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: Reyes, Alexzander J.	**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

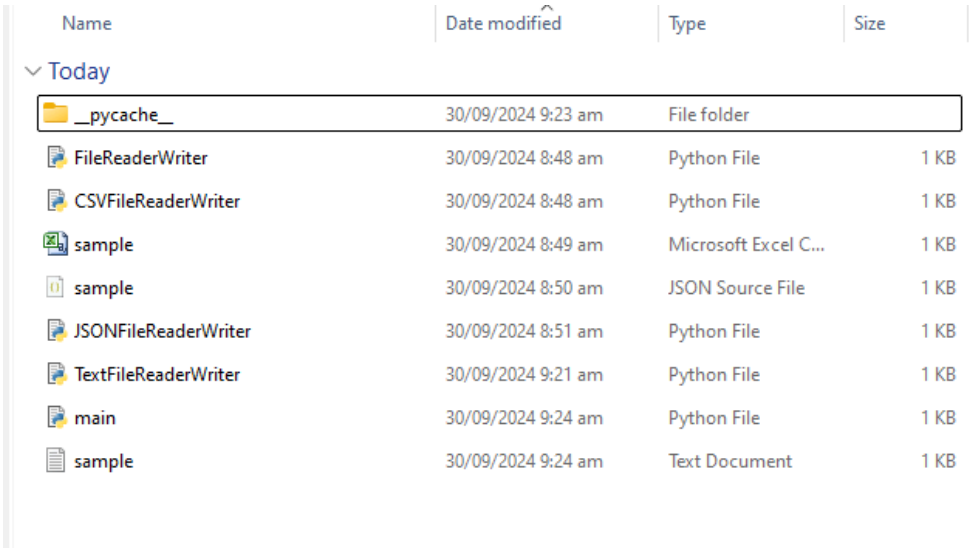
```
from FileReaderWriter import FileReaderWriter
from CSVFileReaderWriter import CSVFileReaderWriter
from JSONFileReaderWriter import JSONFileReaderWriter

# 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.

6. Supplementary Activity:

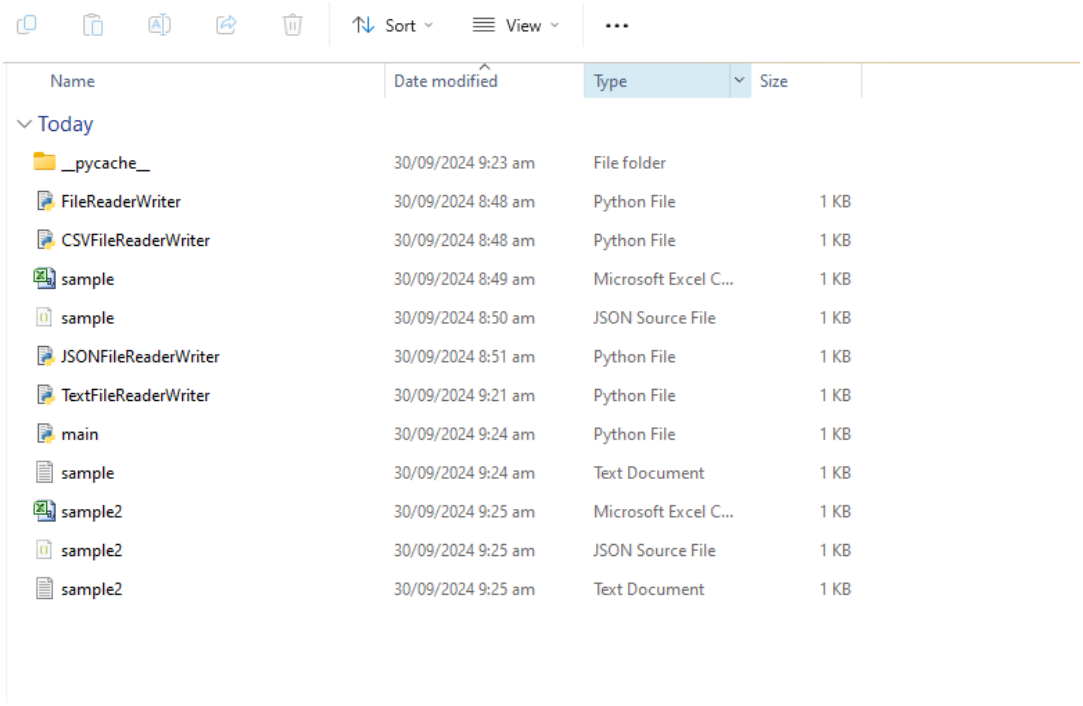
Task

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.



Name	Date modified	Type	Size
Today			
__pycache__	30/09/2024 9:23 am	File folder	
FileReaderWriter	30/09/2024 8:48 am	Python File	1 KB
CSVFileReaderWriter	30/09/2024 8:48 am	Python File	1 KB
sample	30/09/2024 8:49 am	Microsoft Excel C...	1 KB
sample	30/09/2024 8:50 am	JSON Source File	1 KB
JSONFileReaderWriter	30/09/2024 8:51 am	Python File	1 KB
TextFileReaderWriter	30/09/2024 9:21 am	Python File	1 KB
main	30/09/2024 9:24 am	Python File	1 KB
sample	30/09/2024 9:24 am	Text Document	1 KB

When I run the main ( ), the



Sort View ...

Name	Date modified	Type	Size
Today			
__pycache__	30/09/2024 9:23 am	File folder	
FileReaderWriter	30/09/2024 8:48 am	Python File	1 KB
CSVFileReaderWriter	30/09/2024 8:48 am	Python File	1 KB
sample	30/09/2024 8:49 am	Microsoft Excel C...	1 KB
sample	30/09/2024 8:50 am	JSON Source File	1 KB
JSONFileReaderWriter	30/09/2024 8:51 am	Python File	1 KB
TextFileReaderWriter	30/09/2024 9:21 am	Python File	1 KB
main	30/09/2024 9:24 am	Python File	1 KB
sample	30/09/2024 9:24 am	Text Document	1 KB
sample2	30/09/2024 9:25 am	Microsoft Excel C...	1 KB
sample2	30/09/2024 9:25 am	JSON Source File	1 KB
sample2	30/09/2024 9:25 am	Text Document	1 KB

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='|')  # Corrected here
            for row in data:
                print(row)
            return list(data)  # Return a list of rows

    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)
```

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(data, write_file)  # Corrected to json.dump
```

TextReaderWriter.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)
```

sample.csv

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

sample.json

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

sample.txt

```
ALEXZANDER J. REYES
CPE21S4
```

main.py

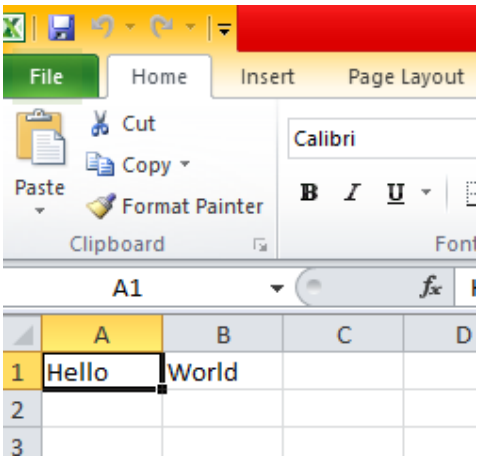
```
from FileReaderWriter import FileReaderWriter
from CSVFileReaderWriter import CSVFileReaderWriter
from JSONFileReaderWriter import JSONFileReaderWriter
from TextFileReaderWriter import TextFileReaderWriter  # Import the TextFileReaderWriter

# 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="This is sample text")
t.read("sample.txt")
```

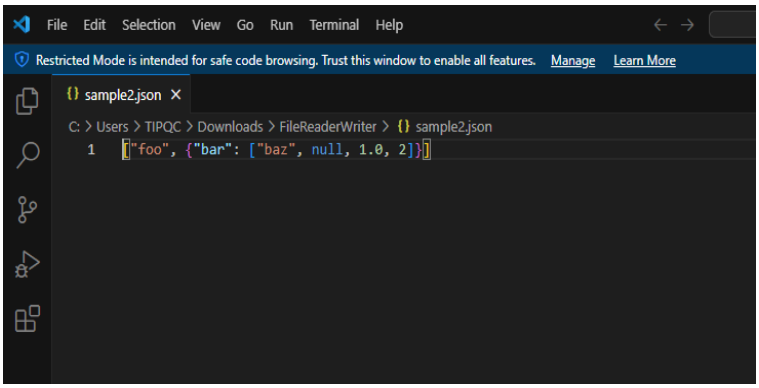
OUTPUT

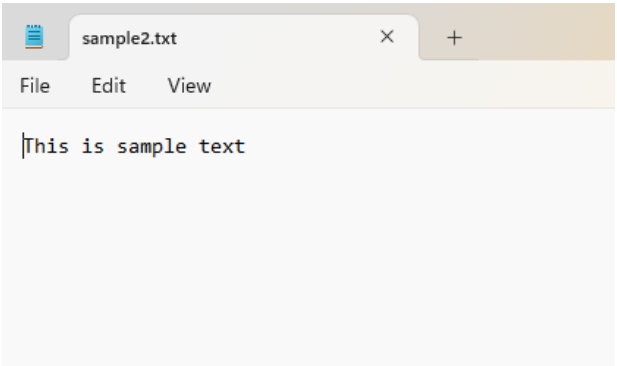
Spyder Output

```
In [38]: runfile('C:/Users/TIPQC/Downloads/FileReaderWriter/main.py', wdir='C:/Users/TIPQC/Downloads/FileReaderWriter')
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'}]}
ALEXZANDER J. REYES
CPE21S4

In [39]:
```

Sample2







Questions

1. Why is Polymorphism important?

   Polymorphism is a fundamental concept in object-oriented programming that significantly enhances code flexibility, maintainability, and usability. By allowing methods to operate on objects of different classes through a unified interface, polymorphism reduces redundancy and promotes code reuse. This means developers can write general code that can handle various data types without altering the underlying structure, making it easier to extend and maintain applications. Furthermore, polymorphism enables dynamic method resolution, allowing the appropriate method to be determined at runtime, which adds adaptability to programs. In team environments, it fosters collaboration by allowing different developers to work on classes implementing the same interface. Ultimately, polymorphism simplifies testing and debugging processes by enabling tests to be written against parent classes while ensuring that subclasses adhere to the required functionality. Overall, its importance lies in its ability to create more flexible and efficient code, essential for modern software development.

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

   Polymorphism in object-oriented programming offers several advantages, including enhanced code flexibility, maintainability, and reusability. It allows developers to write more generalized code that can handle various object types, making it easier to extend applications without modifying existing code. However, there are disadvantages as well. Polymorphism can lead to increased complexity in understanding the code, particularly for those unfamiliar with the various subclasses and their behaviors. Additionally, it may introduce runtime errors if the method resolution is not properly managed, potentially leading to less predictable behavior. Balancing these advantages and disadvantages is crucial for effective software design.

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

   The application that can read and write CSV and JSON files has a number of benefits, such as flexibility in managing various data formats and user-friendliness, which makes data processing easier for users. Along with incorporating error management for resilience, it also encourages data interoperability among systems. However, because of JSON's hierarchical structure, handling both formats may result in an increase in code complexity and performance overhead. Additionally, depending on external libraries might make deployment more difficult and increase the risk of data loss during format conversions. All things considered, even while the program increases flexibility, its disadvantages must be carefully considered.

4. What maybe considered if Polymorphism is to be implemented in an Object-Oriented Program?
   When implementing polymorphism in an Object-Oriented Program, it's important to consider whether to use compile-time (method overloading) or run-time polymorphism (method overriding via inheritance). Ensuring proper design of the class hierarchy and maintaining encapsulation and abstraction is crucial. Polymorphism also promotes loose coupling by allowing objects to interact through base classes or interfaces, enhancing flexibility. However,

it's important to be mindful of performance overhead, particularly with dynamic polymorphism. Additionally, code readability and language-specific nuances (e.g., virtual functions in C++, interfaces in Java) should be carefully addressed for maintainable and efficient implementation.

5. How do you think Polymorphism is used in an actual programs that we use today?
   Polymorphism is widely used in modern software applications to create flexible, scalable, and maintainable systems. In programs we use daily, such as web browsers, mobile apps, and operating systems, polymorphism enables different components to interact seamlessly. For example, in a GUI application, different types of buttons (e.g., submit, cancel) can all be treated as generic "buttons" due to polymorphism, allowing them to be handled uniformly while having distinct behaviors.

7. Conclusion:

In this exercise, we developed and tested file reader/writer classes for various file formats (CSV, JSON, and text files) in order to investigate the idea of polymorphism in object-oriented programming. We showed how methods like read and write in derived classes can be modified to behave differently while keeping a consistent interface in the base class by using polymorphism. This exercise demonstrated the useful advantages of polymorphism, including code reusability, flexibility, and scalability—all of which are critical for effectively handling a variety of file formats in realistic applications. We also noticed that polymorphism makes code expansion and maintenance easier by making it simple to integrate new file formats without changing the old code.

8. Assessment Rubric: