Laporan Praktikum

Pemrograman Berorientasi Objek

Program Studi Teknik Informatika
Institut Teknologi Sumatera
2024

Modul : Inheritence & Polymorphism

Nama : Aulia Putri Sayidina

NIM : 122140060

Kelas (Kelas Asal) : RA (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 berbasis objek, inheritance dan polymorphism adalah prinsip yang sangat penting dalam pembentukan hierarki kelas dan meningkatkan fleksibilitas serta struktur kode. Inheritance, atau yang dikenal sebagai pewarisan, memungkinkan atribut dan metode dari kelas utama diturunkan ke kelas turunannya, memfasilitasi pembagian perilaku dan karakteristik umum antar kelas. Dengan demikian, duplikasi kode dapat dihindari dan kode yang sudah ada dapat digunakan kembali, yang pada gilirannya meningkatkan efisiensi dan keterstrukturan kode.

Di sisi lain, polymorphism memberikan kemampuan objek untuk menunjukkan perilaku yang berbeda tergantung pada konteks penggunaannya. Prinsip ini memungkinkan perubahan atau perluasan perilaku metode yang diwarisi dari kelas utama ke kelas turunannya tanpa mengubah struktur inti kelas utama. Dengan menggabungkan inheritance dan polymorphism, pengembang dapat merancang sistem dengan kode yang modular, mudah dipelihara, dan mengurangi kompleksitas serta duplikasi kode yang tidak diperlukan. Ini membawa dampak positif dalam pengembangan perangkat lunak dengan solusi yang lebih terstruktur dan efisien.

Link Source Code Tugas 1

https://onlinegdb.com/GtNMUM8B1

Source Code Tugas 1

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

    def info(self):
        pass


# child class section
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 f"Processor {self.nama} produksi {self.merk},
{self.jumlah_core} core, {self.kecepatan_processor}"

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

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

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 f"SATA {self.nama}, {self.capacity}, {self.rpm}rpm"

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

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

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

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

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

Dokumentasi Hasil Running Tugas 1

```
Komputer 1
Processor Core i7 7740X produksi Intel, 4 core, 4.3GHz
RAM DDR4 SODimm PC19200/2400MHz, 4GB
SATA HDD 2.5 inch, 500GB, 7200rpm
VGA VGA GTX 1050, 2GB
PSU Corsair V550, 500W
Komputer 2
Processor Ryzen 5 3600 produksi AMD, 4 core, 4.3GHz
RAM DDR4 2400MHz, 4GB
SATA HDD 2.5 inch, 1000GB, 7200rpm
VGA 1060Ti, 8GB
PSU Corsair V550, 500W
```

Gambar 1. Output Tugas 1 Inheritence

Kode di atas merupakan implementasi dari paradigma pemrograman berorientasi objek menggunakan Python. Terdapat parent class `Komputer` yang memiliki atribut nama, jenis, harga, dan merk, serta method `info` yang belum diimplementasikan. Kemudian, terdapat beberapa child class yang mewarisi sifat dan method dari class `Komputer` dengan penyesuaian atribut dan method sesuai jenisnya, seperti Processor, RAM, HDD, VGA, dan PSU. Setiap child class memiliki method `info` yang memberikan informasi spesifik mengenai komponen tersebut. Pada bagian main program, objek-objek dari setiap class dibuat dan dikelompokkan ke dalam list `rakit`, yang kemudian ditampilkan informasinya menggunakan loop for. Dengan demikian, kode tersebut menggambarkan konsep inheritance dan penggunaan method overriding untuk memberikan informasi spesifik dari setiap komponen komputer yang dirakit.

Link Source Code Tugas 2

https://onlinegdb.com/2K9kQT9nm

Source Code Tugas 2

```
import random

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

        self.health = base_health
        self.base_health = base_health

        self.damage = base_damage
        self.base_damage = base_damage

        self.jumlah_kemenangan = 0
        self.turn = 0

    def lakukan_aksi(self):
        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 section
print("Pertandingan robot Yamako")
pilihan = int(input("Pilih robotmu\n1 = Antares\n2 = Alphasetia\n3 =
Lecalicus\nPilihan:"))

if pilihan == 1:
    robotmu = Antares()
elif pilihan == 2:
    robotmu = Alphasetia()
elif pilihan == 3:
    robotmu = Lecalicus()
else:
    print("Pilihan tidak valid.")
```

```
    exit()

while True:
    lawan = random.choice([Antares(), Alphasetia(), Lecalicus()])
    if lawan.nama != robotmu.nama:
        break

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

#proses pertandingan
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)")
```

```
# menentukan pemenang
if robotmu.health <= 0:
    print(f"Robotmu {robotmu.nama} kalah! dan {lawan.nama} menang!")
else:
    print(f"Robotmu {robotmu.nama} menang! dan {lawan.nama} kalah!")
```

Dokumentasi Hasil Running Tugas 2

```
Pertandingan robot Yamako
Pilih robotmu
1 = Antares
2 = Alphasetia
3 = Lecalicus
Pilihan:3
robotmu (Lecalicus - 45000 HP - 5500 ATK)
robot lawan (Alphasetia - 40000 HP - 6000 ATK):

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 (34500 HP)

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

Lecalicus (39000 HP), Alphasetia (34500 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 (34500 HP)
Alphasetia mengaktifkan efek sementara: Health bertambah menjadi 38500 HP
```

Loop program sampai ditentukan pemenang

```
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 (21500 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 (16000 HP)
Lecalicus mengaktifkan efek sementara: Health bertambah menjadi 52000 HP dan Damage meningkat menjadi 11000 DMG

Pilih tangan robotmu (1 = batu, 2 = kertas, 3 = gunting): 1
Lecalicus menyerang sebanyak 11000 DMG
Alphasetia menerima serangan sebanyak 11000 DMG

Lecalicus (52000 HP), Alphasetia (5000 HP)

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

Lecalicus (45000 HP), Alphasetia (-500 HP)
Robotmu Lecalicus menang! dan Alphasetia kalah!
```

Gambar 2. Output Tugas 2 Inheritence & Polymorphisme

Kode di atas adalah implementasi pertandingan antara tiga jenis robot: Antares, Alphasetia, dan Lecalicus. Setiap robot memiliki atribut seperti nama, health, dan damage, serta metode untuk melakukan aksi, menerima aksi, dan menghitung jumlah kemenangan. Konsep inheritance diterapkan di mana ketiga jenis robot merupakan turunan dari kelas Robot yang memiliki atribut dan metode dasar. Pada pertandingan, pemain memilih robotnya dan lawan dipilih secara acak. Selama pertandingan, setiap robot melakukan aksi berdasarkan aturan tertentu, seperti peningkatan damage atau health sementara berdasarkan jumlah kemenangan. Pertandingan dilanjutkan hingga salah satu robot kehilangan seluruh health-nya, dan pemenangnya ditentukan berdasarkan kondisi health yang tersisa. Dengan demikian, kode tersebut mengilustrasikan penggunaan konsep inheritance, pemilihan acak, dan pertandingan dengan mekanisme aksi berdasarkan kondisi tertentu.