# LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK LANJUT

2023

Prepared By:

NAMA : RIFKI PRAMAYANDI MAHESA
NIM : 21051156
KELAS : D (Reguler)

# PEMROGRAMAN BERORIENTASI OBJECK LANJUTAN

Dosen Matakuliah : Freddy Wicaksono, M. Kom

---

Tugas Praktikum :

1. Buatlah masing-masing 2 contoh jenis pewarisan di luar dari contoh yang telah diberikan, beri nama :
   a. Sintaks :
      1) Single-1.py

```
class Komputer:
    def __init__(self, merk, harga):
        self.merk = merk
        self.harga = harga

    def spesifikasi(self):
        print(f"Komputer merk {self.merk} dengan harga {self.harga}.")

class Laptop(Komputer):
    def __init__(self, merk, harga, ukuran_layar):
        super().__init__(merk, harga)
        self.ukuran_layar = ukuran_layar

    def spesifikasi(self):
        print(f"Laptop merk {self.merk} dengan harga {self.harga} dan ukuran
layar {self.ukuran_layar} inci.")

laptop_acer = Laptop("Acer", 5000000, 15.6)
laptop_acer.spesifikasi() # output: "Laptop merk Acer dengan harga 5000000 dan
ukuran layar 15.6 inci."

print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

2) Single-2.py

```
class UnsurAlam:
    def __init__(self, nama):
        self.nama = nama

    def get_info(self):
        print(f"Ini adalah unsur alam {self.nama}.")

class UnsurLogam(UnsurAlam):
    def __init__(self, nama, sifat):
        super().__init__(nama)
        self.sifat = sifat

    def get_info(self):
        print(f"Ini adalah unsur logam {self.nama} yang memiliki sifat
{self.sifat}.")

unsur_1 = UnsurAlam("Oksigen")
unsur_2 = UnsurLogam("Emas", "Mudah ditempa")

unsur_1.get_info() # output: "Ini adalah unsur alam Oksigen."
unsur_2.get_info() # output: "Ini adalah unsur logam Emas yang memiliki sifat
Mudah ditempa."

print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

3) Multiple-1.py

```python
class Switchable:
    def switch_on(self):
        self.is_on = True
        print("Switched on")

    def switch_off(self):
        self.is_on = False
        print("Switched off")

class Dimmable:
    def set_brightness(self, brightness):
        self.brightness = brightness
        print("Brightness set to", brightness)

class Lampu(Switchable, Dimmable):
    def __init__(self):
        self.is_on = False
        self.brightness = 0

lampu1 = Lampu()
lampu1.switch_on()
lampu1.set_brightness(50)
print("Brightness:", lampu1.brightness)

lampu1.switch_off()
print("Is on:", lampu1.is_on)

print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

4) Multiple-2.py

```python
class WebPage:
    def __init__(self, url):
        self.url = url

    def open(self):
        print("Opening webpage:", self.url)

class OperatingSystem:
    def __init__(self, name):
        self.name = name

    def boot_up(self):
        print(self.name, "is booting up")

class Browser(WebPage, OperatingSystem):
    def __init__(self, url, name):
        WebPage.__init__(self, url)
        OperatingSystem.__init__(self, name)

    def display(self):
        self.boot_up()
        self.open()
        print("Browser is ready to use")

# Membuat objek dari kelas Browser
my_browser = Browser("https://www.google.com", "Windows 10")

# Memanggil metode display() pada objek my_browser
my_browser.display()

print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

5) Hierarchy-1.py

```python
class OutputDevice:
    def __init__(self):
        self.connected = False

    def connect(self):
        self.connected = True

    def disconnect(self):
        self.connected = False

class Printer(OutputDevice):
    def __init__(self):
        super().__init__()
        self.print_queue = []

    def print(self, document):
        if self.connected:
            self.print_queue.append(document)
            print(f"Printing {document}")
        else:
            print("Printer is not connected.")

    def cancel_print(self, document):
        if document in self.print_queue:
            self.print_queue.remove(document)
            print(f"{document} has been removed from print queue.")
        else:
            print(f"{document} is not in print queue.")

class Scanner(OutputDevice):
    def __init__(self):
        super().__init__()

    def scan(self):
        if self.connected:
            print("Scanning document...")
        else:
            print("Scanner is not connected.")

# Membuat objek printer dan scanner
printer = Printer()
scanner = Scanner()

# Menghubungkan printer dan scanner
printer.connect()
scanner.connect()
```

```python
# Mencetak dokumen dengan printer
printer.print("Sample Document")

# Membatalkan pencetakan dokumen dengan printer
printer.cancel_print("Sample Document")

# Melakukan scanning dokumen dengan scanner
scanner.scan()

# Memutuskan koneksi printer dan scanner
printer.disconnect()
scanner.disconnect()

print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

6) Hierarchy-2.py

```python
class Hardware:
    def __init__(self):
        self.powered_on = False

    def power_on(self):
        self.powered_on = True
        print("Hardware is now powered on.")

    def power_off(self):
        self.powered_on = False
        print("Hardware is now powered off.")

class InputDevice(Hardware):
    def __init__(self):
        super().__init__()

class OutputDevice(Hardware):
    def __init__(self):
        super().__init__()

class Keyboard(InputDevice):
    def __init__(self):
        super().__init__()

    def type(self, text):
        if self.powered_on:
            print(f"Typed: {text}")
        else:
            print("Keyboard is not powered on.")

class Monitor(OutputDevice):
    def __init__(self):
        super().__init__()

    def display(self, text):
        if self.powered_on:
            print(text)
        else:
            print("Monitor is not powered on.")

# Membuat objek keyboard dan monitor
keyboard = Keyboard()
monitor = Monitor()

# Menghidupkan keyboard dan monitor
keyboard.power_on()
monitor.power_on()
```

```python
# Mengetikkan teks dengan keyboard
keyboard.type("Hello World")

# Menampilkan teks di monitor
monitor.display("Welcome to the world of Python")

# Mematikan keyboard dan monitor
keyboard.power_off()
monitor.power_off()

print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

7) MultiLevel-1.py

```
class Food:
    def __init__(self, name):
        self.name = name

class Ingredient(Food):
    def __init__(self, name, quantity):
        super().__init__(name)
        self.quantity = quantity

class Spice(Ingredient):
    def __init__(self, name, quantity, is_hot):
        super().__init__(name, quantity)
        self.is_hot = is_hot

# membuat objek Spice
cumin = Spice("Cumin", 50, True)

# mencetak atribut dari objek Spice
print(cumin.name)
print(cumin.quantity)
print(cumin.is_hot)

print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

8) MultiLevel-2.py

```
class Computer:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

class Processor(Computer):
    def __init__(self, brand, model, cores):
        super().__init__(brand, model)
        self.cores = cores

class CPU(Processor):
    def __init__(self, brand, model, cores, frequency):
        super().__init__(brand, model, cores)
        self.frequency = frequency

    def print_specs(self):
        print(f"Brand: {self.brand}\nModel: {self.model}\nCores:
{self.cores}\nFrequency: {self.frequency}")

# membuat objek CPU
my_cpu = CPU("Intel", "i7-10700K", 8, "3.8 GHz")

# mencetak spesifikasi dari objek CPU
my_cpu.print_specs()

print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

9) Hybrid-1.py

```
# membuat class Configurate
class Configurate:
    def __init__(self, name, processor):
        self.name = name
        self.processor = processor

    def get_name(self):
        return self.name

    def get_processor(self):
        return self.processor

# membuat class Computing
class Computing:
    def __init__(self, ram, storage):
        self.ram = ram
        self.storage = storage

    def get_ram(self):
        return self.ram

    def get_storage(self):
        return self.storage

# membuat class Configurate Computing dengan menggunakan multiple inheritance
class ConfigurateComputing(Configurate, Computing):
    def __init__(self, name, processor, ram, storage, graphics_card):
        Configurate.__init__(self, name, processor)
        Computing.__init__(self, ram, storage)
        self.graphics_card = graphics_card

    def get_graphics_card(self):
        return self.graphics_card

# membuat objek dari class ConfigurateComputing
pc1 = ConfigurateComputing("Laptop", "Intel Core i5", "8 GB", "256 GB",
"Nvidia GTX 1650")

# memanggil method yang dimiliki oleh class Configurate
print(pc1.get_name())       # Output: Laptop
print(pc1.get_processor())  # Output: Intel Core i5

# memanggil method yang dimiliki oleh class Computing
print(pc1.get_ram())        # Output: 8 GB
print(pc1.get_storage())    # Output: 256 GB
```

```python
# memanggil method yang dimiliki oleh class ConfigurateComputing
print(pc1.get_graphics_card())  # Output: Nvidia GTX 1650

print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

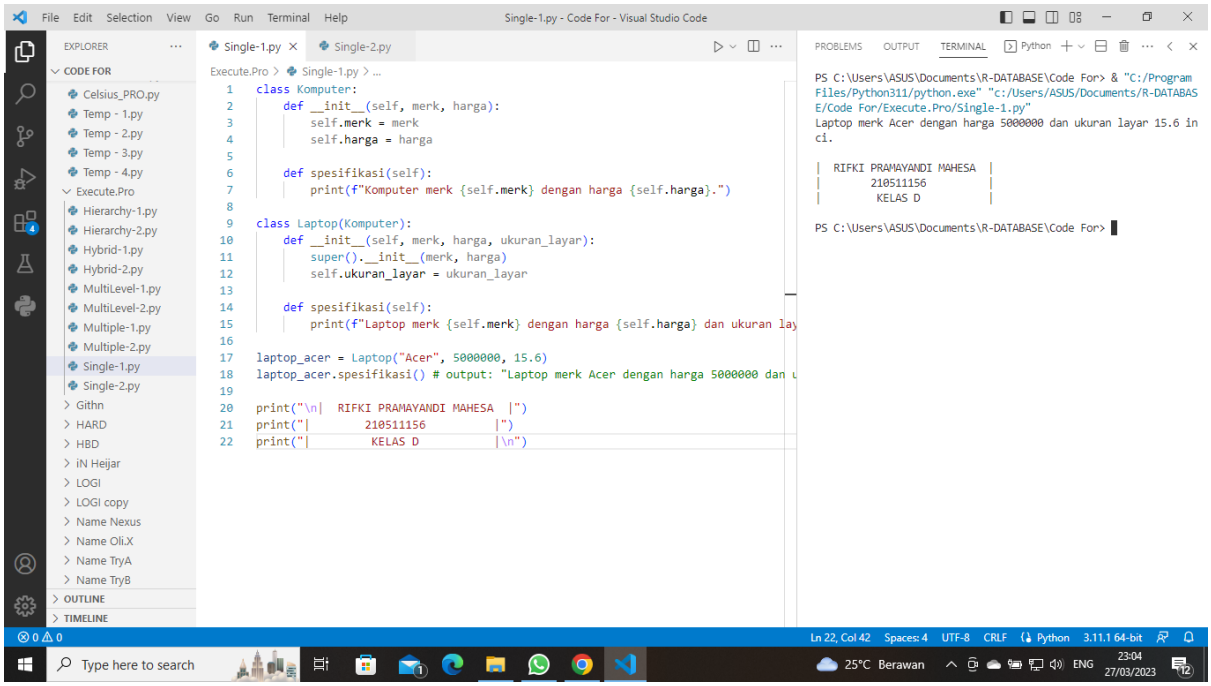
10) Hybrid-2.py

```python
# membuat class Kernel
class Kernel:
    def __init__(self, name, version):
        self.name = name
        self.version = version

    def get_name(self):
        return self.name

    def get_version(self):
        return self.version

# membuat class User Interface
class UserInterface:
    def __init__(self, display_type, theme):
        self.display_type = display_type
        self.theme = theme

    def get_display_type(self):
        return self.display_type

    def get_theme(self):
        return self.theme

# membuat class Operating System dengan menggunakan multiple inheritance
class OperatingSystem(Kernel, UserInterface):
    def __init__(self, name, version, display_type, theme, default_browser):
        Kernel.__init__(self, name, version)
        UserInterface.__init__(self, display_type, theme)
        self.default_browser = default_browser

    def get_default_browser(self):
        return self.default_browser

# membuat objek dari class OperatingSystem
os1 = OperatingSystem("Windows", "10", "GUI", "Light", "Chrome")

# memanggil method yang dimiliki oleh class Kernel
print("Name OS :",os1.get_name())       # Output: Windows
print("Version :",os1.get_version())    # Output: 10

# memanggil method yang dimiliki oleh class UserInterface
print("Display Type :",os1.get_display_type())   # Output: GUI
print("Theme :",os1.get_theme())          # Output: Light

# memanggil method yang dimiliki oleh class OperatingSystem
print("Default Browser :",os1.get_default_browser())    # Output: Chrome
```

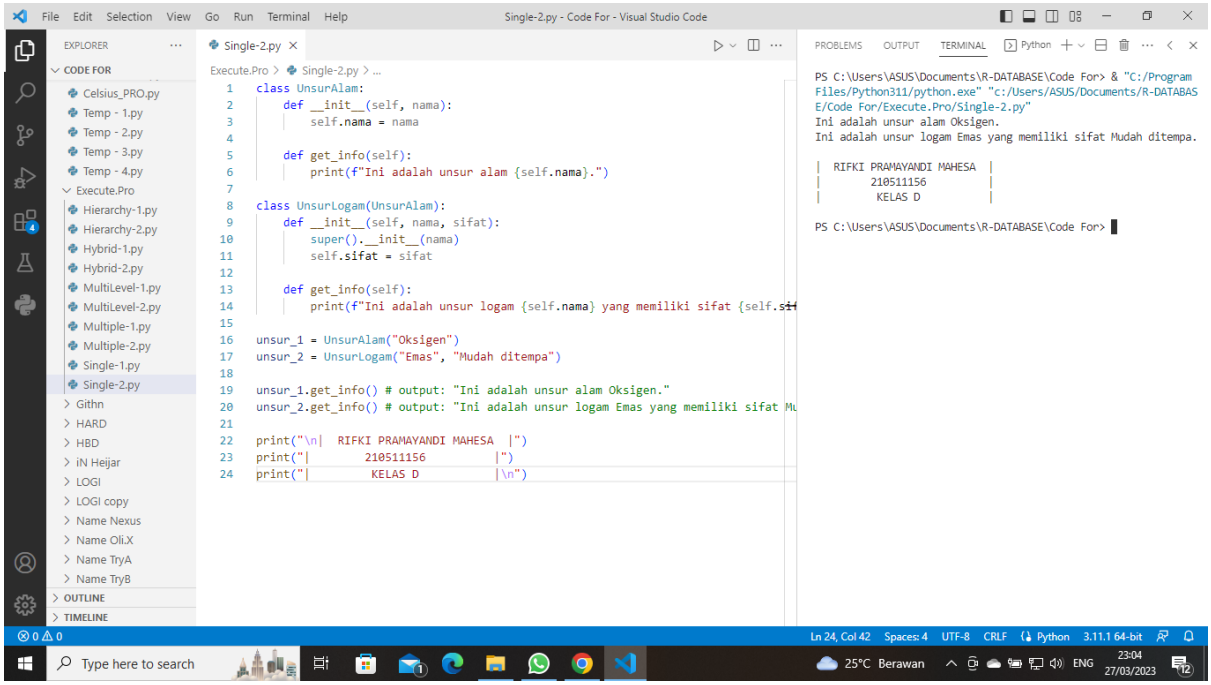
```
print("\n|  RIFKI PRAMAYANDI MAHESA  |")
print("|        210511156          |")
print("|         KELAS D           |\n")
```

b. Hasil Pemrograman :

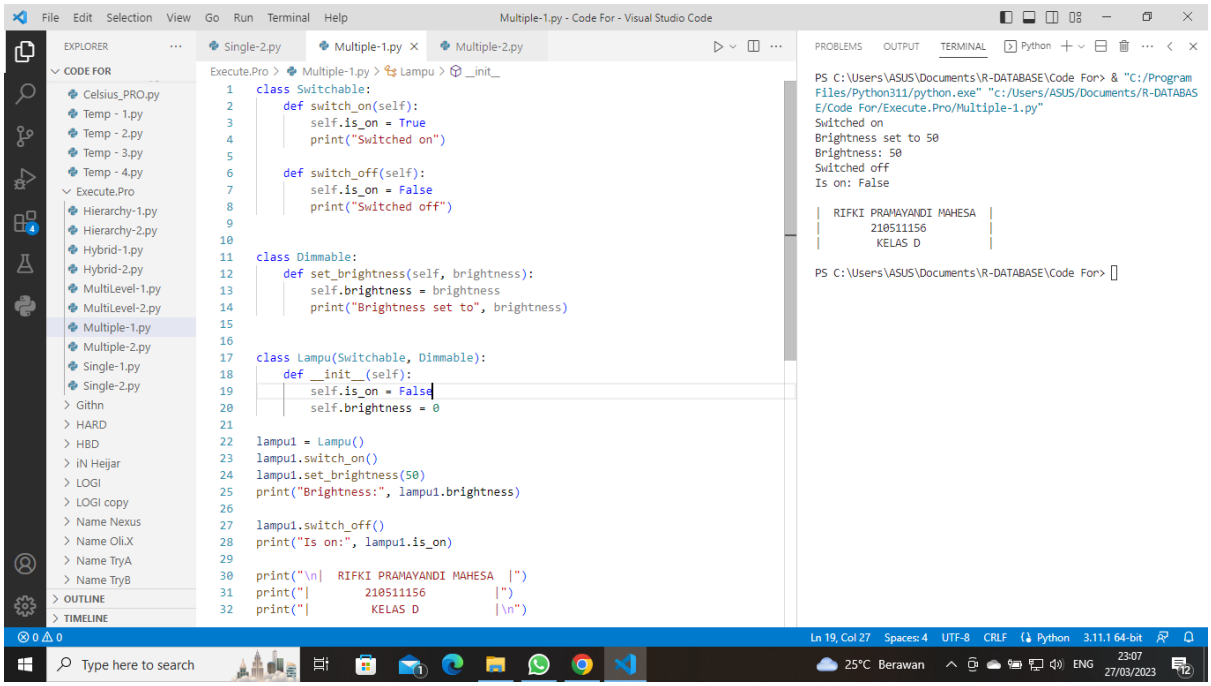
1) Single-1.py



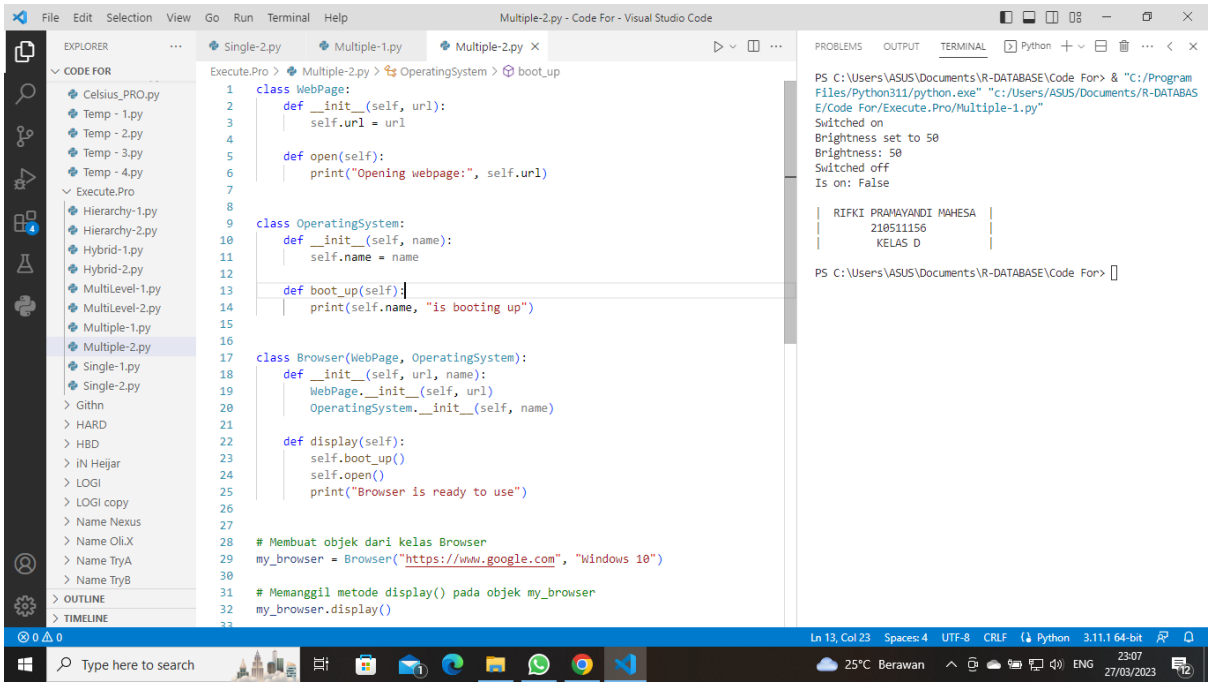

2) Single-2.py



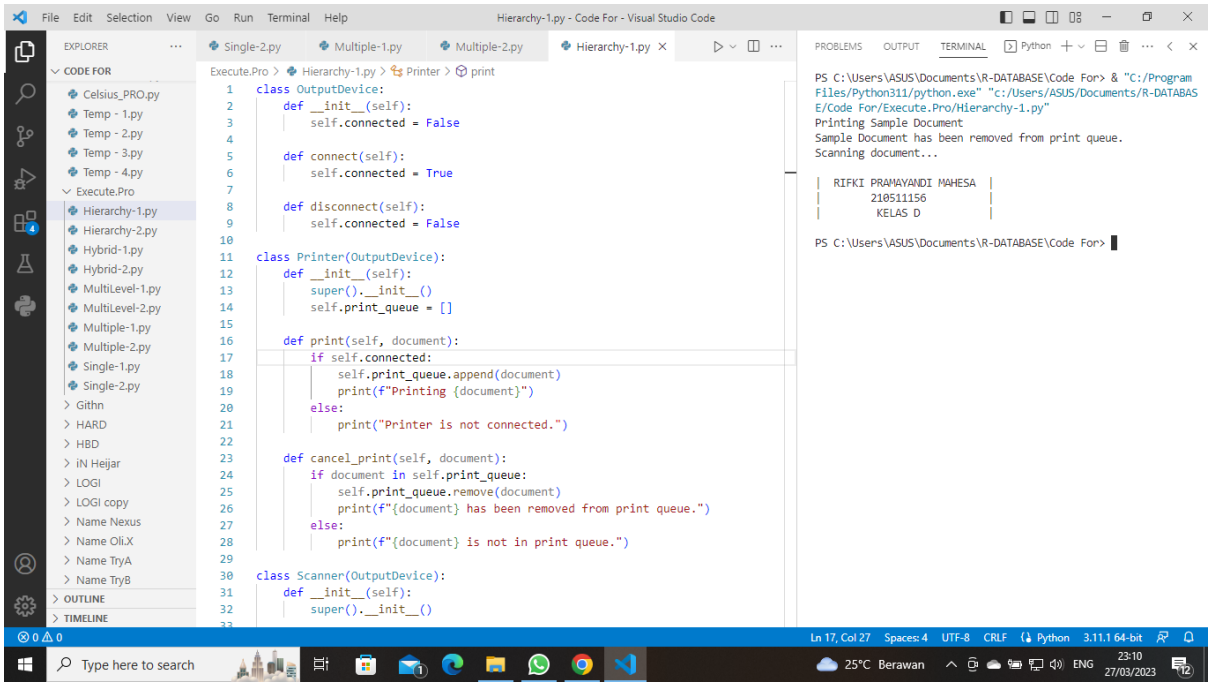
3) Multiple-1.py



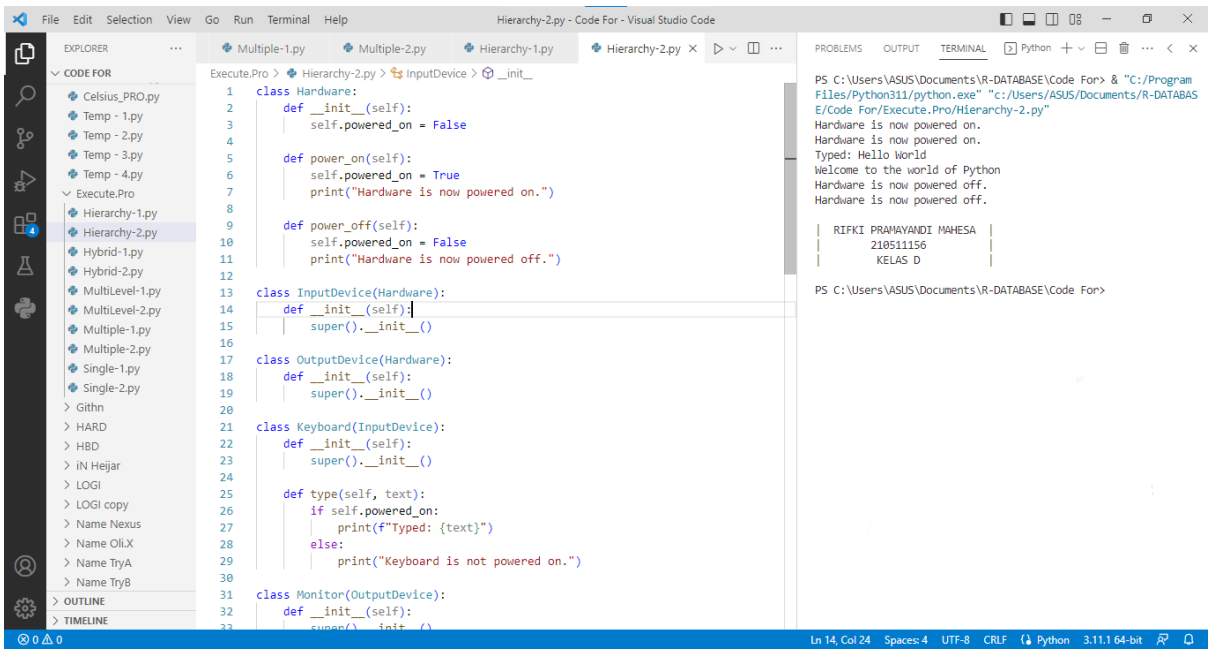

4) Multiple-2.py



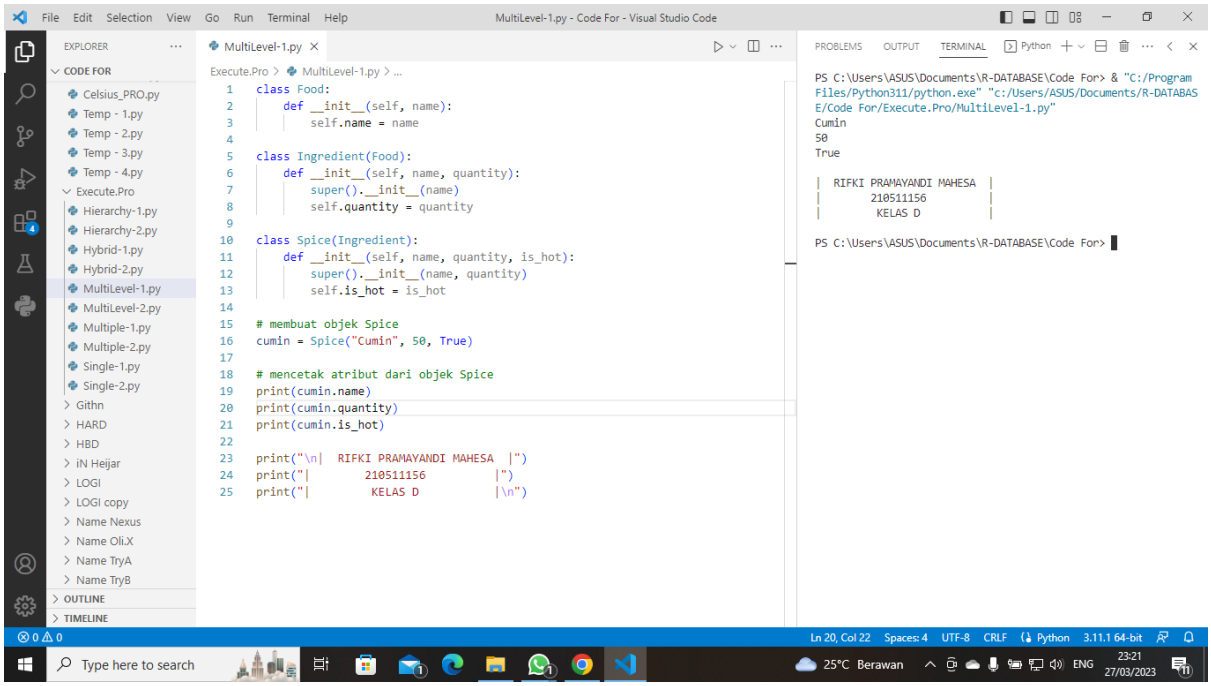
5) Hierarchy-1.py



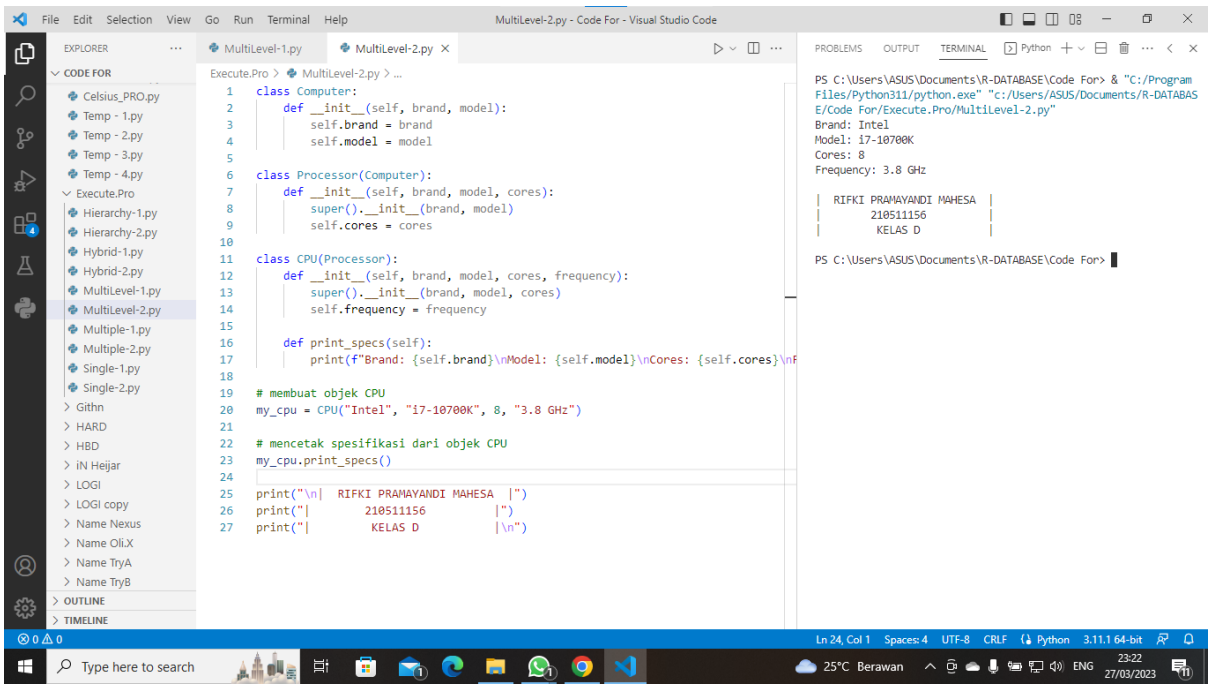

6) Hierarchy-2.py



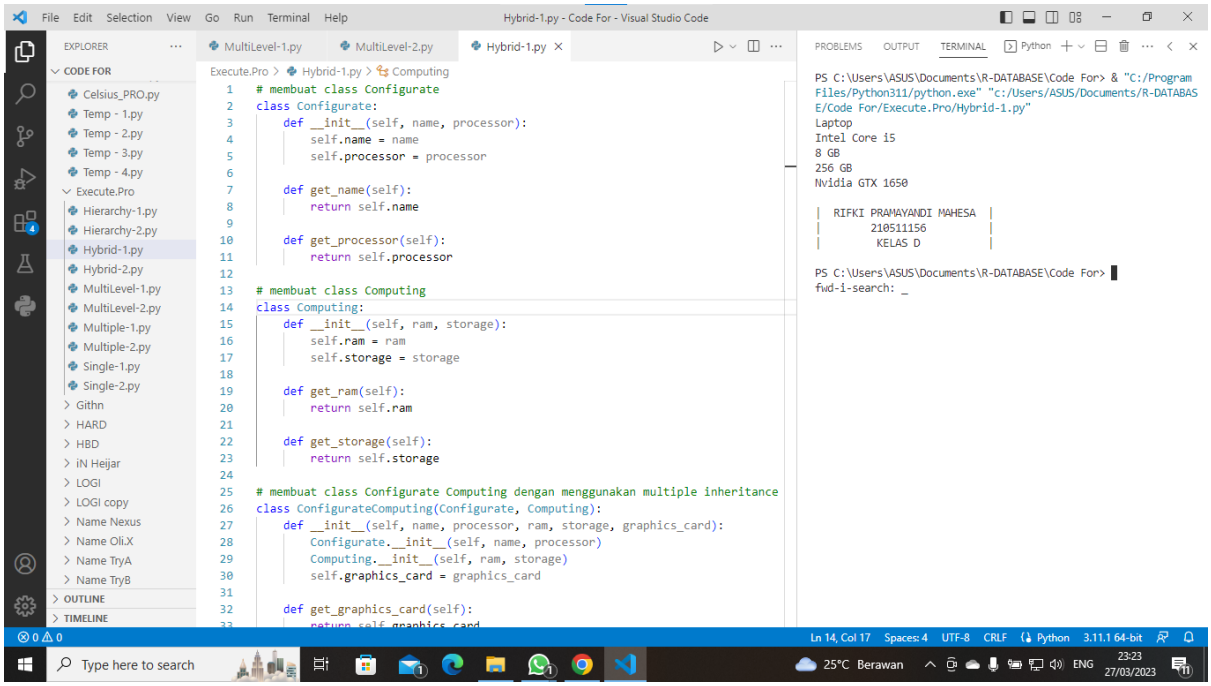
7) MultiLevel-1.py




8) MultiLevel-2.py



9) Hybrid-1.py




10) Hybrid-2.py