NAMA: Silvina Rizqy Nur Auliya

NIM: 121450089

Struktur Data (RC)

Introduction Section

1. Jelaskan definisi object dalam Paradigma Object-Oriented!
2. Jelaskan tiga tujuan dalam Paradigma Object-Oriented!
3. Jelaskan tiga prinsip dalam Paradigma Object-Oriented!
4. Sebutkan contoh design pattern dalam Paradigma Object-Oriented!

Jawab

1. Dalam pemrograman berorientasi objek (OOP), objek adalah hal yang Anda pikirkan pertama kali dalam merancang sebuah program dan juga merupakan unit kode yang pada akhirnya diturunkan dari proses. Di antaranya, setiap objek dibuat menjadi kelas objek generik, dan bahkan lebih banyak kelas generik didefinisikan sehingga objek dapat berbagi model dan menggunakan kembali definisi kelas dalam kodenya. Setiap objek adalah turunan dari kelas atau subkelas tertentu dengan metode atau prosedur dan variabel data kelas itu sendiri. Objek adalah apa yang sebenarnya berjalan di komputer.
2. Modularitas untuk pemecahan masalah yang lebih mudah, Penggunaan kembali kode melalui warisan, Fleksibilitas melalui polimorfisme, Pemecahan masalah yang efektif.
3. Encaptulation, Inheritance (Penurunan Sifat), Abstraction (Abstraksi),Polymorphism.

-Encapsulation pada OOP adalah konsep tentang pengikatan data atau metode berbeda yang disatukan atau “dikapsulkan” menjadi satu unit data.

-Prinsip inheritance pada OOP adalah di mana kita dapat membentuk class baru yang “mewarisi” atau memiliki bagian-bagian dari class yang sudah ada sebelumnya. Konsep ini menggunakan sistem hirarki atau bertingkat.

-prinsip abstract class OOP adalah class-class yang memiliki informasi abstrak dan metode-metode dari sekumpulan data. Abstract Class tidak bisa diubah dan berlaku juga sebagai kerangka dalam penciptaan berbagai subclass (berperan seperti Superclass yang dibahas di konsep Inheritance).

-rinsip polymorphism pada OOP adalah konsep di mana suatu objek berbeda-beda dapat diakses melalui satu interface.

4. Creational Patterns, Structural Patterns, Behavioral Patterns, J2EE Patterns

Class and Object

1. Jelaskan definisi Class!
2. Jelaskan perbedaan Attribute dengan Method!
3. Jelaskan apa itu instansiasi Object!
4. Jelaskan apa itu identifier!

Jawab

1. Class di dalam OOP di gunakan untuk membuat sebuah kerangka kerja dalam membuat program.
2. Attribute merupakan karakteristik dari suatu class. Attribut ini berupa suatu variable yang terletak tepat di dalam class. Attribute dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas, Attribute dapat memiliki hak akses private, public maupun protected. Sementara Method merupakan fungsi yang merupakan kepemilikan dari objek tersebut serta sarana bagi programmer untuk memecah program kompleks menjadi bagian yang kecil-kecil,sehingga nantinya dapat digunakan berulang-ulang.
3. Instance objek adalah kejadian atau fungsi spesifik dari suatu objek.
4. Object Identifier adalah nama jangka panjang yang tidak ambigu untuk semua jenis objek atau entitas, selain itu Identifier (pengenal) merupakan nama yang diberikan untuk mengidentifikasi seperti variabel, fungsi, kelas dan lain sebagainya. Fungsi dari identifier sendiri adalah untuk membedakan antara entitas yang satu dengan entitas lainnya.

Implementing Class

Implementasikan kelas diagram "Kendaraan" berikut ke dalam Python!

Class Kendaraan :

```
- string nomorPlat
- string merk
- string jenis
- string warna
- int tanggalBeli
- int bulanBeli
- int tahunBeli
- double berat
```

- double harga
- showTanggalPembelian()

```
class Kendaraan:
  def __init__ (self, nomorPlat, merk, jenis, warna, tanggalBeli, bulanBeli, tahunBeli, berat
    self.nomorPlat = nomorPlat
    self.merk  = merk
    self.jenis = jenis
    self.warna = warna
    self.tanggalBeli = tanggalBeli
    self.bulanBeli = bulanBeli
    self.tahunBeli = tahunBeli
    self.berat = berat
    self.harga = harga

  def showTanggalPembelian(self):
    print(f"{self.tanggalBeli}/{self.bulanBeli}/{self.tahunBeli}")


KendaraanMotor = Kendaraan("SR 191102 NA ", "Suzuki", "Matic", "Putih", 19, 11, 2002, 450, 45
KendaraanMotor.showTanggalPembelian()
```

```
19/11/2002
```

Main OOP Concept

1. Jelaskan konsep encapsulation dalam Paradigma Object-Oriented!
2. Jelaskan konsep inheritance dalam Paradigma Object-Oriented!
3. Jelaskan perbedaan public method dan private method dalam encapsulation!
4. Jelaskan perbedaan superclass dan subclass dalam inheritance!

Jawaban :

1. Dalam bahasa pemrograman komputer berorientasi objek (OOP), Encapsulation pada OOP adalah konsep tentang pengikatan data atau metode berbeda yang disatukan atau “dikapsulkan” menjadi satu unit data, atau konsep encapsulation yang digunakan untuk membatasi akses method agar tidak bisa diakses dari luar object. Encapsulation dapat memudahkan dalam pembacaan kode karena informasi yang disajikan tidak perlu dibaca secara rinci dan sudah merupakan satu kesatuan.
2. Inheritance dalam konsep OOP adalah kemampuan untuk membentuk class baru yang memiliki fungsi turunan atau mirip dengan fungsi yang ada sebelumnya yang dimana kelas

turunan akan selalu memiliki sifat dan perilaku yang sama dengan kelas induknya. Konsep ini menggunakan sistem hierarki atau bertingkat. yang dimana semakin jauh turunan atau subclass-nya, maka semakin sedikit kemiripan fungsinya.

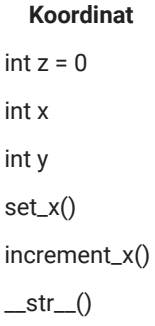
3. Public dapat diakses dari mana saja(luar class/fungsi). Private hanya dapat diakses dari objek itu sendiri atau berfungsi untuk memberikan akses properti yang hanya dapat diakses dari dalam class tersebut
4. Superclass adalah kelas yang ada dari mana kelas baru diturunkan. Superclass digunakan untuk menunjukkan hirarki class yang berarti class dasar dari subclass/class anak.

Subclass adalah kelas baru yang mewarisi properti dan metode Superclass. Suatu subclass tidak lain hanya memperluas (extend) parent class-nya, dan juga class anak atau turunan secara hirarki dari superclass.

Programming Exercise

- Buatlah class diagram dari code python berikut ini:

```
class Koordinat:
    def __init__(self,x,y):
        self.z = 0
        self.x = x
        self.y = y
    def set_x(self,x):
        self.x = x
    def increment_x(self):
        self.x+=1
    def __str__(self):
        return f"x : {self.x} , y:{self.y} , z:{self.z}"
```



- Implementasikan class diagram berikut menjadi program python!

Koordinat

int x

int y

int z

setx()

sety()

setz()

```
class Koordinat:
  def __init__(self, x, y, z):
    self.x = x
    self.y = y
    self.z = z
  def setx(self, x):
    self.x = x
  def sety(self, y):
    self.y = y
  def setz(self, z):
    self.z = z
```

Vector

Koordinat Koordinat

setKoordinat()

getKoordinat()

norm()

```
class Vector:
  def __init__(self, koordinat):
    self.koordinat = koordinat
  def setKoordinat(self, koor):
    self.koordinat = koor
  def getKoordinat(self, y):
    return self.koordinat
  def norm(self):
    norm = 0
    for n in range(len(self.koordinat)):
      norm = norm + self.koordinat[n]**2
    return norm**0.5
```

- fungsi norm() adalah fungsi untuk menghitung norm dari sebuah vector yaitu norm = $\sqrt{x^2 + y^2 + z^2}$. Buatlah fungsi untuk menghitung norm dari vector berdasarkan class yang telah anda buat!

```
class Vector:
  def __init__(self, koor1):
```

```
        self.koor1 = koor1
    def setKoordinat(self, koor):
        self.koor1 = koor
    def getKoordinat(self, y):
        return self.koor1
    def norm(self):
        norm = 0
        for n in range(len(self.koor1)):
          norm = norm + self.koor1[n]**2
        return norm**0.5

koordinat = Vector([7, 24, 25])
koordinat.norm()
```

```
35.35533905932738
```

- Euclidian Distance adalah jarak antara 2 vector dan didefinisikan sebagai

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (x_1 - x_2)^2}$$

buatlah program untuk menghitung euclidian distance antara 2 vector berdasarkan class yang telah anda buat!

```
class Vector:
  def __init__(self, vec1, vec2):
    self.vec1 = vec1
    self.vec2 = vec2
  def euclidian_distance(self):
    distance = 0
    for n in range(len(self.vec1)):
      distance = distance + (self.vec1[n] - self.vec2[n])**2
    return distance**0.5

jarak = Vector([3, 4, 5], [7, 24, 25])
jarak.euclidian_distance()
```

```
28.5657137141714
```

- Perhatikan tabel berikut

Vector	x	y	z	kelas
A	0	1	1	0
B	2	2	2	1
C	1.5	1.2	1.5	1
D	10	9	8	0
E	5	5	5	0
F	8	8	8	0

Vector	x	y	z	kelas
G	6	6	1	0
H	3	3	3	?

Salah satu metode dalam data science untuk melakukan klasifikasi adalah dengan menggunakan k-nearest neighbor, dimana cara menentukan label kelas adalah dengan cara melihat tetangga dengan euclidian distance terdekat. Tentukan kelas dari vector H dengan cara:

- Tentukan 3 tetangga paling dekat dari H
- dari 3 tetangga tersebut, pilih kelas yang paling banyak muncul
- Jadikan kelas yang paling banyak muncul tersebut sebagai kelas dari vector H

Implementasikan algoritma tersebut menggunakan program python dengan paradigma object-oriented programming yang telah anda pelajari!

```
from math import sqrt
import statistics
class Vektor:
  def __init__(self, vektor, x, y, z, namvek):
    self.vektor = vektor
    self.x = x
    self.y = y
    self.z = z
    self.namvek = namvek
    self.klse = 0

  def jarak(self):
    vekt = []
    jarak = []

    for i in self.vektor:
        komponen_vektor = list(self.vektor[i].values())
        def dekat(komponen_vektor):
            distance = 0
            for i in komponen_vektor:
                distance = sqrt((self.x-komponen_vektor[1])**2 + (self.y-komponen_vektor[2])*
            return distance
        vekt = vekt + [self.vektor[i]]
        jarak = jarak + [round(dekat(komponen_vektor), 2)]
        print(f"Jarak vektor {self.vektor[i]['vektor']} dengan vektor H sebesar {round(dekat(
    val_sorted = sorted(jarak)
    tetangga = {}
    for i in range(len(jarak)):
        tetangga[jarak[i]] = vekt[i]
    print("dengan urutan:")
    for j in range(len(val_sorted)):
        print(f"Jarak vektor {tetangga[val_sorted[j]]['vektor']} dengan vektor H sebesar {val
    print("3 Tetangga terdekat:")
```

```
        kls = []
        for k in range(3):
            print(f"Vektor {tetangga[val_sorted[k]]['vektor']} dengan kelas {tetangga[val_sorted[
            kls = kls + [tetangga[val_sorted[k]]['kelas']]
        self.klse = statistics.mode(kls)
        print("Kelas yang paling banyak muncul adalah = ", self.klse)
        print("Kelas dari vektor H = ", self.klse)

      def kelas(self):
        return {'vektor': self.namvek, 'x' : self.x, 'y' : self.y, 'z': self.z, 'kelas': self.kls
```

```
tbl = {
    "vektor1" : {'vektor': 'A', 'x' : 0, 'y' : 1, 'z': 1, 'kelas': 0 },
    "vektor2" : {'vektor': 'B', 'x' : 2, 'y' : 2, 'z': 2, 'kelas': 1},
    "vektor3" : {'vektor': 'C', 'x' : 1.5, 'y' : 1.2, 'z': 1, 'kelas': 1},
    "vektor4" : {'vektor': 'D', 'x' : 10, 'y' : 9, 'z': 0, 'kelas': 0},
    "vektor5" : {'vektor': 'E', 'x' : 5, 'y' : 5, 'z': 5, 'kelas': 0},
    "vektor6" : {'vektor': 'F', 'x' : 8, 'y' : 8, 'z': 0, 'kelas': 0},
    "vektor7" : {'vektor': 'G', 'x' : 6, 'y' : 6, 'z': 0, 'kelas': 0},
}
for key in tbl:
    print(tbl[key])
vektorh = [3, 3, 3, "H"]
print("\n")
print({'vektor': vektorh[3], 'x' : vektorh[0], 'y' : vektorh[1], 'z': vektorh[2], 'kelas': '?
print("\n")
data = Vektor(tbl, vektorh[0], vektorh[1], vektorh[2], vektorh[3])
data.jarak()
print(data.kelas())
tbl["vektor8"] = data.kelas()
print("\n")
for key in tbl:
    print(tbl[key])
```

```
 {'vektor': 'A', 'x': 0, 'y': 1, 'z': 1, 'kelas': 0}
 {'vektor': 'B', 'x': 2, 'y': 2, 'z': 2, 'kelas': 1}
 {'vektor': 'C', 'x': 1.5, 'y': 1.2, 'z': 1, 'kelas': 1}
 {'vektor': 'D', 'x': 10, 'y': 9, 'z': 0, 'kelas': 0}
 {'vektor': 'E', 'x': 5, 'y': 5, 'z': 5, 'kelas': 0}
 {'vektor': 'F', 'x': 8, 'y': 8, 'z': 0, 'kelas': 0}
 {'vektor': 'G', 'x': 6, 'y': 6, 'z': 0, 'kelas': 0}


 {'vektor': 'H', 'x': 3, 'y': 3, 'z': 3, 'kelas': '?'}


 Jarak vektor A dengan vektor H sebesar 4.12
 Jarak vektor B dengan vektor H sebesar 1.73
```

```
Jarak vektor C dengan vektor H sebesar 3.08
Jarak vektor D dengan vektor H sebesar 9.7
Jarak vektor E dengan vektor H sebesar 3.46
Jarak vektor F dengan vektor H sebesar 7.68
Jarak vektor G dengan vektor H sebesar 5.2
dengan urutan:
Jarak vektor B dengan vektor H sebesar 1.73
Jarak vektor C dengan vektor H sebesar 3.08
Jarak vektor E dengan vektor H sebesar 3.46
Jarak vektor A dengan vektor H sebesar 4.12
Jarak vektor G dengan vektor H sebesar 5.2
Jarak vektor F dengan vektor H sebesar 7.68
Jarak vektor D dengan vektor H sebesar 9.7
3 Tetangga terdekat:
Vektor B dengan kelas 1 pada jarak 1.73 dengan vektor H
Vektor C dengan kelas 1 pada jarak 3.08 dengan vektor H
Vektor E dengan kelas 0 pada jarak 3.46 dengan vektor H
Kelas yang paling banyak muncul adalah =  1
Kelas dari vektor H =  1
{'vektor': 'H', 'x': 3, 'y': 3, 'z': 3, 'kelas': 1}


{'vektor': 'A', 'x': 0, 'y': 1, 'z': 1, 'kelas': 0}
{'vektor': 'B', 'x': 2, 'y': 2, 'z': 2, 'kelas': 1}
{'vektor': 'C', 'x': 1.5, 'y': 1.2, 'z': 1, 'kelas': 1}
{'vektor': 'D', 'x': 10, 'y': 9, 'z': 0, 'kelas': 0}
{'vektor': 'E', 'x': 5, 'y': 5, 'z': 5, 'kelas': 0}
{'vektor': 'F', 'x': 8, 'y': 8, 'z': 0, 'kelas': 0}
{'vektor': 'G', 'x': 6, 'y': 6, 'z': 0, 'kelas': 0}
{'vektor': 'H', 'x': 3, 'y': 3, 'z': 3, 'kelas': 1}
```

✓ 0s completed at 11:01 AM