ЗВІТ
про виконання лабораторної роботи № 3
«Розробка ASCII ART генератора для візуалізації текстових даних»
з дисципліни
«Спеціалізовані мови програмування»
ст. групи РІ-31
Танечник Марічки

Мета: створення додатка Генератора ASCII-арту.
Умова завдання:

Завдання 1: Введення користувача
Створіть Python-програму, яка приймає введення користувача для слова або фрази, яку треба перетворити в ASCII-арт.

Завдання 2: Бібліотека ASCII-арту
Інтегруйте бібліотеку ASCII-арту (наприклад, pyfiglet або art) у вашу програму для генерації ASCII-арту з введення користувача

Завдання 3: Вибір шрифту
Дозвольте користувачам вибирати різні стилі шрифтів для свого ASCII-арту. Надайте список доступних шрифтів та дозвольте їм вибрати один.
Завдання 4: Колір тексту
Реалізуйте опцію вибору користувачем кольору тексту для їхнього ASCII-арту. Підтримуйте основний вибір кольорів (наприклад, червоний, синій, зелений).

Завдання 5: Форматування виводу
Переконайтеся, що створений ASCII-арт правильно відформатований та вирівнюється на екрані для зручності читання.

Завдання 6: Збереження у файл
Додайте функціональність для збереження створеного ASCII-арту у текстовому файлі, щоб користувачі могли легко завантажувати та обмінюватися своїми твореннями.

Завдання 7: Розмір ARTу
Дозвольте користувачам вказувати розмір (ширина і висота) ASCII-арту, який вони хочуть створити. Масштабуйте текст відповідно.

Завдання 8: Вибір символів
Дозвольте користувачам вибирати символи, які вони хочуть використовувати для створення ASCII-арту (наприклад, '@', '#', '*', тощо).

Завдання 9: Функція попереднього перегляду
Реалізуйте функцію попереднього перегляду, яка показує користувачам попередній перегляд їхнього ASCII-арту перед остаточним збереженням.

Завдання 10: Інтерфейс, зрозумілий для користувача
Створіть зручний для користувача інтерфейс командного рядку для додатка, щоб зробити його інтуїтивно зрозумілим та легким у використанні.

Текст програми:

```
import random
```

```
import string
from ..shared.validators import Validators
from ..shared.incorrect_character_exception import
IncorrectCharacterException


class Ascii:
    def __init__(self, text, width = 0, height = 0, color = "\033[39m",
shadow = "#", text_s = "#", highlight = "#", justify = "left"):
        self.text = text
        self.shadow = shadow
        self.text_s = text_s
        self.highlight = highlight
        self.height = height
        self.width = Validators.verify_width(width)
        self.color = "\033[" + str(random.randint(31, 39)) + "m" if color
== "random" else color
        self.color_reset = "\033[0m"
        self.justify = justify
        self.font = self.__load_font()

    def print(self):
        art = self.__format_art()
        print(self.color + art + self.color_reset)
        return art

    def __wrap_art(self):
        wrapped_text = []
        length = 0
        current_line = ""
        for char in self.text.upper():
            if char in ["@", "#"]:
                length += 9
            if char in ["M", "W", "4", "*"]:
                length += 8
            elif char in self.font:
                length += 7
```

```
            else:
                raise IncorrectCharacterException("The character " + char
+ " is not a valid character.")
            if length > self.width:
                wrapped_text.append(current_line)
                current_line = char
                length = 8 if char in ["M", "W", "4"] else 7
            else:
                current_line += char
        if current_line:
            wrapped_text.append(current_line)
        return wrapped_text

    def __format_art(self):
        wrapped_art = self.__wrap_art()
        art_lines = []
        for chunk in wrapped_art:
            unsorted_art_list = []
            for art_char in chunk:
                font_art = self.font[art_char.upper()]
                formatted_font_art = (font_art.replace('*', self.highlight)
                                      .replace('#', self.text_s).replace('&',
self.shadow))
                split_font_art = formatted_font_art.splitlines()
                unsorted_art_list.append(split_font_art)
            art_list = []
            for row in zip(*unsorted_art_list):
                row_str = "".join(row)
                match self.justify:
                    case "left":
                        art_list.append(row_str)
                    case "center":
                        width = self.width - len(row_str)
                        padding = " " * (width // 2)
                        art_list.append(padding + row_str + padding)
                    case "right":
                        width = self.width - len(row_str)
```

```
                    padding = " " * width
                    art_list.append(padding + row_str)
            art_lines.append("\n".join(art_list))
        art = "\n\n".join(art_lines)
        art_height = len(art.splitlines())
        height_diff = self.height - art_height
        padding = "\n" * (height_diff // 2) if height_diff > 0 else ""
        return padding + art + padding

    @staticmethod
    def __load_font():
        keys = (list(string.ascii_uppercase) + list(string.digits) +
                ["!", "@", "#", "$", "%", "^", "&", "*", "(", ")", "-", "_", "=",
                 "+", "[", "]", ";", ":", '"', "'", ",", ".", "/", "<", ">", "?", " "])
        font = {}
        with open("Sources/font.txt", "r") as file:
            i = 0
            for line in file:
                if line.strip() == "$":
                    i+= 1
                elif i < len(keys):
                    key = keys[i]
                    font[key] = font.get(key, "") + line
        return font

import random
import textwrap
import GlobalVariables as Global
from BLL.classes.ascii import Ascii
from pyfiglet import FigletFont, figlet_format
from DAL.functions.upload_to_file import file_upload


class Console:
    @staticmethod
    def prompt():
        Ascii.print("ASCIIFY", True)
        while True:
```

```
        prompt = input("1 - enter text\n"
                       "2 - random font\n"
                       "3 - change font\n"
                       "4 - change width and height\n"
                       "5 - change color\n"
                       "Your choice: ")
        match prompt:
            case "1":
                Console.enter_text()
            case "2":
                Console.auto_font()
            case "3":
                Console.change_font()
            case "4":
                Console.change_width_and_height()
            case "5":
                Console.change_color()
            case _:
                return

    @staticmethod
    def enter_text():
        text = input("Enter text: ")
        ftext = Ascii.print(text)
        save_prompt = input("Do you want to save the text? (y/n): 
").lower()
        if save_prompt == "y":
            try:
                file_upload(ftext)
            except IOError:
                print("An error occurred during file upload")

    @staticmethod
    def auto_font():
        text = input("Enter text: ")
        symbols = input("Enter a set of characters that should be in the
ASCII art: ")
```

```
        font_symbols = set(symbols) | {" ", "\n"}
        fonts = FigletFont.getFonts()
        random.shuffle(fonts)
        for font in fonts:
            random_art = figlet_format(text, font=font,
width=Global.width)
            random_art_chars = set(random_art)
            if all(char in [" ", "\n"] for char in random_art_chars):
                continue
            elif all(char in font_symbols for char in random_art_chars):
                print("Found font:" + font)
                Global.font = font
                ftext = Ascii.print(text)
                save_prompt = input("Do you want to save the text? (y/n):
").lower()
                if save_prompt == "y":
                    try:
                        file_upload(ftext)
                    except IOError:
                        print("An error occurred during file upload")
                return
        print("No fonts were found, please try again with a wider set of
characters")

    @staticmethod
    def change_font():
        new_font = input("Enter the new font you want to choose\n"
                         "You can also use 'font' to see all fonts available or
'random' to choose a random font\n"
                         "Your choice: ")
        if new_font in FigletFont.getFonts():
            Global.font = new_font
            print("Font changed successfully")
        elif new_font.lower() == "font":
            print("Available fonts:\n" + textwrap.fill(",
".join(FigletFont.getFonts()), width=Global.width))
        elif new_font.lower() == "random":
```

```
            Global.font = random.choice(FigletFont.getFonts())
            print("Randomly selected font: " + Global.font)
        else:
            print("Invalid font")

    @staticmethod
    def change_width_and_height():
        while True:
            width_prompt = input("Enter the width of an ASCII art\n"
                                 "(any non-positive value will reset it to default
values\n"
                                 "Your choice: ")
            try:
                width = int(width_prompt)
                Global.width = Ascii.verify_width(width)
                print("Width changed successfully")
            except ValueError:
                print("Please enter an integer")
                continue
            height_prompt = input("Enter the height of an ASCII art\n"
                                  "(any non-positive value will reset it to default
values\n"
                                  "Your choice: ")
            try:
                height = int(height_prompt)
                Global.height = height
                print("Height changed successfully")
                break
            except ValueError:
                print("Please enter an integer")
                continue

    @staticmethod
    def change_color():
        color_prompt = input("Enter the color of your ASCII art:\n"
                             "1 - Red\n"
                             "2 - Green\n"
```

```
                    "3 - Yellow\n"
                    "4 - Blue\n"
                    "5 - Magenta\n"
                    "6 - Cyan\n"
                    "7 - Light gray\n"
                    "0 - Default\n"
                    "Your choice: ")
    match color_prompt:
        case "1":
            Global.color = "\033[31m"
        case "2":
            Global.color = "\033[32m"
        case "3":
            Global.color = "\033[33m"
        case "4":
            Global.color = "\033[34m"
        case "5":
            Global.color = "\033[35m"
        case "6":
            Global.color = "\033[36m"
        case "7":
            Global.color = "\033[37m"
        case "0":
            Global.color = "\033[39m"
        case _:
            print("Invalid color choice, please try again.")
            return
    print("Color changed successfully")
```

Висновки: Виконуючи ці завдання, я створила універсальний Генератор ASCII-арту, який дозволить користувачам налаштовувати свої творіння з різними шрифтами, кольорами, розмірами та символами.