# **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА” ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ** **Кафедра ІСМ**

# **Звіт**

# **до лабораторної роботи №5**

# **З дисципліни “Спеціальні мови програмування”**

**Виконав:  
ст.гр.ІТ-31  
Шельвах Максим  
  
Прийняв:  
Щербак С.С.**

# **Львів — 2023**

**Тема роботи:** Розробка ASCII ART генератора для візуалізації 3D-фігур.

**Мета роботи:** Cтворення 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.

Хід роботи

**Реалізований код:**

figure\_art\_interface.py

from abc import ABC, abstractmethod

from generator.figures.figure import Figure

from generator.figures.figures\_3d.cube import Cube

from generator.figures.figures\_2d.square import Square

from generator.colors.colors import textFileSaver

class Command(ABC):

@abstractmethod

def execute(self):

pass

class Generate3DFigureCommand(Command):

def \_\_init\_\_(self, figure\_interface):

self.figure\_interface = figure\_interface

def execute(self):

print(self.figure\_interface.generate3dFigure())

class SetSizeCommand(Command):

def \_\_init\_\_(self, figure\_interface, new\_size):

self.figure\_interface = figure\_interface

self.new\_size = new\_size

def execute(self):

self.figure\_interface.setSize(self.new\_size)

class SetColorCommand(Command):

def \_\_init\_\_(self, figure\_interface, new\_color):

self.figure\_interface = figure\_interface

self.new\_color = new\_color

def execute(self):

self.figure\_interface.setColor(self.new\_color)

class FigureArtInterface(Figure):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.type = "cube"

self.left\_padding = 5

self.top\_padding = 5

self.bottom\_padding = 5

self.commands = {

1: Generate3DFigureCommand(self),

2: SetSizeCommand(self, 0),

3: SetColorCommand(self, "")

}

def setType(self, type):

self.type = type

def setPaddings(self, left\_padding, top\_padding, bottom\_padding):

self.left\_padding = left\_padding

self.top\_padding = top\_padding

self.bottom\_padding = bottom\_padding

def setPrimaryData(self):

size = int(input("figure size: "))

self.setSize(size)

color = input(

"figure color(blue, green, red, magenta, yellow, white, cyan): ")

self.setColor(color)

type = input("figure type(skip for default): ")

if len(type):

self.setType(type)

def generateWithLeftPadding(self, text):

lines = text.split('\n')

padded\_lines = [f"{' ' \* self.left\_padding}{line}" for line in lines]

return '\n'.join(padded\_lines)

def generateWithTopPadding(self, text):

space = self.top\_padding \* '\n'

padded\_lines = space + text

return padded\_lines

def generateWithBottomPadding(self, text):

space = self.bottom\_padding \* '\n'

padded\_lines = text + space

return padded\_lines

def generateWithPaddings(self, text):

return self.generateWithBottomPadding(self.generateWithTopPadding(self.generateWithLeftPadding(text)))

def generate3dFigure(self):

if self.type == 'cube':

cube = Cube()

cube.setSize(self.size)

cube.setColor(self.color)

return self.generateWithPaddings(cube.generateFigure())

return super().generateFigure()

def generate2dFigure(self):

if self.type == 'cube':

square = Square()

square.setSize(self.size)

square.setColor(self.color)

return self.generateWithPaddings(square.generateFigure())

return super().generateFigure()

def saveToFile2d(self):

filename = input("enter filename before saving: ")

textFileSaver(

filename, self.generate2dFigure())

def saveToFile3d(self):

filename = input("enter filename before saving: ")

textFileSaver(

filename, self.generate3dFigure())

@staticmethod

def show\_menu():

print("choose menu option")

print("[ 1 ] - generate 3d figure")

print("[ 2 ] - set size")

print("[ 3 ] - set color(blue, green, red, magenta, yellow, white, cyan)")

print("[ 4 ] - set type(cube)")

print("[ 5 ] - set paddings")

print("[ 6 ] - get 2d version of the figure")

print("[ 7 ] - save to file(3d)")

print("[ 8 ] - save to file(2d)")

print("[ 0 ] - exit")

def loopMenu(self):

while True:

self.show\_menu()

menu\_choice = int(input("menu key: "))

if (menu\_choice == 1):

print(self.generate3dFigure())

elif (menu\_choice == 2):

new\_size = int(input("enter new size: "))

self.setSize(new\_size)

elif (menu\_choice == 3):

new\_color = input("enter new color: ")

self.setColor(new\_color)

elif (menu\_choice == 4):

new\_type = input("enter new type: ")

self.setType(new\_type)

elif (menu\_choice == 5):

left\_padding = int(input("enter left padding: "))

top\_padding = int(input("enter top padding: "))

bottom\_padding = int(input("enter bottom padding: "))

self.setPaddings(left\_padding, top\_padding, bottom\_padding)

elif (menu\_choice == 6):

print(self.generate2dFigure())

elif (menu\_choice == 7):

self.saveToFile3d()

elif (menu\_choice == 8):

self.saveToFile2d()

else:

break

def launch(self):

self.setPrimaryData()

print(self.generate3dFigure())

self.loopMenu()

main.py

from generator.figure\_art\_interface.figure\_art\_interface import FigureArtInterface

def main():

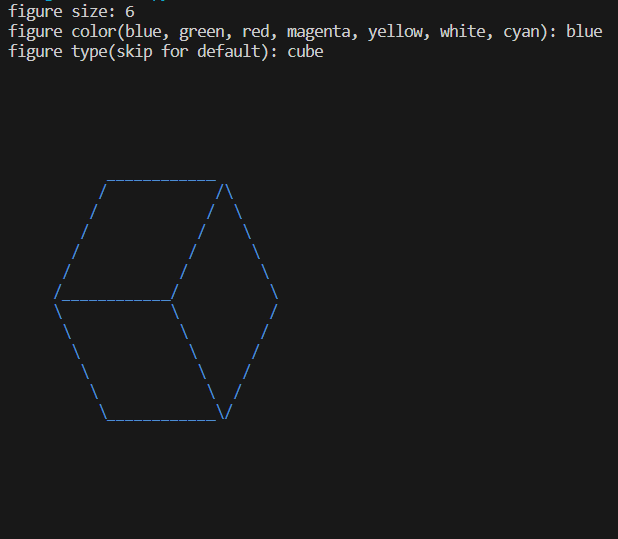
figure\_art\_interface = FigureArtInterface()

figure\_art\_interface.launch()

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Результат виконання програми:**



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