Laboratory Activity No. 3	
Polymorphism	
Course Code: CPE009	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed: 9/30/24**
Section: CPE 009B - CPE21S4	**Date Submitted: 9/30/24**
Name: CALVIN EARL PLANTA	**Instructor: Prof. 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

```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

```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.

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

C: > Users > TIPQC > Downloads > uup > sample2.csv

```
1   Hello,World
2
```

C: > Users > TIPQC > Downloads > uup > {} sample2.json > ...

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

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

```
class TextFileReaderWriter:
        def __init__(self, file_path):
        self.file_path = file_path

        def read(self):
        # Attempt to read the file and handle failure with a message
        with open(self.file_path, 'r') as file:
        return file.read()

        def write(self, content):
        with open(self.file_path, 'w') as file:
        file.write(content)
        return "Write operation successful."

if __name__ == "__main__":
        reader_writer = TextFileReaderWriter('sample.csv')

        print(reader_writer.write("This is now the new content of sample.csv\n123, TIP, Lab2, +-=."))

        try:
        print("File Content:")
        print(reader_writer.read())
        except FileNotFoundError:
        print(f"Error: The file '{reader_writer.file_path}' was not found.")
```

```
File Content:
This is now the new content of sample.csv
123, TIP, Lab2, +-=.
PS C:\Users\TIPQC\Downloads\uup>
```

C: > Users > TIPQC > Downloads > uup > sample.csv

```
1   This is now the new content of sample.csv
2   123, TIP, Lab2, +-=.
```

Questions

1. Why is Polymorphism important?

   Because it enables objects of various kinds to be treated consistently through a common interface, polymorphism in OOP is crucial for improving flexibility, reusability, and scalability. It encourages extensibility by allowing new kinds to be smoothly incorporated, lowers complexity by doing away with conditional type-checking, and simplifies code by enabling functions to work on a variety of object types without requiring modification.

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

   By providing a common interface for several object types, polymorphism improves code reusability, flexibility, and simplification in object-oriented programming, among other benefits. But if used improperly, it can also result in too generalized or less understandable code, increase debugging complexity, and add performance overhead from dynamic method dispatch.

3. What may be the advantages and disadvantages of the program we wrote to read and write csv and json files?

   The program's ability to read and write CSV and JSON files has the benefit of facilitating simple data sharing and storing in commonly-used formats, which increases its adaptability and compatibility with other programs and systems. A drawback, though, would be the possibility of performance problems with big datasets because JSON can grow memory-intensive and CSV lacks structure, which could cause processing to lag or use more resources.

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

   When adding polymorphism to an object-oriented program, it's crucial to take into account the program's scalability and flexibility, make sure that a common method dispatch is handled dynamically, and make sure the code is still understandable and maintainable. Planning ahead is crucial to prevent overcomplication and improper use of polymorphism, which can obfuscate the design and make debugging more challenging.

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

   Modern programs frequently use polymorphism to improve maintainability and flexibility. For instance, buttons and sliders are handled consistently in graphical user interfaces (GUIs), allowing for consistent functionality. It is also used in frameworks and APIs, where it simplifies functionality and communication by enabling the processing of many objects using the same methods. The interface or base class is used efficiently. Developers need to consider performance costs as well.

7. Conclusion:

In conclusion, polymorphism is a fundamental concept in object-oriented programming that significantly enhances the flexibility, reusability, and maintainability of code. By allowing different object types to be treated uniformly through a common interface, polymorphism simplifies complex systems and fosters extensibility. However, its implementation must be approached with care to avoid potential drawbacks, such as reduced code clarity, increased debugging complexity, and performance overhead. When applied thoughtfully, polymorphism can lead to robust software design, as seen in real-world applications ranging from GUIs to various APIs, where it facilitates consistent behavior and interactions across diverse object types. Ultimately, the key to leveraging polymorphism effectively lies in balancing its advantages with careful design considerations, ensuring that the code remains comprehensible and efficient.