

Program Studi Teknik Informatika Institut Teknologi Sumatera

2024

**Pemrograman Berorientasi Objek**

**Laporan Praktikum**

|  |  |
| --- | --- |
| **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**](https://highlight.hohli.com/index.php)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

*# 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

|  |
| --- |
| **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*

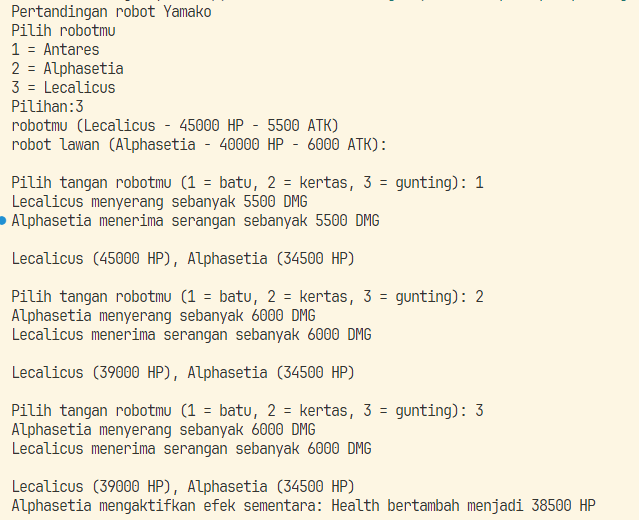
if robotmu.health <= 0:

    print(**f**"Robotmu {robotmu.nama} kalah! dan {lawan.nama} menang!")

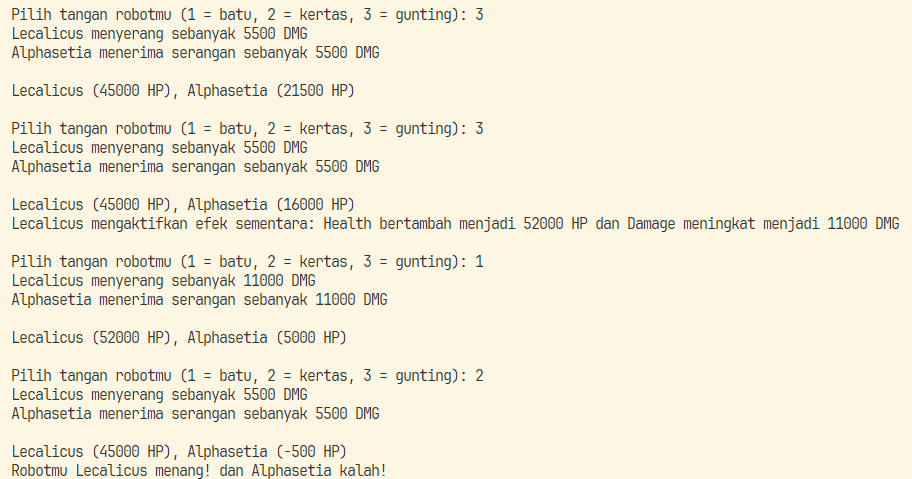
else:

    print(**f**"Robotmu {robotmu.nama} menang! dan {lawan.nama} kalah!")

# Dokumentasi Hasil Running Tugas 2

****

Loop program sampai ditentukan pemenang



**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.