Name: Philip Liu
Date: February 28, 2024
Course: IT FDN 110 A

Assignment07 – Course Registration Program Menu with Data Handers.

Intro

Our Course Registration Program menu product line is ever so quickly experiencing iterations, we now have added Data Handling. *Data Handler* will be marked to an existing document to showcase implementation details.

1.1: SoC, Data Layer. Accessing Data at Rest and in Different Forms. Read & Write to File is Wrapped for Error Handling

```
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------
'''

FILE_NAME: str = "Enrollments.json"  # File name Constant

student_data: list = []  # List of data entered by user
menu_choice: str = ''  # Hold the choice made by the user
```

Continue to Page 2.

1.2: Main Data Object, Also Known As: Parent. *Data handler*
Contextually, imagine: A Person enters school's administration office. Administrators have no idea why until this Person states their business/purpose to be best profiled and then support accordingly.

```python
class Person:
    """
    Changelog: Philip Liu. 2/28/2024. Created Class.
    Base class of all (sub)classes
    that will inherit from this class
    Philip Liu. 2/28/2024. Created: Properties
    Philip Liu. 2/28/2024. Created: Setters
    """

    def __init__(self, first_name: str = "", last_name: str = ""):
        self.first_name = first_name # set the attribute using the property to provide validation
        self.last_name = last_name # set the attribute using the property to provide validation

    @property
    def first_name(self):
        return self._first_name.title()

    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha() or value == "":
            self._first_name = value
        else:
            raise ValueError("The first name should not contain numbers.")

        @property
        def last_name(self):
            return self._last_name.title()

        @last_name.setter
        def last_name(self, value: str):
            if value.isalpha() or value == "":
                self._last_name = value
            else:
                raise ValueError("The last name should not contain numbers.")

    def __str__(self):
        return f"{self.first_name} {self.last_name}"
```

Continue to Page 3.

1.2: Inheriting Data Based on Main Data Object. Also Known As: Child.
Contextually, imagine: The Person has stated their business/purpose, to register for classes. After registration process the Person is now a Student.

```python
class Student(Person):
    """
    Changelog: Philip Liu. 2/28/2024. Created Class.
    Subclass of Person class. Inherits from Person class.
    """
    def __init__(self, first_name: str = "", last_name: str = "", course_name: str = ""):
        super().__init__(first_name=first_name, last_name=last_name)
        self.course_name = course_name
```

Continue to Page 3.

1.4: Class for I/O – Input & Output. For this Section - Output of Mentioned Constant Menu. Input Function, start point for User Entries. Input is Coupled with Error Handling within the Bounds of Menu Options.

```python
class IO:
    """
    ChangeLog: Philip Liu. 2/21/2024. Created Class
    Collection of input, output, display functions to communicate with users
    """
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        print(message)

    @staticmethod
    def output_menu():
        """
        Display menu of choices
        """
        print(MENU)

    @staticmethod
    def input_menu_choice():
        """
        Get menu choice from the user
        """
        choice = 0
        try:
            choice = input("Enter your menu of choice: ")
            if choice not in ("1", "2", "3", "4"):
                raise ValueError("Please choose only options: 1, 2, 3, or 4")
        except ValueError as e:
            IO.output_error_messages(str(e))
        return choice  # Return the choice value
```

Continue to Page 5.

1.4: Function of Class I/O. Focused on Input from User with Error Handling. Goal: Set Boundaries of User Input.

```python
@staticmethod
def input_student_data():
    """
    Get student data from the user
    """
    try:
        student_first_name = input("Enter first name: ")
        if not student_first_name.isalpha():
            raise ValueError("No numbers, please. ")
        student_last_name = input("Enter last name: ")
        if not student_last_name.isalpha():
            raise ValueError("No numbers, please. ")
        course_name = input("Enter course name: ")
        students = {"First Name": student_first_name, "Last Name": student_last_name,
                    "Course": course_name}
        student_data.append(students)
    except ValueError as e:
        print("Error:", e)
    return student_data
```

1.5: Function of Class I/O. Focused on Output Reflecting Users Input.

```python
@staticmethod
def output_student_data(student_data: list):
    """
    Displays user's data
    """
    print("-" * 50)
    for student in student_data:
        if isinstance(student, Student):
            print(
                f"{student.first_name} {student.last_name} is enrolled into {student.course_name}"
            )
        else:
            print(student)
    print("-" * 50)
```

Continue to Page 6.

1.6: Application Layer Based on SoC, Separation of Concern. Computation Layer/Logic Layer.

```python
### Application Layer
def main():
    student_data = FileProcessor.read_data_from_file(FILE_NAME)

    while True:
        IO.output_menu()
        menu_choice = IO.input_menu_choice()

        if menu_choice == "1":
            student = IO.input_student_data()
            if student:
                student_data.append(student)
        elif menu_choice == "2":
            print("\nThe current data is: ")
            IO.output_student_data(student_data)
        elif menu_choice == "3":
            FileProcessor.write_data_to_file(FILE_NAME, student_data)
        elif menu_choice == "4":
            print("Exiting program...")
            break

        else:
            print("Please only choose options 1, 2, 3 or 4")



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

Conclusion

Data Handling enables us to structure our code. Build, from our elementary understanding point towards a mature and reusable state., and a more simplified hand-off. Iterations, demonstrates our step-by-step journey, and our passion to upscale, version up, be better than we started from.