**LAPORAN**

**INHERITANCE**

**Logo

Description automatically generatedUNIVERSITAS TEKNOLOGI YOGYAKARTA**

**Oleh :**

**Naufal Firmansyah\_5210411166**

**INHERITANCE**

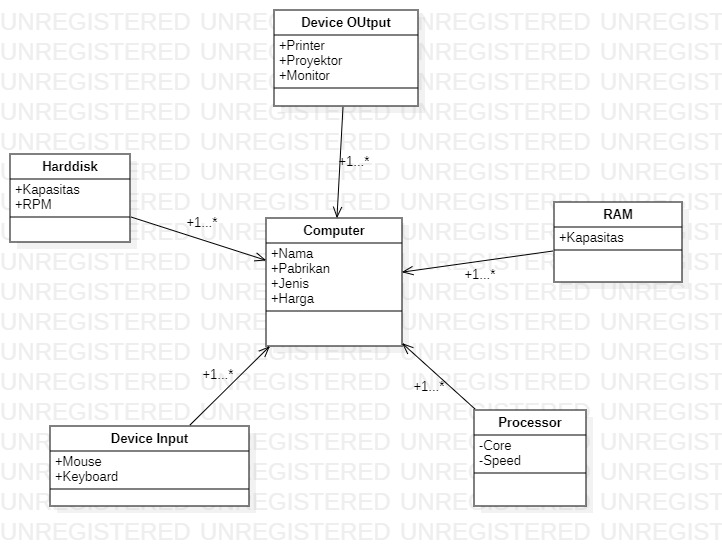
**Pengertian**

**INHERITANCE** (Pewarisan) merupakan salah satu dari tiga konsep dasar OOP. Konsep inheritance ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan. Dengan konsep inheritance, sebuah class dapat mempunyai class turunan.

**Tujuan**

**Inheritance** atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat 'menurunkan' property dan method yang dimilikinya kepada class lain. Konsep **inheritance** digunakan untuk memanfaatkan fitur 'code reuse' untuk menghindari duplikasi kode program.

**CLASS DIAGRAM**



**PROGRAM**

**HIRARKI**

class ComputerPart:

    def \_\_init\_\_(self, pabrikan, nama, jenis, harga):

        self.pabrikan = pabrikan

        self.nama  = nama

        self.jenis = jenis

        self.harga = harga

class Processor(ComputerPart):

    def \_\_init\_\_(self, pabrikan, nama, harga, jumlah\_core, speed):

        super().\_\_init\_\_(pabrikan, nama, 'processor', harga)

        self.jumlah\_core = jumlah\_core

        self.speed = speed

class RandomAccessMemory(ComputerPart):

    def \_\_init\_\_(self, pabrikan, nama, harga, kapasitas):

        super().\_\_init\_\_(pabrikan, nama, 'RAM', harga)

        self.kapasitas = kapasitas

class HardDiskSATA(ComputerPart):

    def \_\_init\_\_(self, pabrikan, nama, harga, kapasitas, rpm):

        super().\_\_init\_\_(pabrikan, nama, 'SATA', harga)

        self.kapasitas = kapasitas

        self.rpm = rpm

p    = Processor('Intel', 'Core i7 7740X',4350000, 4,'4.2GHz')

m    = RandomAccessMemory('V-Gen', 'DDR4 SODimm PC19200/2400MHz', 328000, '4GB')

hdd  = HardDiskSATA('Seagate', 'HDD 2.5 inch', 295000, '500GB', 7200)

parts = [p,m,hdd]

for x in parts:

    print('{} {} produksi {}'.format(x.jenis, x.nama, x.pabrikan))

**Penjelasan :**

Suatu kelas dapat dibuat sebagai kelas yang benar-benar baru atau dapat dibentuk dari kelas yang sudah ada. Proses pembentukan kelas baru dari sebuah kelas yang sudah ada disebut sebagai pewarisan (inheritance). Kelas yang menjadi basis bagi pembentukan kelas lainnya disebut kelas super. Sedangkan kelas yang terbentuk dinamakan kelas sub.

Proses pewarisan inilah yang akan membentuk hirarki kelas. Kelas yang terletak pada puncak hirarki disebut sebagai kelas abstrak. Dimana sifat dari kelas abstrak hanya berisi deskripsi data dan method yang sangat umum. Sehingga tidak mungkin diimplementasikan secara terperinci. Oleh karena itu kelas abstrak ini tidak dapat langsung di instansiasi menjadi objek.

**MULTIPLE**

class prosessor() :

    def \_\_init\_\_(self, prosesor) -> None:

        self.prosesor = prosesor

class RandomAkksesmemori() :

    def \_\_init\_\_(self, detail\_ram) -> None:

        self.ram = detail\_ram

class Storage() :

    def \_\_init\_\_(self, hdd) -> None:

        self.hdd = hdd

class Monitor() :

    def \_\_init\_\_(self, mntr) -> None:

        self.monitor = mntr

*# CHILD*

class Computerpart(prosessor, Storage, RandomAkksesmemori, Monitor) :

    def \_\_init\_\_(self, prosesor, ram, hdd, mntr) -> None:

        prosessor.\_\_init\_\_(self, prosesor)

        Storage.\_\_init\_\_(self, hdd)

        RandomAkksesmemori.\_\_init\_\_(self, ram)

        Monitor.\_\_init\_\_(self, mntr)

this\_pc = Computerpart("Intel® Core™ i9‑11900H", "DOMINATOR® PLATINUM RGB 32 GB (2 x 16 GB) DDR4 DRAM 3200MHz C16 Memory Kit", "HDD Toshiba 1 TB", "Monitor 24 inch Samsung S24F350FHE")

print("\n",

f'''

Saya mempunyai PC/Komputer dengan Spesifikasi Sebagai Berikut :

Processor           : {this\_pc.prosesor}

Random Acces Memory : {this\_pc.ram}

Storage             : {this\_pc.hdd}

Monitor             : {this\_pc.monitor}

 ''')

**Penjelasan :**

Pada konsep multiple inheritance, sebuah kelas turunan akan memiliki semua atribut dan fitur dari **semua kelas yang menjadi induknya**.

**MULTILEVEL**

class computerPart:

    def \_\_init\_\_(self, nama, harga):

        self.nama = nama

        self.harga = harga

class storage(computerPart):

    def \_\_init\_\_(self, nama, harga, jenis):

        super().\_\_init\_\_(nama, harga)

        self.jenis = jenis

class hdd(storage):

    def \_\_init\_\_(self,nama, harga, kapasistas, rpm):

        super().\_\_init\_\_(nama, harga, 'SSD')

        self.kapasitas = kapasistas

        self.rpm = rpm

pc = hdd('Seagate Harddisk Internal Barracuda',750000,'1 TB', '7200 rpm')

print(f'Alhamdulillah Saya Mampu Memiliki Perangkat Computer dengan penyimpanan jenis {pc.jenis} yang berkapasitas {pc.kapasitas} dan memiliki RPM {pc.rpm}')

**Penjelasan :**

penurunan class berjenjang dari satu class ke class lain. Misalkan kita memiliki class A yang diturunkan kepada class B. Kemudian class B ini juga diturunkan kepada class C, sehingga sudah terjadi sebuah multilevel inheritance.

**OUTPUT :**

**HIRAIKI**

**Text

Description automatically generated**

**MULTIPLE**

**Text

Description automatically generated**

**MULTILEVEL**

**Text

Description automatically generated**

**Kesimpulan :**

**Saya menyimpulkan bahwa INHERITANCE** :

1. **Mempermudah programmer dalam mengspesifikan variable tertentu**
2. **Mempermudah programmer dalam mencari sebuah isi dari variable**
3. **Jika Teknik ini digunakan oleh pemula akan sangat membingungkan.**