LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK LANJUT

2023

Prepared By:

Nama : Zidan Khoirul Rizki

NIM : 210511049

Kelas : R2

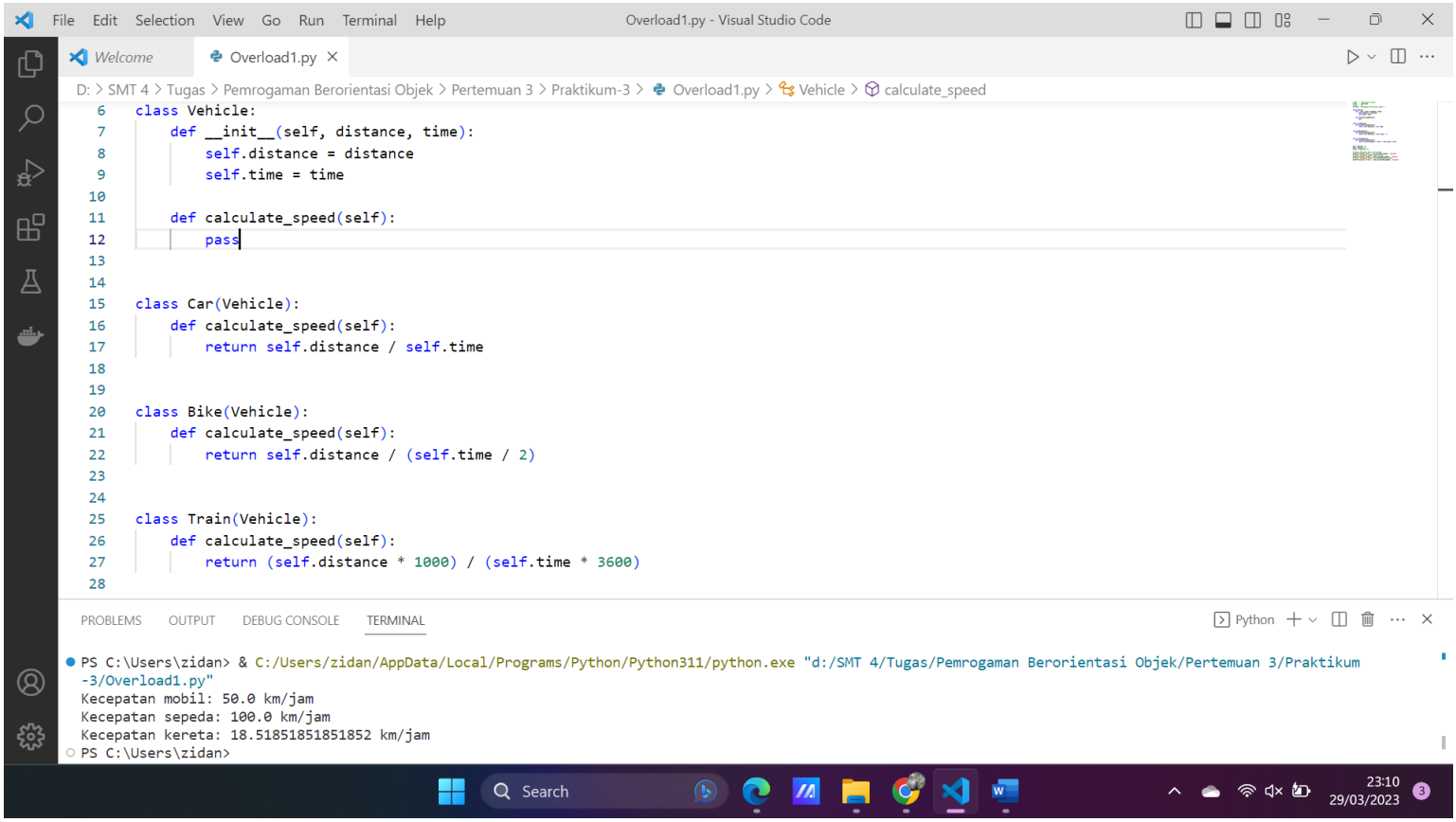
PRAKTIKUM 3

Buatlah masing-masing 2 contoh polymorphism statis (overload) dan polymorphism dinamis (overriding).

1. Overload1.py

```
class Vehicle:
    def __init__(self, distance, time):
        self.distance = distance
        self.time = time

    def calculate_speed(self):
        pass


class Car(Vehicle):
    def calculate_speed(self):
        return self.distance / self.time


class Bike(Vehicle):
    def calculate_speed(self):
        return self.distance / (self.time / 2)


class Train(Vehicle):
    def calculate_speed(self):
        return (self.distance * 1000) / (self.time * 3600)


car = Car(100, 2)
bike = Bike(50, 1)
train = Train(200, 3)

# Output: Kecepatan mobil: 50.0 km/jam
print("Kecepatan mobil:", car.calculate_speed(), "km/jam")
# Output: Kecepatan sepeda: 100.0 km/jam
print("Kecepatan sepeda:", bike.calculate_speed(), "km/jam")
# Output: Kecepatan kereta: 18.51851851851852 km/jam
print("Kecepatan kereta:", train.calculate_speed(), "km/jam")
```

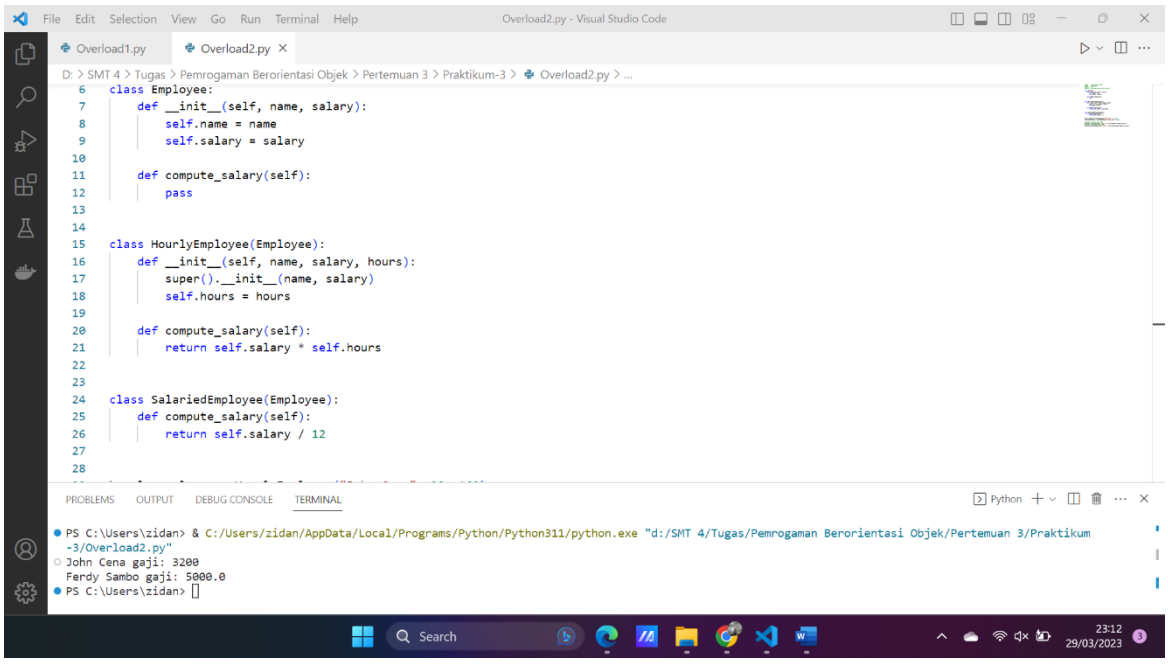
OUTPUT


File Edit Selection View Go Run Terminal Help
Overload1.py - Visual Studio Code
Welcome
Overload1.py
D: > SMT 4 > Tugas > Pemrogaman Berorientasi Objek > Pertemuan 3 > Praktikum-3 > Overload1.py > Vehicle > calculate_speed
6 class Vehicle:
7 def __init__(self, distance, time):
8 self.distance = distance
9 self.time = time
10
11 def calculate_speed(self):
12 pass
13
14
15 class Car(Vehicle):
16 def calculate_speed(self):
17 return self.distance / self.time
18
19
20 class Bike(Vehicle):
21 def calculate_speed(self):
22 return self.distance / (self.time / 2)
23
24
25 class Train(Vehicle):
26 def calculate_speed(self):
27 return (self.distance * 1000) / (self.time * 3600)
28
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
PS C:\Users\zidan> & C:/Users/zidan/AppData/Local/Programs/Python/Python311/python.exe "d:/SMT 4/Tugas/Pemrogaman Berorientasi Objek/Pertemuan 3/Praktikum -3/Overload1.py"
Kecepatan mobil: 50.0 km/jam
Kecepatan sepeda: 100.0 km/jam
Kecepatan kereta: 18.51851851851852 km/jam
PS C:\Users\zidan>
Search
23:10
29/03/2023

2. Overload2.py

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

    def compute_salary(self):
        pass


class HourlyEmployee(Employee):
    def __init__(self, name, salary, hours):
        super().__init__(name, salary)
        self.hours = hours

    def compute_salary(self):
        return self.salary * self.hours


class SalariedEmployee(Employee):
    def compute_salary(self):
        return self.salary / 12


hourly_employee = HourlyEmployee("John Cena", 20, 160)
salaried_employee = SalariedEmployee("Ferdy Sambo", 60000)

# Output: John Cena gaji: 3200
print(hourly_employee.name, "gaji:", hourly_employee.compute_salary())
# Output: Ferdy Sambo gaji: 5000.0
print(salaried_employee.name, "gaji:", salaried_employee.compute_salary())
```

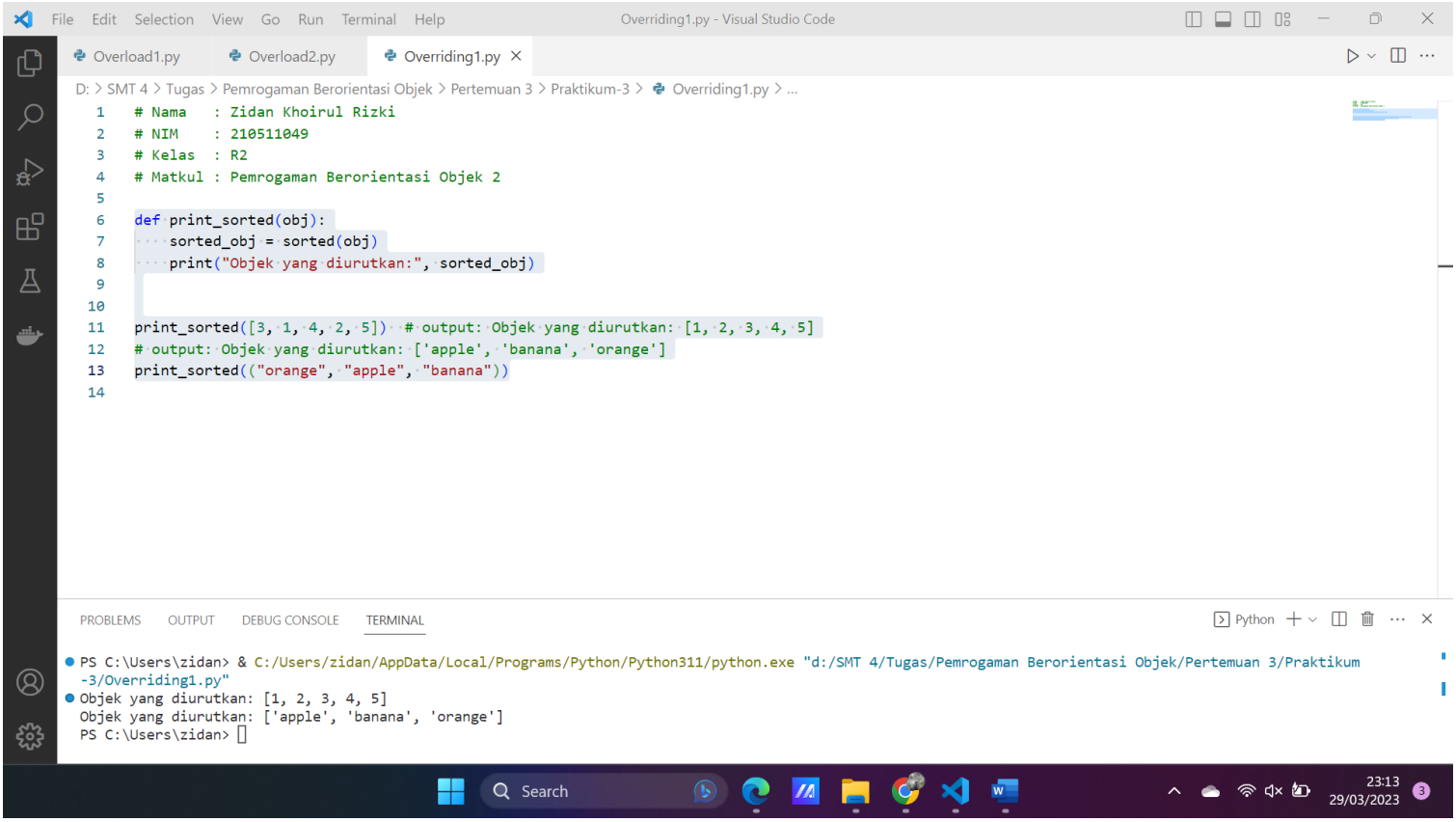
OUTPUT



3. Overriding1.py

```python
def print_sorted(obj):
    sorted_obj = sorted(obj)
    print("Objek yang diurutkan:", sorted_obj)


print_sorted([3, 1, 4, 2, 5])  # output: Objek yang diurutkan: [1, 2, 3, 4, 5]
# output: Objek yang diurutkan: ['apple', 'banana', 'orange']
print_sorted(("orange", "apple", "banana"))
```

OUTPUT



4. Overriding2.py

```
class Runnable:
    def run(self):
        pass


class Car(Runnable):
    def run(self):
        print("Mobil berjalan.")


class Bike(Runnable):
    def run(self):
        print("Sepeda berjalan.")


class Bus(Runnable):
    def run(self):
        print("Bus berjalan.")


def run_all(objects):
    for obj in objects:
        obj.run()


objects = [Car(), Bike(), Bus()]
run_all(objects)
```

OUTPUT