LATIHAN

1. Latihan overriding, overloading, dan polymorphism

   Buatlah sebuah kelas **Karyawan** yang memiliki atribut berupa **nama_karyawan**, **gaji_karyawan**, dan **jabatan_karyawan** (default sebagai: Staff). Kelas Karyawan tersebut memiliki fungsi utama yaitu **lihat_info()**. Lalu, buatlah kelas turunan dari Karyawan yaitu kelas **Manajer** dan **Teknisi**, dengan ketentuan sebagai berikut:

Karyawan	Apabila fungsi **lihat_info()** dieksekusi maka akan menampilkan "*nama_karyawan* memiliki jabatan berupa *jabatan_karyawan* dengan gaji Rp *gaji_karyawan*"
Manajer	Memiliki **gaji** default **12 juta** dengan **jabatan** "**Manajer**" dan atribut tambahan yaitu **nama_cabang** Apabila fungsi **lihat_info()** dieksekusi maka akan menampilkan "*nama_karyawan* memiliki jabatan berupa *jabatan_karyawan* dengan gaji Rp *gaji_karyawan* pada **nama_cabang**"
Teknisi	Memiliki **gaji** default **7 juta** dengan **jabatan** "**Teknisi**" dan fungsi tambahan yaitu **maintenance()** Fungsi **lihat_info()** mengikuti kelas Karyawan, lalu fungsi **maintenance(nama_cabang)** akan menampilkan "*nama_karyawan* sedang mengadakan maintenance (perbaikan) di cabang **nama_cabang**"

   Contoh kode main program:

```
Malika = Karyawan("Malika", "5 000 000")
Adrian = Manajer("Adrian", "Tanjung Karang")
Syarif = Teknisi("Syarif")

# Duck typing untuk fungsi lihat_info()
for karyawan in Malika, Adrian, Syarif:
    karyawan.lihat_info()
# Hanya ada pada kelas Teknisi
Syarif.maintenance("Tanjung Karang")
```

   Contoh output program:

```
Malika memiliki jabatan berupa Staff dengan gaji Rp 5 000 000
Adrian memiliki jabatan berupa Manajer dengan gaji Rp 12 000 000 pada cabang Tanjung Karang
Syarif memiliki jabatan berupa Teknisi dengan gaji Rp 7 000 000
Syarif sedang mengadakan maintenance (perbaikan) di cabang Tanjung Karang
```

2. Latihan multiple inheritance

   Herman tinggal di sebuah Rumah yang juga berfungsi sebagai Toko tempat usahanya. Buatlah sebuah kelas turunan dari **Rumah** dan **Toko** untuk membantu Herman belajar bahasa Python dengan ketentuan kelas sebagai berikut:

Rumah	Memiliki atribut berupa **biaya_hidup** dan **jumlah_penghuni** (ditentukan saat inisiasi kelas), serta sebuah fungsi **hitung_pengeluaran()** yang akan menghitung pengeluaran bulanan dengan rumus biaya_hidup * jumlah_penghuni
Toko	Memiliki atribut berupa **pendapatan** dan **jumlah_hari** (ditentukan saat inisiasi kelas), serta sebuah fungsi **hitung_pendapatan()** yang akan menghitung pendapatan bulanan dengan rumus pendapatan * jumlah_hari
Ruko	Merupakan kelas turunan dari Rumah dan Toko. Memiliki fungsi tambahan berupa **hitung_laba_bersih()** yang akan menghitung pendapatan bersih bulanan dengan rumus hitung_pendapatan() - hitung_pengeluaran()

   Contoh main program:

```
Herman = Ruko(biaya_hidup = 2000000, jumlah_penghuni = 3, pendapatan = 3000000, jumlah_hari = 22)
print(Herman.hitung_pengeluaran())
print(Herman.hitung_pendapatan())
Herman.hitung_laba_bersih()
```

   Contoh output:

```
6000000
66000000
Ruko bulan ini memiliki laba bersih senilai Rp 60000000
```

TUGAS

1. Single inheritance

   Om Bambang memiliki sebuah toko komputer yang menjual komputer rakitannya sendiri. Bantulah Om Bambang untuk membuat sebuah program sederhana dari sebuah kelas induk **Komputer** dan kelas turunan **Processor, RAM, HDD, VGA,** dan **PSU**.

Komputer	Merupakan sebuah **induk(*parent*)** yang memiliki atribut berupa **nama**, **jenis**, **harga**, dan **merk.**
Processor	Merupakan kelas turunan dari **Komputer** dengan atribut tambahan berupa **jumlah_core** dan **kecepatan_processor.**
RAM	Merupakan kelas turunan dari **Komputer** dengan atribut tambahan berupa **capacity (kapasitas)** dari RAM tersebut.
HDD	Merupakan kelas turunan dari **Komputer** dengan atribut tambahan berupa **capacity (kapasitas)** dan **rpm** dari HDD tersebut.
VGA	Merupakan kelas turunan dari **Komputer** dengan atribut tambahan berupa **capacity (kapasitas)** dari VGA tersebut.
PSU	Merupakan kelas turunan dari **Komputer** dengan atribut tambahan berupa **daya** dari PSU tersebut.

   Tujuan akhir program adalah menampilkan seluruh komponen yang Om Bambang gunakan untuk masing-masing komputer yang dirakitnya. Kalian dapat memanfaatkan sebuah **list** yang dapat menampung komponen komputer tersebut untuk setiap komputer yang telah dirakit oleh Om Bambang.

   Contoh visualisasi list:

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

   Contoh objek:

```
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')
```

   Output yang diharapkan:

```
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
```

2. Inheritance dan polymorphism

   Yamako merupakan sebuah perusahaan Robot yang memiliki 3 tipe robot turunan utama yaitu Antares, Alphasetia, dan Lecalicus. Dimana ketiga robot tersebut mempunyai karakteristik yang berbeda-beda dengan detail seperti tabel di bawah:

Robot	Memiliki variabel instance berupa **nama** (sesuai nama kelas), **health** (darah) dan **damage** (serangan). Memiliki variabel kelas berupa **jumlah_turn** dengan nilai mula-mula yaitu 0. Memiliki fungsi **lakukan_aksi()** yang bila dieksekusi akan menyerang lawan sekaligus mengeksekusi aksi lainnya seperti menambah darah (bila sudah waktunya); serta fungsi **terima_aksi()** untuk menerima serangan (bila menurutmu perlu diimplementasikan)
Antares	Memiliki base health 50000 HP dengan base attack 5000 DMG. Tiap 3x turn (aksi apabila menang), damage dari Antares akan meningkat menjadi 1.5x lipat secara sementara (1 turn)
Alphasetia	Memiliki base health 40000 HP dengan base attack 6000 DMG. Tiap 2x turn (aksi apabila menang), health Alphasetia akan bertambah sebanyak 4000 HP.
Lecalicus	Memiliki base health 45000 HP dengan base attack 5500 DMG. Tiap 4x turn (aksi apabila menang), health Lecalicus akan bertambah sebanyak 7000 HP dan damage nya akan menjadi 2x lipat secara sementara (1 turn)

   Bantulah perusahaan Yamako untuk menentukan robot terkuat yang dimilikinya dengan menyusun **permainan gunting-batu-kertas** serta **pemain berupa para robot**. Apabila suatu robot menang gunting-batu-kertas pada suatu turn, maka fungsi **lakukan_aksi()** pada robot yang menang tersebut akan dieksekusi. Contoh output program (tidak mesti sama persis, aksi lawan bisa saja diotomatisasi):

```
Selamat datang di pertandingan robot Yamako
Pilih robotmu (1 = Antares, 2 = Alphasetia, 3 = Lecalicus): 1
Pilih robot lawan (1 = Antares, 2 = Alphasetia, 3 = Lecalicus): 2
Selanjutnya, pilih 1 untuk batu, 2 untuk kertas, dan 3 untuk gunting
```

```
Turn saat ini: 1
Robotmu (Antares - 50000 HP), robot lawan (Alphasetia - 40000 HP)
Pilih tangan robotmu (Antares): 1
Pilih tangan robot lawan (Alphasetia): 2
# fungsi lakukan_aksi() pada robot lawan dieksekusi
Robot lawan (Alphasetia) menyerang sebanyak 6000 DMG

Turn saat ini: 2
Robotmu (Antares - 44000 HP), robot lawan (Alphasetia - 40000 HP)
Pilih tangan robotmu (Antares): 3
Pilih tangan robot lawan (Alphasetia): 2
# fungsi lakukan_aksi() pada robotmu dieksekusi
Robotmu (Antares) menyerang sebanyak 5000 DMG

Turn saat ini: 3
Robotmu (Antares - 44000 HP), robot lawan (Alphasetia - 35000 HP)
Pilih tangan robotmu (Antares): 2
Pilih tangan robot lawan (Alphasetia): 1
# fungsi lakukan_aksi() pada robotmu dieksekusi
Robotmu (Antares) menyerang sebanyak 7500 DMG

Turn saat ini: 4
Robotmu (Antares - 44000 HP), robot lawan (Alphasetia - 27500 HP)
Pilih tangan robotmu (Antares): 1
Pilih tangan robot lawan (Alphasetia): 2
# fungsi lakukan_aksi() pada robot lawan dieksekusi
Robot lawan (Alphasetia) menambah darah sebanyak 4000 HP
Robot lawan (Alphasetia) menyerang sebanyak 6000 DMG

# Dan seterusnya... permainan berakhir apabila darah salah satu robot mencapai nol
# Output apabila menang disesuaikan
```

3. Eksplorasi mandiri

   Carilah sebuah kasus yang menurutmu menarik dan dapat diselesaikan melalui implementasi konsep **inheritance** (single/multiple; topik bebas). Gunakan juga konsep **magic method**, **type casting**, dan/atau **operator overloading** pada kelas tersebut jika memungkinkan (tidak mesti semuanya, boleh pilih salah satu). Poin penilaian bergantung pada kreativitas (tidak pasaran) dan kompleksitas pada program nantinya. Jangan lupa untuk memberikan komentar atau penjelasan pada program!