LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 7

2411102441052

Angga Maulana Saputra

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

EMAIL : 2411102441052@umkt.ac.id

Latihan Praktikum 7:

a. Screenshoot kode pertemuan_7.py (pustaka math)

```python
import math # Impor pustaka math

class KalkulatorLingkaran:
    def __init__(self, radius):
        self.__radius = 0
        self.set_radius(radius)
        print(f"Objek lingkaran dengan radius {self.__radius} dibuat.")

    def set_radius(self, radius):
        if radius > 0:
            self.__radius = radius
        else:
            print("Error: Radius harus lebih besar dari 0.")
            self.__radius = 1 # Nilai default jika input salah

    def hitung_luas(self):
        # Menggunakan konstanta pi dari pustaka math
        luas = math.pi * (self.__radius ** 2)
        return luas

    def hitung_keliling(self):
        # Menggunakan konstata pi lagi
        keliling = 2 * math.pi * self.__radius
        return keliling

# --- Bagian Utama Program ---
lingkaran_1 = KalkulatorLingkaran(7)
luas_lingkaran = lingkaran_1.hitung_luas()
keliling_lingkaran = lingkaran_1.hitung_keliling()

print(f"\nRadius: 7")
print(f"Luas Lingkarang: {luas_lingkaran:.2f}") # Format 2 angka di belakang koma
print(f"Keliling Lingkaran: {keliling_lingkaran:.2f}")
```

b. Screenshoot hasil eksekusi program

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
>_bash  pertemuan7  332ms
>> python pertemuan_7.py
Objek lingkaran dengan radius 7 dibuat.

Radius: 7
Luas Lingkarang: 153.94
Keliling Lingkaran: 43.98
>_bash  pertemuan7  384ms
>>
```

c. Screenshoot kode pertemuan_7.py (pustaka datetime)

```python
import datetime # Impor pustaka datetime

class LogPesan:
    def __init__(self, pengirim, isi_pesan):
        self.__pengirim = pengirim
        self.__isi_pesan = isi_pesan
        # Secara otomatis mendapatkan waktu saat ini ketika objek dibuat
        self.__timestamp = datetime.datetime.now()

    def tampilkan_log(self):
        # Memformat timestamp menjadi string yang mudah dibaca
        waktu_terformat = self.__timestamp.strftime("%d %B %Y, Pukul %H:%M:%S")
        print("--- Log Pesan Masuk ---")
        print(f"Pengirim : {self.__pengirim}")
        print(f"Waktu    : {waktu_terformat}")
        print(f"Pesan    : {self.__isi_pesan}")

# --- Bagian Utama Program ---
pesan_1 = LogPesan("Admin", "Server akan segera di-restart untuk maintenance")
pesan_1.tampilkan_log()

# Tunggu beberapa detik dan buat pesan lain
# (Untuk simulasi, kita bisa tambahkan time.sleep jika diinginkan)
pesan_2 = LogPesan("User01", "Pekerjaan saya sudah disimpan, silahkan restart.")
pesan_2.tampilkan_log()
```

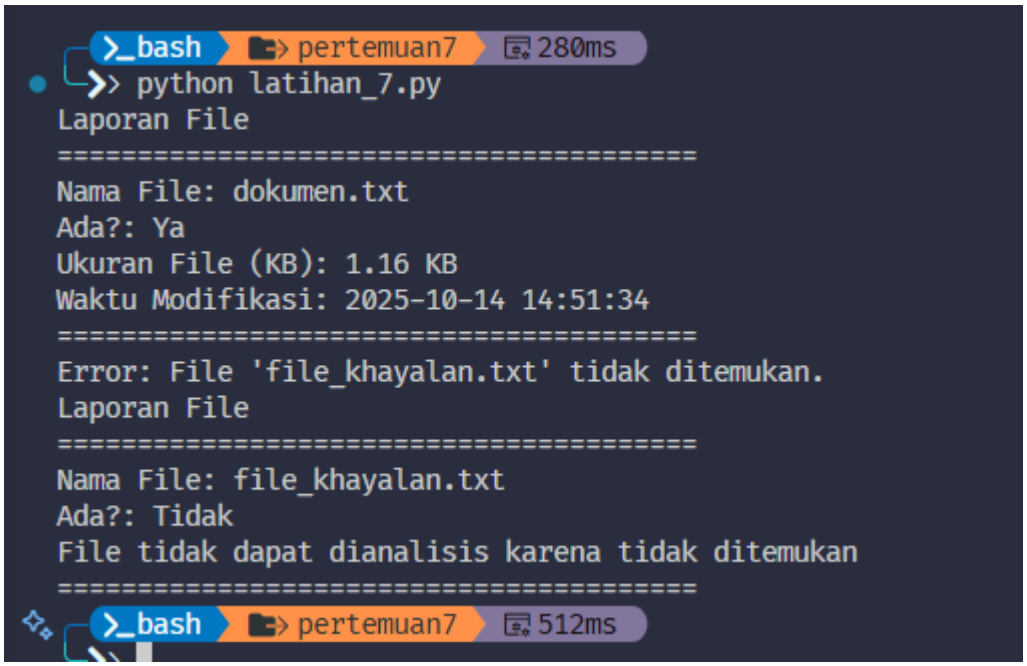
d. Screenshoot hasil eksekusi program

```
>_bash  pertemuan7  0ms
>> python pertemuan_7.py
--- Log Pesan Masuk ---
Pengirim : Admin
Waktu    : 14 October 2025, Pukul 12:23:20
Pesan    : Server akan segera di-restart untuk maintenance
--- Log Pesan Masuk ---
Pengirim : User01
Waktu    : 14 October 2025, Pukul 12:23:20
Pesan    : Pekerjaan saya sudah disimpan, silahkan restart.
>_bash  pertemuan7  338ms
>>
```

e. Screenshoot kode latihan_7.py

```python
import os
from datetime import datetime

class FileAnalyzer:
    def __init__(self, file_path):
        self.__file_path = file_path
        self.__file_ada = os.path.exists(file_path)
        if self.__file_ada:
            self.__file_size = os.path.getsize(file_path)
        else:
            print(f"Error: File '{file_path}' tidak ditemukan.")
            self.__file_size = False

    def get_file_size(self, unit="bytes"):
        if self.__file_ada:
            if unit.upper() == "KB":
                return round(self.__file_size / 1024, 2)
        else:
            return None
        return self.__file_size

    def get_modification_time(self):
        mod_time = os.path.getatime(self.__file_path)
        return datetime.fromtimestamp(mod_time).strftime("%Y-%m-%d %H:%M:%S")

    def analyze(self):
        print("Laporan File")
        print("="*40)
        print(f"Nama File: {self.__file_path}")
        print(f"Ada?: {'Ya' if self.__file_ada else 'Tidak'}")
        if self.__file_ada:
            print(f"Ukuran File (KB): {self.get_file_size('KB')} KB")
            print(f"Waktu Modifikasi: {self.get_modification_time()}")
        else:
            print("File tidak dapat dianalisis karena tidak ditemukan")
        print("="*40)

analyzer1 = FileAnalyzer("dokumen.txt")
analyzer1.analyze()

analyzer2 = FileAnalyzer("file_khayalan.txt")
analyzer2.analyze()
```

f. Screenshoot hasil eksekusi program.



g. Refleksi singkat: Mengintegrasikan pustaka standar ke dalam sebuah class membuat kode lebih terstruktur, mudah dipelihara, dan mudah dikembangkan karena data dan perilaku terkait disatukan dalam satu kesatuan (encapsulation). Dengan class, kita bisa membuat objek yang menyimpan data, memiliki state, serta menyediakan method-method khusus yang relevan, sehingga kode menjadi lebih modular dan reusable dibandingkan sekadar serangkaian fungsi terpisah.