

LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK LANJUT

2023



Prepared By:

Nama : Zidan Khoirul Rizki

NIM : 210511049

Kelas : R2

PRAKTIKUM 2

Buatlah masing-masing 2 contoh jenis pewarisan di luar dari contoh yang telah diberikan, beri nama :

1. Single1.py

```
class Animal:

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

    def speak(self):
        print("")

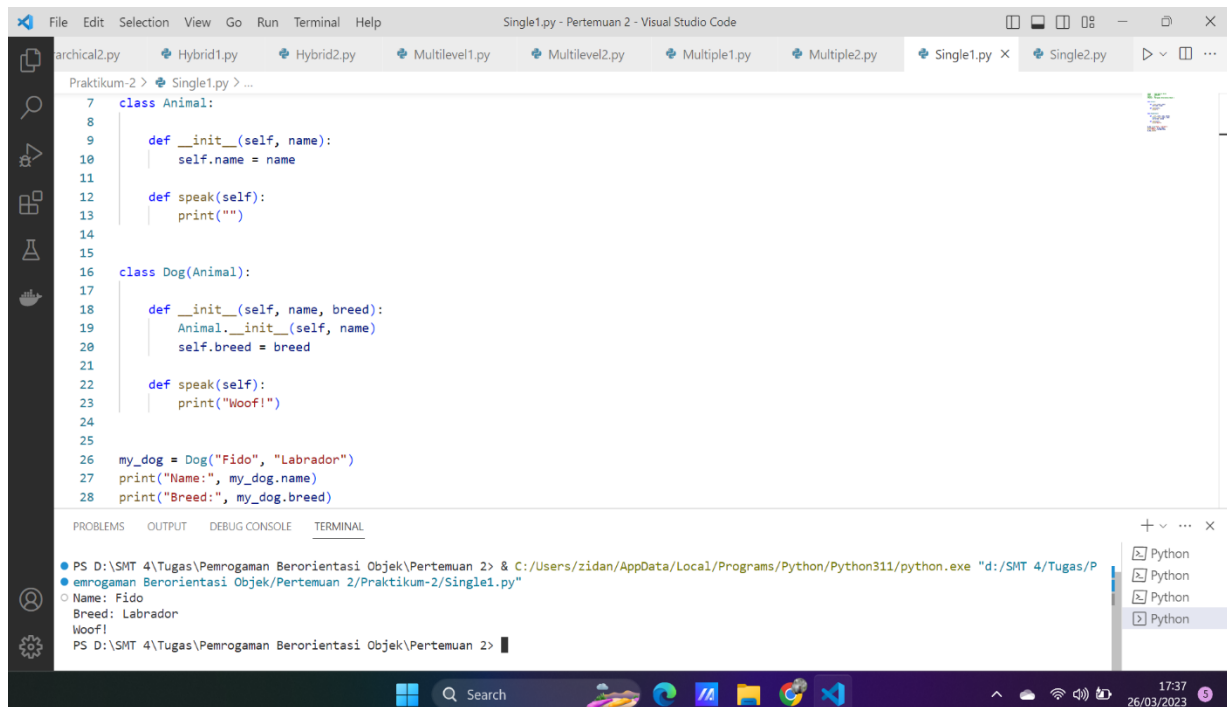
class Dog(Animal):

    def __init__(self, name, breed):
        Animal.__init__(self, name)
        self.breed = breed

    def speak(self):
        print("Woof!")

my_dog = Dog("Fido", "Labrador")
print("Name:", my_dog.name)
print("Breed:", my_dog.breed)
my_dog.speak()
```

OUTPUT



The screenshot shows the Visual Studio Code interface. The editor window displays the Python code for the 'Single1.py' file. The code defines an 'Animal' class with an '.__init__' method and a 'speak' method. A 'Dog' class inherits from 'Animal' and overrides the '.__init__' and 'speak' methods. An instance 'my_dog' is created with the name 'Fido' and breed 'Labrador'. The code then prints the name and breed, and calls the 'speak' method. The terminal window at the bottom shows the output of the code execution: 'Name: Fido', 'Breed: Labrador', and 'Woof!'. The status bar at the bottom indicates the file is located at 'PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2>' and the current time is 17:37 on 26/03/2023.

```
File Edit Selection View Go Run Terminal Help Single1.py - Pertemuan 2 - Visual Studio Code
Praktikum-2 > Single1.py > ...
7 class Animal:
8
9     def __init__(self, name):
10         self.name = name
11
12     def speak(self):
13         print("")
14
15
16 class Dog(Animal):
17
18     def __init__(self, name, breed):
19         Animal.__init__(self, name)
20         self.breed = breed
21
22     def speak(self):
23         print("Woof!")
24
25
26 my_dog = Dog("Fido", "Labrador")
27 print("Name:", my_dog.name)
28 print("Breed:", my_dog.breed)
29
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2> C:\Users\zidan\AppData\Local\Programs\Python\Python311\python.exe "d:/SMT 4/Tugas/P
emrograman Berorientasi Objek/Pertemuan 2/Praktikum-2/Single1.py"
Name: Fido
Breed: Labrador
Woof!
PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2>
```

2. Single2.py

```
class Vehicle:

    def __init__(self, color):
        self.color = color

    def start(self):
        print("Starting vehicle...")

class Car(Vehicle):

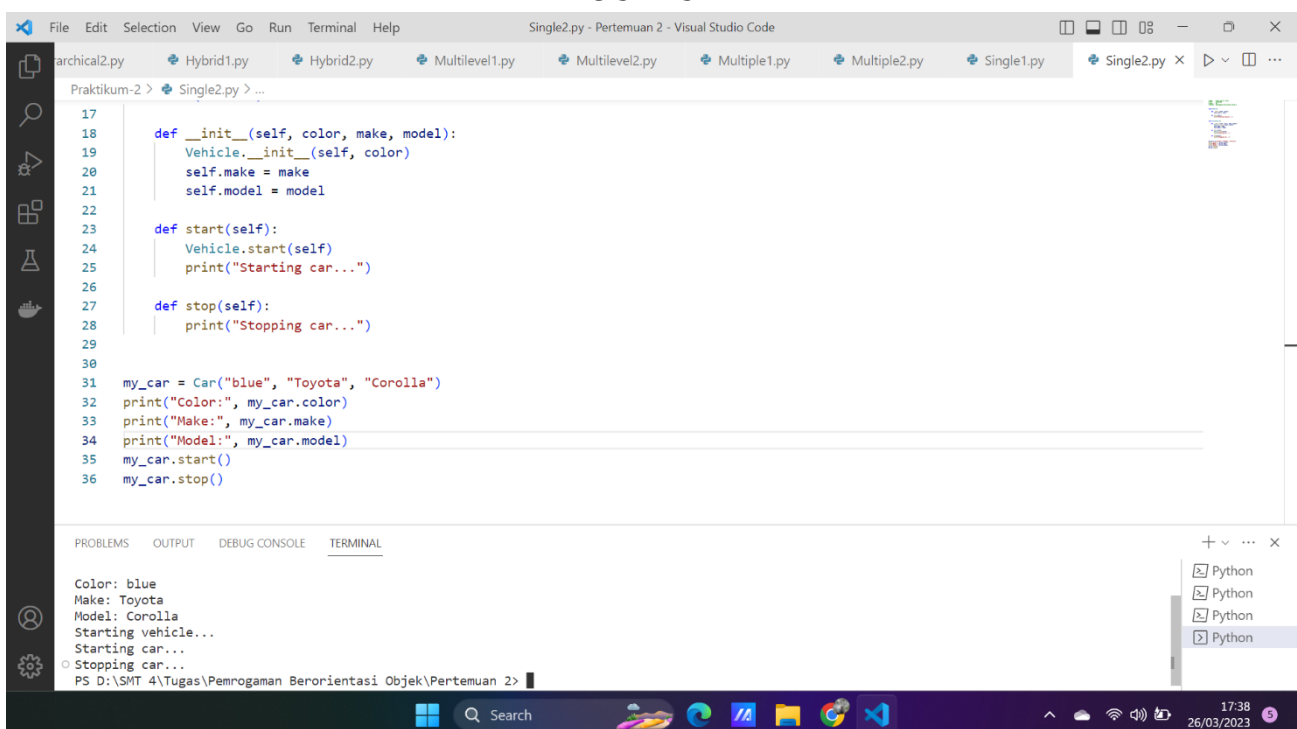
    def __init__(self, color, make, model):
        Vehicle.__init__(self, color)
        self.make = make
        self.model = model

    def start(self):
        Vehicle.start(self)
        print("Starting car...")

    def stop(self):
        print("Stopping car...")

my_car = Car("blue", "Toyota", "Corolla")
print("Color:", my_car.color)
print("Make:", my_car.make)
print("Model:", my_car.model)
my_car.start()
my_car.stop()
```

OUTPUT



```
File Edit Selection View Go Run Terminal Help Single2.py - Pertemuan 2 - Visual Studio Code
Praktikum-2 > Single2.py > ...
17
18     def __init__(self, color, make, model):
19         Vehicle.__init__(self, color)
20         self.make = make
21         self.model = model
22
23     def start(self):
24         Vehicle.start(self)
25         print("Starting car...")
26
27     def stop(self):
28         print("Stopping car...")
29
30
31 my_car = Car("blue", "Toyota", "Corolla")
32 print("Color:", my_car.color)
33 print("Make:", my_car.make)
34 print("Model:", my_car.model)
35 my_car.start()
36 my_car.stop()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Color: blue
Make: Toyota
Model: Corolla
Starting vehicle...
Starting car...
Stopping car...
PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2>
```

3. Multiple1.py

```
class Person:
```

```
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

```
    def show_info(self):
        print("Name:", self.name)
        print("Age:", self.age)
```

```
class Employee:
```

```
    def __init__(self, salary, job_title):
        self.salary = salary
        self.job_title = job_title
```

```
    def show_info(self):
        print("Salary:", self.salary)
        print("Job Title:", self.job_title)
```

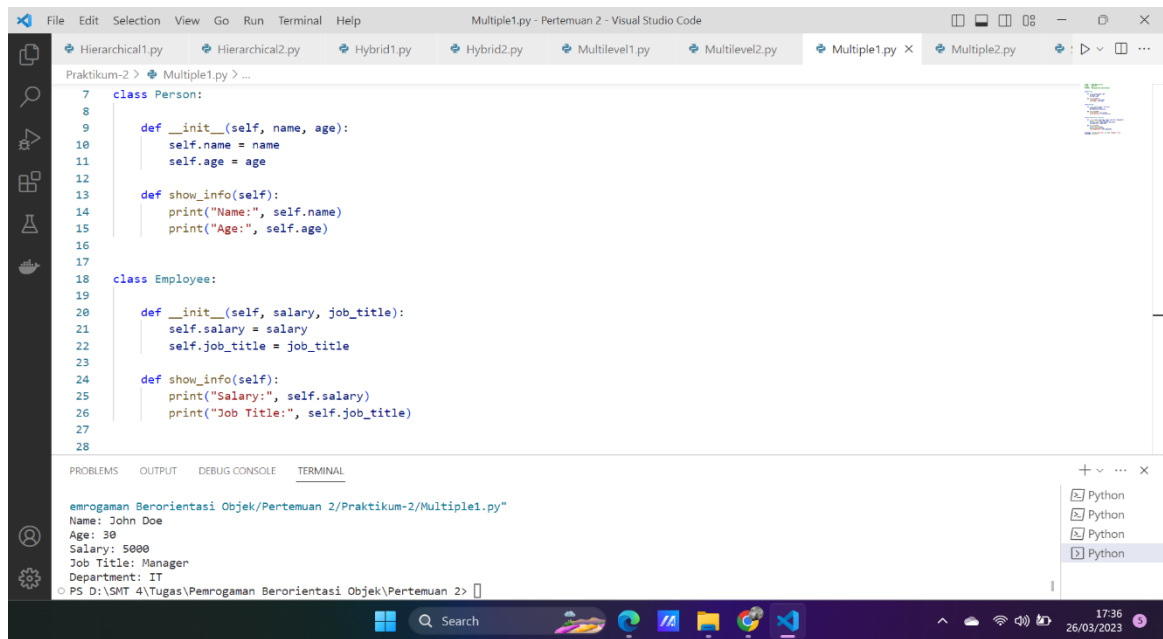
```
class Manager(Person, Employee):
```

```
    def __init__(self, name, age, salary, job_title, department):
        Person.__init__(self, name, age)
        Employee.__init__(self, salary, job_title)
        self.department = department
```

```
    def show_info(self):
        Person.show_info(self)
        Employee.show_info(self)
        print("Department:", self.department)
```

```
my_manager = Manager("John Doe", 30, 5000, "Manager", "IT")
my_manager.show_info()
```

OUTPUT



The screenshot shows the Visual Studio Code interface with a Python file named 'Multiple1.py'. The code defines two classes: 'Person' and 'Employee'. The 'Person' class has an '.__init__' method that takes 'name' and 'age' as arguments and a 'show_info' method that prints the name and age. The 'Employee' class has an '.__init__' method that takes 'salary' and 'job_title' as arguments and a 'show_info' method that prints the salary and job title. The terminal output shows the execution of the code, displaying the name 'John Doe', age '30', salary '5000', and job title 'Manager'.

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

    def show_info(self):
        print("Name:", self.name)
        print("Age:", self.age)

class Employee:
    def __init__(self, salary, job_title):
        self.salary = salary
        self.job_title = job_title

    def show_info(self):
        print("Salary:", self.salary)
        print("Job Title:", self.job_title)
```

emrogaman Berorientasi Objek/Pertemuan 2/Praktikum-2/Multiple1.py
Name: John Doe
Age: 30
Salary: 5000
Job Title: Manager
Department: IT

4. Multiple2.py

```
class Shape:
```

```
    def __init__(self, color):
        self.color = color
```

```
class Fillable:
```

```
    def __init__(self, is_filled):
        self.is_filled = is_filled
```

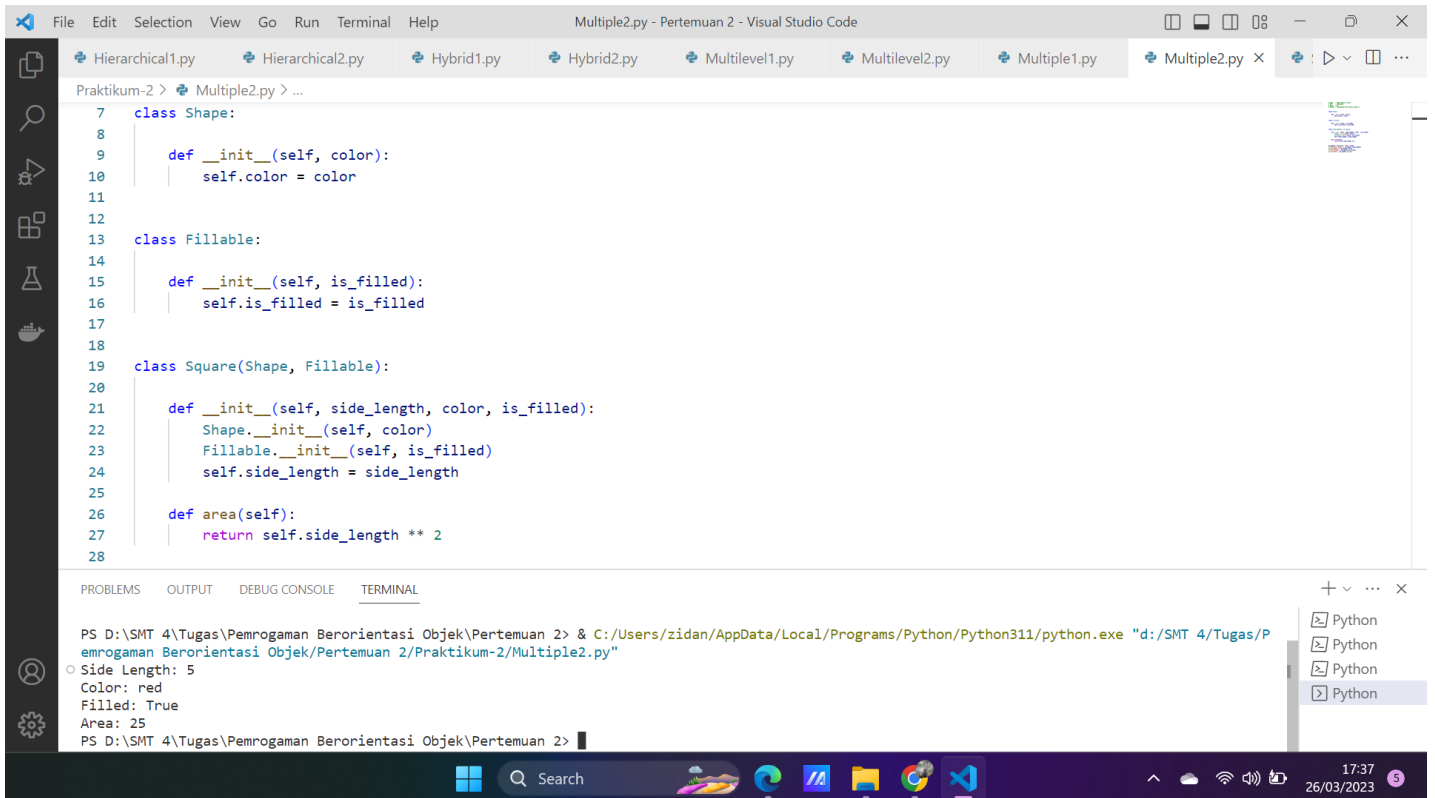
```
class Square(Shape, Fillable):
```

```
    def __init__(self, side_length, color, is_filled):
        Shape.__init__(self, color)
        Fillable.__init__(self, is_filled)
        self.side_length = side_length
```

```
    def area(self):
        return self.side_length ** 2
```

```
my_square = Square(5, "red", True)
print("Side Length:", my_square.side_length)
print("Color:", my_square.color)
print("Filled:", my_square.is_filled)
print("Area:", my_square.area())
```

OUTPUT



```
class Shape:
    def __init__(self, color):
        self.color = color

class Fillable:
    def __init__(self, is_filled):
        self.is_filled = is_filled

class Square(Shape, Fillable):
    def __init__(self, side_length, color, is_filled):
        Shape.__init__(self, color)
        Fillable.__init__(self, is_filled)
        self.side_length = side_length

    def area(self):
        return self.side_length ** 2
```

PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2> & C:/Users/zidan/AppData/Local/Programs/Python/Python311/python.exe "d:/SMT 4/Tugas/Pemrograman Berorientasi Objek/Pertemuan 2/Praktikum-2/Multiple2.py"

Side Length: 5
Color: red
Filled: True
Area: 25

PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2>

5. Hierarchical1.py

```
class Employee:
```

```
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
```

```
    def display_info(self):
        print("Name:", self.name)
        print("Salary:", self.salary)
```

```
class Manager(Employee):
```

```
    def __init__(self, name, salary, department):
        super().__init__(name, salary)
        self.department = department
```

```
    def display_info(self):

        super().display_info()
        print("Department:", self.department)
```

```
class Engineer(Employee):
```

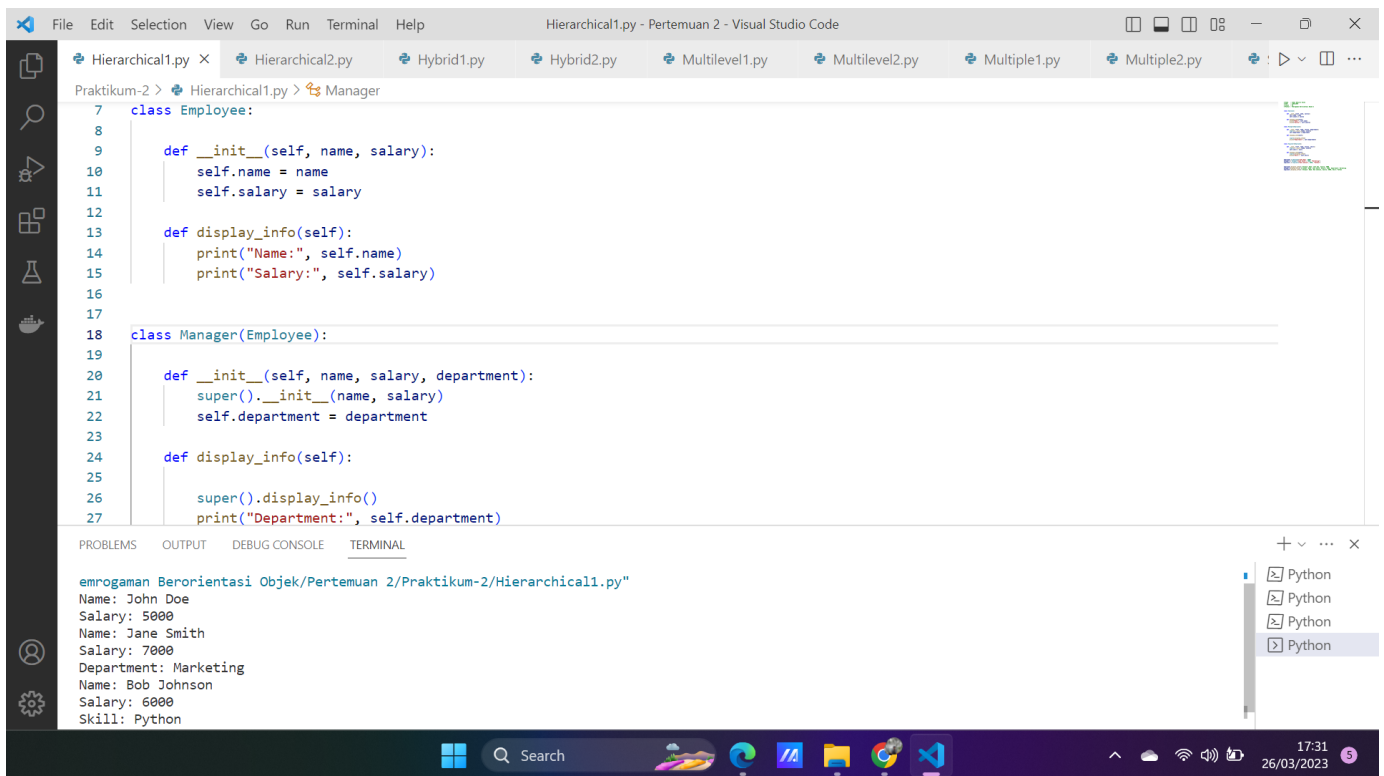
```
    def __init__(self, name, salary, skill):
        super().__init__(name, salary)
        self.skill = skill
```

```
    def display_info(self):
        super().display_info()
        print("Skill:", self.skill)
```

```
employee1 = Employee("John Doe", 5000)
manager1 = Manager("Jane Smith", 7000, "Marketing")
engineer1 = Engineer("Bob Johnson", 6000, "Python")
```

```
employee1.display_info() # Output: Name: John Doe, Salary: 5000
manager1.display_info() # Output: Name: Jane Smith, Salary: 7000, Department:
Marketing
engineer1.display_info() # Output: Name: Bob Johnson, Salary: 6000, Skill:
Python
```

OUTPUT



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a terminal at the bottom. The editor displays a Python file named 'Hierarchical1.py' with the following code:

```
7 class Employee:
8
9     def __init__(self, name, salary):
10         self.name = name
11         self.salary = salary
12
13     def display_info(self):
14         print("Name:", self.name)
15         print("Salary:", self.salary)
16
17
18 class Manager(Employee):
19
20     def __init__(self, name, salary, department):
21         super().__init__(name, salary)
22         self.department = department
23
24     def display_info(self):
25
26         super().display_info()
27         print("Department:", self.department)
```

The terminal at the bottom shows the output of the program, which is the result of running the code in the file 'Praktikum-2/Hierarchical1.py' by user 'emrogaman'. The output is as follows:

```
emrogaman Berorientasi Objek/Permuan 2/Praktikum-2/Hierarchical1.py"
Name: John Doe
Salary: 5000
Name: Jane Smith
Salary: 7000
Department: Marketing
Name: Bob Johnson
Salary: 6000
Skill: Python
```

6. Hierarchical2.py

```
class Vehicle:

    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def info(self):
        print(f"This is a {self.brand} {self.model}")

class Car(Vehicle):

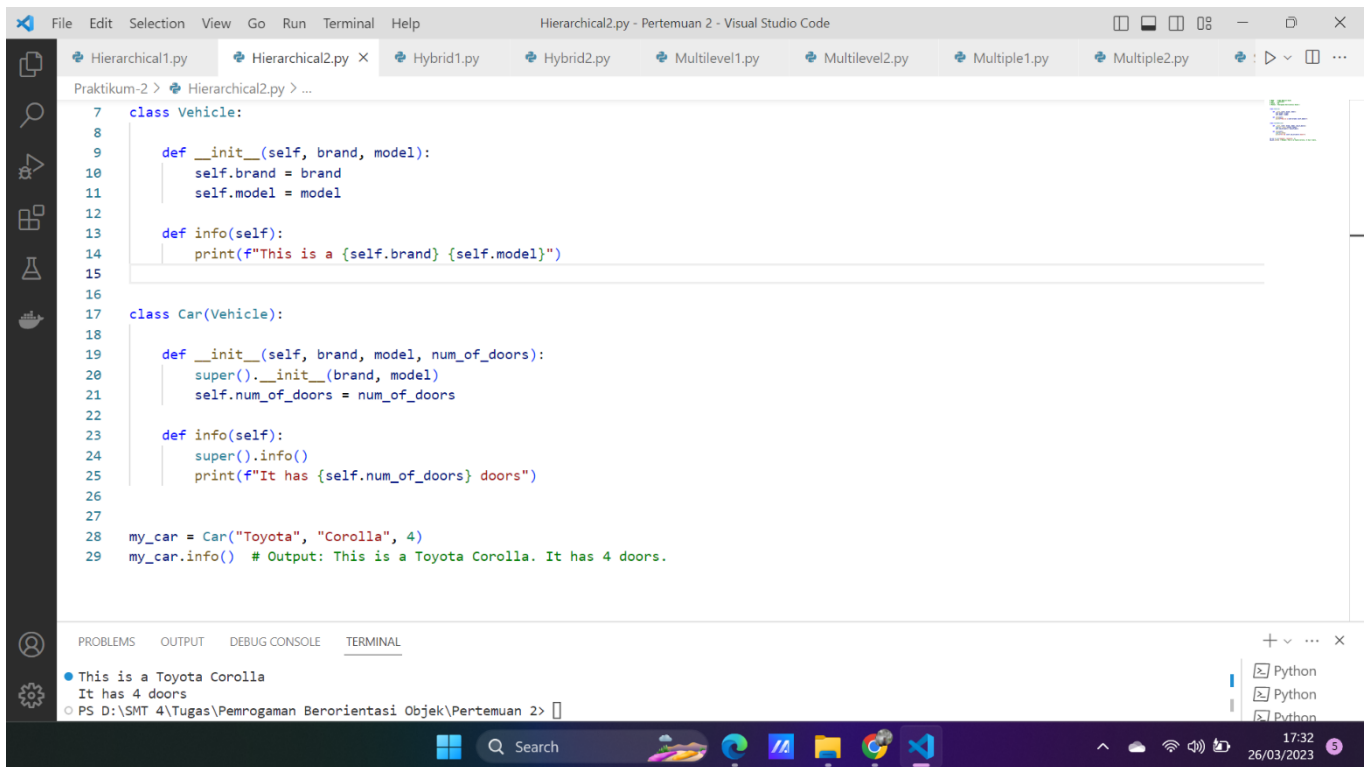
    def __init__(self, brand, model, num_of_doors):
        super().__init__(brand, model)
        self.num_of_doors = num_of_doors

    def info(self):
        super().info()
        print(f"It has {self.num_of_doors} doors")
```



```
my_car = Car("Toyota", "Corolla", 4)
my_car.info() # Output: This is a Toyota Corolla. It has 4 doors.
```

OUTPUT



The screenshot shows a Visual Studio Code window with a file explorer on the left and a code editor in the center. The code editor displays a Python script with two classes: `Vehicle` and `Car`. The `Vehicle` class has an `__init__` method that takes `brand` and `model` as arguments and an `info` method that prints a string. The `Car` class inherits from `Vehicle` and has an `__init__` method that takes `brand`, `model`, and `num_of_doors` as arguments, and an `info` method that prints a string. The script also includes a line of code to create a `Car` object and call its `info` method. The output of the script is shown in the terminal at the bottom of the window.

```
7 class Vehicle:
8
9     def __init__(self, brand, model):
10         self.brand = brand
11         self.model = model
12
13     def info(self):
14         print(f"This is a {self.brand} {self.model}")
15
16
17 class Car(Vehicle):
18
19     def __init__(self, brand, model, num_of_doors):
20         super().__init__(brand, model)
21         self.num_of_doors = num_of_doors
22
23     def info(self):
24         super().info()
25         print(f"It has {self.num_of_doors} doors")
26
27
28 my_car = Car("Toyota", "Corolla", 4)
29 my_car.info() # Output: This is a Toyota Corolla. It has 4 doors.
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

● This is a Toyota Corolla
It has 4 doors
PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2>

7. Multilevel1.py

```
class Vehicle:

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

    def start(self):
        print("Starting", self.name)

class Car(Vehicle):

    def __init__(self, name, color):
        Vehicle.__init__(self, name)
```

```

        self.color = color

    def drive(self):
        print("Driving", self.name, "in", self.color, "color")

class Sedan(Car):

    def __init__(self, name, color, brand):
        Car.__init__(self, name, color)
        self.brand = brand

    def model(self):
        print("This", self.name, "is a", self.brand, "model")

my_sedan = Sedan("Civic", "Black", "Honda")
my_sedan.start()
my_sedan.drive()
my_sedan.model()

```

OUTPUT

The screenshot shows the Visual Studio Code interface with a Python file named 'Multilevel1.py' open. The code defines three classes: Vehicle, Car, and Sedan. The Vehicle class has a start method. The Car class inherits from Vehicle and has a drive method. The Sedan class inherits from Car and has a model method. The code is executed, and the terminal shows the output of the program.

```

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

    def start(self):
        print("Starting", self.name)

class Car(Vehicle):
    def __init__(self, name, color):
        Vehicle.__init__(self, name)
        self.color = color

    def drive(self):
        print("Driving", self.name, "in", self.color, "color")

class Sedan(Car):
    def __init__(self, name, color, brand):
        Car.__init__(self, name, color)
        self.brand = brand

    def model(self):
        print("This", self.name, "is a", self.brand, "model")

my_sedan = Sedan("Civic", "Black", "Honda")
my_sedan.start()
my_sedan.drive()
my_sedan.model()

```

The terminal output is as follows:

```

PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2> & C:/Users/zidan/AppData/Local/Programs/Python/Python311/python.exe "d:/SMT 4/Tugas/Pemrograman Berorientasi Objek/Pertemuan 2/Praktikum-2/Multilevel1.py"
Starting Civic
Driving Civic in Black color
This Civic is a Honda model
PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2>

```

8. Multilevel2.py

```
class Animal:

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

    def eat(self):
        print(self.name, "is eating")

class Dog(Animal):

    def __init__(self, name, breed):
        Animal.__init__(self, name)
        self.breed = breed

    def bark(self):
        print(self.name, "is barking")

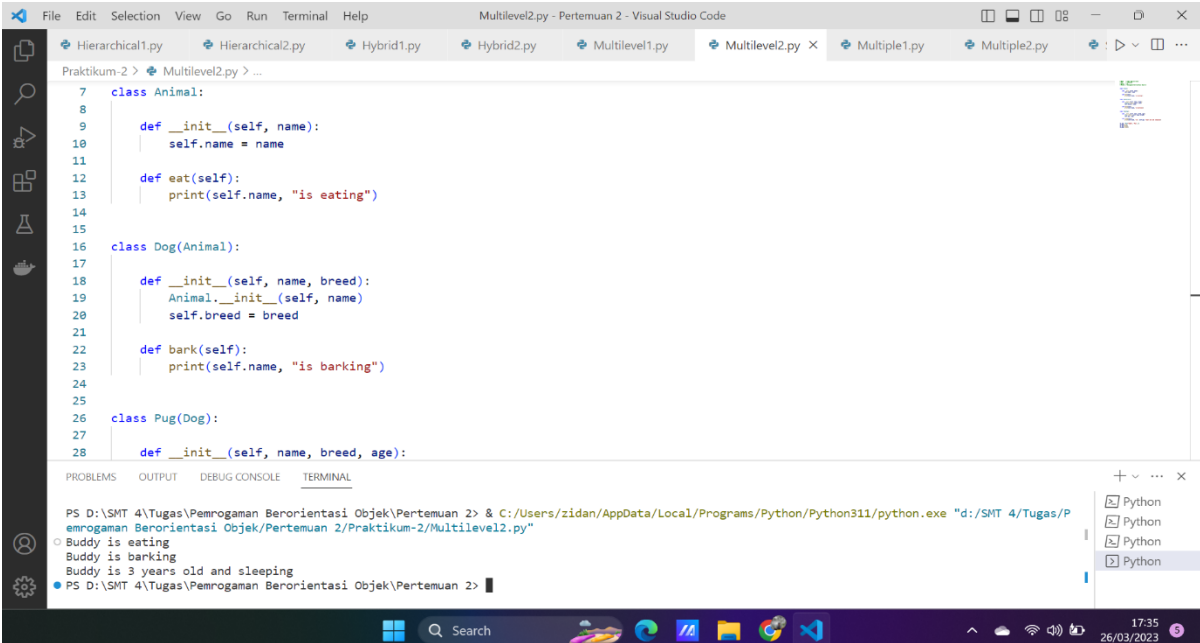
class Pug(Dog):

    def __init__(self, name, breed, age):
        Dog.__init__(self, name, breed)
        self.age = age

    def sleep(self):
        print(self.name, "is", self.age, "years old and sleeping")

my_pug = Pug("Buddy", "Pug", 3)
my_pug.eat()
my_pug.bark()
my_pug.sleep()
```

OUTPUT



The screenshot shows the Visual Studio Code interface with the file 'Multilevel2.py' open. The code in the editor matches the provided text. The bottom panel shows the 'OUTPUT' window with the following text:

```
PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2> & C:/Users/zidan/AppData/Local/Programs/Python/Python311/python.exe "d:/SMT 4/Tugas/Pemrograman Berorientasi Objek/Pertemuan 2/Praktikum-2/Multilevel2.py"
Buddy is eating
Buddy is barking
Buddy is 3 years old and sleeping
PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2>
```

9. Hybrid1.py

```
class Animal:

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

    def speak(self):
        pass

class Mammal(Animal):

    def __init__(self, name):
        super().__init__(name)

    def give_birth(self):
        pass

class Dog(Mammal):

    def __init__(self, name, breed):
        super().__init__(name)
        self.breed = breed

    def speak(self):
        return "Woof"

class Cat(Mammal):

    def __init__(self, name, breed):
        super().__init__(name)
        self.breed = breed

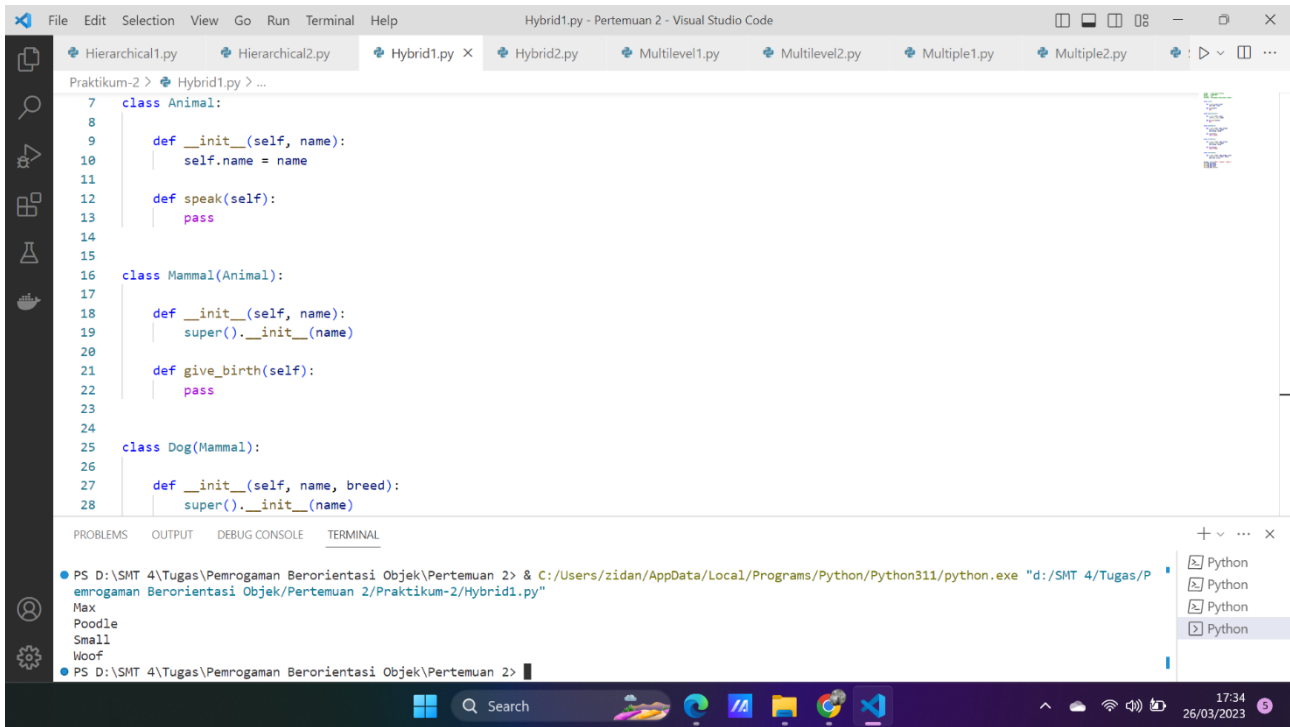
    def speak(self):
        return "Meow"

class Poodle(Dog):

    def __init__(self, name, breed, size):
        super().__init__(name, breed)
        self.size = size
my_dog = Poodle("Max", "Poodle", "Small")
print(my_dog.name)
print(my_dog.breed)
```

```
print(my_dog.size)
print(my_dog.speak())
```

OUTPUT



10. Hybrid2.py

```
class Vehicle:
```

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

    def start(self):
        print("Starting", self.name)
```

```
class Engine:
```

```
    def __init__(self, fuel_type):
        self.fuel_type = fuel_type

    def fuel(self):
        print("Using", self.fuel_type, "as fuel")
```

```

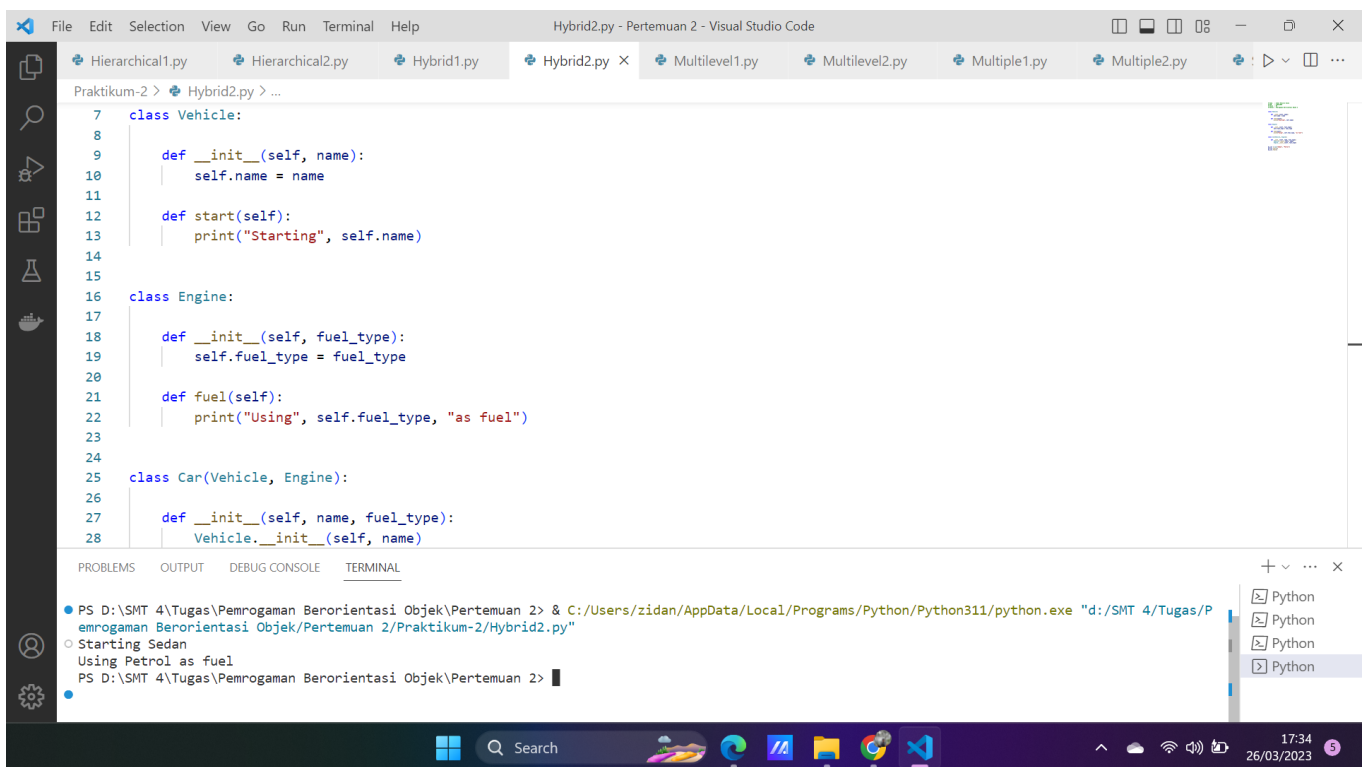
class Car(Vehicle, Engine):

    def __init__(self, name, fuel_type):
        Vehicle.__init__(self, name)
        Engine.__init__(self, fuel_type)

my_car = Car("Sedan", "Petrol")
my_car.start()
my_car.fuel()

```

OUTPUT



The screenshot shows the Visual Studio Code interface with a Python file named 'Hybrid2.py' open. The code defines two classes, `Vehicle` and `Engine`, and a `Car` class that inherits from both. The `Car` class has methods `start()` and `fuel()`. The output shows the execution of the `Car` class, resulting in 'Starting Sedan' and 'Using Petrol as fuel'.

```

7 class Vehicle:
8
9     def __init__(self, name):
10         self.name = name
11
12     def start(self):
13         print("Starting", self.name)
14
15
16 class Engine:
17
18     def __init__(self, fuel_type):
19         self.fuel_type = fuel_type
20
21     def fuel(self):
22         print("Using", self.fuel_type, "as fuel")
23
24
25 class Car(Vehicle, Engine):
26
27     def __init__(self, name, fuel_type):
28         Vehicle.__init__(self, name)

```

Terminal Output:

```

PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2> & C:/Users/zidan/AppData/Local/Programs/Python/Python311/python.exe "d:/SMT 4/Tugas/P
emrograman Berorientasi Objek/Pertemuan 2/Praktikum-2/Hybrid2.py"
Starting Sedan
Using Petrol as fuel
PS D:\SMT 4\Tugas\Pemrograman Berorientasi Objek\Pertemuan 2>

```