PRELIM OUTPUT

Dequito, Kyla Dessirei L.
M001 || CP102 || MW – 4:00 – 6:30

Task 1: Object-Oriented Programming (OOP) in Python

Description:

- This simple Tattoo Appointment Booker program utilizes Object-Oriented Programming (OOP) principles to manage tattoo artists and appointments. It allows users to add tattoo artists, view available artists, book tattoo appointments, and view scheduled appointments. The program employs two classes: TattooArtist, which includes attributes like artist name, specialization, and hourly rate, and TattooAppointment, which stores client name, tattoo design, appointment duration, and calculates the total cost based on the artist's hourly rate. The use of private attributes, getters, and setters ensures encapsulation, allowing safe access and modification of object properties.

Objectives:

- Demonstrate the use of OOP concepts such as classes, attributes, constructors, methods, and encapsulation.
- Utilize private attributes with getter and setter methods to enforce controlled access.
- Allow users to interact with a menu system for adding tattoo artists, booking appointments, and viewing the details of both.

Source Code (OOP):

Tattoo Artist Class

```python
# -- CLASS TATTOO ARTIST --
class TattooArtist:
    def __init__(self, name: str, specialization: str, hourly_rate: float):
        self.__name = name
        self.__specialization = specialization
        self.__hourly_rate = hourly_rate

    # Getters
    @property
    def name(self):
        return self.__name

    @property
    def specialization(self):
        return self.__specialization

    @property
    def hourly_rate(self):
        return self.__hourly_rate

    # Setters
    @name.setter
    def name(self, new_name):
        if isinstance(new_name, str) and new_name.strip():
            self.__name = new_name
        else:
            raise ValueError("Invalid NAME!")

    @specialization.setter
    def specialization(self, new_specialization):
        if isinstance(new_specialization, str) and new_specialization.strip():
            self.__specialization = new_specialization
        else:
            raise ValueError("Invalid SPECIALIZATION!")

    @hourly_rate.setter
    def hourly_rate(self, new_rate):
        if isinstance(new_rate, (int, float)) and new_rate > 0:
            self.__hourly_rate = new_rate
        else:
            raise ValueError("Invalid HOURLY RATE! Must be a positive number.")

    # Methods
    def display_artist_info(self):
        return (f'Tattoo Artist: {self.__name}\n'
                f'Artist Specialization: {self.__specialization}\n'
                f'Hourly Rate: ₱{self.__hourly_rate:.2f}')
```

Tattoo Appointment Class

```python
# -- CLASS TATTOO APPOINTMENT --
class TattooAppointment:
    def __init__(self, client_name: str, tat_design: str, duration: float, artist: TattooArtist):
        self.__client_name = client_name
        self.__tat_design = tat_design
        self.__duration = duration
        self.__artist = artist

    # Getters
    @property
    def client_name(self):
        return self.__client_name

    @property
    def tat_design(self):
        return self.__tat_design

    @property
    def duration(self):
        return self.__duration

    @property
    def artist(self):
        return self.__artist

    # Setters
    @client_name.setter
    def client_name(self, new_name):
        if isinstance(new_name, str) and new_name.strip():
            self.__client_name = new_name
        else:
            raise ValueError("Invalid CLIENT NAME!")

    @tat_design.setter
    def tat_design(self, new_design):
        if isinstance(new_design, str) and new_design.strip():
            self.__tat_design = new_design
        else:
            raise ValueError("Invalid TATTOO DESIGN!")

    @tat_design.setter
    def tat_design(self, new_design):
        if isinstance(new_design, str) and new_design.strip():
            self.__tat_design = new_design
        else:
            raise ValueError("Invalid TATTOO DESIGN!")

    @duration.setter
    def duration(self, new_duration):
        if isinstance(new_duration, (int, float)) and new_duration > 0:
            self.__duration = new_duration
        else:
            raise ValueError("Invalid DURATION! Must be a positive number.")

    @artist.setter
    def artist(self, new_artist: TattooArtist):
        if isinstance(new_artist, TattooArtist):
            self.__artist = new_artist
        else:
            raise ValueError("Invalid ARTIST!")

    # Methods
    def appointment_details(self):
        return (f'Appointment for {self.__client_name}\n'
                f'Tattoo Design: {self.__tat_design}\n'
                f'Duration: {self.__duration} hour(s)\n'
                f'Tattoo Artist: {self.__artist.name} ({self.__artist.specialization})')

    def calculate_cost(self):
        return self.__artist.hourly_rate * self.__duration

```

Main Program

```python
# -- MAIN PROGRAM --
artists = []
appointments = []

def main():
    while True:
        print("\nTattoo Studio Booking System")
        print("[1] Add Tattoo Artist")
        print("[2] View Tattoo Artists")
        print("[3] Book Tattoo Appointment")
        print("[4] View Appointments")
        print("[5] Exit")

        choice = input("Enter your choice: ").strip()

        if choice == "1":
            add_tattoo_artist()
        elif choice == "2":
            display_artists()
        elif choice == "3":
            add_tattoo_appointment()
        elif choice == "4":
            display_appointments()
        elif choice == "5":
            print("\nExiting... Have a great day!")
            break
        else:
            print("Invalid choice. Please enter a number between 1-5.")


def add_tattoo_artist():
    print('\nAdd Tattoo Artist')
    name = input("Enter artist name: ").strip()
    specialization = input("Enter specialization: ").strip()

    while True:
        try:
            hourly_rate = float(input("Enter hourly rate (₱): "))
            if hourly_rate > 0:
                break
            else:
                print("Hourly rate must be a positive number.")
        except ValueError:
            print("Invalid input! Please enter a numeric value.")

    artist = TattooArtist(name, specialization, hourly_rate)
    artists.append(artist)
    print(f'Artist "{name}" has been added successfully!\n')

```

```
def add_tattoo_appointment():
    if not artists:
        print('\nNo tattoo artists available! Add an artist first.\n')
        return

    print('\nBook a Tattoo Appointment')
    client_name = input("Enter client name: ").strip()
    tat_design = input("Enter tattoo design: ").strip()

    while True:
        try:
            duration = float(input("Enter duration (in hours): "))
            if duration > 0:
                break
            else:
                print("Duration must be a positive number.")
        except ValueError:
            print("Invalid input! Please enter a numeric value.")

    print('\nAvailable Artists:')
    for i, artist in enumerate(artists, 1):
        print(f'{i}. {artist.name} - {artist.specialization} (₱{artist.hourly_rate}/hr)')

    while True:
        try:
            artist_choice = int(input("Select an artist by number: ")) - 1
            if 0 <= artist_choice < len(artists):
                break
            else:
def display_artists():
    if not artists:
        print("\nNo tattoo artists available.\n")
        return
    print("\nAvailable Tattoo Artists:")
    for i, artist in enumerate(artists, 1):
        print(f"{i}. {artist.display_artist_info()}\n")


def display_appointments():
    if not appointments:
        print("\nNo appointments scheduled.\n")
        return
    print("\nScheduled Appointments:")
    for i, appointment in enumerate(appointments, 1):
        print(f"{i}. {appointment.appointment_details()}")
        print(f"Total Cost: ₱{appointment.calculate_cost():.2f}\n")


# -- RUN PROGRAM --
if __name__ == "__main__":
    main()
```

Sample Output:

```
PS C:\Users\User\Desktop\CP102 PRELIM EXAM>
py"

Tattoo Studio Booking System
[1] Add Tattoo Artist
[2] View Tattoo Artists
[3] Book Tattoo Appointment
[4] View Appointments
[5] Exit
Enter your choice: 1

Add Tattoo Artist
Enter artist name: Weaver
Enter specialization: Blackwork
Enter hourly rate (₱): 1750
Artist "Weaver" has been added successfully


Tattoo Studio Booking System
[1] Add Tattoo Artist
[2] View Tattoo Artists
[3] Book Tattoo Appointment
[4] View Appointments
[5] Exit
Enter your choice: 1

Add Tattoo Artist
Enter artist name: Luna
Enter specialization: Japanese Traditional
Enter hourly rate (₱): 1600
Artist "Luna" has been added successfully!
```

```
Tattoo Studio Booking System
[1] Add Tattoo Artist
[2] View Tattoo Artists
[3] Book Tattoo Appointment
[4] View Appointments
[5] Exit
Enter your choice: 2

Available Tattoo Artists:
1. Tattoo Artist: Weaver
Artist Specialization: Blackwork
Hourly Rate: ₱1750.00

2. Tattoo Artist: Luna
Artist Specialization: Japanese Traditional
Hourly Rate: ₱1600.00
```

```
Tattoo Studio Booking System
[1] Add Tattoo Artist
[2] View Tattoo Artists
[3] Book Tattoo Appointment
[4] View Appointments
[5] Exit
Enter your choice: 3

Book a Tattoo Appointment
Enter client name: Namei
Enter tattoo design: Wave-like Koi
Enter duration (in hours): 3

Available Artists:
1. Weaver - Blackwork (₱1750.0/hr)
2. Luna - Japanese Traditional (₱1600.0/hr)
Select an artist by number: 2

Appointment for Namei booked with Luna!


Tattoo Studio Booking System
[1] Add Tattoo Artist
[2] View Tattoo Artists
[3] Book Tattoo Appointment
[4] View Appointments
[5] Exit
Enter your choice: 3

Book a Tattoo Appointment
Enter client name: Tato
Enter tattoo design: Tribal
Enter duration (in hours): 1

Available Artists:
1. Weaver - Blackwork (₱1750.0/hr)
2. Luna - Japanese Traditional (₱1600.0/hr)
Select an artist by number: 1

Appointment for Tato booked with Weaver!
```

```
Tattoo Studio Booking System
[1] Add Tattoo Artist
[2] View Tattoo Artists
[3] Book Tattoo Appointment
[4] View Appointments
[5] Exit
Enter your choice: 4

Scheduled Appointments:
1. Appointment for Namei
Tattoo Design: Wave-like Koi
Duration: 3.0 hour(s)
Tattoo Artist: Luna (Japanese Traditional)
Total Cost: ₱4800.00

2. Appointment for Tato
Tattoo Design: Tribal
Duration: 1.0 hour(s)
Tattoo Artist: Weaver (Blackwork)
Total Cost: ₱1750.00


Tattoo Studio Booking System
[1] Add Tattoo Artist
[2] View Tattoo Artists
[3] Book Tattoo Appointment
[4] View Appointments
[5] Exit
Enter your choice: 5

Exiting... Have a great day!
PS C:\Users\User\Desktop\CP102 PRELIM EXAM>
```

Task 2: Regular Expressions (RegEx) in Python

Description:

This program was designed to make use of the functions of Regular Expressions (RegEx) on a text file called ‘bee_movie_and_bee_facts.txt’. Using built-in file handling functions the file will be opened and closed safely once the program is done with it and read the contents of the file without issue using the encoding method UTF-8. The program utilizes both *search()* and *findall()* functions alongside RegEx patterns to identify facets of the text file such as instances of the word bee and honey or lines that start with the word honey regardless of sentence-case. It also performs data processing by computing the sum of the numbers found, its average and the min/max of the numbers found. Data processing was also done when the amount of the word bee and honey were found, which was then added for its sum and computed for its average.

Objectives:

- Perform file handling for reading and processing of text files.
- Make use of various metacharacters to define search patterns.
- Make use of RegEx functions (*search()* and *findall()*) to extract relevant data.
- Perform data processing operations from the extracted data. (counting matches, sums, min/max values, averages)

Source Code (RegEx):

```
37
38  #Displaying outputs
39  print("-_-" * 25 + '\n')
40
41  print(f"Unique Movie Characters ({len(unique_characters)} Found):\n{sorted(unique_characters)}")
42
43  print("-_-" * 25 + '\n')
44
45  print(f'All instances of the word Bee found: {len(bee_list)}')
46  print(f'All instances of the word Honey-related words found: {len(honey_list)}')
47  print(f'The sum of bees and honey: {len(bee_list) + len(honey_list)}')
48  print(f'The average of bees and honey: {(len(bee_list) + len(honey_list))/2:.2f}')
49
50  print("-_-" * 25 + '\n')
51
52  print(f'The amount of numbers found: {len(num_list)}')
53  print(f'The sum of all numbers: {sum(num_list)}')
54  print(f'The highest number: {max(num_list)}')
55  print(f'The lowest number: {min(num_list)}')
56  print(f'The average of the numbers: {sum(num_list)/len(num_list):.2f}')
```

Sample Output:

```
PS C:\Users\User\Desktop\CP102 PRELIM EXAM> & C:/Users/User/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/U
sers/User/Desktop/CP102 PRELIM EXAM/Dequito - Task 2 (RegEx).py"
er/Desktop/CP102 PRELIM EXAM/Dequito - Task 2 (RegEx).py"
-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-

Everytime Honey appears at the start of a line.

Honey begins when our valiant Pollen
Honey!
```

```
1   import re
2
3   unique_characters = set()
4   num_list = []
5   bee_list =[]
6   honey_list = []
7
8   with open("bee_movie_and_bee_facts.txt", 'r', encoding='utf-8') as file:
9
10      print("-_-" * 25 + "\n")
11      print("Everytime Honey appears at the start of a line.\n")
12      for line in file:
13          line = line.rstrip()
14
15          #Searches for lines that start with the word Honey
16          if re.search(r'^honey', line, re.IGNORECASE):
17              print(line)
18
19          #Finds all unique characters in the film
20          matches = re.findall(r'^[A-Z0-9]{2,}:$', line)
21          unique_characters.update({match[:-1].capitalize() for match in matches}) #Updates to set to avoid dupes, removes colon, capitalizes
22
23          #Finds all instances of the word bee regardless or case or if it is in another word (Ex. honeybee, bee-ing, etc.)
24          bee = re.findall(r'bee', line, re.IGNORECASE)
25          if len(bee) > 0:
26              bee_list += bee
27
28          #Finds all instances of the word honey regardless or case or if it is in another word (Ex. honeycomb, honey, etc.)
29          honey = re.findall(r'honey', line, re.IGNORECASE)
30          if len(honey) > 0:
31              honey_list += honey
32
33          #Finds all digits in the text file
34          num = re.findall(r'\d+(?:,\d+)*', line)
35          for n in num:
36              num_list.append(int(n.replace(',', ''))) #Removes commas and convert to int
```

Source Text (excerpt):

bee_movie_and_bee_facts.txt

```
Bee Movie
By Jerry Seinfeld

NARRATOR:
(Black screen with text; The sound of buzzing bees can be heard)
According to all known laws
of aviation,
 :
there is no way a bee
should be able to fly.
 :
Its wings are too small to get
its fat little body off the ground.
 :
The bee, of course, flies anyway
 :
because bees don't care
what humans think is impossible.
BARRY BENSON:
(Barry is picking out a shirt)
Yellow, black. Yellow, black.
Yellow, black. Yellow, black.
 :
Ooh, black and yellow!
Let's shake it up a little.
JANET BENSON:
Barry! Breakfast is ready!
BARRY:
Coming!
 :
Hang on a second.
(Barry uses his antenna like a phone)
 :
Hello?
ADAM FLAYMAN:

(Through phone)
- Barry?
BARRY:
- Adam?
```

References:

Bee Movie Script (University of Washington. (2020). Bee Movie Script. Retrieved from https://courses.cs.washington.edu/courses/cse163/20wi/files/lectures/L04/bee-movie.txt)
Bee Facts (ChatGPT. (2025). Bee Facts. OpenAI. Retrieved from https://chat.openai.com)