МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА  ІСМ



ЗВІТ

про виконання лабораторної роботи №5

з дисципліни

«Спеціалізовані мови програмування»

студента групи ІТ-32

Ткачишина Юрія

Прийняв Щербак С. С.

Львів - 2023

**Мета роботи:** Створення додатка для малювання 3D-фігур у ASCII-арті на основі об’єктно-орієнтованого підходу та мови Python.

**Індивідуальне завдання**

Завдання 1: Проектування класів Розробіть структуру класів для вашого генератора 3D ASCII-арту. Визначте основні компоненти, атрибути та методи, необхідні для програми.

Завдання 2: Введення користувача Створіть методи у межах класу для введення користувача та вказання 3D-фігури, яку вони хочуть намалювати, та її параметрів (наприклад, розмір, кольори).

Завдання 3: Представлення фігури Визначте структури даних у межах класу для представлення 3D-фігури. Це може включати використання списків, матриць або інших структур даних для зберігання форми фігури та її властивостей.

Завдання 4: Проектування з 3D в 2D Реалізуйте метод, який перетворює 3D-представлення фігури у 2D-представлення, придатне для ASCII-арту.

Завдання 5: Відображення ASCII-арту Напишіть метод у межах класу для відображення 2D-представлення 3D-фігури як ASCII-арту. Це може включати відображення кольорів і форми за допомогою символів ASCII.

Завдання 6: Інтерфейс, зрозумілий для користувача Створіть зручний для користувача командний рядок або графічний інтерфейс користувача (GUI) за допомогою об'єктно-орієнтованих принципів, щоб дозволити користувачам спілкуватися з програмою.

Завдання 7: Маніпуляція фігурою Реалізуйте методи для маніпулювання 3D-фігурою, такі масштабування або зміщення, щоб надавати користувачам контроль над її виглядом.

Завдання 8: Варіанти кольорів Дозвольте користувачам вибирати варіанти кольорів для їхніх 3D ASCII-арт-фігур. Реалізуйте методи для призначення кольорів різним частинам фігури.

Завдання 9: Збереження та експорт Додайте функціональність для зберігання згенерованого 3D ASCII-арту у текстовий файл

Завдання 10: Розширені функції Розгляньте можливість додавання розширених функцій, таких як тінь, освітлення та ефекти перспективи, для підвищення реалізму 3D ASCII-арту.

**Хід виконання:**

На рис. 1 зображено знімок екрану виконання програми.



*Рис. 1 Виконання програми*

Код:  
Файл Runner.py:  
from generator3D import \*

from services import file\_handler

def input\_mark():

while True:

mark = input("Please input a mark for the shape: ")

if not AbstractShape.valid\_mark(mark):

print("Only one mark is required!")

else:

return mark

def select\_color\_code():

while True:

try:

color\_code = int(input("Select a color code: "))

if color\_code not in palette.keys():

print("Select a valid color code!")

else:

return color\_code

except ValueError:

print("A numeric value is required!")

def input\_edge\_length():

while True:

try:

edge\_length = int(input("Input the edge length: "))

if edge\_length <= 0:

print("Edge length must be greater than zero!")

else:

return edge\_length

except ValueError:

print("A numeric value is required!")

def input\_scaling():

while True:

try:

scaling = float(input("Set the scale: "))

if scaling <= 0:

print("Scale must be greater than zero!")

else:

return scaling

except ValueError:

print("A numeric value is required!")

file\_2d = "2D.txt"

file\_3d = "3D.txt"

def main():

shape\_created = False

two\_d\_ready = False

three\_d\_ready = False

while True:

print("Options:")

print("1 - Generate a Square")

print("2 - Show 2D")

print("3 - Show 3D")

print("4 - Save 2D")

print("5 - Save 3D")

print("0 - Quit")

user\_choice = input("What your choice?: ")

match user\_choice:

case "1":

mark = input\_mark()

print("Available colors:")

show\_palette()

color\_code = select\_color\_code()

edge\_length = input\_edge\_length()

scaling = input\_scaling()

try:

shape = Square(edge\_length, mark, color\_code)

shape\_created = True

except ValueError as error:

print(error)

shape\_created = False

case "2":

if shape\_created:

representation\_2D = shape.draw\_2d()

for line in representation\_2D:

print(line)

two\_d\_ready = True

else:

print("No shape created yet!")

case "3":

if shape\_created:

representation\_3D = shape.draw\_3d(scaling)

print(representation\_3D)

three\_d\_ready = True

else:

print("No shape created yet!")

case "4":

if two\_d\_ready:

try:

file\_handler.write\_into\_file(file\_2d, "".join(representation\_3D))

except PermissionError:

print("Insufficient permissions to write to file!")

except FileNotFoundError:

print("File not found!")

else:

print("No shape created yet!")

case "5":

if three\_d\_ready:

try:

file\_handler.write\_into\_file(file\_3d, representation\_3D)

except PermissionError:

print("Insufficient permissions to write to file!")

except FileNotFoundError:

print("File not found!")

else:

print("No shape created yet!")

case "0":

break

case \_:

print("Please select a valid option!")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Файл file\_handler.py:

#Запис у файл

def write\_into\_file(file\_path: str, text: str) -> None:

with open(file\_path, "w") as file:

file.write(text)

#Зчитувати файл

def read\_from\_file(file\_path: str) -> str:

with open(file\_path, "r") as file:

return file.read()

Файл generator3D.py:

from abc import ABC, abstractmethod

import colorama

from colorama import Fore

colorama.init(autoreset=True)

palette = {index: color for index, color in enumerate(sorted(Fore.\_\_dict\_\_.keys())) if not color.startswith('\_')}

def show\_palette() -> None:

for index, color in palette.items():

print(f"{index}. {color}")

class AbstractShape(ABC):

def \_\_init\_\_(self, mark: str, color\_code: int):

if color\_code not in palette:

raise ValueError("Color code should be within the palette range")

elif not self.valid\_mark(mark):

raise ValueError("Only a single mark is acceptable")

self.\_mark = mark

self.\_color\_code = color\_code

@abstractmethod

def draw\_2d(self) -> list:

pass

@abstractmethod

def draw\_3d(self) -> str:

pass

@staticmethod

def valid\_mark(mark: str) -> bool:

return len(mark) == 1

class Square(AbstractShape):

def \_\_init\_\_(self, edge: int, mark: str, color\_code: int):

if edge <= 0:

raise ValueError("Edge length must be positive")

super().\_\_init\_\_(mark, color\_code)

self.\_edge = edge

self.\_midpoint = int(edge / 2 + 1)

def draw\_2d(self) -> list:

sketch = ""

for i in range(self.\_edge):

for j in range(self.\_edge):

if i in [0, self.\_edge - 1] or j in [0, self.\_edge - 1]:

sketch += f"{self.\_mark} "

else:

sketch += " "

sketch += "\n"

return [Fore.\_\_getattribute\_\_(palette[self.\_color\_code]) + "\n" + sketch]

def draw\_3d(self, scaling: float = 1.0) -> str:

new\_edge = int(self.\_edge \* scaling) if self.\_edge \* scaling >= 2 else self.\_edge

new\_midpoint = int(new\_edge / 2 + 1)

sketch = ""

for row in range(new\_midpoint - 1):

for col in range(new\_edge + new\_midpoint - 1):

if (row + col == new\_midpoint - 1) or (row == 0 and col > new\_midpoint - 1):

sketch += f"{self.\_mark}" + (

"" if col == new\_edge + new\_midpoint - 2 and row == 0 else " ")

elif new\_edge + new\_midpoint - row == col + 2:

sketch += f"{self.\_mark}"

elif col == new\_edge + new\_midpoint - 2:

sketch += f" {self.\_mark}"

else:

sketch += " "

sketch += "\n"

for row in range(new\_edge):

for col in range(new\_edge + new\_midpoint):

if ((row == 0 or row == new\_edge - 1) and col < new\_edge or (

col == 0 or col == new\_edge - 1) and row < new\_edge and col < new\_edge):

sketch += f"{self.\_mark}" + (

"" if row == new\_edge - 1 and col == new\_edge - 1 else " ")

elif row + col == (new\_edge - 1) \* 2 and col < new\_edge + new\_midpoint - 1:

sketch += " " \* (new\_edge - row - 2) + f"{self.\_mark}"

elif col < new\_edge and row < new\_edge:

sketch += " "

elif row < new\_edge - new\_midpoint and col > new\_edge:

if col == new\_midpoint + new\_edge - 1:

sketch += f"{self.\_mark}"

else:

sketch += " "

sketch += "\n"

return Fore.\_\_getattribute\_\_(palette[self.\_color\_code]) + "\n" + sketch

**Висновок:** Виконуючи ці завдання, я створив високорівневий об'єктно-орієнтований генератор 3D ASCII-арту, який дозволяє користувачам проектувати, відображати та маніпулювати 3D- фігурами в ASCII-арті. Цей проект надав мені глибоке розуміння об'єктно-орієнтованого програмування і алгоритмів графіки.