**Name:** Hamsa M Ahmed
**Subject:** OOP Practical
**Department:** Computer Network System

# Lecture 12: File Handling in Object-Oriented Programming

## 1. Introduction

File handling is a key feature for any programming language, allowing programs to read from and write to files. Integrating file operations within classes helps organize and encapsulate related functionalities.

## 2. Basic File Operations in Python

Python provides built-in functions to open, read, write, and close files.

```
# Writing to a file
with open('example.txt', 'w') as file:
    file.write("Hello, World!")

# Reading from a file
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)
```

## 3. Encapsulating File Operations in a Class

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

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

    def read_data(self):
        with open(self.filename, 'r') as file:
            return file.read()
```

**Name:** Hamsa M Ahmed
**Subject:** OOP Practical
**Department:** Computer Network System

# 4. Usage Example

```
file = FileHandler('test.txt')
file.write_data("This is a test.")
content = file.read_data()
print(content)
```

# 5. Exception Handling in File Operations

To handle file-related errors:

```
try:
    file = FileHandler('nonexistent.txt')
    content = file.read_data()
except FileNotFoundError:
    print("File not found!")
```

# 6. Exercises

1. Extend `FileHandler` to append data to existing files.
2. Add method to delete the file safely.
3. Implement a method to check if a file exists.

# 7. Summary

File handling combined with OOP concepts allows for modular, maintainable code that manages file operations effectively with error handling.

# 8. References

1. Python Docs - Reading and Writing Files
2. Real Python - Working with Files in Python