МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Лабораторна робота №4

з дисципліни

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

на тему

РОЗРОБКА ASCII ART ГЕНЕРАТОРА ДЛЯ ВІЗУАЛІЗАЦІЇ 2D-ФІГУР

Виконала:

ст. гр. ІТ-21сп

Борисов П.Р.

Прийняв:

Щербак С.С.

Львів-2023

**Мета роботи:** Створення Генератора ASCII-арту без використання зовнішніх бібліотек.

**План роботи**

**Завдання 1:** Введення користувача

Створіть програму Python, яка отримує введення користувача щодо слова або фрази, яку вони хочуть перетворити в ASCII-арт.

**Завдання 2:** Набір символів

Визначте набір символів (наприклад, '@', '#', '\*', тощо), які будуть використовуватися для створення ASCII-арту. Ці символи будуть відображати різні відтінки.

**Завдання 3:** Розміри Art-у

Запитайте у користувача розміри (ширина і висота) ASCII-арту, який вони хочуть створити. Переконайтеся, що розміри в межах керованого діапазону

**Завдання 4:** Функція генерації Art-у

Напишіть функцію, яка генерує ASCII-арт на основі введення користувача, набору символів та розмірів. Використовуйте введення користувача, щоб визначити, які символи використовувати для кожної позиції в Art-у.

**Завдання 5:** Вирівнювання тексту

Реалізуйте опції вирівнювання тексту (ліво, центр, право), щоб користувачі могли вибирати, як їх ASCII-арт розміщується на екрані.

**Завдання 6:** Відображення мистецтва

Відобразіть створений ASCII-арт на екрані за допомогою стандартних функцій друку Python.

**Завдання 7:** Збереження у файл

Додайте можливість зберігати створений ASCII-арт у текстовий файл, щоб користувачі могли легко завантажувати та обмінюватися своїми творіннями.

**Завдання 8:** Варіанти кольорів

Дозвольте користувачам вибирати опції кольорів (чорно-білий, відтінки сірого) для свого ASCII-арту.

**Завдання 9:** Функція попереднього перегляду

Реалізуйте функцію попереднього перегляду, яка показує користувачам попередній перегляд їх ASCII-арту перед остаточним збереженням

**Завдання 10:** Інтерфейс, зрозумілий для користувача

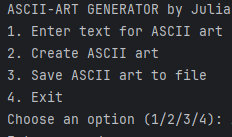
Створіть інтерфейс для користувача у командному рядку, щоб зробити програму легкою та інтуїтивно зрозумілою для використання.

**Код програми:**  
Клас Menu.py  
from UI.MenuBuilder.ASCIIArtGenerator import ASCIIArtGenerator  
from Shared.ReaderWriter import ReaderWriter  
from Shared.Validator import Validator  
from Shared.ascii\_art\_library import ascii\_art  
  
class Menu:  
 # Constructor for the Menu class  
 def \_\_init\_\_(self, generator, reader\_writer, validator):  
 self.generator = generator  
 self.reader\_writer = reader\_writer  
 self.validator = validator  
 self.ascii\_art\_created = False  
  
 # Method to display the menu options  
 def display\_options(self):  
 print("1. Enter text for ASCII art")  
 print("2. Create ASCII art")  
 print("3. Save ASCII art to file")  
 print("4. Exit")  
  
 # Method to run the menu loop  
 def run(self):  
 print("ASCII-ART GENERATOR by Julia")  
 while True:  
 self.display\_options()  
 choice = self.validator.validate\_input("Choose an option (1/2/3/4): ", ['1', '2', '3', '4'])  
 if choice == '1':  
 self.generator.get\_user\_input()  
 elif choice == '2':  
 if not self.generator.text:  
 print("Error: Enter text first (option 1).")  
 else:  
 self.ascii\_art\_created = True  
 self.generator.create\_ascii\_art()  
 elif choice == '3':  
 if not self.ascii\_art\_created:  
 print("Error: Create ASCII art first (option 2).")  
 else:  
 filename = self.reader\_writer.read\_input("Enter a filename to save the ASCII art: ")  
 ascii\_art\_no\_color = self.generator.remove\_color\_codes(self.generator.ascii\_art)  
 self.reader\_writer.write\_output(filename, ascii\_art\_no\_color)  
 print(f"ASCII art saved in file {filename}")  
 elif choice == '4':  
 print("Goodbye!")  
 break  
 else:  
 print("Invalid choice. Try again.")  
  
# Main function to create the necessary objects and run the menu  
if \_\_name\_\_ == '\_\_main\_\_':  
 reader\_writer = ReaderWriter()  
 validator = Validator()  
 generator = ASCIIArtGenerator(ascii\_art, reader\_writer)  
 menu = Menu(generator, reader\_writer, validator)  
 menu.run()

Клас ASCIIArtGenerator.py  
from Shared.Validator import Validator  
from Shared.color\_util import ColorUtil  
  
  
# Constructor for the ASCIIArtGenerator class  
class ASCIIArtGenerator:  
 def \_\_init\_\_(self, ascii\_art, reader\_writer):  
 self.ascii\_art = ascii\_art  
 self.text = ''  
 self.validator = Validator()  
 self.reader\_writer = reader\_writer  
  
 # Method to get user input for the text to be converted to ASCII art  
 def get\_user\_input(self):  
 self.text = self.reader\_writer.read\_input('Enter a word: ')  
  
 # Method to get user input for the symbol to be used in the ASCII art  
 def get\_user\_symbol(self):  
 symbol\_mapping = {  
 '\*': 'Star (\*)',  
 '#': 'Hash character (#)',  
 '@': 'Special character (@)',  
 '?': 'Question mark (?)',  
 '=': 'Equal sign (=)',  
 '-': 'Hyphen (-)'  
 }  
  
 print("Available symbols for ASCII art:")  
 for symbol, description in symbol\_mapping.items():  
 print(f"{symbol}: {description}")  
  
 user\_choice = self.reader\_writer.read\_input('Choose a symbol for ASCII art: ')  
 if user\_choice in symbol\_mapping:  
 self.user\_symbol = user\_choice  
 else:  
 print("Invalid choice.")  
  
 # Method to get user input for the size of the ASCII art  
 def get\_art\_size(self):  
 self.width = int(self.reader\_writer.read\_input('Enter the width of the ASCII art (2-40): '))  
 self.height = int(self.reader\_writer.read\_input('Enter the height of the ASCII art (2-40): '))  
  
 # Method to get user input for the alignment of the ASCII art  
 def get\_alignment(self):  
 alignment\_options = {  
 'left': 'left',  
 'center': 'center',  
 'right': 'right',  
 }  
 choice = self.reader\_writer.read\_input('Choose text alignment (left/center/right): ')  
 if choice in alignment\_options:  
 self.alignment = alignment\_options[choice]  
 else:  
 print("Invalid choice.")  
  
 # Method to get user input for the colors of the symbols in the ASCII art  
 def get\_colors(self):  
 self.colors = []  
 for i in range(len(self.user\_symbol)):  
 color\_options = {  
 'white': 'white',  
 'grey': 'grey',  
 }  
 choice = self.reader\_writer.read\_input(f'Choose a color for symbol {self.user\_symbol[i]}: ')  
 while choice not in color\_options:  
 print("Invalid choice.")  
 choice = self.reader\_writer.read\_input(f'Choose a color for symbol {self.user\_symbol[i]}: ')  
 self.colors.append(choice)  
  
 # Method to generate ASCII art based on user input  
 def generate\_art(self):  
 ascii\_art = ''  
 for char in self.text:  
 if char.upper() in self.ascii\_art:  
 art = self.ascii\_art[char.upper()]  
 art = [line.replace('\*', self.user\_symbol) for line in art]  
 art = art[:self.height]  
 while len(art) < self.height:  
 art.append(' ' \* self.width)  
 ascii\_art += '\n'.join(art) + '\n'  
 else:  
 ascii\_art += f"ASCII art for the letter {char} is not available.\n"  
  
 if self.alignment == 'center':  
 lines = ascii\_art.strip().split('\n')  
 ascii\_art = '\n'.join(line.center(self.width) for line in lines)  
 elif self.alignment == 'right':  
 lines = ascii\_art.strip().split('\n')  
 ascii\_art = '\n'.join(line.rjust(self.width) for line in lines)  
  
 colored\_art = ''  
 for symbol, color in zip(self.user\_symbol, self.colors):  
 colored\_art += ColorUtil.colorize(symbol, color)  
  
 ascii\_art = ascii\_art.replace(self.user\_symbol, colored\_art)  
 return ascii\_art  
  
 # Method to create ASCII art based on user input  
 def create\_ascii\_art(self):  
 self.get\_user\_symbol()  
 self.get\_art\_size()  
 self.get\_alignment()  
 self.get\_colors()  
 self.ascii\_art = self.generate\_art()  
 self.display\_art(self.ascii\_art)  
  
 # Method to display the generated ASCII art  
 def display\_art(self, ascii\_art):  
 print("Created ASCII art:")  
 print(ascii\_art)  
  
 # Method to save the generated ASCII art to a file  
 def save\_to\_file(self):  
 filename = self.reader\_writer.read\_input("Enter a filename to save the ASCII art: ")  
 ascii\_art\_no\_color = self.remove\_color\_codes(self.ascii\_art)  
 self.reader\_writer.write\_output(filename, ascii\_art\_no\_color)  
 print(f"ASCII art saved in file {filename}")  
  
 # Method to remove color codes from the text  
 def remove\_color\_codes(self, text):  
 while '\033[' in text:  
 start = text.find('\033[')  
 end = text.find('m', start)  
 if end != -1:  
 text = text[:start] + text[end+1:]  
 else:  
 break  
 return text

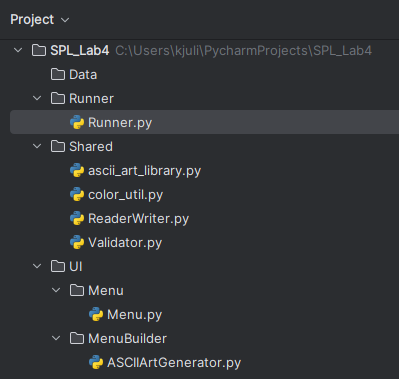
Клас Runner.py  
from Shared.ascii\_art\_library import ascii\_art  
from UI.Menu.Menu import Menu  
from Shared.ReaderWriter import ReaderWriter  
from Shared.Validator import Validator  
from UI.MenuBuilder.ASCIIArtGenerator import ASCIIArtGenerator  
  
  
class Runner:  
 def \_\_init\_\_(self):  
 self.reader\_writer = ReaderWriter()  
 self.validator = Validator()  
 self.generator = ASCIIArtGenerator(ascii\_art, self.reader\_writer)  
 self.menu = Menu(self.generator, self.reader\_writer, self.validator)  
  
 def run(self):  
 self.menu.run()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 runner = Runner()  
 runner.run()

На рисунку 1 зображено результат виконання всіх 10 завдань.



**Рис. 1** Результат завдань

На рисунку 2 зображена архітектура проєкту.



**Рис. 2** Архітектура проєкту

**Висновок:** я створив генератор ASCII-арту без використання зовнішніх бібліотек.