Laboratory Activity No. 3	
Polymorphism	
Course Code: CPE009	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed: 30/09/2024**
Section: CPE21S4	**Date Submitted: 30/09/2024**
Name: TITONG, LEE IVAN B	**Instructor: Prof. Maria Rizette Sayo**

1. Objective(s):

This activity aims to familiarize students with the concepts of Polymorphism in Object-Oriented Programming

2. Intended Learning Outcomes (ILOs):

The students should be able to:

2.1 Identify the use of Polymorphism in Object-Oriented Programming

2.2 Implement an Object-Oriented Program that applies Polymorphism

3. Discussion:

Polymorphism is a core principle of Object-Oriented that is also called “method overriding”. Simply stated the principles says
that a method can be redefined to have a different behavior in different derived classees.

For an example, consider a **base file reader/writer** class then three derived classes **Text file reader/writer**, **CSV file reader/ writer**, and **JSON file reader/writer**. The base file reader/writer class has the methods: **read**(filepath=””) , **write**(filepath=””). The three derived classes (classes that would inherit from the base class) should have behave differently when their read, write methods are invoked.

CSV stands for **Comma Separated Values** while **JSON** stands for **Javascript Server Object Notation**. These are the standard file formats and structures used by applications and systems to transfer/exchange data between their systems. For example, you may visit this online api http://dummy.restapiexample.com/api/v1/employees (note that the data is fake) but this url provides data that another system can consume and use in their system.

4. Materials and Equipment:

Desktop Computer with
Anaconda Python Windows
Operating System

5. Procedure:

Creating the Classes

1. Create a folder named oopfa1<lastname>_lab8
2. Open your IDE in that folder.
3. Create the base FileReaderWriter .py file and Class using the code below:

FileReaderWriter.py > ...

```
1  class FileReaderWriter():
2      def read(self):
3          print("This is the default read method")
4
5      def write(self):
6          print("This is the default write method")
```

4. Create the CSVFileReaderWriter .py and Class using the code below:

CSVFileReaderWriter.py > ...

```python
from FileReaderWriter import FileReaderWriter
import csv

class CSVFileReaderWriter(FileReaderWriter):
    def read(self, filepath):
        with open(filepath, newline='') as csvfile:
            data = csv.reader(csvfile, delimiter=',', quotechar='|')
            for row in data:
                print(row)
            return data

    def write(self, filepath, data):
        with open(filepath, 'w', newline='') as csvfile:
            writer = csv.writer(csvfile, delimiter=',',
                                quotechar='|', quoting=csv.QUOTE_MINIMAL)
            writer.writerow(data)
```

5. Create the JSONFileReaderWriter Class using the code below

JSONFileReaderWriter.py > ...

```python
from FileReaderWriter import FileReaderWriter
import json

class JSONFileReaderWriter(FileReaderWriter):
    def read(self, filepath):
        with open(filepath, "r") as read_file:
            data = json.load(read_file)
            print(data)
            return data

    def write(self, filepath, data):
        with open(filepath, "w") as write_file:
            json.dump(obj=data, fp=write_file)
```

Testing and Observing Polymorphism

1. Create a .csv file named sample.csv with the following content. (you may use the IDE or plain notepad)

sample.csv

```
Apple,Banana,Mango,Orange,Cherry
```

2. Create a .json file named sample.json with the following content. (you may use the IDE or plain notepad)

sample.json

```
{
    "description":"This is a JSON Sample",
    "accounts": [
        {"id":1,"name":"Jack"},
        {"id":2,"name":"Rose"}
    ]
}
```

3. Create the main.py that will test the functionality of the classes.

main.py

```
from FileReaderWriter import FileReaderWriter
from CSVFileReaderWriter import CSVFileReaderWriter
from JSONFileReaderWriter import JSONFileReaderWriter

# Test the default class
df = FileReaderWriter()
df.read()
df.write()

# Test the polymoprhed methods
c = CSVFileReaderWriter()
c.read("sample.csv")
c.write(filepath="sample2.csv", data=["Hello","World"])

j = JSONFileReaderWriter()
j.read("sample.json")
j.write(data=['foo', {'bar': ('baz', None, 1.0, 2)}],filepath="sample2.json")
```

4. Run the program and observe the output carefully the values in sample2.csv and sample2.json

```
In [23]: runfile('C:/Users/TIPQC/Desktop/oopfa1TITONG_lab8/main.py', wdir='C:/Users/TIPQC/Desktop/
oopfa1TITONG_lab8')
This is the default read method
This is the default read method
['Apple', ' Banana', ' Mango', ' Orange', ' Cherry']
{'description': 'This is a JSON Sample', 'accounts': [{'id': 1, 'name': 'Jack'}, {'id': 2, 'name': 'Rose'}]}

In [24]:
```

sample2.csv

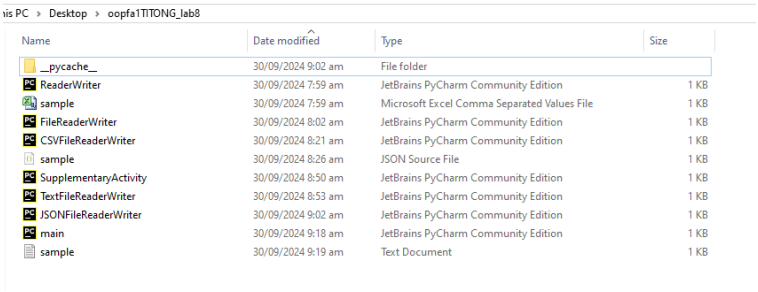
sample2.json

.

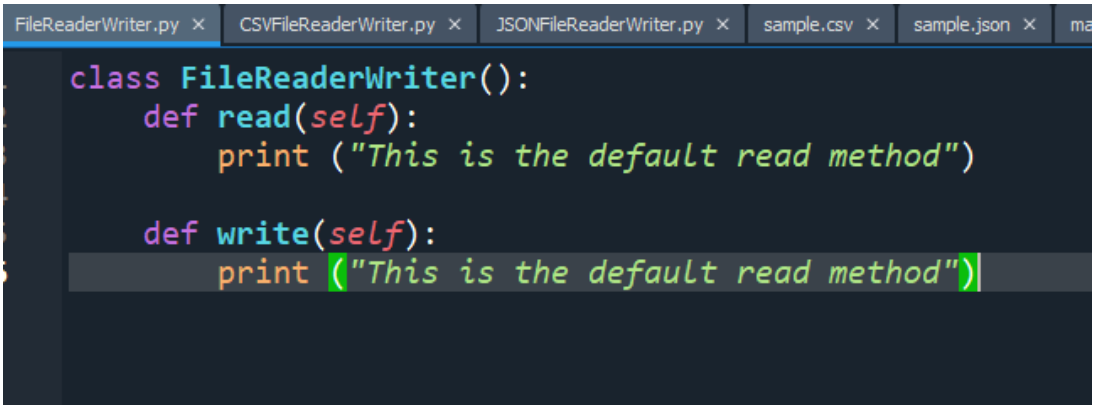
6. Supplementary Activity:

Task

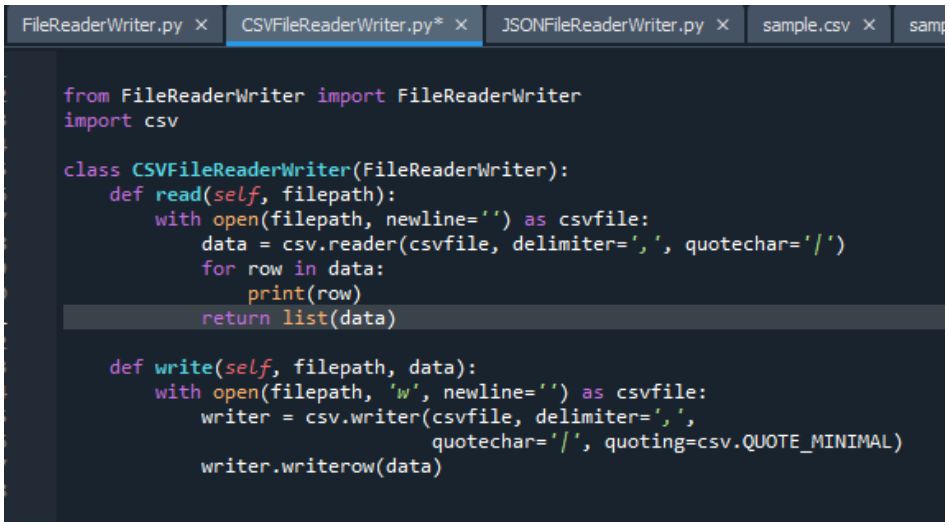
Create a simple TextFileReaderWriter .py file and Class that will be able to **read** from and **write** (override) to a text file. The read and write method should be overriden according to the requirement of Text File Reading and Writing as performed in Laboratory Activity 5.



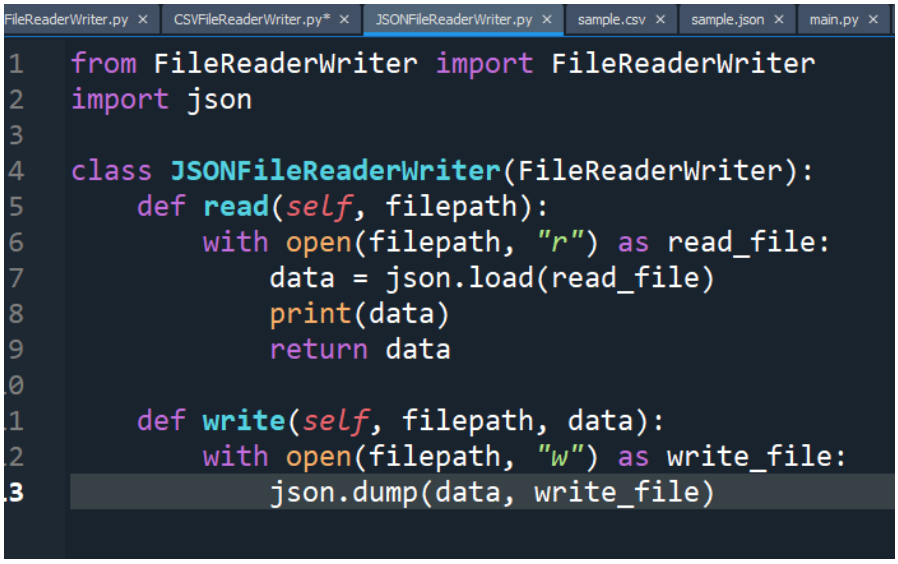
FileReaderWriter .py



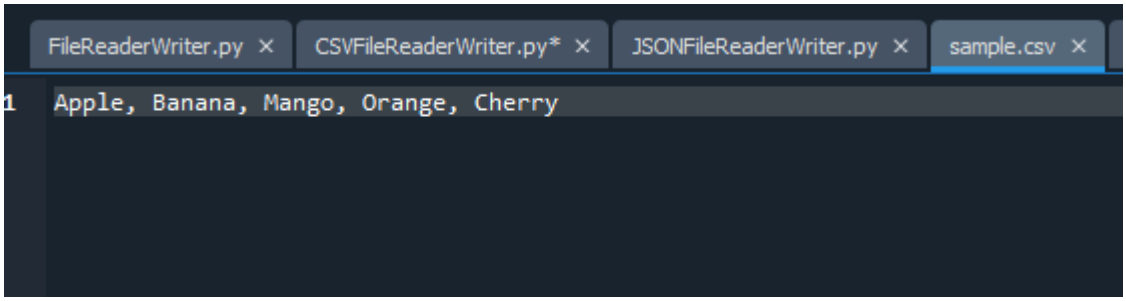
CSVFileReaderWriter .py



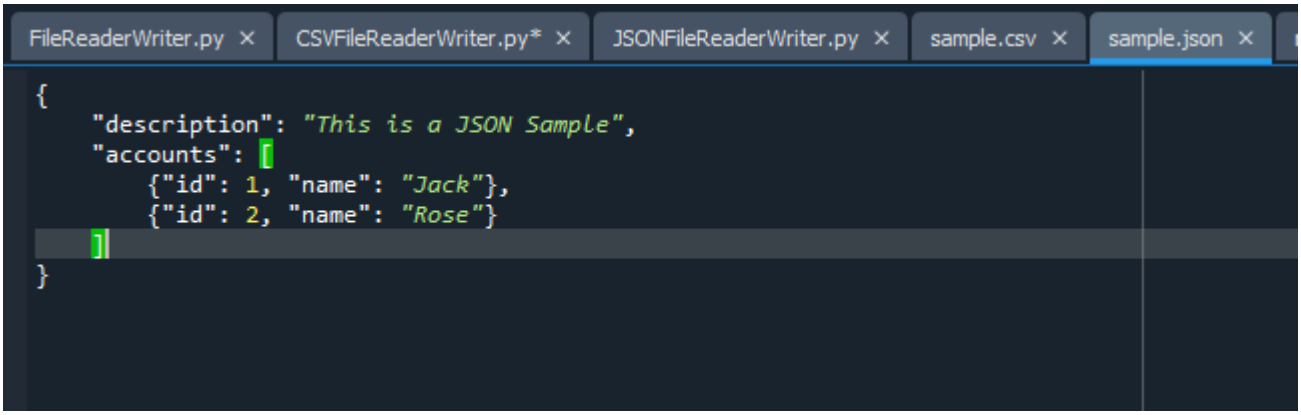
JSONFileReaderWriter



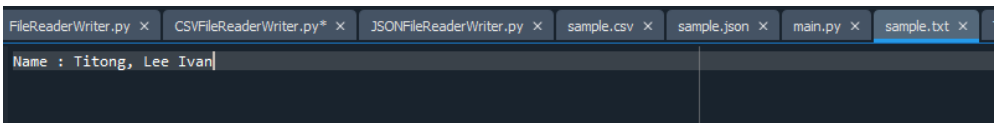
Sample.csv



sample.json



sample.txt



TextFileReaderWriter.py

FileReaderWriter.py | CSVFileReaderWriter.py* | JSONFileReaderWriter.py | sample.csv | sample.json | main.py | sample.txt | TextFileReaderWriter.py

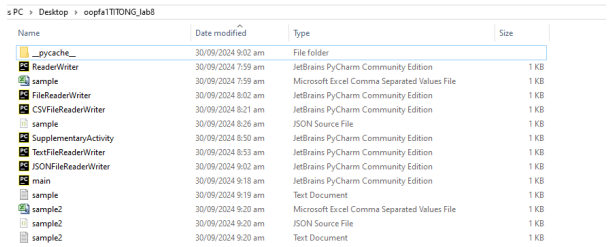
```python
from FileReaderWriter import FileReaderWriter

class TextFileReaderWriter(FileReaderWriter):
    def read(self, filepath):
        with open(filepath, 'r') as file:
            content = file.read()
            print(content)
            return content

    def write(self, filepath, data):
        with open(filepath, 'w') as file:
            file.write(data)

```

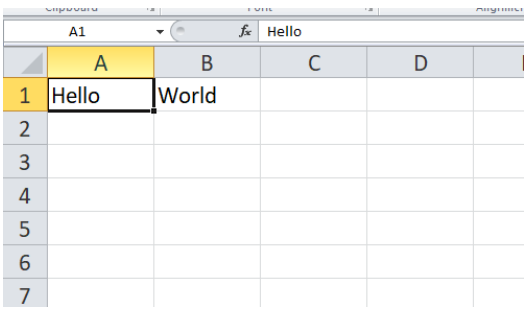
Main.py

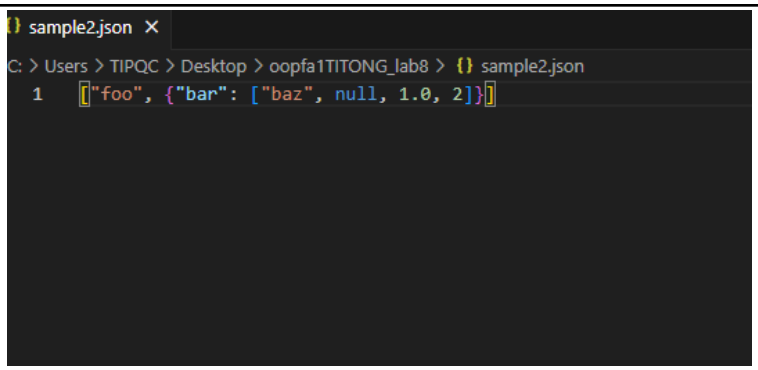
FileReaderWriter.py | CSVFileReaderWriter.py | JSONFileReaderWriter.py | sample.csv | sample.json | main.py | sample.txt | TextFileReaderWriter.py

```python
from FileReaderWriter import FileReaderWriter
from CSVFileReaderWriter import CSVFileReaderWriter
from JSONFileReaderWriter import JSONFileReaderWriter
from TextFileReaderWriter import TextFileReaderWriter  # Import the TextFileRea

# Test FileReaderWriter (base class)
df = FileReaderWriter()
df.read()
df.write()

# Test CSVFileReaderWriter
c = CSVFileReaderWriter()
c.read("sample.csv")
c.write(filepath="sample2.csv", data=["Hello", "World"])

# Test JSONFileReaderWriter
j = JSONFileReaderWriter()
j.read("sample.json")
# Replace the set with a list
j.write(data=['foo', {'bar': ['baz', None, 1.0, 2]}], filepath="sample2.json")

# Test TextFileReaderWriter
t = TextFileReaderWriter()
t.write(filepath="sample2.txt", data="ggwp.")
t.read("sample.txt")

```

OUTPUT








```
In [28]: runfile('C:/Users/TIPQC/Desktop/oopfa1TITONG_lab8/main.py', wdir='C:/Users/TIPQC/Desktop/
oopfa1TITONG_lab8')
Reloaded modules: FileReaderWriter, CSVFileReaderWriter, JSONFileReaderWriter, TextFileReaderWriter
This is the default read method
This is the default read method
['Apple', ' Banana', ' Mango', ' Orange', ' Cherry']
{'description': 'This is a JSON Sample', 'accounts': [{'id': 1, 'name': 'Jack'}, {'id': 2, 'name': 'Rose'}]}
Name : Titong, Lee Ivan

In [29]:
```

Questions

1. Why is Polymorphism important?

   Polymorphism is important because it allows for flexibility and reusability in code. By enabling objects of different classes to be treated as objects of a common superclass, polymorphism allows methods to operate on objects of various types, making code more adaptable to change and reducing redundancy. This facilitates easier maintenance and extension of programs.

2. Explain the advantages and disadvantages of using applying Polymorphism in an Object-Oriented Program.

   Advantages:

- Code Reusability: Common interfaces can be reused for different classes, reducing code duplication.
- Maintainability: Changes in one part of the code are less likely to require changes in others, as methods can be invoked on common interfaces.
- Flexibility: New classes can be added with minimal disruption to existing code, promoting scalability.

  Disadvantages:

- Complexity: Understanding the relationships between classes can be challenging, especially in

large systems.

- Performance Overhead: Polymorphism may introduce some performance costs due to dynamic dispatch and method resolution at runtime.
- Debugging Difficulty: Errors can be harder to trace when behavior is determined at runtime based on the object type.

3. What maybe the advantage and disadvantage of the program we wrote to read and write csv and json files?

Advantages:

- Reusability: The base class can be extended for different file types, allowing for easy addition of new file handlers.
- Code Organization: Each file format is handled by its own class, which keeps the code organized and focused on specific tasks.

Disadvantages:

- Limited Error Handling: If not properly handled, reading/writing files can lead to runtime errors (e.g., file not found, invalid format).
- Lack of Flexibility: The current implementation is specific to CSV and JSON formats. Adding support for other formats requires creating new classes.

4. What maybe considered if Polymorphism is to be implemented in an Object-Oriented Program?

- **Design Structure: The class hierarchy should be designed to allow for clear relationships between classes.**
- **Interface Definition: Define common interfaces that encapsulate the expected behaviors of the polymorphic classes.**
- **Method Overriding: Ensure that subclasses properly override methods from the superclass to provide the desired functionality.**
- **Type Safety: Consider using type checks or documentation to clarify expected behaviors, helping to prevent runtime errors.**
-

5. How do you think Polymorphism is used in an actual programs that we use today?

- GUI Frameworks: Different UI components (buttons, sliders, etc.) can be treated as generic "widgets," allowing for flexible layouts and interactions.
- Game Development: Different character types can be treated as a general "character" type, enabling unified behaviors (like movement and interaction) while allowing unique character-specific features.
- Data Processing Libraries: Libraries like Pandas allow different types of data to be processed uniformly, such as DataFrames and Series being treated as general data containers.

- Web Frameworks: Many frameworks support polymorphism in routing, where different URL paths can map to different controller actions, allowing for flexible request handling.

7. Conclusion:

Polymorphism is an important concept in object-oriented programming that allows different classes to be treated as the same type. This makes code more flexible and easier to reuse. While it can add some complexity, the advantages—like easier maintenance and the ability to add new features—make it valuable in software development.

In everyday applications, like file handling or user interfaces, polymorphism helps organize code and simplifies interactions between different components. Overall, it enables developers to build better, more adaptable software that can grow and change as needed.

8. Assessment Rubric: