

LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK LANJUT

2023



Prepared By:

Aini Salsabila

210511065 / R2

Soal Praktikum 2

Buatlah masing-masing dua jenis pewarisan di luar dari contoh yang di berikan:

1. Single Inheritance:

a. contoh 1:

Nama : Aini Salsabila

NIM : 210511065

Kelas : R2/B

```
print("\nSingle Inheritance_NASI GORENG\n\n")
```

```
class Menu:
```

```
    def __init__(self,menu,level):
```

```
        self.menu = menu
```

```
        self.level = level
```

```
    def info(self):
```

```
        print('Menu\t\t: ',self.menu)
```

```
        print('Level\t\t: ',self.level)
```

```
class Pesan(Menu):
```

```
    def __init__(self,menu,level,topping,tambahan):
```

```
        super().__init__(menu,level)
```

```
        self.topping = topping
```

```
        self.tambahan = tambahan
```

```
    def pesan(self):
```

```
        print("Topping\t\t: ",self.topping)
```

```
        print("Tambahan\t\t: ",self.tambahan)
```

```
pesan1 = Pesan("Nasi Goreng",3,"Telur Dadar","Tidak pake sayuran\n")
```

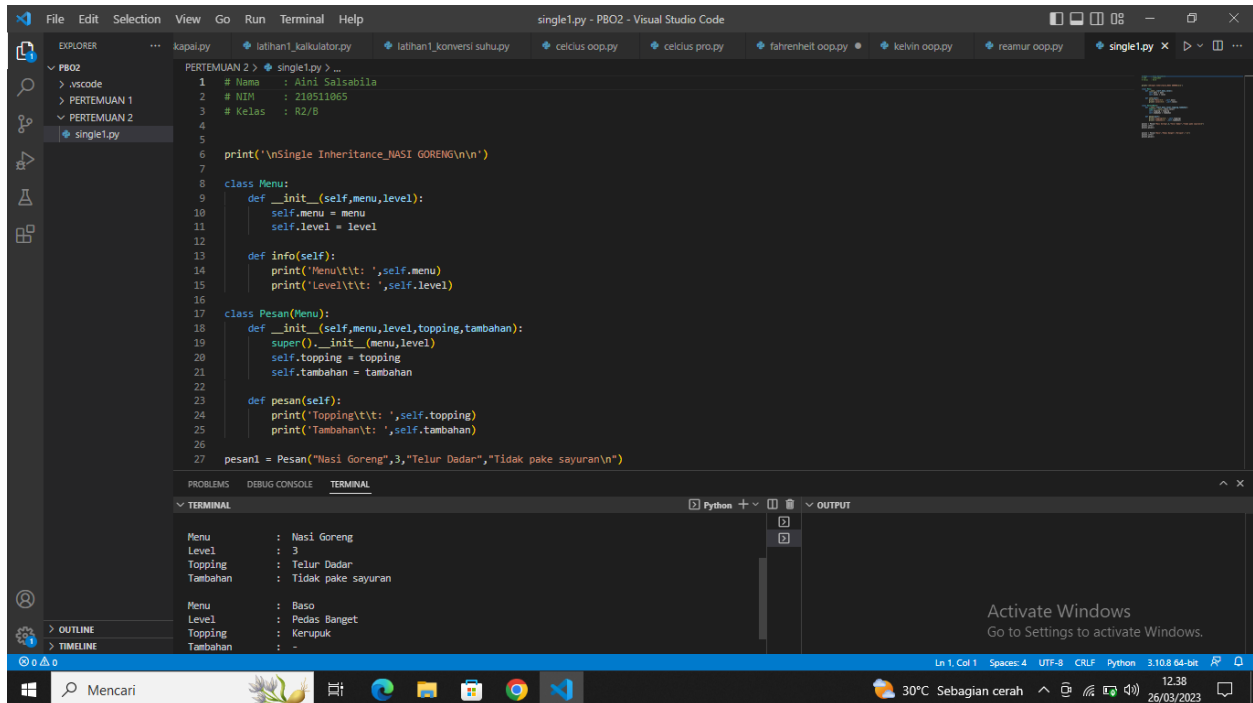
```
pesan1.info()
```

```
pesan1.pesan()
```

```
pesan2 = Pesan("Baso", "Pedas Banget", "Kerupuk", "-\n")
pesan2.info()
```

```
pesan2.pesan()
```

output:



The screenshot shows a Visual Studio Code window with a Python file named 'single1.py'. The code defines a 'Menu' class with an 'info' method and a 'Pesan' class that inherits from 'Menu'. The 'Pesan' class has a 'pesan' method. An instance of 'Pesan' is created with the name 'Nasi Goreng', level 3, topping 'Telur Dadar', and no vegetables. The terminal output shows the results of calling 'info' and 'pesan' on this instance.

```
1 # Nama : Aini Salsabila
2 # NIM : 210511065
3 # Kelas : R2/B
4
5
6 print('\nSingle Inheritance_NASI GORENG\n\n')
7
8 class Menu:
9     def __init__(self, menu, level):
10         self.menu = menu
11         self.level = level
12
13     def info(self):
14         print('Menu\t\t: ', self.menu)
15         print('Level\t\t: ', self.level)
16
17 class Pesan(Menu):
18     def __init__(self, menu, level, topping, tambahan):
19         super().__init__(menu, level)
20         self.topping = topping
21         self.tambahan = tambahan
22
23     def pesan(self):
24         print('Topping\t\t: ', self.topping)
25         print('Tambahan\t: ', self.tambahan)
26
27 pesan1 = Pesan("Nasi Goreng", 3, "Telur Dadar", "Tidak pake sayuran\n")
```

Terminal Output:

```
Menu : Nasi Goreng
Level : 3
Topping : Telur Dadar
Tambahan : Tidak pake sayuran

Menu : Baso
Level : Pedas Banget
Topping : Kerupuk
Tambahan : -
```

b. contoh 2 :

```
# Nama : Aini Salsabila
# NIM : 210511065
# Kelas : R2/B
```

```
print("\nSingle Inheritance_DATA\n\n")
```

class diri:

```
def __init__(self):
    self.nama = "Aini Salsabila"
    self.umur = 21

def info(self):
    print('Nama\t\t: ', self.nama)
    print(f'Umur\t\t: {self.umur} Tahun')
```

class data(diri):

```

def __init__(self):
    super().__init__()
    self.status = "Mahasiswa"
    self.univ = "Universitas Muhammadiyah Cirebon"
    self.prodi = "Teknik Informatika"
    self.alamat = "Brebes\n"

def display(self):
    print('Status\t\t: ',self.status)
    print('Universitas\t: ',self.univ)
    print('Jurusan\t\t: ',self.prodi)
    print('Alamat\t\t: ',self.alamat)

```

```

a = data()
a.info()
a.display()

```

output:

The screenshot shows a Visual Studio Code window with a file named 'single2.py'. The code defines a base class 'diri' and a derived class 'data'. The 'data' class inherits from 'diri' and overrides the 'display' method. The terminal output shows the execution of the 'data' class, displaying the status, university, faculty, and address.

```

1 # Nama : Aini Salsabila
2 # NIM : 210511065
3 # Kelas : R2/B
4
5
6 print('\nSingle Inheritance_DATA\n\n')
7
8 class diri:
9
10     def __init__(self):
11         self.nama = "Aini Salsabila"
12         self.umur = 21
13
14     def info(self):
15         print('Nama\t\t: ',self.nama)
16         print('Umur\t\t: {self.umur} Tahun')
17
18 class data(diri):
19     def __init__(self):
20         super().__init__()
21         self.status = "Mahasiswa"
22         self.univ = "Universitas Muhammadiyah Cirebon"
23         self.prodi = "Teknik Informatika"
24         self.alamat = "Brebes\n"
25
26     def display(self):
27         print('Status\t\t: ',self.status)

```

TERMINAL OUTPUT:

```

Nama      : Aini Salsabila
Umur      : 21 Tahun
Status    : Mahasiswa
Universitas : Universitas Muhammadiyah Cirebon
Jurusan   : Teknik Informatika
Alamat    : Brebes

```

2. Multiple Inheritance :

a. contoh 1:

```
print("\nMultiple Inheritance_NCT\n")
```

```

class NCT:
    def __init__(self, nama, asal):
        self.nama = nama

```

```

        self.asal = asal
    def display_info(self):
        print(f"Nama: {self.nama}")
        print(f"Asal: {self.asal}")

class Dream:
    def __init__(self, posisi, line):
        self.posisi = posisi
        self.line = line
    def display_info(self):
        print(f"Posisi: {self.posisi}")
        print(f"Line: {self.line}")

class Ilichil:
    def __init__(self, posisi1, line1):
        self.posisi1 = posisi1
        self.line1 = line1
    def display_info(self):
        print(f"Posisi: {self.posisi1}")
        print(f"Line: {self.line1}")

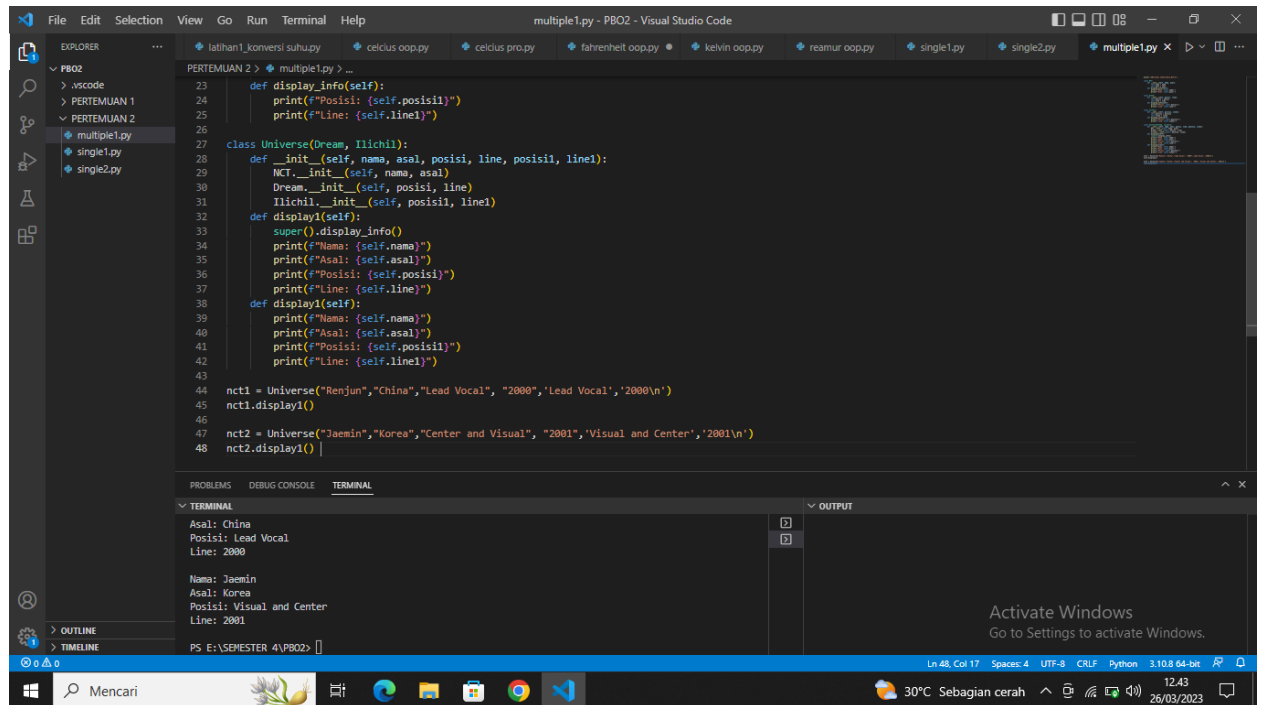
class Universe(Dream, Ilichil):
    def __init__(self, nama, asal, posisi, line, posisi1, line1):
        NCT.__init__(self, nama, asal)
        Dream.__init__(self, posisi, line)
        Ilichil.__init__(self, posisi1, line1)
    def display1(self):
        super().display_info()
        print(f"Nama: {self.nama}")
        print(f"Asal: {self.asal}")
        print(f"Posisi: {self.posisi}")
        print(f"Line: {self.line}")
    def display1(self):
        print(f"Nama: {self.nama}")
        print(f"Asal: {self.asal}")
        print(f"Posisi: {self.posisi1}")
        print(f"Line: {self.line1}")

nct1 = Universe("Renjun", "China", "Lead Vocal", "2000", 'Lead Vocal', '2000\n')
nct1.display1()

nct2 = Universe("Jaemin", "Korea", "Center and Visual", "2001", 'Visual and Center', '2001\n')
nct2.display1()

```

output:



b. contoh 2:

```
print('\nMultiple Inheritance_FILM\n')
```

class Film:

```
def __init__(self, judul, tahun, asal):
    self.judul = judul
    self.tahun = tahun
    self.asal = asal
```

```
def display(self):
    print('Judul\t\t: ', self.judul)
    print('Tahun\t\t: ', self.tahun)
    print('Produksi\t\t: ', self.asal)
```

class Jenis:

```
def __init__(self, jenis, genre):
    self.jenis = jenis
    self.genre = genre
```

```
def display(self):
    print('Jenis\t\t: ', self.jenis)
    print('Genre\t\t: ', self.genre)
```

class FilmDetail(Film, Jenis):

```
def __init__(self, judul, tahun, asal, jenis, genre):
    Film.__init__(self, judul, tahun, asal)
```

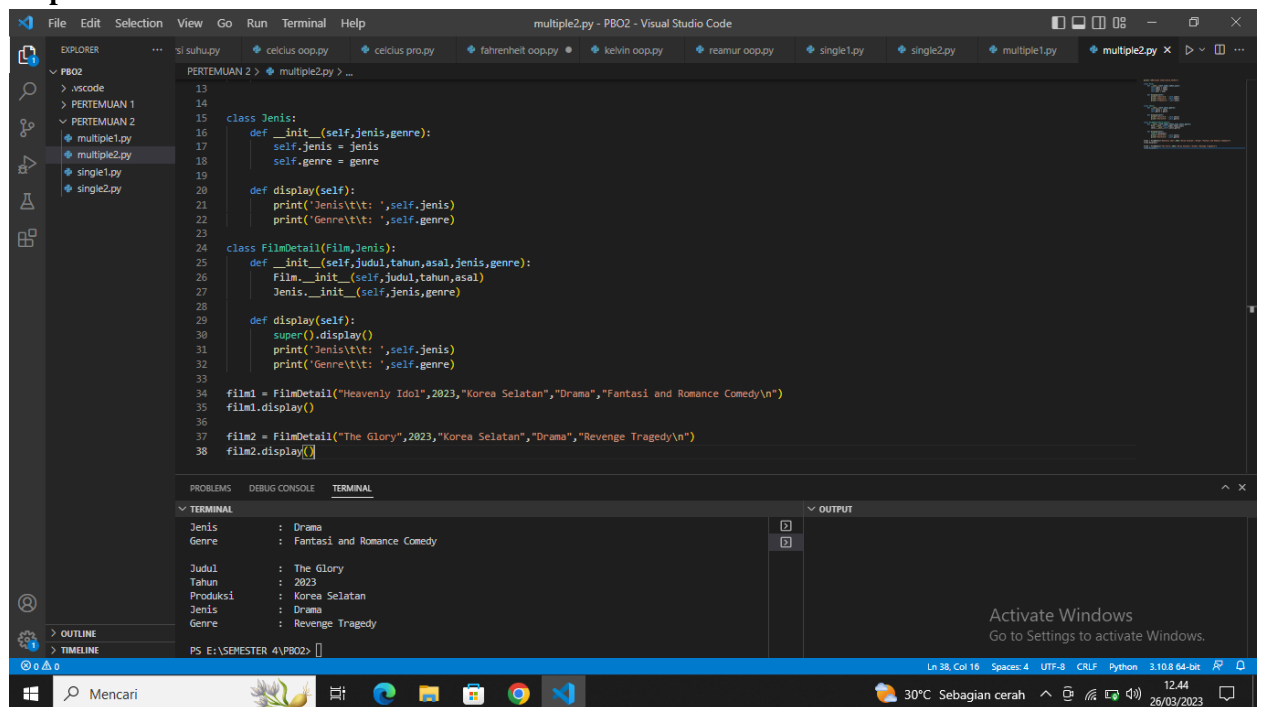
```
Jenis.__init__(self,jenis,genre)
```

```
def display(self):  
    super().display()  
    print('Jenis\t\t: ',self.jenis)  
    print('Genre\t\t: ',self.genre)
```

```
film1 = FilmDetail("Heavenly Idol",2023,"Korea Selatan","Drama","Fantasi and Romance  
Comedy\n")  
film1.display()
```

```
film2 = FilmDetail("The Glory",2023,"Korea Selatan","Drama","Revenge Tragedy\n")  
film2.display()
```

output:



The screenshot shows a Visual Studio Code window with a Python file named 'multiple2.py'. The code defines two classes: 'Jenis' and 'FilmDetail'. 'Jenis' has an '__init__' method that takes 'jenis' and 'genre' as arguments and a 'display' method that prints them. 'FilmDetail' inherits from 'Jenis' and has an '__init__' method that takes 'judul', 'tahun', 'asal', 'jenis', and 'genre' as arguments, and a 'display' method that calls 'super().display()' and prints the 'judul' and 'tahun' attributes. The script creates two instances of 'FilmDetail': 'film1' with details for 'Heavenly Idol' and 'film2' with details for 'The Glory'. The terminal output shows the results of calling 'display()' on both instances.

```
13  
14  
15 class Jenis:  
16     def __init__(self,jenis,genre):  
17         self.jenis = jenis  
18         self.genre = genre  
19  
20     def display(self):  
21         print('Jenis\t\t: ',self.jenis)  
22         print('Genre\t\t: ',self.genre)  
23  
24 class FilmDetail(Film,Jenis):  
25     def __init__(self,judul,tahun,asal,jenis,genre):  
26         Film.__init__(self,judul,tahun,asal)  
27         Jenis.__init__(self,jenis,genre)  
28  
29     def display(self):  
30         super().display()  
31         print('Judul\t\t: ',self.judul)  
32         print('Tahun\t\t: ',self.tahun)  
33  
34 film1 = FilmDetail("Heavenly Idol",2023,"Korea Selatan","Drama","Fantasi and Romance Comedy\n")  
35 film1.display()  
36  
37 film2 = FilmDetail("The Glory",2023,"Korea Selatan","Drama","Revenge Tragedy\n")  
38 film2.display()
```

TERMINAL

```
Jenis      : Drama  
Genre      : Fantasi and Romance Comedy  
  
Judul      : The Glory  
Tahun      : 2023  
Produksi   : Korea Selatan  
Jenis      : Drama  
Genre      : Revenge Tragedy
```

3. Hierarchical Inheritance :

a. contoh 1:

```
print("\nHierarchical Inheritance_KPOP\n\n")
```

```
class Grup:  
    def __init__(self, grup, anggota):  
        self.grup = grup  
        self.anggota = anggota  
  
    def ket(self):
```

```

        print(f'{self.grup} beranggotakan {self.anggota} orang\n\n')

    def getGrup(self):
        return self.grup

    def getAnggota(self):
        return self.anggota

class Gen(Grup):
    def __init__(self, grup, anggota, gen):
        super().__init__(grup, anggota)
        self.gen = gen

    def detail(self):
        print(f'Grup {self.grup} merupakan Generasi Ke-{self.gen} Kpop\n')

    def getGen(self):
        return self.gen

class Agensi(Gen):
    def __init__(self, grup, anggota, gen, fandom, agensi):
        super().__init__(grup, anggota, gen)
        self.fandom = fandom
        self.agenisi = agensi

    def keterangan(self):
        print(f'Grup: {self.grup}\nAnggota: {self.anggota} Orang\nFandom: {self.fandom}\nGen: {self.gen}\nAgenisi: {self.agenisi}\n')

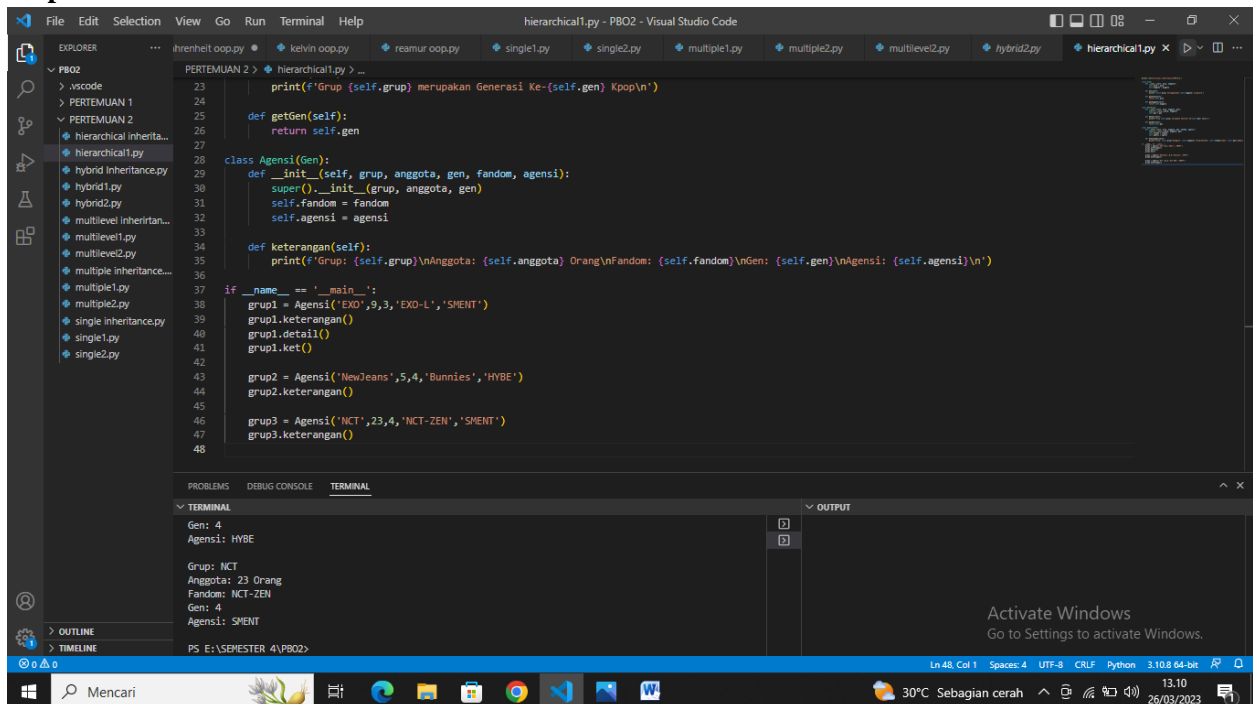
if __name__ == '__main__':
    grup1 = Agensi('EXO',9,3,'EXO-L','SMENT')
    grup1.keterangan()
    grup1.detail()
    grup1.ket()

    grup2 = Agensi('NewJeans',5,4,'Bunnies','HYBE')
    grup2.keterangan()

    grup3 = Agensi('NCT',23,4,'NCT-ZEN','SMENT')
    grup3.keterangan()

```


output:



```
File Edit Selection View Go Run Terminal Help
hierarchical1.py - PBO2 - Visual Studio Code

EXPLORER
PBO2
  > .vscode
  > PERTEMUAN 1
  > PERTEMUAN 2
    > hierarchical inherita...
      hierarchical1.py
    > hybrid inheritance.py
    > hybrid1.py
    > hybrid2.py
    > multilevel inheritan...
    > multilevel1.py
    > multilevel2.py
    > multiple inheritance...
    > multiple1.py
    > multiple2.py
    > single inheritance.py
    > single1.py
    > single2.py

hierarchical1.py
23 print(f'Grup {self.grup} merupakan Generasi Ke-{self.gen} Kpop\n')
24
25 def getGen(self):
26     return self.gen
27
28 class Agensi(Gen):
29     def __init__(self, grup, anggota, gen, fandom, agensi):
30         super().__init__(grup, anggota, gen)
31         self.fandom = fandom
32         self.agenisi = agensi
33
34     def keterangan(self):
35         print(f'Grup: {self.grup}\nAnggota: {self.anggota} Orang\nFandom: {self.fandom}\nGen: {self.gen}\nAgensi: {self.agenisi}\n')
36
37 if __name__ == '__main__':
38     grup1 = Agensi('EXO', 9, 3, 'EXO-L', 'SMENT')
39     grup1.keterangan()
40     grup1.detail()
41
42     grup2 = Agensi('NewJeans', 5, 4, 'Bunnies', 'HYBE')
43     grup2.keterangan()
44
45     grup3 = Agensi('NCT', 23, 4, 'NCT-ZEN', 'SMENT')
46     grup3.keterangan()
47
48
PROBLEMS DEBUG CONSOLE TERMINAL
TERMINAL
Gen: 4
Agensi: HYBE
Grup: NCT
Anggota: 23 Orang
Fandom: NCT-ZEN
Gen: 4
Agensi: SMENT
PS E:\SEMESTER 4\PBO2>
OUTPUT
Activate Windows
Go to Settings to activate Windows.
Ln 48, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.8 64-bit 13:10 26/03/2023
```

b. contoh 2:

```
print("\nHierarchical Inheritance_Mahasiswa\n\n")
```

```
class Mahasiswa:
```

```
    def __init__(self, name, nim):
```

```
        self.name = name
```

```
        self.nim = nim
```

```
    def ket(self):
```

```
        print(f'{self.name} adalah Mahasiswa UMC dengan NIM {self.nim}\n')
```

```
    def getName(self):
```

```
        return self.name
```

```
    def getNim(self):
```

```
        return self.nim
```

```
class Fakultas(Mahasiswa):
```

```
    def __init__(self, name, nim, fakultas):
```

```
        super().__init__(name, nim)
```

```
        self.fakultas = fakultas
```

```
    def detail(self):
```

```
print(f'Nama: {self.name}\nNim: {self.nim}\nFakultas: {self.fakultas}\n')
```

```
def getFakultas(self):  
    return self.fakultas
```

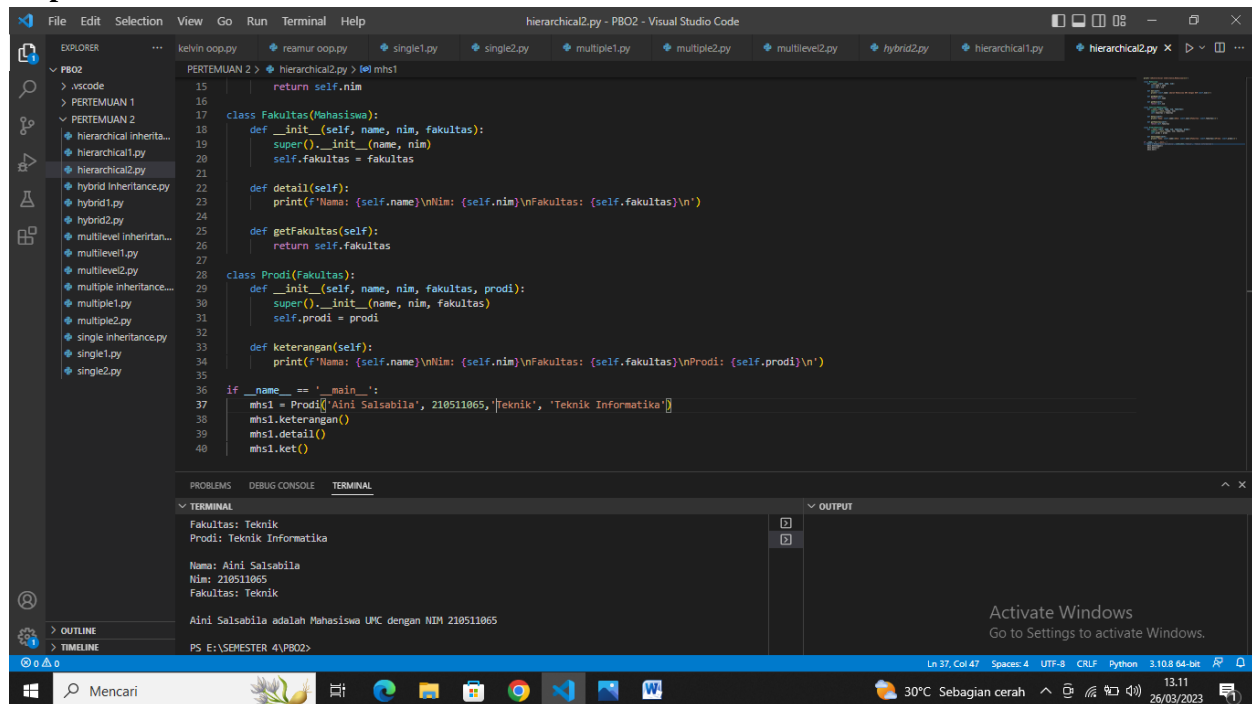
```
class Prodi(Fakultas):
```

```
    def __init__(self, name, nim, fakultas, prodi):  
        super().__init__(name, nim, fakultas)  
        self.prodi = prodi
```

```
    def keterangan(self):  
        print(f'Nama: {self.name}\nNim: {self.nim}\nFakultas: {self.fakultas}\nProdi: {self.prodi}\n')
```

```
if __name__ == '__main__':  
    mhs1 = Prodi('Aini Salsabila', 210511065, 'Teknik', 'Teknik Informatika')  
    mhs1.keterangan()  
    mhs1.detail()  
    mhs1.ket()
```

output:



The screenshot shows a Visual Studio Code window with a Python file named 'hierarchical2.py'. The code defines a base class 'Fakultas' and a derived class 'Prodi'. The 'Fakultas' class has attributes 'name', 'nim', and 'fakultas', and methods 'detail' and 'getFakultas'. The 'Prodi' class inherits from 'Fakultas' and adds a 'prodi' attribute and a 'keterangan' method. The main block creates an instance 'mhs1' of the 'Prodi' class and calls its 'keterangan', 'detail', and 'ket' methods.

The terminal output shows the following text:

```
Fakultas: Teknik  
Prodi: Teknik Informatika  
  
Nama: Aini Salsabila  
Nim: 210511065  
Fakultas: Teknik  
  
Aini Salsabila adalah Mahasiswa UMC dengan NIM 210511065  
PS E:\SEMESTER 4\PB02>
```

4. Multilevel Inheritance :

a. Contoh 1:

```
class Animal:
```

```
    def __init__(self, name):
        self.name = name
    def speak(self):
        print("The animal speaks")
```

```
class Cat(Animal):
```

```
    def __init__(self, name, breed):
        super().__init__(name)
        self.breed = breed
    def speak(self):
        print("The cat groaned")
```

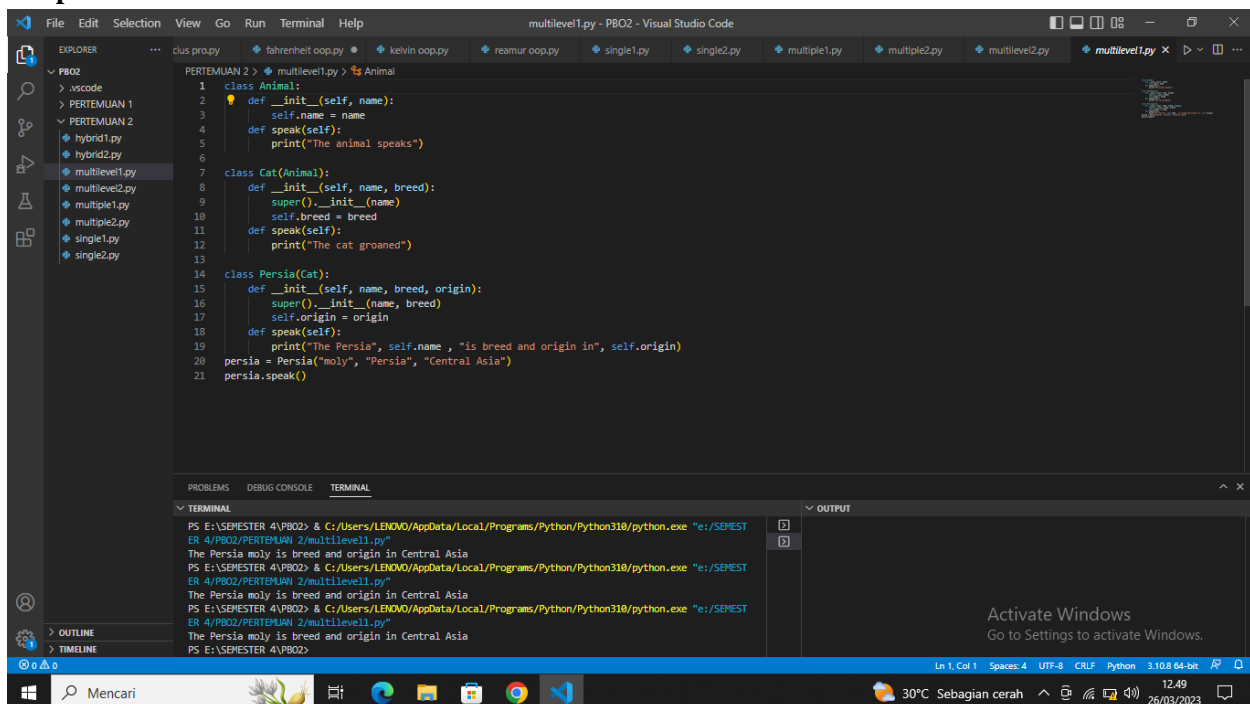
```
class Persia(Cat):
```

```
    def __init__(self, name, breed, origin):
        super().__init__(name, breed)
        self.origin = origin
    def speak(self):
        print("The Persia", self.name, "is breed and origin in", self.origin)
```

```
persia = Persia("moly", "Persia", "Central Asia")
```

```
persia.speak()
```

Output:



The screenshot shows a Visual Studio Code window with a Python file named `multilevel1.py`. The code defines three classes: `Animal`, `Cat`, and `Persia`, demonstrating multilevel inheritance. The `Persia` class inherits from `Cat`, which inherits from `Animal`. The `Persia` class has an additional attribute `origin` and a `speak` method that prints the name, breed, and origin. The script creates an instance of `Persia` named `persia` with the attributes "moly", "Persia", and "Central Asia", and then calls the `speak` method.

```
1 class Animal:
2     def __init__(self, name):
3         self.name = name
4     def speak(self):
5         print("The animal speaks")
6
7 class Cat(Animal):
8     def __init__(self, name, breed):
9         super().__init__(name)
10        self.breed = breed
11    def speak(self):
12        print("The cat groaned")
13
14 class Persia(Cat):
15     def __init__(self, name, breed, origin):
16         super().__init__(name, breed)
17         self.origin = origin
18     def speak(self):
19         print("The Persia", self.name, "is breed and origin in", self.origin)
20
21 persia = Persia("moly", "Persia", "Central Asia")
22 persia.speak()
```

The terminal output shows the execution of the script, resulting in the following lines:

```
PS E:\SEMESTER 4\PB02> C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe "E:/SEMESTER 4/PB02/PERTEMUAN 2/multilevel1.py"
The Persia moly is breed and origin in Central Asia
PS E:\SEMESTER 4\PB02> C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe "E:/SEMESTER 4/PB02/PERTEMUAN 2/multilevel1.py"
The Persia moly is breed and origin in Central Asia
PS E:\SEMESTER 4\PB02> C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe "E:/SEMESTER 4/PB02/PERTEMUAN 2/multilevel1.py"
The Persia moly is breed and origin in Central Asia
PS E:\SEMESTER 4\PB02>
```

An "Activate Windows" watermark is visible in the bottom right corner of the terminal window.

b. Contoh 2:

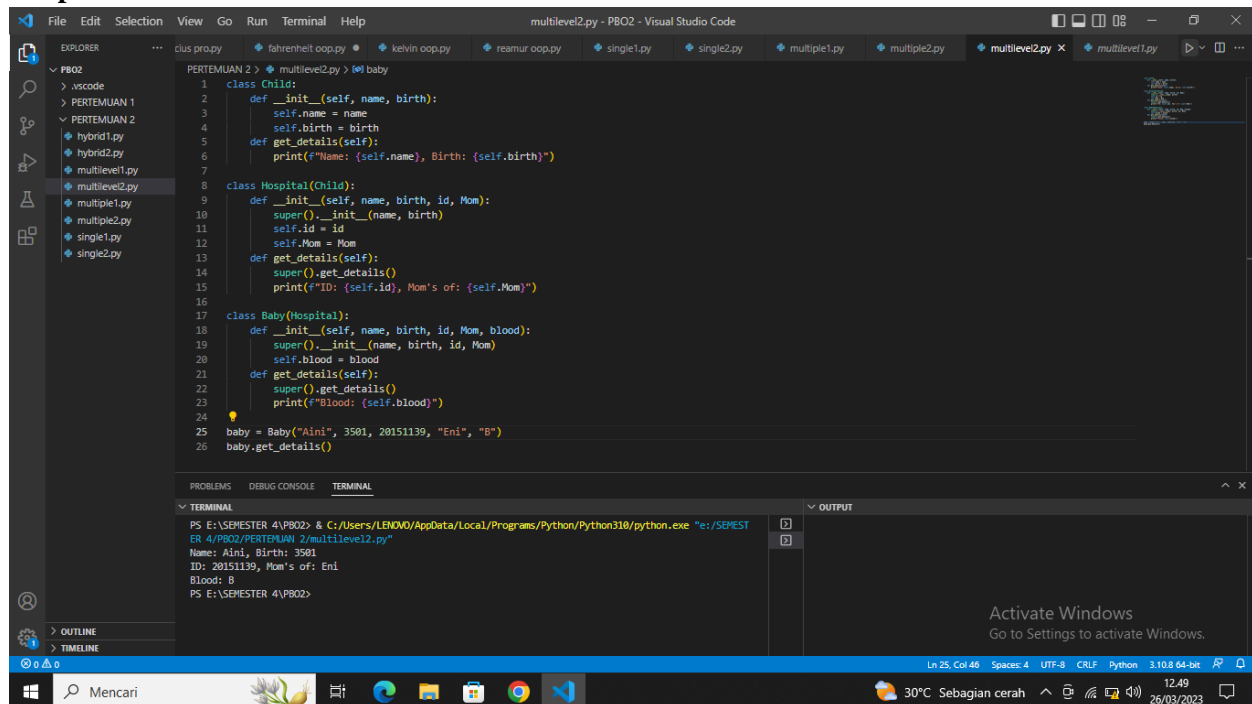
```
class Child:
    def __init__(self, name, birth):
        self.name = name
        self.birth = birth
    def get_details(self):
        print(f"Name: {self.name}, Birth: {self.birth}")

class Hospital(Child):
    def __init__(self, name, birth, id, Mom):
        super().__init__(name, birth)
        self.id = id
        self.Mom = Mom
    def get_details(self):
        super().get_details()
        print(f"ID: {self.id}, Mom's of: {self.Mom}")

class Baby(Hospital):
    def __init__(self, name, birth, id, Mom, blood):
        super().__init__(name, birth, id, Mom)
        self.blood = blood
    def get_details(self):
        super().get_details()
        print(f"Blood: {self.blood}")

baby = Baby("Aini", 3501, 20151139, "Eni", "B")
baby.get_details()
```

Output:



```
1 class Child:
2     def __init__(self, name, birth):
3         self.name = name
4         self.birth = birth
5     def get_details(self):
6         print(f"Name: {self.name}, Birth: {self.birth}")
7
8 class Hospital(Child):
9     def __init__(self, name, birth, id, Mom):
10        super().__init__(name, birth)
11        self.id = id
12        self.Mom = Mom
13    def get_details(self):
14        super().get_details()
15        print(f"ID: {self.id}, Mom's of: {self.Mom}")
16
17 class Baby(Hospital):
18     def __init__(self, name, birth, id, Mom, blood):
19        super().__init__(name, birth, id, Mom)
20        self.blood = blood
21    def get_details(self):
22        super().get_details()
23        print(f"Blood: {self.blood}")
24
25 baby = Baby("Aini", 3501, 20151139, "Eni", "B")
26 baby.get_details()
```

PS E:\SEMESTER 4\PROG2 & C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe "E:\SEMESTER 4\PROG2\PERTEMUAN 2\multilevel2.py"

Name: Aini, Birth: 3501
ID: 20151139, Mom's of: Eni
Blood: B
PS E:\SEMESTER 4\PROG2>

5. Hybrid Inheritance :

a. Contoh 1:

class GameObject:

```
def __init__(self, x, y):
    self.x = x
    self.y = y
```

class Drawable:

```
def draw(self):
    print("Drawing object at: ", self.x, self.y)
```

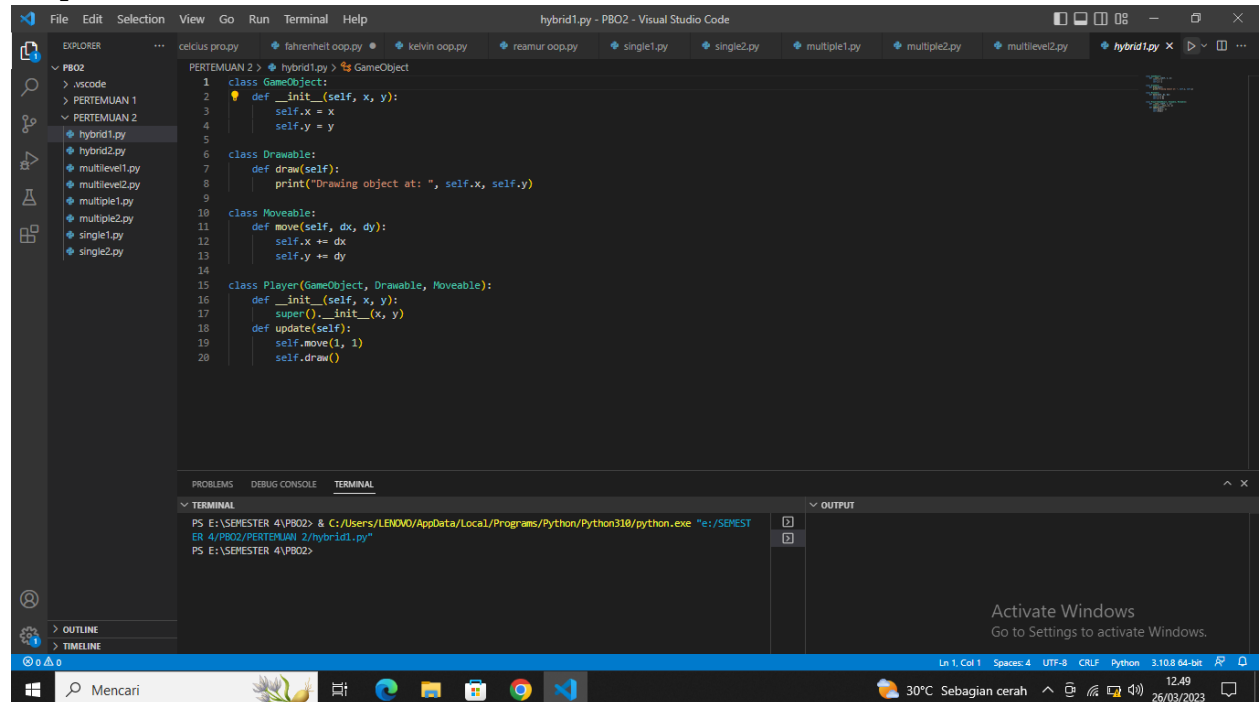
class Moveable:

```
def move(self, dx, dy):
    self.x += dx
    self.y += dy
```

class Player(GameObject, Drawable, Moveable):

```
def __init__(self, x, y):
    super().__init__(x, y)
def update(self):
    self.move(1, 1)
    self.draw()
```

Output:



b. Contoh 2:

class Seseorang:

```
def __init__(self, name, age, address):
    self.name = name
    self.age = age
    self.address = address
def get_info(self):
    print("Name:", self.name)
    print("Age:", self.age)
    print("Address:", self.address)
```

class Siswa(Seseorang):

```
def __init__(self, name, age, address, student_id):
    super().__init__(name, age, address)
    self.student_id = student_id
def get_info(self):
    super().get_info()
    print("Student ID:", self.student_id)
```

class Employee(Seseorang):

```
def __init__(self, name, age, address, employee_id, salary):
    super().__init__(name, age, address)
    self.employee_id = employee_id
    self.salary = salary
```

```
def get_info(self):
    super().get_info()
    print("Employee ID:", self.employee_id)
    print("Salary:", self.salary)
```

class Pengarang(Employee, Siswa):

```
def __init__(self, name, age, address, employee_id, salary, student_id, published_books):
    Employee.__init__(self, name, age, address, employee_id, salary)
    Siswa.__init__(self, name, age, address, student_id)
    self.published_books = published_books
def get_info(self):
    super().get_info()
    print("Student ID:", self.student_id)
    print("Published Books:", self.published_books)
```

Output:

The screenshot shows a Visual Studio Code editor with a Python file named `hybrid2.py`. The code defines a base class `Seseorang` and two subclasses, `Siswa` and `Employee`, both inheriting from `Seseorang`. The `Siswa` class adds a `student_id` attribute and a `get_info` method. The `Employee` class adds `employee_id` and `salary` attributes and a `get_info` method. The `hybrid2.py` file is open in the editor, and the terminal shows the output of running the code.

```
1 class Seseorang:
2     def __init__(self, name, age, address):
3         self.name = name
4         self.age = age
5         self.address = address
6     def get_info(self):
7         print("Name:", self.name)
8         print("Age:", self.age)
9         print("Address:", self.address)
10
11 class Siswa(Seseorang):
12     def __init__(self, name, age, address, student_id):
13         super().__init__(name, age, address)
14         self.student_id = student_id
15     def get_info(self):
16         super().get_info()
17         print("Student ID:", self.student_id)
18
19 class Employee(Seseorang):
20     def __init__(self, name, age, address, employee_id, salary):
21         super().__init__(name, age, address)
22         self.employee_id = employee_id
23         self.salary = salary
24     def get_info(self):
25         super().get_info()
26         print("Employee ID:", self.employee_id)
27         print("Salary:", self.salary)
```

The terminal output shows the execution of the code, displaying the name, age, and address for the `Siswa` and `Employee` objects, followed by their respective `student_id` and `employee_id` and `salary`.

```
PS E:\SEMESTER 4\PROG2> & C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe "e:\SEMESTER 4\PROG2\PERTEMUAN 2\hybrid1.py"
PS E:\SEMESTER 4\PROG2> & C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe "e:\SEMESTER 4\PROG2\PERTEMUAN 2\hybrid2.py"
PS E:\SEMESTER 4\PROG2>
```

The bottom of the screenshot shows the Windows taskbar with the date and time as 26/03/2023, 12:49, and the system temperature as 30°C.