Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 5

“ Розробка ASCII ART генератора для візуалізації 3D-фігур ”

з дисципліни **“ Спеціалізовані мови програмування”**

Виконала студентка групи ІТ-32
ДЕНИСЯК С.-М. Т.
Прийняв:
ЩЕРБАК С. С.

Львів – 2023

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

План роботи

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.

Код програми:

runner.py:

```
from Figures3D import *
from src.utility import FileProcessor


def get_character_input():
    while True:
        character = input("Enter a character to represent in the shape: ")
        if Figure3D.is_appropriate_character(character) is False:
            print("You should have entered one character!")
        else:
            return character


def get_color_position_input():
    while True:
        try:
            color = int(input("Enter a number of color: "))
            if color not in range(len(colors)):
                print("You should have entered a color option which is
available!")
            else:
                return color
        except ValueError:
            print("You should have entered an integer number!")


def get_length_input():
    while True:
        try:
            length = int(input("Enter a length: "))
            if length <= 0:
                print("You should have entered a length greater than 0!")
            else:
                return length
        except ValueError:
            print("You should have entered an integer number!")


def get_scale_input():
    while True:
        try:
            scale = float(input("Enter a scale for figure: "))
            if scale <= 0:
                print("You should have entered a scale greater than 0!")
            else:
                return scale
        except ValueError:
            print("You should have entered a float number!")


representation_2d_file = "2d.txt"
representation_3d_file = "3d.txt"


def main():
    is_figure_available: bool = False
    is_2d_representation_available = False
    is_3d_representation_available = False

    while True:

        print("1 - Create a cube")
```

```
        print("2 - Display 2D")
        print("3 - Display 3D")
        print("4 - Save 2D")
        print("5 - Save 3D")
        print("0 - Exit")
        option = str(input("Enter an option: "))

        match option:
            case "1":
                character = get_character_input()
                print("There are such colors available:")
                display_colors()
                color_position = get_color_position_input()
                length = get_length_input()
                scale = get_scale_input()
                try:
                    figure = Cube(length, character, color_position)
                    is_figure_available = True
                except ValueError as e:
                    print(e)
                    is_figure_available = False
            case "2":
                if is_figure_available is True:
                    representation_2d = figure.get_2d_representation()
                    [print(item) for item in representation_2d]
                    is_2d_representation_available = True
                else:
                    print("There is no figure available!")
            case "3":
                if is_figure_available is True:
                    representation_3d =
figure.get_3d_representation(scale=scale)
                    print(representation_3d)
                    is_3d_representation_available = True
                else:
                    print("There is no figure available!")
            case "4":
                if is_2d_representation_available is True:
                    try:
                        FileProcessor.write_into_file(representation_2d_file,
"".join(representation_2d))
                    except PermissionError:
                        print("You do not have permission to write to the
file!")
                    except FileNotFoundError:
                        print("The file does not exist!")
                else:
                    print("There is no figure available!")
            case "5":
                if is_3d_representation_available is True:
                    try:
                        FileProcessor.write_into_file(representation_3d_file,
representation_3d)
                    except PermissionError:
                        print("You do not have permission to write to the
file!")
                    except FileNotFoundError:
                        print("The file does not exist!")
                else:
                    print("There is no figure available!")
            case "0":
                break
            case _:
                print("Invalid option!")
```

```
if __name__ == "__main__":
    main()
```

Figures3D:

```
from abc import ABC, abstractmethod
import colorama
from colorama import Fore

colorama.init(autoreset=True)

colors = dict(enumerate(sorted(Fore.__dict__.keys())))


def display_colors() -> None:
    for i in colors:
        print(str(i) + ". " + colors[i])


class Figure3D(ABC):
    def __init__(self, character: str, color_position: int):
        if not colors.__contains__(color_position):
            raise ValueError("Color position must be in range of available
colors")
        elif self.is_appropriate_character(character) is False:
            raise ValueError("It supposed to be only one character instead")
        self._character = character
        self._color_position = color_position

    @abstractmethod
    def get_2d_representation(self) -> list:
        pass

    @abstractmethod
    def get_3d_representation(self) -> str:
        pass

    @staticmethod
    def is_appropriate_character(character: str) -> bool:
        return len(character) == 1


class Cube(Figure3D):
    def __init__(self, length: int, character: str, color_position: int):
        if length <= 0:
            raise ValueError("Length must be greater than 0")
        super().__init__(character, color_position)
        self.__length = length
        self.__offset = int(length / 2 + 1)

    def get_2d_representation(self) -> list:
        result = ""
        for row in range(self.__length):
            for col in range(self.__length):
                if row == 0 or row == self.__length - 1:
                    result += f"{self._character}  "
                elif col == 0 or col == self.__length - 1:
                    result += f"{self._character}  "
                else:
                    result += "   "
            result += "\n"

        return [(Fore.__getattribute__(colors[self._color_position]) + "\n" +
result) for _ in range(6)]
```

```
    def get_3d_representation(self, scale: float = 1.0) -> str:
        modified_length = int(self.__length * scale) if self.__length * scale >=
2 else self.__length
        modified_offset = int(modified_length / 2 + 1)
        result = ""

        for row in range(modified_offset - 1):
            for col in range(modified_length + modified_offset - 1):
                if (row + col == modified_offset - 1) or (row == 0 and col >
modified_offset - 1):
                    result += f"{self._character}" + (
                        "" if col == modified_length + modified_offset - 2 and
row == 0 else "  ")
                elif modified_length + modified_offset - row == col + 2:
                    result += f"{self._character}"
                elif col == modified_length + modified_offset - 2:
                    result += f"  {self._character}"
                else:
                    result += "   "
            result += "\n"

        for row in range(modified_length):
            for col in range(modified_length + modified_offset):
                if ((row == 0 or row == modified_length - 1) and col <
modified_length or (
                        col == 0 or col == modified_length - 1) and row <
modified_length and col < modified_length):
                    result += f"{self._character}" + (
                        "" if row == modified_length - 1 and col ==
modified_length - 1 else "  ")
                elif row + col == (modified_length - 1) * 2 and col <
modified_length + modified_offset - 1:
                    result += "   " * (modified_length - row - 2) +
f"{self._character}"
                elif col < modified_length and row < modified_length:
                    result += "   "
                elif row < modified_length - modified_offset and col >
modified_length:
                    if col == modified_offset + modified_length - 1:
                        result += f"{self._character}"
                    else:
                        result += "   "

            result += "\n"

        return Fore.__getattribute__(colors[self._color_position]) + "\n" +
result
```

Висновки. Виконуючи ці завдання, ви створите високорівневий об'єктно-орієнтований генератор 3D ASCII-арту, який дозволить користувачам проектувати, відображати та маніпулювати 3D-фігурами в ASCII-арті. Цей проект надасть вам глибоке розуміння об'єктно-орієнтованого програмування і алгоритмів графіки, сприятиме творчому підходу до створення ASCII-арту.