



# **ALGORITMA DAN STRUKTUR DATA**

**INF1083**

## **LAPORAN PRAKTIKUM 3 : Penjabaran Kelas**

Oleh :

*Akhmad Qasim*

*2211102441237*

Teknik Informatika  
Fakultas Sains & Teknologi  
Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2023

# Laporan Praktikum 3:

# Penjabaran Kelas

## Pokok Bahasan:

- ❖ Pengendali Struktur
- ❖ Pengendali Pengecualian
- ❖ Penjabaran Fungsi
- ❖ Penjabaran Kelas

## Tujuan Pembelajaran:

- ✓ Pengendali Struktur.
- ✓ Pengendali Pengecualian.
- ✓ Penjabaran Fungsi.
- ✓ Penjabaran Kelas.

## Percobaan & Latihan:

### 3.1 Pengendali Struktur

#### a) Berikan tampilan output dan keterangan penjelasan!

Syntax:

```
1 counter = 1 # Memberi nilai 1 pada variabel counter
2 while counter <= 5: # Melakukan perulangan selama nilai counter kurang dari sama dengan 5
3     print("Halo, nim saya : 2211102441237") # Menampilkan teks Halo, nim saya : 2211102441237
4     counter = counter + 1 # Menambahkan nilai 1 pada variabel counter
```

Tampilan Output:

```
Halo, nim saya : 2211102441237
Halo, nim saya : 2211102441237
Halo, nim saya : 2211102441237
Halo, nim saya : 2211102441237
Halo, nim saya : 2211102441237
```

### 3.2 Pengendali Struktur

#### a) Berikan tampilan output dan keterangan penjelasan!

Syntax:

```
1 counter = 0 # Memberi nilai 0 pada variabel counter
2 while counter < 10: # Melakukan perulangan selama nilai counter kurang dari 10
3     print(counter, "Kurang dari 10") # Menampilkan variabel counter Kurang dari 10
4     counter = counter + 1 # Menambahkan nilai 1 pada variabel counter
5 else: # Jika perulangan selesai
6     print(counter, "Sama dengan 10") # Menampilkan teks counter Sama dengan 10
```

Tampilan Output:

```
0 Kurang dari 10
1 Kurang dari 10
2 Kurang dari 10
3 Kurang dari 10
4 Kurang dari 10
5 Kurang dari 10
6 Kurang dari 10
7 Kurang dari 10
8 Kurang dari 10
9 Kurang dari 10
10 Sama dengan 10
```

b) Apa yang terjadi jika iterasi terjadi hingga  $\leq 10$  ? (berikan output & penjelasan)

Output:

```
0 Kurang dari 10
1 Kurang dari 10
2 Kurang dari 10
3 Kurang dari 10
4 Kurang dari 10
5 Kurang dari 10
6 Kurang dari 10
7 Kurang dari 10
8 Kurang dari 10
9 Kurang dari 10
10 Kurang dari 10
11 Sama dengan 10
```

Tidak terjadi error, tetapi perulangan akan di eksekusi sebanyak 12 kali karena perulangan kurang dari sama dengan 10 yang artinya pada saat berhenti di angka 11 masuk ke percabangan else yang memberikan output berupa sebuah pernyataan yang tidak sesuai dengan yang ditampilkan.

### 3.3 Pengendali Struktur

a) Berikan tampilan output dan keterangan penjelasan!

Syntax:

```
1 daftarkata = ["saya", 'anda', "dia"] # Variabel dengan tipe data list
2 daftarhuruf = [] # Variabel dengan tipe data list
3 for setiapkata in daftarkata: # Perulangan untuk setiapkata in daftarkata
4     for setiaphuruf in setiapkata: # Perulangan untuk setiaphuruf in setiapkata
5         daftarhuruf.append(setiaphuruf) # Menambahkan setiaphuruf pada variabel daftarhuruf
6 print(daftarhuruf) # Menampilkan variabel daftarhuruf
```

Tampilan Output:

```
['s', 'a', 'y', 'a', 'a', 'n', 'd', 'a', 'd', 'i', 'a']
```

### 3.4 Pengendali Struktur

a) Buatlah sebuah perintah program input untuk memasukkan nilai variabel n, kemudian jalankan perintah pada gambar diatas!

Syntax:

```
1 import math # Memanggil modul math
2
3 n = int(input("Masukkan angka")) # Meminta input angka
4 if n < 0: # Jika angka yang di input negatif
5     print("Maaf, nilai yang di input adalah negatif") # Maka akan menampilkan pesan error
6 else: # Jika angka yang di input positif
7     print(math.sqrt(n)) # Maka akan menampilkan hasil akar dari angka yang di input
```

b) Berikan 2 nilai input variabel (n = -2) dan (n = 2)!

```
Masukkan angka -2
```

```
Masukkan angka 2
```

c) Berikan tampilan output dan penjelasannya!

```
Masukkan angka -2
Maaf, nilai yang di input adalah negatif
```

Jika angka -2 dimasukkan maka akan menampilkan pesan error karena angka yang di input negatif.

```
Masukkan angka 2
1.4142135623730951
```

Jika angka 2 dimasukkan kedalam variable n, maka akan menampilkan hasil 1.4142135623730951 karena 2 adalah bilangan positif dan akar dari 2 adalah 1.4142135623730951.

### 3.5 Pengendalian Struktur

**a) Berikan tampilan output!**

Syntax:

```
1 SQList = [] # list kosong
2 # generate list dengan for loop dan dimasukkan ke SQList
3 for x in range(1, 11): # x = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
4     SQList.append(x * x) # x * x = 1, 4, 9, 16, 25, 36, 49, 64, 81, 100
5
6 print(SQList) # menampilkan SQList
7
8 STlist = [x * x for x in range(1, 11)] # generate list dengan list comprehension
9 print(STlist) # menampilkan STlist
```

Tampilan Output:

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

**b) Jelaskan perbedaan kedua metode dalam pemberian sebuah nilai di variabel tersebut (SQList dan STlist)!**

Perbedaan antara list comprehension dan for loop adalah:

1. List comprehension lebih singkat.
2. List comprehension lebih cepat.
3. List comprehension lebih mudah dibaca.

**c) Buatlah variabel S, V, M dengan menggunakan metode List Comprehension sehingga membuat output seperti gambar dibawah ini:**

```
11 s = [x * x for x in range(10)]
12 v = [2 ** i for i in range(13)]
13 m = [x * 2 for x in range(5)]
14
15 print(s)
16 print(v)
17 print(m)
```

### 3.6 Pengendali Struktur

**a) Berikan tampilan output dan penjelasannya!**

Syntax:

```
1 # Dilakukan perulangan untuk setiap huruf pada string
2 # Jika huruf tersebut bukan huruf vokal, maka akan ditambahkan ke dalam list
3 # Huruf akan diubah menjadi huruf kapital
4 namaTanpaKonsonan = [ch.upper() for ch in "Akhmad Qasim" if ch not in "aeiou"]
5 print(namaTanpaKonsonan) # Menampilkan list namaTanpaKonsonan
```

Tampilan Output:

```
['A', 'K', 'H', 'M', 'D', 'Q', 'S', 'M']
```

### 3.7 Pengendali Pengecualian

a) Jelaskan mengapa terjadi ValueError diperintah tersebut!

Jika nilai yang di input -23 maka akan muncul pesan error karena nilai yang di input negatif.

b) Berikan pengendali pengecualian menggunakan metode try & except untuk mengatasi masalah diatas!

```
1 import math
2
3 anumber = int(input("Masukan nilai integer"))
4 # cara mengatasi hal tersebut adalah dengan menambahkan fungsi try dan except
5 try:
6     print(math.sqrt(anumber))
7 except:
8     print("Mohon Maaf, nilai yang anda input adalah bilangan negatif")
```

### 3.8 Penjabaran Fungsi

a) Berikan tampilan output dan penjelasannya!

Syntax:

```
1 def kuadrat(n): # Membuat fungsi kuadrat dengan parameter n
2     return n**2 # Mengembalikan nilai (n pangkat 2)
3
4 print(kuadrat(237)) # Akan menampilkan 56169
```

Tampilan Output:

```
56169
```

b) Buatlah fungsi-fungsi yang menyatakan menjabarkan fungsi berikut:

I. Luas lingkaran (n = jari-jari).

```
8 # rumus luas lingkaran
new *
9 def luasLingkaran(r): # Membuat fungsi luasLingkaran dengan parameter r
10     return 3.14 * r ** 2 # Mengembalikan nilai (3.14 * r pangkat 2)
```

II. Luas segitiga siku-siku (a = alas, t = tinggi).

```
13 # rumus luas segitiga siku-siku
new *
14 def luasSegitiga(a, t): # Membuat fungsi luasSegitiga dengan parameter a dan t
15     return 0.5 * a * t # Mengembalikan nilai (0.5 * a * t)
```

### 3.9 Penjabaran Kelas

a) Berikan tampilan output dan penjelasan keterangan output tersebut!

Syntax:

```
1 class Fraksi: # Definisi kelas Fraksi
    new *
2     def __init__(self, top, bottom): # Membuat fungsi __init__ dengan parameter top dan bottom
3         self.num = top # Membuat atribut num dengan nilai top
4         self.den = bottom # Membuat atribut den dengan nilai bottom
5
6
7 myFraksi = Fraksi(3, 5) # Membuat objek myFraksi dengan nilai top = 3 dan bottom = 5
8 print(myFraksi) # Menampilkan objek myFraksi
```

Tampilan Output:

```
<__main__.Fraksi object at 0x000001A0B12B3880>
```

b) Apa peran ‘\_\_init\_\_’ pada baris ketiga pada perintah diatas?

Peran \_\_init\_\_ adalah untuk menginisialisasi objek yang baru dibuat

### 3.10 Penjabaran Kelas

a) Tambahkan fungsi diatas kedalam kelas Fraksi yang terdapat di Percobaan & Latihan 3.9!

Syntax:

```
1 class Fraksi: # Definisi kelas Fraksi
    new *
2     def __init__(self, top, bottom): # Membuat fungsi __init__ dengan parameter top dan bottom
3         self.num = top # Membuat atribut num dengan nilai top
4         self.den = bottom # Membuat atribut den dengan nilai bottom
5
6     new *
7     def __str__(self): # Berfungsi untuk mengambil nilai dari objek
8         return str(self.num) + "/" + str(self.den) # Mengembalikan nilai berupa string
9
10    new *
11    def show(self): # Berfungsi untuk menampilkan nilai dari objek
12        print(self.num, "/", self.den)
13
14 myFraksi = Fraksi(3, 5) # Membuat objek myFraksi dengan nilai top = 3 dan bottom = 5
15 print(myFraksi) # Menampilkan objek myFraksi
```

b) Bagaimana cara menampilkan output menggunakan fungsi diatas agar tampilan output menghasilkan “myFraksi = 3/5” ?

Syntax:

```
def show(self): # Berfungsi untuk menampilkan nilai dari objek
    print("myFraksi = ", self.num, "/", self.den)
```

Tampilan Output:

```
3/5
myFraksi = 3 / 5
```

### 3.11 Penjabaran Kelas

a) Tambahkan fungsi diatas kedalam kelas Fraksi yang terdapat di percobaan & Latihan 3.9!

Syntax:

```
new *
6 def __str__(self): # Berfungsi untuk mengambil nilai dari objek
7   return str(self.num) + "/" + str(self.den) # Mengembalikan nilai berupa string
8
```

b) Jalankan perintah dibawah ini:

```
myf = Fraksi(3,5)
print(myf)
print("Saya makan", myf, "dari kue hamparan tatak")
```

Syntax:

```
1 class Fraksi: # Definisi kelas Fraksi
2     new *
3     def __init__(self, top, bottom): # Membuat fungsi __init__ dengan parameter top dan bottom
4         self.num = top # Membuat atribut num dengan nilai top
5         self.den = bottom # Membuat atribut den dengan nilai bottom
6
7     # Fungsi __str__ akan dipanggil ketika objek di print
8     new *
9     def __str__(self): # Berfungsi untuk mengambil nilai dari objek dan
10        return str(self.num) + "/" + str(self.den) # Mengembalikan nilai berupa string
11
12 myFraksi = Fraksi(3, 5) # Membuat objek myFraksi dengan nilai top = 3 dan bottom = 5
13 print(myFraksi) # Menampilkan objek myFraksi
14
15 myf = Fraksi(3, 5) # Membuat objek myf dengan nilai top = 3 dan bottom = 5
16 print(myf) # Menampilkan objek myf
17 print("Saya makan", myf, "dari kue hamparan tatak") # Menampilkan objek myf
```

Tampilan Output:

```
3/5
3/5
Saya makan 3/5 dari kue hamparan tatak
```



c) Apa peran ‘\_\_str\_\_’ yang terdapat di fungsi tersebut?

```
6      # Fungsi __str__ akan dipanggil ketika objek di print
      new *
7      def __str__(self): # Berfungsi untuk mengambil nilai dari objek dan
8          return str(self.num) + "/" + str(self.den) # Mengembalikan nilai berupa string
```

### 3.12 Penjabaran Kelas

a) Berikan tampilan output dan jelaskan mengapa terjadi Error?

Syntax:

```
1  class Fraksi: # Definisi kelas Fraksi
    new *
2      def __init__(self, top, bottom): # Membuat fungsi __init__ dengan parameter top dan bottom
3          self.num = top # Membuat atribut num dengan nilai top
4          self.den = bottom # Membuat atribut den dengan nilai bottom
5
6
7  myFraksi = Fraksi(3, 5) # Membuat objek myFraksi dengan nilai top = 3 dan bottom = 5
8  print(myFraksi) # Menampilkan objek myFraksi
9
10 f1 = Fraksi(1, 4) # Membuat objek f1 dengan nilai top = 1 dan bottom = 4
11 f2 = Fraksi(1, 2) # Membuat objek f2 dengan nilai top = 1 dan bottom = 2
12 print(f1 + f2) # Terjadi error karena tidak ada fungsi __add__ yang didefinisikan
```

Tampilan Output:

```
Traceback (most recent call last):
  File "C:\Development\Algoritma-dan-Struktur-Data\Laporan 03\Source Code\3.12.py", line 12, in <module>
    print(f1 + f2) # Terjadi error karena tidak ada fungsi __add__ yang didefinisikan
TypeError: unsupported operand type(s) for +: 'Fraksi' and 'Fraksi'
<__main__.Fraksi object at 0x000002682BF438E0>
```

b) Tambahkan fungsi berikut ini kedalam kelas Fraksi yang terdapat di Percobaan & Latihan 3.9!

```
def __add__(self, otherfraction):

    newnum = self.num * otherfraction.den + self.den * otherfraction.num
    newden = self.den * otherfraction.den

    return Fraksi(newnum, newden)
```

Syntax:

```
6      def __add__(self, otherfraction): # Membuat fungsi __add__ dengan parameter otherFraction
7          # Membuat objek baru dengan nilai top dan bottom yang baru
8          newnum = self.num * otherfraction.den + self.den * otherfraction.num
9          newden = self.den * otherfraction.den # Membuat atribut newnum dan newden
10
11      return Fraksi(newnum, newden) # Mengembalikan objek baru
```

c) Jalankan kembali perintah print diatas!

Tampilan Output:

```
<__main__.Fraksi object at 0x000001840A2F3850>  
<__main__.Fraksi object at 0x000001840A2F3B20>
```

**d) Apa peran ‘\_\_add\_\_’ yang terdapat di fungsi tersebut?**

Fungsi dari \_\_add\_\_ pada kode tersebut adalah untuk menambahkan dua objek Fraksi. Metode ini mendefinisikan perilaku operasi penjumlahan (+) pada objek Fraksi. Dalam hal ini, self merepresentasikan objek Fraksi yang saat ini dipanggil, sementara otherfraction merepresentasikan objek Fraksi lain yang digunakan sebagai argumen pada operasi penjumlahan.

Operasi penjumlahan antara dua objek Fraksi melibatkan penjumlahan numerik pembilang dan penyebut dari kedua objek tersebut. Fungsi \_\_add\_\_ melakukan operasi ini dengan menghitung numerik pembilang dan penyebut dari objek Fraksi baru, yang kemudian digunakan untuk membuat objek Fraksi baru yang merupakan hasil penjumlahan kedua objek Fraksi.

Pada akhirnya, fungsi \_\_add\_\_ mengembalikan objek Fraksi baru yang merupakan hasil penjumlahan dari dua objek Fraksi.

### 3.13 Penjabaran Kelas

**a) Jalankan perintah berikut!**

```
f1 = Fraksi(1, 4)  
f2 = Fraksi(1, 2)  
print(f1 + f2)
```

Syntax:

```

1  class Fraksi: # Definisi kelas Fraksi
    new *
2      def __init__(self, top, bottom): # Membuat fungsi __init__ dengan parameter top dan bottom
3          self.num = top # Membuat atribut num dengan nilai top
4          self.den = bottom # Membuat atribut den dengan nilai bottom
5
6      new *
7      def __add__(self, otherfraction): # Membuat fungsi __add__ dengan parameter otherfraction
8          newnum = self.num * otherfraction.den + self.den * otherfraction.num # Membuat atribut newnum
9          newden = self.den * otherfraction.den # Membuat atribut newden
10         common = gcd(newnum, newden) # Membuat atribut common dengan nilai gcd dari newnum dan newden
11         return Fraksi(newnum // common, newden // common) # Mengembalikan objek baru
12
13     new *
14     def gcd(m, n): # Membuat fungsi gcd dengan parameter m dan n
15         while m % n != 0: # Membuat perulangan while dengan kondisi m mod n tidak sama dengan 0
16             oldm = m # Membuat atribut oldm dengan nilai m
17             oldn = n # Membuat atribut oldn dengan nilai n
18
19             m = oldn # Mengubah nilai m dengan nilai oldn
20             n = oldm % oldn # Mengubah nilai n dengan nilai oldm mod oldn
21
22         return n # Mengembalikan nilai n
23
24     myFraksi = Fraksi(3, 5) # Membuat objek myFraksi dengan nilai top = 3 dan bottom = 5
25     print(myFraksi) # Menampilkan objek myFraksi
26
27     f1 = Fraksi(1, 4) # Membuat objek f1 dengan nilai top = 1 dan bottom = 4
28     f2 = Fraksi(1, 2) # Membuat objek f2 dengan nilai top = 1 dan bottom = 2
29     print(f1 + f2) # Menampilkan objek f1 + f2

```

Tampilan Output:

```

<__main__.Fraksi object at 0x000002687C8D3A30>
<__main__.Fraksi object at 0x000002687C8D3B50>

```

b) Jelaskan mengapa perintah diatas mempunyai hasil yang berbeda dengan hasil pada Percobaan & Latihan 3.12c?

Karena terdapat perbedaan antara fungsi `__add__` yang ada didalam kelas Fraksi, dimana pada Percobaan & Latihan 3.13a terdapat tambahan syntax.

### 3.14 Penjabaran Kelas

a) Berikan tampilan output dan penjelasannya?

Syntax:

```

1 class Fraksi: # Definisi kelas Fraksi
2     # Akhmad Qasim
3     def __init__(self, top, bottom): # Membuat fungsi __init__ dengan parameter top dan bottom
4         self.num = top # Membuat atribut num dengan nilai top
5         self.den = bottom # Membuat atribut den dengan nilai bottom
6
7     # Akhmad Qasim
8     def __eq__(self, other): # Membuat fungsi __eq__ dengan parameter other
9         firstnum = self.num * other.den # Membuat atribut firstnum
10        secondnum = other.num * self.den # Membuat atribut secondnum
11
12        return firstnum == secondnum # Mengembalikan nilai firstnum == secondnum
13
14 myFraksi = Fraksi(3, 5) # Membuat objek myFraksi dengan nilai top = 3 dan bottom = 5
15 print(myFraksi) # Menampilkan objek myFraksi
16
17 x = Fraksi(1, 2) # Membuat objek x dengan nilai top = 1 dan bottom = 2
18 y = Fraksi(2, 3) # Membuat objek y dengan nilai top = 2 dan bottom = 3
19 print(x + y) # Terdapat error karena tidak ada fungsi __add__ yang didefinisikan
20 print(x == y) # Menampilkan objek x == y

```

Tampilan Output:

```

<__main__.Fraksi object at 0x000001B3CFA53850>
Traceback (most recent call last):
  File "C:\Development\Algoritma-dan-Struktur-Data\Laporan 03\Source Code\3.14.py", line 18, in <module>
    print(x + y) # Terdapat error karena
TypeError: unsupported operand type(s) for +: 'Fraksi' and 'Fraksi'

```

### Kesimpulan:

Kelas dalam Python merupakan sebuah blueprint atau template yang digunakan untuk membuat objek. Kelas mendefinisikan atribut dan metode yang akan dimiliki oleh objek yang dibuat. Berikut adalah penjelasan mengenai beberapa konsep yang terkait dengan kelas dalam Python:

1. Pengendali struktur: Pengendali struktur seperti if, while, dan list comprehension dapat digunakan dalam kelas untuk mengatur alur program dalam objek yang dibuat. Misalnya, kita bisa menggunakan if dalam metode kelas untuk menentukan apakah suatu objek memenuhi syarat tertentu atau tidak.
2. Pengendali pengecualian: Pengendali pengecualian seperti else, try-except dapat digunakan dalam kelas untuk menangani kesalahan atau kondisi yang tidak diinginkan dalam objek. Misalnya, kita bisa menggunakan try-except dalam metode kelas untuk menangani kesalahan saat memproses data dalam objek.
3. Metode: Kelas memiliki metode yang merupakan fungsi yang terkait dengan objek yang dibuat dari kelas tersebut. Dalam kelas, kita dapat mendefinisikan metode

khusus seperti `__init__` yang digunakan untuk menginisialisasi objek, atau metode kustom lainnya seperti `fraction` yang digunakan untuk melakukan operasi pada objek.

4. Atribut: Kelas juga memiliki atribut, yaitu variabel yang terkait dengan objek yang dibuat dari kelas tersebut. Atribut dapat digunakan untuk menyimpan informasi tentang objek, seperti nama, usia, atau nilai-nilai lainnya.
5. Inheritance: Inheritance adalah konsep di mana kelas baru dibuat dengan cara menurunkan sifat dan perilaku dari kelas yang sudah ada. Kelas baru ini dapat menambahkan atribut atau metode tambahan, atau mengubah atau memperluas metode dan atribut yang ada pada kelas yang diturunkan.

Dengan menggunakan konsep-konsep ini, kita dapat membuat kelas yang kompleks dan fleksibel dalam Python. Kelas memungkinkan kita untuk membuat objek dengan perilaku dan sifat yang sesuai dengan kebutuhan kita, sehingga memudahkan pengembangan program yang lebih besar dan kompleks.

### **Tugas Mandiri:**

```
Enter Pin A input for gate G1-->0
Enter Pin B input for gate G1-->1
Enter Pin A input for gate G2-->1
Enter Pin B input for gate G2-->1
0
```

Tugas mandiri menjelaskan pengantar konsep pemrograman berorientasi objek dengan Python, khususnya menggunakan teknik inheritance (pewarisan) untuk membuat kelas dalam gerbang logika. Kelas `LogicGate` adalah kelas utama yang mendefinisikan fungsionalitas dasar gerbang, dan `BinaryGate` dan `UnaryGate` adalah subkelas yang menambahkan pin input ke gerbang. Kelas `AndGate`, `OrGate`, dan `NotGate` adalah subkelas selanjutnya yang menentukan perilaku spesifik setiap gerbang, menggunakan metode `performGateLogic` untuk menghitung output gerbang. Penggunaan pewarisan memungkinkan programmer menggunakan kembali kode dan ekstensi agar menjadi lebih mudah berdasarkan dari hierarki kelas untuk memasukkan jenis gerbang baru.