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.