Laporan Praktikum

Pemrograman Berorientasi Objek

Program Studi Teknik Informatika
Institut Teknologi Sumatera
2024

Modul : Inheritence & Polymorphism

Nama : Akhdan Arif Prayoga

NIM : 122140099

Kelas (Kelas Asal) : RB

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

Dalam paradigma pemrograman berorientasi objek, inheritance dan polymorphism merupakan prinsip penting yang mendukung pembentukan hierarki kelas dan meningkatkan fleksibilitas serta modularitas kode. Inheritance, yang juga dikenal sebagai pewarisan, memfasilitasi transfer atribut dan metode dari kelas induk ke kelas turunannya. Hal ini memungkinkan pembagian perilaku dan sifat umum di antara berbagai kelas, menghindari duplikasi kode dan memungkinkan penggunaan kembali kode yang sudah ada. Dengan demikian, proses pengembangan menjadi lebih efisien dan terstruktur.

Sementara itu, polymorphism, atau sering disebut polimorfisme, memberikan kemampuan objek untuk menampilkan perilaku yang berbeda tergantung pada konteks penggunaannya. Prinsip ini memungkinkan penggantian atau perluasan perilaku metode yang diwarisi dari kelas induk ke kelas turunannya tanpa mengubah struktur kelas induknya. Dengan menggabungkan inheritance dan polymorphism, pengembang dapat merancang sistem dengan kode yang modular, mudah untuk dipelihara, dan mengurangi kompleksitas serta duplikasi kode yang tidak perlu. Hasilnya adalah solusi yang lebih elegan dan efisien dalam pengembangan perangkat lunak.

Link Source Code Tugas 1

https://onlinegdb.com/Y2UD0J9mR

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

class Processor(Komputer):
    def __init__(self, nama, jenis, harga, merk, jumlahCore, kecepatanProcessor):
        super().__init__(nama, jenis, harga, merk)
        self.core = jumlahCore
        self.proci = kecepatanProcessor

    def __str__(self):
        return f"Processor {self.jenis} produksi {self.merk}"

class RAM(Komputer):
```

```
    def __init__(self, nama, jenis, harga, merk, capacity):
        super().__init__(nama, jenis, harga, merk)
        self.kapasitasRam = capacity

    def __str__(self):
        return f"RAM {self.jenis} produksi {self.merk}"

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

    def __str__(self):
        return f"SATA {self.jenis} produksi {self.merk}"

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

    def __str__(self):
        return f"VGA {self.jenis} produksi {self.merk}"

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

    def __str__(self):
        return f"PSU {self.jenis} produksi {self.merk}"

p1 = Processor('Intel','Core i7 7740X',4350000,'Intel',4,'4.3GHz')
p2 = Processor('AMD','Ryzen 5 3600',250000,'AMD',4,'4.3GHz')
ram1 = RAM('V-Gen','DDR4 SODimm PC19200/2400MHz',328000,'V-Gen','4GB')
ram2 = RAM('G.SKILL','DDR4 2400MHz',328000,'G.SKILL','4GB')
hdd1 = HDD('Seagate','HDD 2.5 inch',295000,'Seagate','500GB',7200)
hdd2 = HDD('Seagate','HDD 2.5 inch',295000,'Seagate','1000GB',7200)
vga1 = VGA('Asus','VGA GTX 1050',250000,'Asus','2GB')
vga2 = VGA('Asus','1060Ti',250000,'Asus','8GB')
psu1 = PSU('Corsair','Corsair V550',250000,'Corsair','500W')
psu2 = PSU('Corsair','Corsair V550',250000,'Corsair','500W')

rakit = [
    [p1,ram1,hdd1,vga1,psu1],
```

```
    [p2,ram2,hdd2,vga2,psu2]
]

def tampilkan_komponen(komputer_num, komponen_list):
    print(f"Komputer {komputer_num}")
    for komponen in komponen_list:
        print(komponen)
    print("\n")

for i, komponen_list in enumerate(rakit, 1):
    tampilkan_komponen(i, komponen_list)
```

Dokumentasi Hasil Running Tugas 1

```
Komputer 1
Processor Core i7 7740X produksi Intel
RAM DDR4 SODimm PC19200/2400MHz produksi V-Gen
SATA HDD 2.5 inch produksi Seagate
VGA VGA GTX 1050 produksi Asus
PSU Corsair V550 produksi Corsair


Komputer 2
Processor Ryzen 5 3600 produksi AMD
RAM DDR4 2400MHz produksi G.SKILL
SATA HDD 2.5 inch produksi Seagate
VGA 1060Ti produksi Asus
PSU Corsair V550 produksi Corsair
```

Gambar 1. Output Tugas 1 Inheritence

Kode di atas menggambarkan sebuah sistem rakitan komputer yang menggunakan konsep pewarisan (inheritance) dalam pemrograman berorientasi objek. Kelas utama adalah `Komputer` yang memiliki atribut umum seperti nama, jenis, harga, dan merk. Selanjutnya, terdapat kelas-kelas turunan seperti `Processor`, `RAM`, `HDD`, `VGA`, dan `PSU` yang mewarisi atribut-atribut tersebut dari kelas `Komputer` serta menambahkan atribut-atribut spesifik seperti jumlah core dan kecepatan processor untuk `Processor`, kapasitas RAM untuk `RAM`, kapasitas HDD dan rpm untuk `HDD`, kapasitas VGA untuk `VGA`, dan daya untuk `PSU`. Setelah mendefinisikan kelas-kelas tersebut, dilakukan pembuatan objek-objek yang mewakili komponen-komponen komputer seperti processor, RAM, HDD, VGA, dan PSU dengan detail spesifik masing-masing. Selanjutnya, dilakukan pengelompokan objek-objek ini ke dalam sebuah list yang merepresentasikan komputer-komputer yang akan dirakit. Terakhir, terdapat fungsi `tampilkan_komponen` yang digunakan untuk menampilkan komponen-komponen dari setiap komputer yang akan dirakit, dengan menggunakan enumerasi untuk memberikan nomor pada setiap komputer yang dihasilkan.

Link Source Code Tugas 2

https://onlinegdb.com/FuetngZU8t

Source Code Tugas 2

```
import random

class Robot:
    jumlah_turn = 0

    def __init__(self, nama, health, damage):
        self.nama = nama
        self.health = health
        self.damage = damage
        self.jumlah_menang = 0

    def lakukan_aksi(self):
        pass

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

class Antares(Robot):
    def __init__(self):
        super().__init__('Antares', 50000, 5000)
        self.turn_counter = 0

    def lakukan_aksi(self):
        super().lakukan_aksi()

        if self.turn_counter > 0 and self.turn_counter % 3 == 0:
            self.damage *= 1.5
            print(f"{self.nama} secret pasif aktif: Peningkatan damage sementara
menjadi {self.damage}")

        self.turn_counter += 1

class Alphasetia(Robot):
    def __init__(self):
        super().__init__('Alphasetia', 40000, 6000)
        self.turn_counter = 0

    def lakukan_aksi(self):
        super().lakukan_aksi()
```

```
        if self.turn_counter > 0 and self.turn_counter % 2 == 0:
            self.health += 4000
            print(f"{self.nama} secret pasif aktif: Menambah darah menjadi
{self.health}")

        self.turn_counter += 1

class Lecalius(Robot):
    def __init__(self):
        super().__init__('Lecalicus', 45000, 5500)
        self.turn_counter = 0

    def lakukan_aksi(self):
        super().lakukan_aksi()

        if self.turn_counter > 0 and self.turn_counter % 4 == 0:
            self.health += 7000
            self.damage *= 2
            print(f"{self.nama} secret pasif aktif: Menambah darah menjadi
{self.health} dan Peningkatan damage menjadi {self.damage}")

        self.turn_counter += 1


yeah = True
robot = None
while yeah:
    print("Selamat datang di pertandingan robot Yamako")
    pilih = int(input("| 1 -> Antares | 2 -> Alphasetia | 3 -> Lecalicus |
pilihan robotmu: "))
    if pilih ==  1:
        robot = Antares()
        yeah = False
    elif pilih == 2:
        robot = Alphasetia()
        yeah = False
    elif pilih == 3:
        robot = Lecalius()
        yeah = False
    else:
        print("Robot yang dipilih tidak ada, silahkan pilih kembali :D")

while True:
    robotLawan = random.choice([Antares(), Alphasetia(), Lecalius()])
```

```
    if robotLawan.nama != robot.nama:
        break

print(f"Robot mu ({robot.nama} - {robot.health} HP - {robot.damage} DMG)\nRobot
lawan ({robotLawan.nama} - {robotLawan.health} HP - {robotLawan.damage} DMG):")

while robot.health > 0 and robotLawan.health > 0:
    robot.lakukan_aksi()
    robotLawan.lakukan_aksi()

    # Setelah kedua robot melakukan aksi, tambahkan baris ini:
    Robot.jumlah_turn += 1

    tanganRobot = int(input("\n| 1 -> Batu | 2 -> Gunting | 3 -> Kertas |\nPilih
tangan robotmu: "))
    if tanganRobot not in [1, 2, 3]:
        print("Pilihanmu tidak valid, silahkan pilih kembali")
        continue

    tanganLawan = random.randint(1, 3)

    if tanganRobot == tanganLawan:
        print("Seri!!")

    elif (tanganRobot == 1 and tanganLawan == 2) or (tanganRobot == 2 and
tanganLawan == 3) or (tanganRobot == 3 and tanganLawan == 1):
        robotLawan.terima_aksi(robot.damage)
        print(f"{robot.nama} menyerang dengan {robot.damage} DMG")
        print(f"{robotLawan.nama} menerima serangan {robot.damage} DMG")
    else:
        robot.terima_aksi(robotLawan.damage)
        print(f"{robotLawan.nama} menyerang dengan {robotLawan.damage} DMG")
        print(f"{robot.nama} menerima serangan {robotLawan.damage} DMG")
    print(f"\n{robot.nama} ({robot.health} Health), {robotLawan.nama}
({robotLawan.health} Health)")

if robot.health <= 0:
    print(f"{robotLawan.nama} menang")
    print(f"{robot.nama} kalah")
else:
    print(f"{robot.nama} menang")
    print(f"{robotLawan.nama} kalah")
```

Dokumentasi Hasil Running Tugas 2

```
Selamat datang di pertandingan robot Yamako
| 1 -> Antares | 2 -> Alphasetia | 3 -> Lecalicus | pilihan robotmu: 1
Robot mu (Antares - 50000 HP - 5000 DMG)
Robot lawan (Lecalicus - 45000 HP - 5500 DMG):

| 1 -> Batu | 2 -> Gunting | 3 -> Kertas |
Pilih tangan robotmu: 2
Antares menyerang dengan 5000 DMG
Lecalicus menerima serangan 5000 DMG

Antares (50000 Health), Lecalicus (40000 Health)

| 1 -> Batu | 2 -> Gunting | 3 -> Kertas |
Pilih tangan robotmu: 3
Antares menyerang dengan 5000 DMG
Lecalicus menerima serangan 5000 DMG

Antares (50000 Health), Lecalicus (35000 Health)

| 1 -> Batu | 2 -> Gunting | 3 -> Kertas |
Pilih tangan robotmu: 4
Pilihanmu tidak valid, silahkan pilih kembali
Antares secret pasif aktif: Peningkatan damage sementara menjadi 7500.0

| 1 -> Batu | 2 -> Gunting | 3 -> Kertas |
Pilih tangan robotmu: 2
Seri!!

Antares (50000 Health), Lecalicus (35000 Health)
Lecalicus secret pasif aktif: Menambah darah menjadi 42000 dan Peningkatan damage menjadi 11000
```

Program akan terus dijalankan hingga salah 1 robot darahnya habis

```
| 1 -> Batu | 2 -> Gunting | 3 -> Kertas |
Pilih tangan robotmu: 3
Antares menyerang dengan 16875.0 DMG
Lecalicus menerima serangan 16875.0 DMG

Antares (39000 Health), Lecalicus (20875.0 Health)

| 1 -> Batu | 2 -> Gunting | 3 -> Kertas |
Pilih tangan robotmu: 3
Antares menyerang dengan 16875.0 DMG
Lecalicus menerima serangan 16875.0 DMG

Antares (39000 Health), Lecalicus (4000.0 Health)
Antares secret pasif aktif: Peningkatan damage sementara menjadi 25312.5
Lecalicus secret pasif aktif: Menambah darah menjadi 11000.0 dan Peningkatan damage menjadi 44000

| 1 -> Batu | 2 -> Gunting | 3 -> Kertas |
Pilih tangan robotmu: 1
Lecalicus menyerang dengan 44000 DMG
Antares menerima serangan 44000 DMG

Antares (-5000 Health), Lecalicus (11000.0 Health)
Lecalicus menang
Antares kalah
```

Gambar 2. Output Tugas 2 Inheritence & Polymorphisme

Kode di atas mengimplementasikan simulasi pertandingan antara tiga jenis robot: Antares, Alphasetia, dan Lecalius, yang memiliki kemampuan unik yang diaktifkan setelah sejumlah giliran tertentu. Setiap robot diwakili oleh kelas yang mewarisi atribut dan metode dasar dari kelas `Robot`, seperti nama, health, damage, dan jumlah turn. Setiap robot juga memiliki atribut tambahan, yaitu `turn_counter` yang digunakan untuk menghitung giliran pertandingan. Saat pertandingan dimulai, pengguna diminta untuk memilih robot yang akan dikendalikan. Robot lawan dipilih secara acak dari antara tiga jenis robot yang tersedia. Selama pertandingan berlangsung, setiap robot melakukan aksi sesuai dengan aturan tertentu, seperti peningkatan damage atau penambahan health, tergantung pada jumlah giliran yang telah berlalu. Pengguna juga diminta untuk memilih tangan (batu, gunting, atau kertas) untuk bertarung melawan robot lawan. Hasil pertandingan ditampilkan berdasarkan sisa health masing-masing robot.

Perlu dicatat bahwa kode tersebut menunjukkan penggunaan konsep inheritance, polymorphism, serta penggunaan variabel dan metode kelas. Selain itu, penggunaan loop `while` memastikan bahwa pertandingan terus berlanjut hingga salah satu robot kehabisan health. Keseluruhan, kode ini menciptakan simulasi pertandingan robot yang interaktif dan dinamis, di mana setiap robot memiliki strategi unik berdasarkan kondisi pertandingan.