Laboratory Activity No. 3	
Polymorphism	
Course Code: CPE009	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed: Oct 02, 2024**
Section: CPE21S1	**Date Submitted: Oct 03, 2024**
Name: Katarina Nicole R. Delavin	**Instructor: 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 > ...

```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 > ...

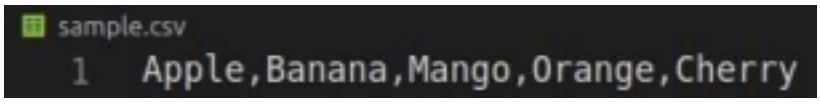
```python
from FileReaderWriter import FileReaderWriter
import json

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)



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.

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.

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

    def read(self):
        """
        Reads the contents of the text file.
        """
        try:
            with open(self.filename, 'r') as file:
                contents = file.read()
            return contents
        except FileNotFoundError:
            return ""

    def write(self, content):
        """
        Writes the given content to the text file.
        """
        with open(self.filename, 'w') as file:
            file.write(content)

if __name__ == "__main__":
    file_handler = TextFileReaderWriter("my_file.txt")

    contents = file_handler.read()
    print("File contents:", contents)

    new_content = "This is some new content."
    file_handler.write(new_content)

    updated_contents = file_handler.read()
    print("Updated file contents:", updated_contents)
```

Questions

1. Why is Polymorphism important?

- Because polymorphism makes code more reusable by enabling generic programming, it also makes code management easier by eliminating the need for complicated conditionals, improves maintainability by centralizing behavior in base classes or interfaces, makes it easier to override methods to customize behaviors in subclasses, and supports abstraction by concealing object-specific details.

  Combining these benefits makes polymorphism a potent and crucial component of object-oriented programming, resulting in code that is easier to read, write, and scale.

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

- By allowing objects of different classes to be treated as instances of the same superclass, polymorphism in object-oriented programming facilitates flexibility and encourages code reuse. Nevertheless, this may also result in more difficult debugging because it could be more difficult to determine the true type of object during runtime. Additionally, because to dynamic method resolution, polymorphism occasionally causes performance overhead.

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

- A program that can read and write CSV and JSON files has the benefit of being versatile; it makes it simple to move data across many systems or formats, including tabular and structured data (JSON and CSV). This can make integrating with different databases, APIs, and apps easier. The drawback, however, could be the possibility of performance problems with huge datasets because CSV and JSON files are text-based and not speed-optimized, and processing diverse formats necessitates thorough error handling and validation to prevent inconsistent or corrupted data.

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

- Make sure that derived classes logically extend base classes by carefully planning the inheritance hierarchy under Class Design and Inheritance: Inheritance Structure. Only methods that should exhibit distinct behaviors across subclasses should be overridden. You may make sure that polymorphism is implemented in an Object-Oriented Program in a way that makes code flexible, scalable, and maintainable by taking these considerations into account.

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

- In real applications, polymorphism enables flexibility by allowing various objects to be treated in the same way via a common interface. For instance, polymorphism in a graphics software allows the application to handle many forms, such as triangles, squares, and circles, using the same drawing technique, even though each shape has a unique implementation for how it's drawn.

7. Conclusion:

To sum up, polymorphism increases code flexibility by enabling objects of various kinds to be treated consistently through a shared interface. It enhances code reuse and simplicity by allowing methods from several classes to carry out distinct operations under the same method name. Programs can be made more scalable and maintainable by using polymorphism, which enables the addition of new object types without changing the existing code.

8. Assessment Rubric: