

Program Studi Teknik Informatika Institut Teknologi Sumatera

2024

**Pemrograman Berorientasi Objek**

**Laporan Praktikum**

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

|  |
| --- |
| **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")

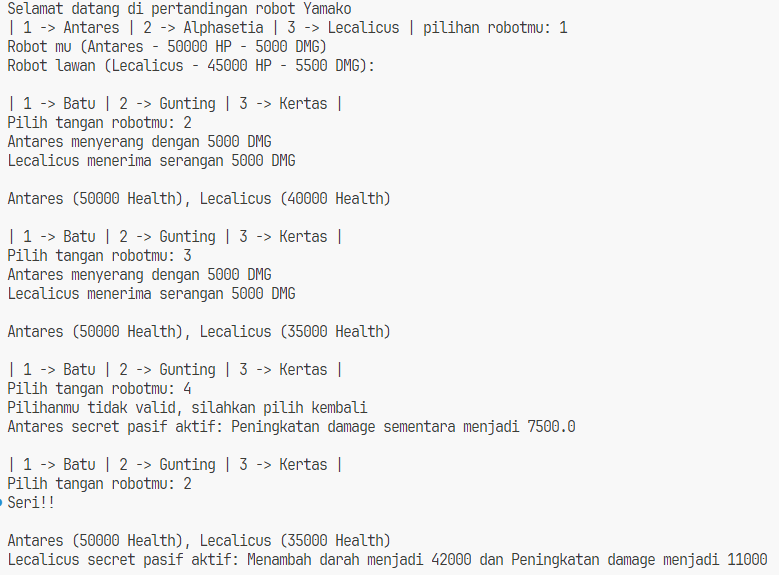
    print(f"{robot.nama} kalah")

else:

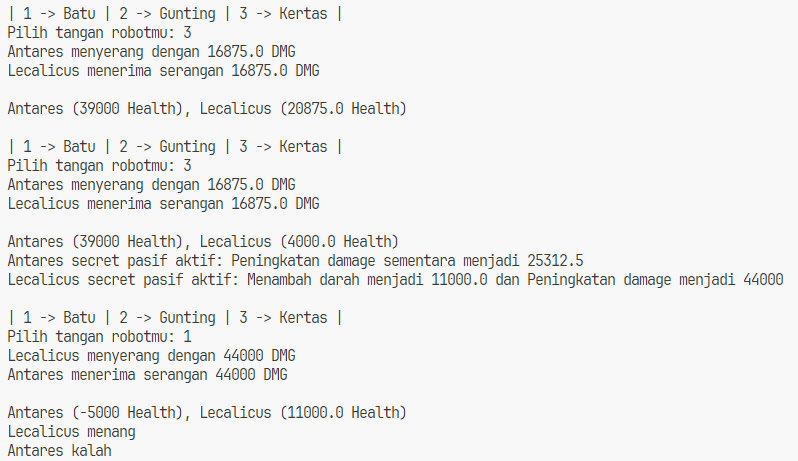
    print(f"{robot.nama} menang")

    print(f"{robotLawan.nama} kalah")

# Dokumentasi Hasil Running Tugas 2

****

Program akan terus dijalankan hingga salah 1 robot darahnya habis



**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.