Laboratory Activity
No. 3

Polymorphism

Course Code: CPE009	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed: 09/30/24**
Section: CPE 21S4	**Date Submitted: 09/30/24**
Name: VIRTUCIO, DOMINIC JOSEPH	**Instructor: Ma'am Say0**

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.

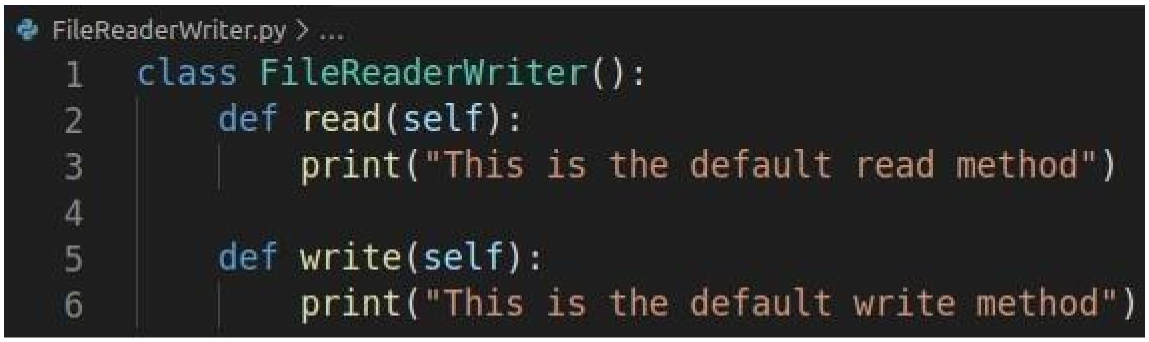
4. Materials and Equipment:

Desktop Computer with Anaconda
Python Windows Operating System

5. Procedure:

Creating the Classes

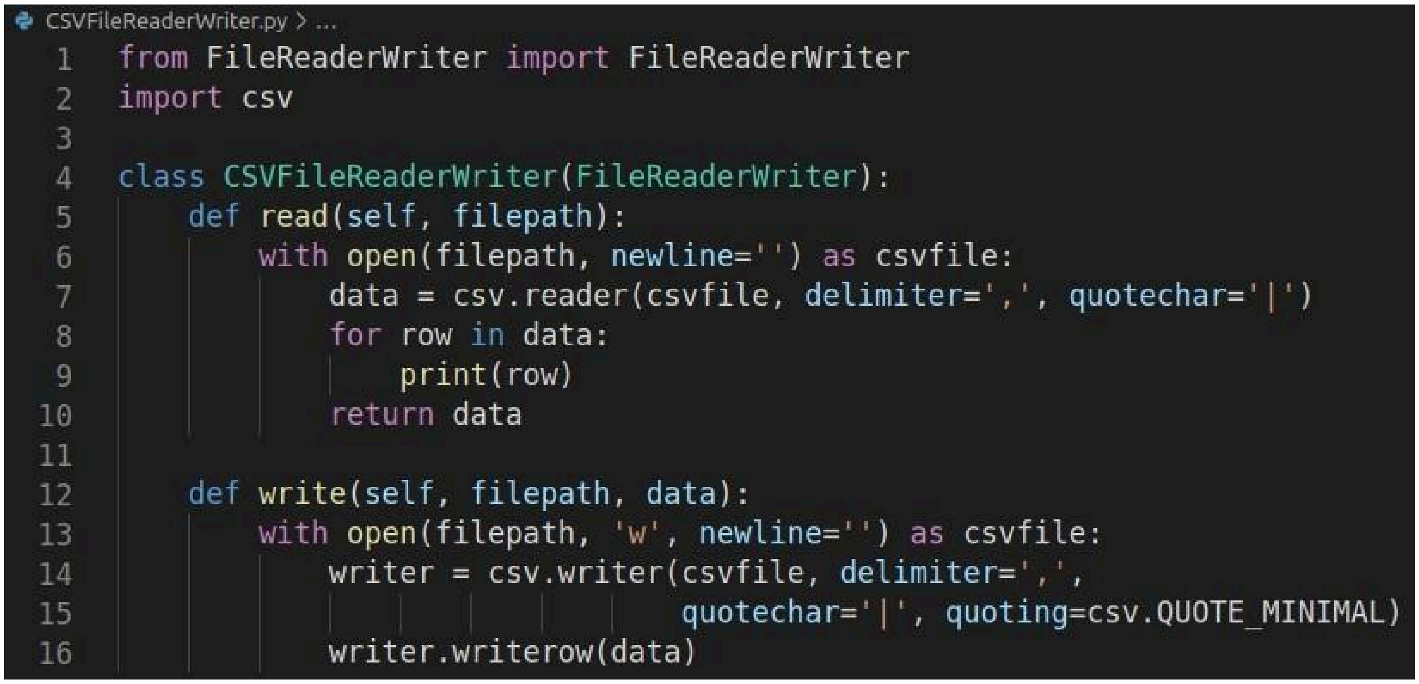
1. Create a folder named oopfa1<lastname>_lab8
2. Open your IDE in that folder.
3. Create the base FileReaderWriter .py file and Class using the code below:



FileReaderWriter.py > ...

```python
class FileReaderWriter():
    def read(self):
        print("This is the default read method")

    def write(self):
        print("This is the default write method")
```

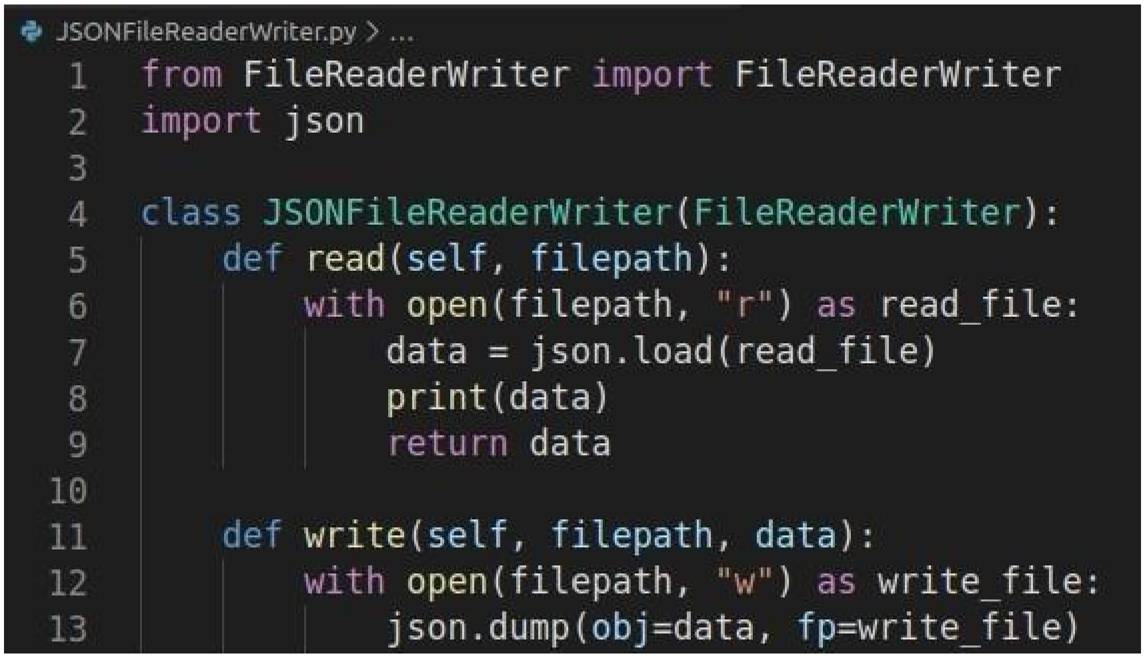
4. Create the CSVFileReaderWriter .py and Class using the code below:



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

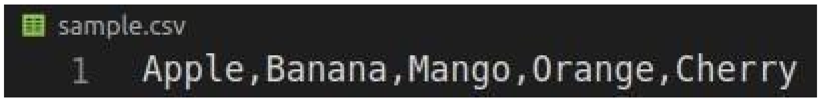


JSONFileReaderWriter.py > ...

```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

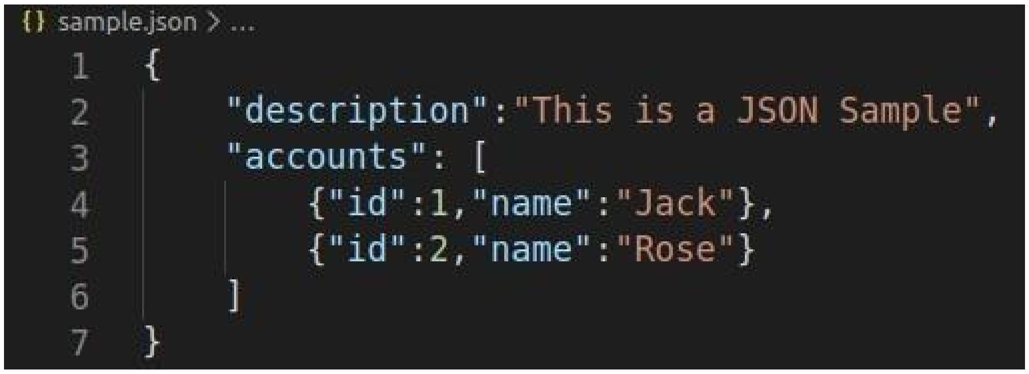
1. Create a .csv file named sample.csv with the following content. (you may use the IDE or plain notepad)



sample.csv

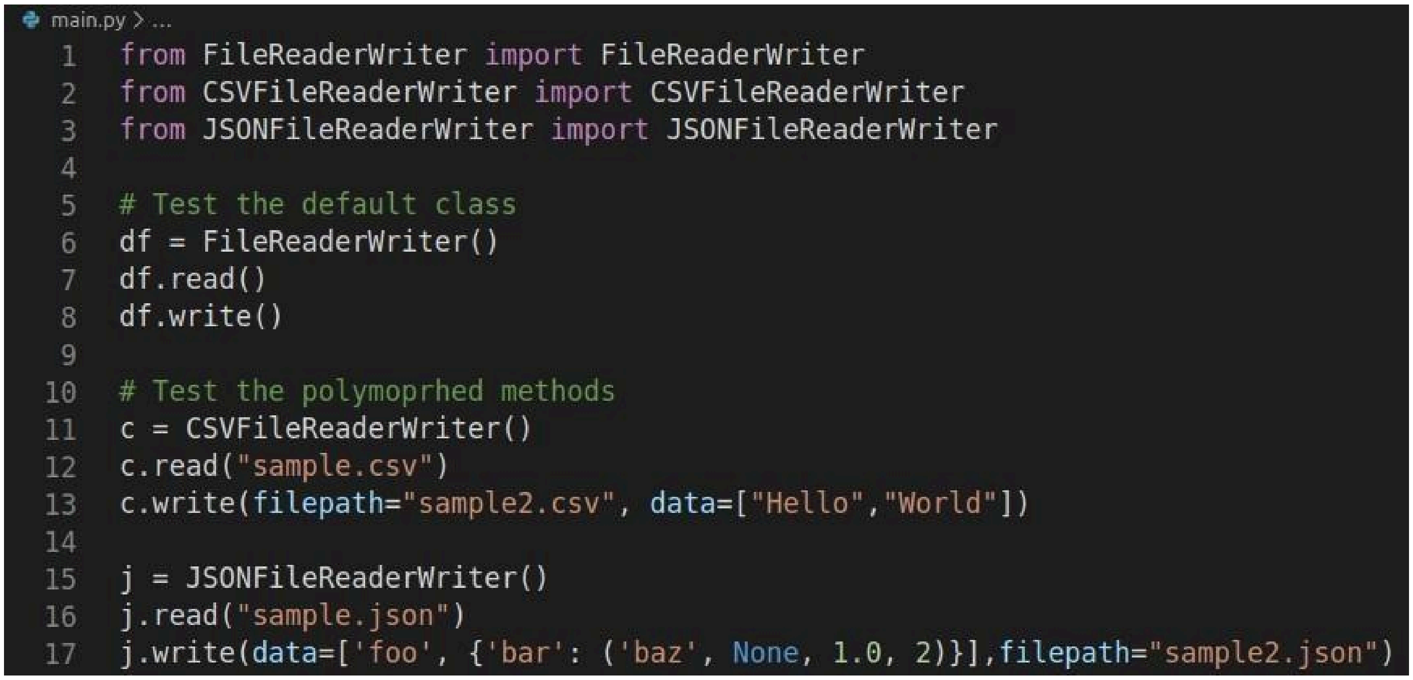
```
Apple,Banana,Mango,Orange,Cherry
```

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



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

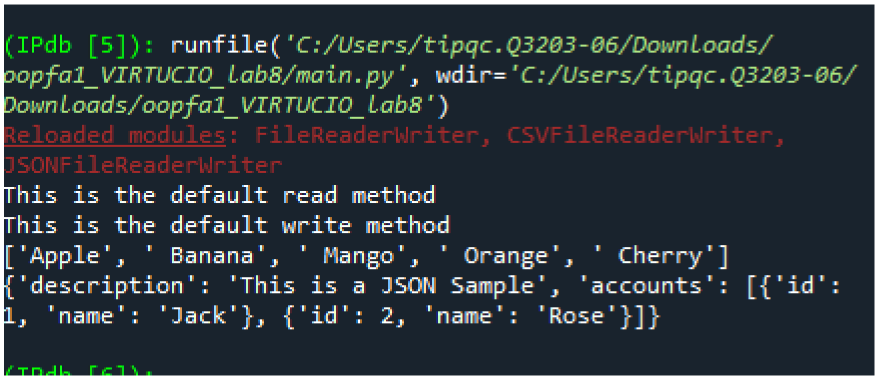
3. Create the main.py that will test the functionality of the classes.



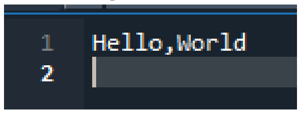
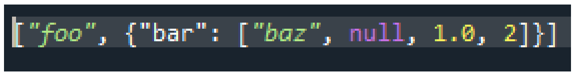
```python
from FileReaderWriter import FileReaderWriter
from CSVFileReaderWriter import CSVFileReaderWriter
from JSONFileReaderWriter import JSONFileReaderWriter

# 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.

OUTPUTS:



```
(IPdb [5]): runfile('C:/Users/tipqc.Q3203-06/Downloads/
oopfa1_VIRTUCIO_lab8/main.py', wdir='C:/Users/tipqc.Q3203-06/
Downloads/oopfa1_VIRTUCIO_lab8')
Reloaded modules: FileReaderWriter, CSVFileReaderWriter,
JSONFileReaderWriter
This is the default read method
This is the default write method
['Apple', ' Banana', ' Mango', ' Orange', ' Cherry']
{'description': 'This is a JSON Sample', 'accounts': [{'id':
1, 'name': 'Jack'}, {'id': 2, 'name': 'Rose'}]}

(IPdb [6]):
```

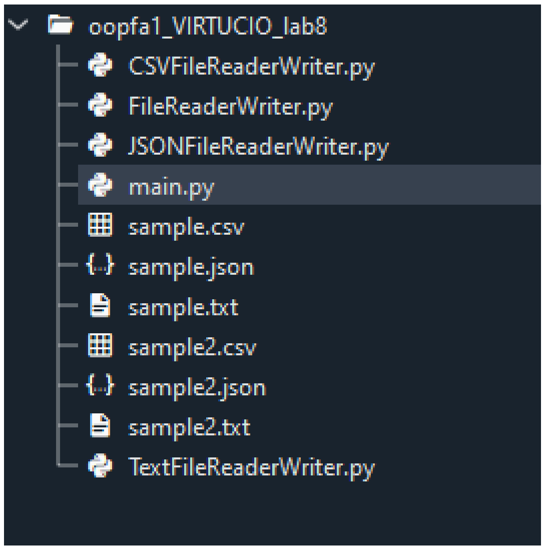
sample2.csv	sample2.json
	

```
Hello,World
```

```
["foo", {"bar": ["baz", null, 1.0, 2]}]
```

6. Supplementary Activity:

Files below can be seen in this gdrive link:
https://drive.google.com/drive/folders/1aotPp4A2O67BRjajoXL4kziPqQCdURhR?usp=sharing



```
oopfa1_VIRTUCIO_lab8
    CSVFileReaderWriter.py
    FileReaderWriter.py
    JSONFileReaderWriter.py
    main.py
    sample.csv
    sample.json
    sample.txt
    sample2.csv
    sample2.json
    sample2.txt
    TextFileReaderWriter.py
```

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 overridden according to the requirement of Text File Reading and Writing as performed in Laboratory Activity 5.

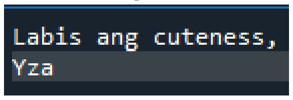
TextFIleReaderWriter.py

```python
from FileReaderWriter import FileReaderWriter

class TextFileReaderWriter(FileReaderWriter):
    def read(self, filepath):
        with open(filepath, 'r') as txtfile:
            data = txtfile.readlines()
            for line in data:
                print(line.strip())
            return data
    def write(self, filepath, data):
        with open(filepath, 'w') as write_file:
            write_file.write(data)
```

TextFileReaderWriter CODE in main.py

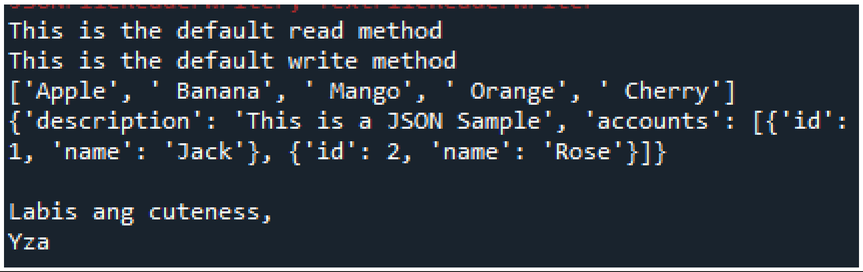
```python
t = TextFileReaderWriter()
t.read("sample.txt")
t.write(data="Hello Madlang Pipol", filepath="sample2.txt")
```

sample.txt



```
Labis ang cuteness,
Yza
```

final main.py output with txt:



```
This is the default read method
This is the default write method
['Apple', ' Banana', ' Mango', ' Orange', ' Cherry']
{'description': 'This is a JSON Sample', 'accounts': [{'id':
1, 'name': 'Jack'}, {'id': 2, 'name': 'Rose'}]}

Labis ang cuteness,
Yza
```

Questions

1. **Why is Polymorphism important?**

- Polymorphism is important because they include the ability to treat objects of different classes as though they were members of the same superclass, code reuse that enables users to write more generic and reusable code, and the ability to write functions and methods that operate on objects of different classes as long as they are members of the same superclass. By allowing functions to work on objects of different types via a common interface, it also makes code simpler. This idea allows for dynamic method binding, in which the actual type of the object determines at runtime which method is called. Moreover, polymorphism encourages abstraction and encapsulation, which makes code more modular and easier to maintain.

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

- Polymorphism in object-oriented programming has various benefits, including increased code reusability, maintainability, and extensibility by allowing the same interface to function with different underlying forms (data types), making the code more flexible and manageable. However, it adds complexity and potential performance overhead owing to dynamic method resolution during runtime, which can complicate debugging and testing processes. Furthermore, inappropriate usage of polymorphism can result in difficult-to-understand and maintain code, negating some of its advantages. Despite these disadvantages, when utilized effectively, polymorphism is an effective tool for developing adaptive and scalable software.

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

- The program we created to read and write CSV and JSON files has both advantages and disadvantages. CSV files are straightforward to use, work with most programs, and are ideal for simple tabular data. However, they require proper character encoding and cannot handle hierarchical data. JSON files handle structured data more effectively and allow layered structures, making them ideal for complex datasets. However, they use more resources and are prone to syntactic problems. Thus, CSVs are simpler and faster for simple data, whereas JSONs provide more flexibility for complicated data at the expense of increased complexity.

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

- When implementing polymorphism in an object-oriented program, consider the following: inheritance, where a base class provides a common interface for subclasses; method overriding, where subclasses can offer specific behaviors; and interface implementation, where different classes can use the same methods. Be aware of dynamic method resolution, which increases flexibility but may reduce performance and add complexity. Additionally, thorough design and testing are required to maintain robust, manageable, and efficient code.

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

- In modern software development, polymorphism is frequently used to improve flexibility and reuse. For instance, it makes data processing simpler in web development by allowing functions to handle many data types (such JSON, XML, and CSV) via a single interface. The ability to handle buttons, text fields, and other components uniformly in GUI frameworks makes event handling and interface updates easier. It allows the creation of several query handlers in databases, each of which can handle different types of queries with a same interface. All things considered, polymorphism makes it possible to create scalable, maintainable, and adaptable solutions for a variety of applications.

7. Conclusion:

- I now know that a key idea in object-oriented programming is polymorphism, which makes Python more versatile by enabling the same code to operate on different kinds of data. Using polymorphism in my code has demonstrated to me how, like the ideas of abstraction and encapsulation, it enhances flexibility and reusability. Code that has polymorphism can operate in "many forms," which increases its flexibility and manageability in a variety of situations. This knowledge has brought to light how crucial it is to write and test reliable, effective, and maintainable code.

8. Assessment Rubric: