apply XOR encryption with the specified key to the input text."""  
 key\_bytes = self.key.encode()  
 text\_bytes = text.encode()  
 result = bytearray()  
 for i, byte in enumerate(text\_bytes):  
 result.append(byte ^ key\_bytes[i % len(key\_bytes)])  
 return result.decode("latin1")  
  
  
class Encryptor:  
 """Class to manage encryption using a specified strategy."""  
  
 # pylint: disable=too-few-public-methods  
  
 def \_\_init\_\_(self, strategy: EncryptionStrategy):  
 self.strategy = strategy  
  
 def set\_strategy(self, strategy: EncryptionStrategy):  
 """Set the encryption strategy."""  
 self.strategy = strategy  
  
 def encrypt(self, text):  
 """Encrypt the text using the current strategy."""  
 return self.strategy.encrypt(text)  
  
  
def run\_encryption():  
 """Prompt user to encrypt a text file using a chosen strategy."""  
 while True:  
 input\_file = input("Enter input file name (e.g., input.txt): ")  
 if os.path.isfile(input\_file):  
 break  
 print("File does not exist. Try again.")  
  
 try:  
 with open(input\_file, "r", encoding="utf-8") as f:  
 text = f.read()  
 except (FileNotFoundError, IOError) as e:  
 print(f"Error reading file: {e}")  
 return  
  
 print("Available encryption methods:")  
 print("1. Remove vowels")  
 print("2. Caesar cipher")  
 print("3. XOR cipher")  
  
 while True:  
 choice = input("Choose method (1-3): ")  
 if choice in ["1", "2", "3"]:  
 break  
 print("Invalid choice. Try again.")  
  
 encryptor = Encryptor(RemoveVowelsStrategy())  
  
 if choice == "1":  
 encryptor.set\_strategy(RemoveVowelsStrategy())  
 elif choice == "2":  
 while True:  
 try:  
 shift = int(input("Enter shift for Caesar cipher (1-25): "))  
 if 1 <= shift <= 25:  
 encryptor.set\_strategy(CaesarCipherStrategy(shift))  
 break  
 print("Shift must be between 1 and 25.")  
 except ValueError:  
 print("Invalid input. Enter a number.")  
 elif choice == "3":  
 key = input("Enter key for XOR cipher: ")  
 encryptor.set\_strategy(XORCipherStrategy(key if key else "secret"))  
  
 encrypted\_text = encryptor.encrypt(text)  
  
 output\_file = input("Enter output file name (e.g., output.txt): ")  
  
 try:  
 with open(output\_file, "w", encoding="utf-8") as f:  
 f.write(encrypted\_text)  
 print(f"Encrypted text written to {output\_file}")  
 print(f"Encrypted text: {encrypted\_text}")  
 except (FileNotFoundError, IOError) as e:  
 print(f"Error writing file: {e}")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 run\_encryption()

**Спецификация ввода:**

 Имя входного файла (например, input.txt).

 Метод шифрования (1 – удаление гласных, 2 – шифр Цезаря, 3 – XOR).

 Для шифра Цезаря: величина сдвига (1–25).

 Для XOR: ключ шифрования.

 Имя выходного файла (например, output.txt).

**Пример:**

Enter input file name (e.g., input.txt): input.txt

Available encryption methods:

1. Remove vowels

2. Caesar cipher

3. XOR cipher

Choose method (1-3): 2

Enter shift for Caesar cipher (1-25): 4

Enter output file name (e.g., output.txt): output.txt

**Спецификация вывода:**

 Подтверждение записи зашифрованного текста в файл.

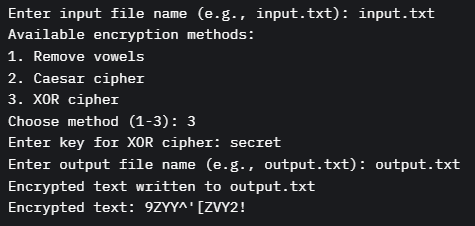
 Зашифрованный текст.

**Пример:**

Encrypted text written to output.txt

Encrypted text: Lipps, Asvph!

**Рисунки с результатами работы программы:**

****

**Вывод:** закрепил знания паттернов Python при решении практических задач