Laporan Praktikum

Pemrograman Berorientasi Objek

Program Studi Teknik Informatika
Institut Teknologi Sumatera
2024

Modul : Inheritence & Polymorphism

Nama : Muhammad Yusuf

NIM : 122140193

Kelas (Kelas Asal) : RA

Instruksi sederhana :

- Disarankan kepada **Praktikan Pemrograman Berorientasi Objek** untuk mengeditnya menggunakan Google Docs agar tidak berantakan dan rapi,
- Silahkan mengganti **Nama Modul** baik yang ada pada **Cover** dan **Header** sesuai dengan materi praktikum,
- Gunakan text styling seperti **Heading 1**, **Normal Text** yang telah terformat / Text Style lainnya yang digunakan untuk menjaga estetika laporan,
- Gunakan **Syntax Highlighter** untuk merapikan kode yang sudah Praktikan buat ke dalam Laporan Praktikum.

Materi Praktikum

Inheritance dan polymorphism adalah dua konsep kunci dalam pemrograman berorientasi objek yang memungkinkan untuk membuat hierarki kelas dan meningkatkan fleksibilitas serta modularitas kode. Inheritance (pewarisan) memungkinkan kelas turunan untuk mewarisi atribut dan metode dari kelas induknya, sehingga memungkinkan untuk membagikan perilaku dan karakteristik umum antar kelas. Hal ini mengurangi duplikasi kode dan memungkinkan penggunaan kembali kode yang ada. Sementara itu, polymorphism (polimorfisme) memungkinkan objek untuk menunjukkan perilaku yang berbeda tergantung pada konteks penggunaannya. Dengan menggunakan konsep ini, kita dapat mengganti atau memperluas perilaku metode dari kelas induk dalam kelas turunan tanpa mengubah kelas induknya, sehingga meningkatkan fleksibilitas dan kemudahan pengembangan sistem. Dengan menggabungkan inheritance dan polymorphism, pemrogram dapat merancang kode yang lebih modular, mudah dipelihara, dan meminimalkan duplikasi serta kompleksitas yang tidak perlu.

Link Source Code Tugas 1

https://onlinegdb.com/OEU-FJwv_

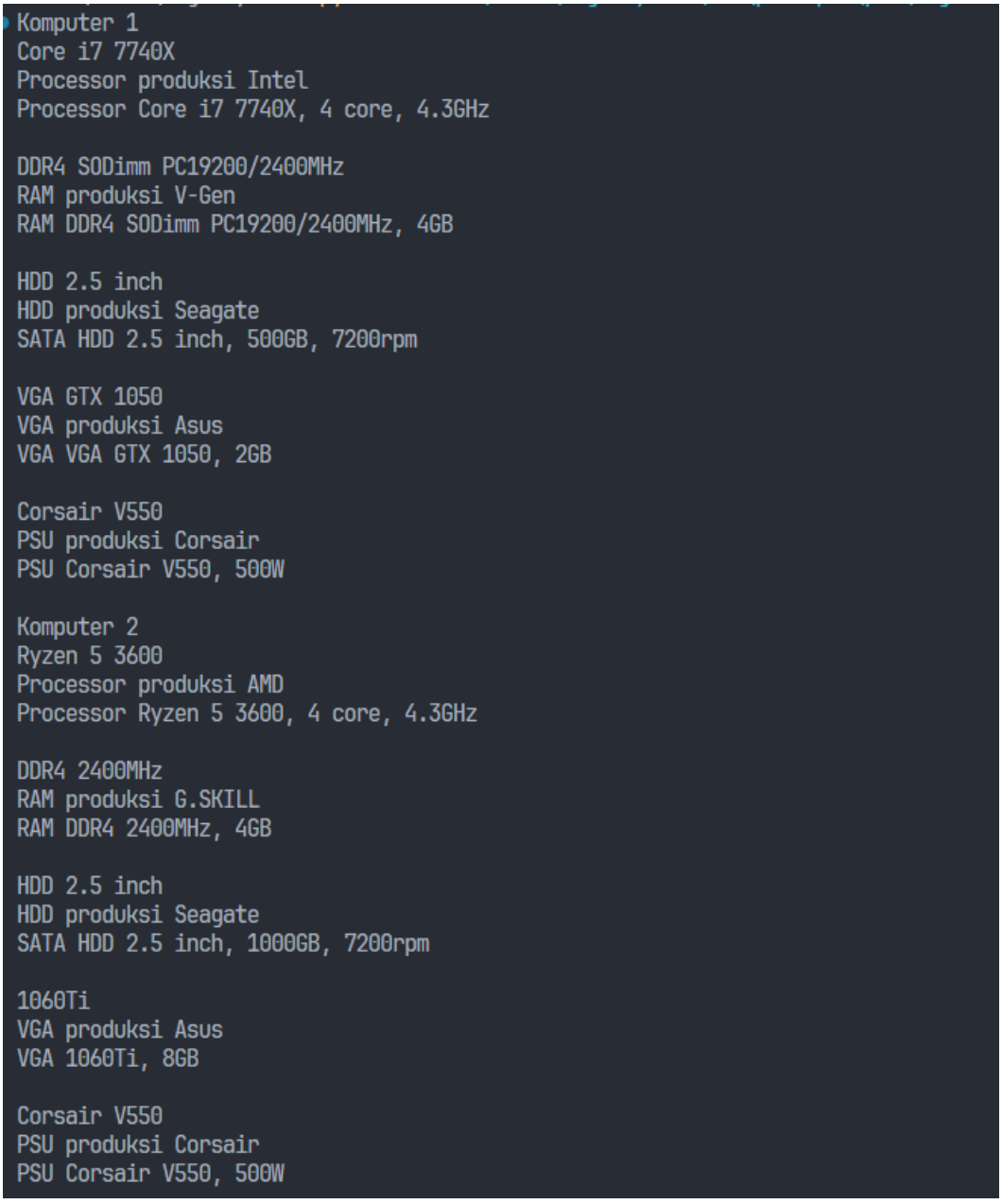
Source Code Tugas 1

```python
# Muhammad Yusuf_122140193
# Tugas 1

# parent class
class Komputer:
    def __init__(self, nama, jenis, harga, merk):
        self.nama = nama
        self.jenis = jenis
        self.harga = harga
        self.merk = merk

    def info(self):
        return f"{self.nama}\n{self.jenis} produksi {self.merk}\n"


# child class start
class Processor(Komputer):
    def __init__(self, merk, nama, harga, jumlah_core, kecepatan_processor):
        super().__init__(nama, 'Processor', harga, merk)
        self.jumlah_core = jumlah_core
```

```
        self.kecepatan_processor = kecepatan_processor

    def info(self):
        return super().info() + f"Processor {self.nama}, {self.jumlah_core}
core, {self.kecepatan_processor}\n"


class RAM(Komputer):
    def __init__(self, merk, nama, harga, capacity):
        super().__init__(nama, 'RAM', harga, merk)
        self.capacity = capacity

    def info(self):
        return super().info() + f"RAM {self.nama}, {self.capacity}\n"


class HDD(Komputer):
    def __init__(self, merk, nama, harga, capacity, rpm):
        super().__init__(nama, 'HDD', harga, merk)
        self.capacity = capacity
        self.rpm = rpm

    def info(self):
        return super().info() + f"SATA {self.nama}, {self.capacity},
{self.rpm}rpm\n"


class VGA(Komputer):
    def __init__(self, merk, nama, harga, capacity):
        super().__init__(nama, 'VGA', harga, merk)
        self.capacity = capacity

    def info(self):
        return super().info() + f"VGA {self.nama}, {self.capacity}\n"


class PSU(Komputer):
    def __init__(self, merk, nama, harga, daya):
        super().__init__(nama, 'PSU', harga, merk)
        self.daya = daya

    def info(self):
        return super().info() + f"PSU {self.nama}, {self.daya}\n"

# child class end
```

```
# main program start

# generate object
p1 = Processor('Intel', 'Core i7 7740X', 4350000, 4, '4.3GHz')
ram1 = RAM('V-Gen', 'DDR4 SODimm PC19200/2400MHz', 328000, '4GB')
hdd1 = HDD('Seagate', 'HDD 2.5 inch', 295000, '500GB', 7200)
vga1 = VGA('Asus', 'VGA GTX 1050', 250000, '2GB')
psu1 = PSU('Corsair', 'Corsair V550', 250000, '500W')

p2 = Processor('AMD', 'Ryzen 5 3600', 250000, 4, '4.3GHz')
ram2 = RAM('G.SKILL', 'DDR4 2400MHz', 328000, '4GB')
hdd2 = HDD('Seagate', 'HDD 2.5 inch', 295000, '1000GB', 7200)
vga2 = VGA('Asus', '1060Ti', 250000, '8GB')
psu2 = PSU('Corsair', 'Corsair V550', 250000, '500W')

rakit = [[p1, ram1, hdd1, vga1, psu1], [p2, ram2, hdd2, vga2, psu2]]

for index, komputer in enumerate(rakit, start=1): # sama seperti for i in
range(1, len(rakit)+1)
    print(f"Komputer {index}")
    for komponen in komputer:
        print(komponen.info())
# main program end
```

Dokumentasi Hasil Running Tugas 1



Gambar 1. Output Tugas 1 Inheritence

Di atas, kita memiliki beberapa kelas yang mewarisi kelas `Komputer`: `Processor`, `RAM`, `HDD`, `VGA`, dan `PSU`. Setiap kelas ini memiliki metode `info()` yang mengembalikan informasi spesifik tentang komponen komputer yang bersangkutan. Pada bagian `MAIN

PROGRAM`, kita membuat beberapa objek dari kelas-kelas tersebut dengan memberikan nilai-nilai yang sesuai. Kemudian kita mengatur objek-objek ini ke dalam list `rakit`, yang merupakan rakitan komputer yang terdiri dari beberapa komponen. Selanjutnya, kita melakukan iterasi melalui setiap rakitan komputer dalam list `rakit` dan menampilkan informasi setiap komponen untuk setiap komputer yang dirakit menggunakan metode `info()` yang sudah didefinisikan sebelumnya dalam setiap kelas. Ini memungkinkan kita untuk melihat detail spesifik dari setiap komponen yang digunakan dalam setiap rakitan komputer.

Link Source Code Tugas 2

https://onlinegdb.com/G6NSYH-JQ

Source Code Tugas 2

```
# Muhammad Yusuf_122140193
# Tugas 2

import random

class Robot:
    def __init__(self, nama, base_health, base_damage):
        self.nama = nama

        self.health = base_health #health yang akan berubah nantinya
        self.base_health = base_health #health yang tidak akan berubah

        self.damage = base_damage #damage yang akan berubah nantinya
        self.base_damage = base_damage #damage yang tidak akan berubah

        self.jumlah_kemenangan = 0 #jumlah kemenangan yang akan bertambah
setiap kali menang
        self.turn = 0

    def lakukan_aksi(self):
        #membatasi jumlah turn sebanyak 1 kali
        if self.turn > 0:
            self.turn -= 1

    def terima_aksi(self, damage_terima):
        self.health -= damage_terima

    def menang(self):
        self.turn = 1
        self.jumlah_kemenangan += 1

class Antares(Robot):
    def __init__(self):
        super().__init__('Antares', 50000, 5000)

    def lakukan_aksi(self):
        if self.jumlah_kemenangan % 3 == 0 and self.turn > 0:
            self.damage *= 1.5
```

```
            print(f"{self.nama} mengaktifkan efek sementara: Damage meningkat
menjadi {self.damage} DMG")
        else:
            self.damage = self.base_damage

        super().lakukan_aksi()

class Alphasetia(Robot):
    def __init__(self):
        super().__init__('Alphasetia', 40000, 6000)

    def lakukan_aksi(self):
        if self.jumlah_kemenangan % 2 == 0 and self.turn > 0:
            self.health += 4000

            print(f"{self.nama} mengaktifkan efek sementara: Health bertambah
menjadi {self.health} HP")
        super().lakukan_aksi()

class Lecalicus(Robot):
    def __init__(self):
        super().__init__('Lecalicus', 45000, 5500)

    def lakukan_aksi(self):
        if self.jumlah_kemenangan % 4 == 0 and self.turn > 0:
            self.health += 7000
            self.damage *= 2

            print(f"{self.nama} mengaktifkan efek sementara: Health bertambah
menjadi {self.health} HP dan Damage meningkat menjadi {self.damage} DMG")
        else:
            self.health = self.base_health
            self.damage = self.base_damage

        super().lakukan_aksi()

# main program start
print("Selamat datang di pertandingan robot Yamako")
pilihan = int(input("Pilih robotmu (1 = Antares, 2 = Alphasetia, 3 =
Lecalicus): "))
if pilihan == 1:
    robotmu = Antares()
elif pilihan == 2:
    robotmu = Alphasetia()
elif pilihan == 3:
```

```
    robotmu = Lecalicus()
else:
    print("Pilihan tidak valid.")
    exit()

#memilih robot lawan secara random selain robot yang dipilih
while True:
    lawan = random.choice([Antares(), Alphasetia(), Lecalicus()])
    if lawan.nama != robotmu.nama:
        break

#info dasar robot
print(f"robotmu ({robotmu.nama} - {robotmu.health} HP - {robotmu.base_damage}
ATK)\nrobot lawan ({lawan.nama} - {lawan.health} HP - {lawan.base_damage}
ATK):")

while robotmu.health > 0 and lawan.health > 0:
    robotmu.lakukan_aksi()
    lawan.lakukan_aksi()

    tangan_robotmu = int(input("\nPilih tangan robotmu (1 = batu, 2 = kertas, 3
= gunting): "))

    if tangan_robotmu not in [1, 2, 3]:
        print("Pilihan tidak valid.")
        continue

    tangan_lawan = random.randint(1, 3)

    if tangan_robotmu == tangan_lawan:
        print("Seri!")
    elif (tangan_robotmu == 1 and tangan_lawan == 3) or (tangan_robotmu == 2
and tangan_lawan == 1) or (tangan_robotmu == 3 and tangan_lawan == 2):
        robotmu.menang()
        lawan.terima_aksi(robotmu.damage)

        print(f"{robotmu.nama} menyerang sebanyak {robotmu.damage} DMG")
        print(f"{lawan.nama} menerima serangan sebanyak {robotmu.damage} DMG")
    else:
        lawan.menang()
        robotmu.terima_aksi(lawan.damage)

        print(f"{lawan.nama} menyerang sebanyak {lawan.damage} DMG")
        print(f"{robotmu.nama} menerima serangan sebanyak {lawan.damage} DMG")
```

```
    print(f"\n{robotmu.nama} ({robotmu.health} HP), {lawan.nama}
({lawan.health} HP)")

if robotmu.health <= 0:
    print(f"{robotmu.nama} kalah!")
else:
    print(f"{robotmu.nama} menang!")
# main program end
```

Dokumentasi Hasil Running Tugas 2

Robot Kalah

```
Antares (8000 HP), Alphasetia (9500.0 HP)

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 1
Seri!

Antares (8000 HP), Alphasetia (9500.0 HP)

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 2
Alphasetia menyerang sebanyak 6000 DMG
Antares menerima serangan sebanyak 6000 DMG

Antares (2000 HP), Alphasetia (9500.0 HP)
Alphasetia mengaktifkan efek sementara: Health bertambah menjadi 13500.0 HP

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 1
Alphasetia menyerang sebanyak 6000 DMG
Antares menerima serangan sebanyak 6000 DMG

Antares (-4000 HP), Alphasetia (13500.0 HP)
Antares kalah!
```

Robot menang

```
Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 1
Alphasetia menyerang sebanyak 6000 DMG
Lecalicus menerima serangan sebanyak 6000 DMG

Lecalicus (39000 HP), Alphasetia (16000 HP)
Alphasetia mengaktifkan efek sementara: Health bertambah menjadi 20000 HP

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 3
Lecalicus menyerang sebanyak 5500 DMG
Alphasetia menerima serangan sebanyak 5500 DMG

Lecalicus (45000 HP), Alphasetia (14500 HP)

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 3
Alphasetia menyerang sebanyak 6000 DMG
Lecalicus menerima serangan sebanyak 6000 DMG

Lecalicus (39000 HP), Alphasetia (14500 HP)

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 1
Lecalicus menyerang sebanyak 5500 DMG
Alphasetia menerima serangan sebanyak 5500 DMG

Lecalicus (45000 HP), Alphasetia (9000 HP)

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 1
Lecalicus menyerang sebanyak 5500 DMG
Alphasetia menerima serangan sebanyak 5500 DMG

Lecalicus (45000 HP), Alphasetia (3500 HP)

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 2
Seri!

Lecalicus (45000 HP), Alphasetia (3500 HP)

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 3
Lecalicus menyerang sebanyak 5500 DMG
Alphasetia menerima serangan sebanyak 5500 DMG

Lecalicus (45000 HP), Alphasetia (-2000 HP)
Lecalicus menang!
```

Gambar 2. Output Tugas 2 Inheritence & Polymorphisme

Pada kode di atas, kita mendefinisikan beberapa kelas yang menerapkan konsep warisan (inheritance) dalam pemrograman berorientasi objek. Kelas `Robot` menjadi kelas dasar yang memiliki atribut dan metode dasar seperti health, damage, dan aksi yang dapat dilakukan. Selanjutnya, kelas-kelas turunan seperti `Antares`, `Alphasetia`, dan `Lecalicus` mewarisi atribut dan metode dari kelas `Robot` tetapi juga menambahkan perilaku khusus yang unik untuk setiap robot. Dalam `main program `, kita melihat implementasi dari kelas-kelas ini dengan membuat objek robot sesuai dengan pilihan pengguna dan mempertemukannya dengan lawan secara acak. Dengan ini, program menciptakan simulasi pertandingan antara robot yang memiliki atribut dan aksi yang berbeda-beda berdasarkan kelas turunannya, menunjukkan penggunaan konsep inheritance dalam pengembangan permainan atau simulasi yang lebih kompleks.