Laporan Praktikum

Pemrograman Berorientasi Objek

Program Studi Teknik Informatika
Institut Teknologi Sumatera
2024

Modul : Inheritence & Polymorphism

Nama : Rayhan Fadel Irwanto

NIM : 122140236

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 prinsip utama dalam paradigma pemrograman berorientasi objek yang memfasilitasi pembentukan hierarki kelas dan meningkatkan fleksibilitas serta modularitas kode. Inheritance, atau pewarisan, memungkinkan kelas turunan untuk menerima atribut dan metode dari kelas induknya, memungkinkan adanya pembagian perilaku dan sifat umum di antara kelas-kelas tersebut. Dengan demikian, duplikasi kode dapat dihindari dan kode yang sudah ada dapat digunakan kembali, mengarah pada pengembangan yang lebih efisien dan terstruktur.

Di sisi lain, polymorphism, atau polimorfisme, memungkinkan objek untuk menunjukkan perilaku yang bervariasi tergantung pada konteks penggunaannya. Konsep ini memungkinkan penggantian atau perluasan perilaku metode dari kelas induk di kelas turunan tanpa memodifikasi kelas induknya secara langsung. Dengan menggabungkan inheritance dan polymorphism, para pengembang dapat merancang sistem dengan kode yang modular, mudah dipelihara, dan mengurangi duplikasi serta kompleksitas yang tidak perlu, menghasilkan solusi yang lebih elegan dan efisien.

Link Source Code Tugas 1

https://onlinegdb.com/Zk4uFJ8Zu

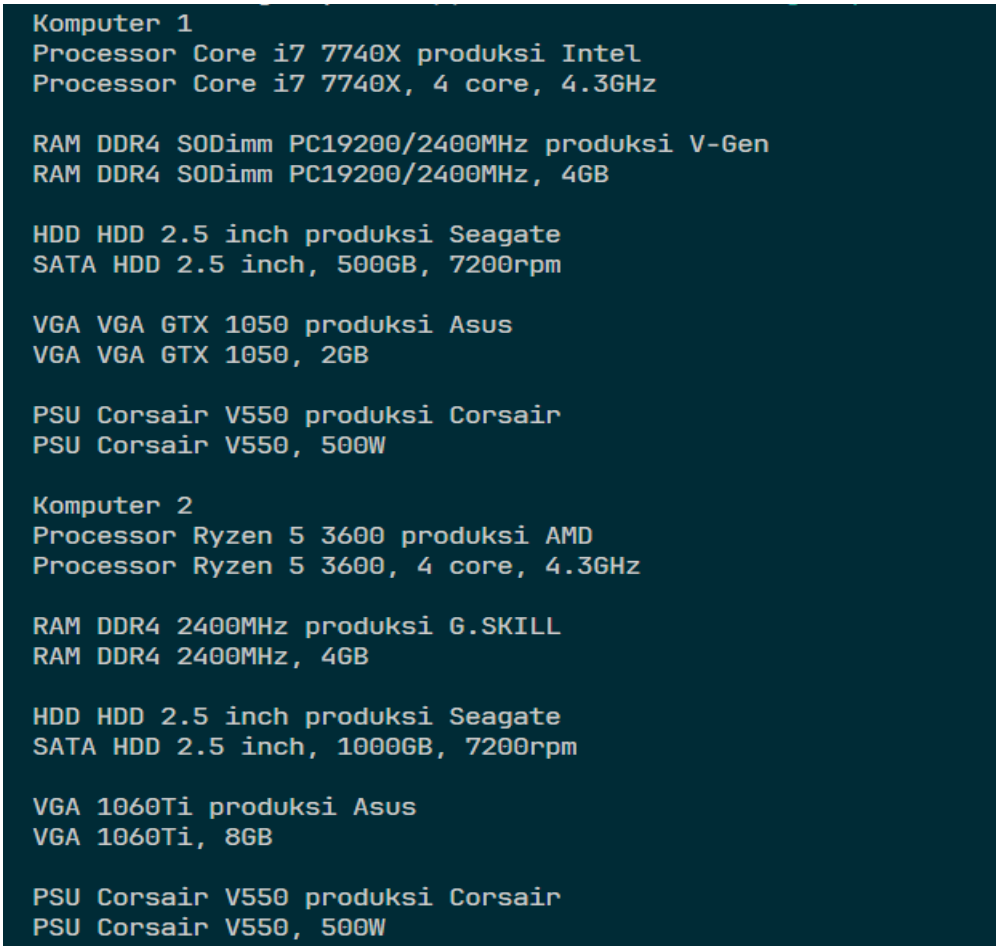
Source Code Tugas 1

```
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.jenis} {self.nama} produksi {self.merk}\n"


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"{self.jenis} {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"{self.jenis} {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"{self.jenis} {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"{self.jenis} {self.nama}, {self.daya}\n"


#main program
p1 = Processor('Intel', 'Core i7 7740X', 4350000, 4, '4.3GHz')
p2 = Processor('AMD', 'Ryzen 5 3600', 250000, 4, '4.3GHz')
ram1 = RAM('V-Gen', 'DDR4 SODimm PC19200/2400MHz', 328000, '4GB')
```

```
ram2 = RAM('G.SKILL', 'DDR4 2400MHz', 328000, '4GB')
hdd1 = HDD('Seagate', 'HDD 2.5 inch', 295000, '500GB', 7200)
hdd2 = HDD('Seagate', 'HDD 2.5 inch', 295000, '1000GB', 7200)
vga1 = VGA('Asus', 'VGA GTX 1050', 250000, '2GB')
vga2 = VGA('Asus', '1060Ti', 250000, '8GB')
psu1 = PSU('Corsair', 'Corsair V550', 250000, '500W')
psu2 = PSU('Corsair', 'Corsair V550', 250000, '500W')

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

for index in range(len(rakit)):
    print(f"Komputer {index + 1}")
    for komponen in rakit[index]:
        print(komponen.info())
```

Dokumentasi Hasil Running Tugas 1



Gambar 1. Output Tugas 1 Inheritence

Dalam kode di atas, terdapat beberapa kelas yang mewarisi kelas `Komputer`, yaitu `Processor`, `RAM`, `HDD`, `VGA`, dan `PSU`. Setiap kelas ini memiliki fungsi `info()` yang memberikan informasi khusus tentang komponen komputer yang terkait. Di bagian `PROGRAM UTAMA`, kita membuat beberapa objek dari kelas-kelas ini dengan memberikan nilai-nilai yang sesuai. Objek-objek ini kemudian disusun ke dalam list `rakit`, yang merupakan kompilasi komponen-komponen komputer yang berbeda. Setelah itu, kita melakukan perulangan melalui setiap rakitan komputer dalam list `rakit` dan menampilkan detail informasi setiap komponen untuk setiap komputer yang telah dirakit dengan menggunakan fungsi `info()` yang telah didefinisikan sebelumnya dalam masing-masing kelas. Hal ini memungkinkan kita untuk memeriksa detail spesifik dari masing-masing komponen yang digunakan dalam setiap rakitan komputer.

Link Source Code Tugas 2

https://onlinegdb.com/fTWh8McSR

Source Code Tugas 2

```
import random

class Robot:
    def __init__(self, nama, base_health, base_damage):
        self.nama = nama
        self.health = base_health
        self.damage = base_damage
        self.base_health = base_health
        self.base_damage = base_damage
        self.jumlah_turn = 0
        self.jumlah_kemenangan = 0

    def lakukan_aksi(self):
        if self.jumlah_turn > 0:
            self.jumlah_turn -= 1 #menghilangkan efek sementara

    def terima_aksi(self, damage_terima):
        self.health -= damage_terima  # Mengurangi health robot sebesar damage
yang diterima

    def menang(self):
        self.jumlah_turn = 1  # Menandakan bahwa robot ini menang pada giliran
ini
        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.jumlah_turn > 0:
            self.damage *= 1.5  # Efek sementara damage meningkat 1.5x lipat
            print(f"{self.nama} mengaktifkan efek sementara: Damage meningkat
menjadi {self.damage} DMG")
        else:
            self.damage = self.base_damage #mengembalikan damage ke nilai awal
setelah efek sementara selesai
```

```
        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.jumlah_turn > 0:
            self.health += 4000  # Efek sementara tambah darah 4000 HP
            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.jumlah_turn > 0:
            self.health += 7000  # Efek sementara tambah darah 7000 HP
            self.damage *= 2  # Efek sementara damage meningkat 2x lipat
            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 #mengembalikan health ke nilai awal
setelah efek sementara selesai
            self.damage = self.base_damage #mengembalikan damage ke nilai awal
setelah efek sementara selesai

        super().lakukan_aksi()

# ======== MAIN PROGRAM ========
print("Selamat datang di pertandingan robot Yamako")  # Menampilkan pesan
selamat datang di pertandingan.
pilihan = int(input("Pilih robotmu (1 = Antares, 2 = Alphasetia, 3 =
Lecalicus): "))  # Meminta input dari pengguna untuk memilih robot.
if pilihan == 1:  # Jika pilihan pengguna adalah 1, membuat objek Antares.
    robotmu = Antares()
elif pilihan == 2:  # Jika pilihan pengguna adalah 2, membuat objek Alphasetia.
    robotmu = Alphasetia()
elif pilihan == 3:  # Jika pilihan pengguna adalah 3, membuat objek Lecalicus.
    robotmu = Lecalicus()
else:  # Jika pilihan pengguna tidak valid, menampilkan pesan kesalahan dan
keluar dari program.
    print("Pilihan tidak valid.")
```

```
    exit()

while True:  # Melakukan perulangan untuk memilih lawan hingga lawan yang
dipilih tidak sama dengan robotmu.
    lawan = random.choice([Antares(), Alphasetia(), Lecalicus()])  # Memilih
lawan secara acak dari list robot.
    if lawan.nama != robotmu.nama:  # Memastikan nama robot lawan tidak sama
dengan nama robot pemain.
        break  # Jika nama lawan sudah berbeda, keluar dari perulangan.

print(f"robotmu ({robotmu.nama} - {robotmu.health} HP - {robotmu.base_damage}
ATK)\nrobot lawan ({lawan.nama} - {lawan.health} HP - {lawan.base_damage}
ATK):")  # Menampilkan informasi robotmu dan lawan sebelum pertandingan.

while robotmu.health > 0 and lawan.health > 0:  # Melakukan perulangan selama
kedua robot masih memiliki health di atas 0.
    robotmu.lakukan_aksi()  # Memanggil fungsi lakukan_aksi untuk robotmu.
    lawan.lakukan_aksi()  # Memanggil fungsi lakukan_aksi untuk lawan.

    tangan_robotmu = int(input("\nPilih tangan robotmu (1 = batu, 2 = kertas, 3
= gunting): "))  # Meminta input pengguna untuk memilih tangan robotmu.
    if tangan_robotmu not in [1, 2, 3]:  # Memeriksa apakah input pengguna
valid.
        print("Pilihan tidak valid.")  # Jika tidak valid, tampilkan pesan
kesalahan.
        continue  # Lanjut ke iterasi berikutnya dari perulangan.

    tangan_lawan = random.randint(1, 3)  # Memilih tangan lawan secara acak.

    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)  # Menyerang lawan
        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)  # Menyerang robotmu
        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)")  # Menampilkan health robotmu dan lawan setelah serangan.

if robotmu.health <= 0:  # Jika health robotmu habis, tampilkan pesan bahwa
robotmu kalah.
    print(f"{robotmu.nama} kalah!")
else:  # Jika health lawan habis, tampilkan pesan bahwa robotmu menang.
    print(f"{robotmu.nama} menang!")
```

Dokumentasi Hasil Running Tugas 2

```
Selamat datang di pertandingan robot Yamako
Pilih robotmu (1 = Antares, 2 = Alphasetia, 3 = Lecalicus): 2
robotmu (Alphasetia - 40000 HP - 6000 ATK)
robot lawan (Antares - 50000 HP - 5000 ATK):

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

Alphasetia (40000 HP), Antares (44000 HP)

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

Alphasetia (40000 HP), Antares (38000 HP)
Alphasetia mengaktifkan efek sementara: Health bertambah menjadi 44000 HP
```

Program terus berulang hingga salah satu darah robot habis

```
Alphasetia (2000 HP), Antares (2000 HP)
Alphasetia mengaktifkan efek sementara: Health bertambah menjadi 6000 HP

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

Alphasetia (6000 HP), Antares (2000 HP)

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 3
Antares menyerang sebanyak 5000 DMG
Alphasetia menerima serangan sebanyak 5000 DMG

Alphasetia (1000 HP), Antares (2000 HP)

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

Alphasetia (1000 HP), Antares (-4000 HP)
Alphasetia menang!
```

Gambar 2. Output Tugas 2 Inheritence & Polymorphisme

Dalam contoh kode yang diberikan, kami menyaksikan penerapan konsep warisan

(inheritance) dalam pemrograman berorientasi objek. Kelas `Robot` dijadikan sebagai kelas induk yang memiliki atribut dan metode dasar seperti health, damage, dan berbagai aksi yang dapat dijalankan. Kemudian, kelas-kelas turunan seperti `Antares`, `Alphasetia`, dan `Lecalicus` mewarisi sifat-sifat tersebut dari kelas `Robot`, sambil menambahkan perilaku khusus yang unik pada masing-masing robot. Melalui `MAIN PROGRAM`, program menunjukkan bagaimana kelas-kelas ini diimplementasikan dengan menciptakan objek robot sesuai pilihan pengguna dan menyusun pertandingan dengan lawan secara acak. Dengan demikian, program ini menciptakan simulasi pertandingan antara robot-robot yang memiliki atribut dan aksi yang beragam berdasarkan jenis kelas turunannya, menggambarkan penggunaan konsep inheritance dalam pembangunan permainan atau simulasi yang lebih kompleks.

Kode tersebut mencerminkan struktur yang memanfaatkan hierarki kelas untuk mengatur dan mengelola berbagai perilaku robot, memperlihatkan kemampuan inheritance dalam mempermudah pengelolaan dan pengembangan perangkat lunak yang melibatkan entitas-entitas yang terkait dan saling berhubungan.