RESUME PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

Oleh :

Giovanni Lucy Faustine S (121140060)

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

LAMPUNG SELATAN
2023

Pengenalan Pemrograman Phyton

Bahasa pemrograman phyton dibuat oleh Guido Van Rossum pada akhir 1980-an. Phyton mendukung banyak paradigma pemrograman seperti object-oriented, functional, dan structured. Bahasa pemrograman Phyton menjadi populer karena sintaksnya yang mudah, didukung oleh library yang berlimpah, dan bisa dipakai untuk pemrograman dekstop maupun mobile, CLI, GUI, web, automatasi, hacking, dan sebagainya.

Phyton tidak menggunakan kurung kurawal ({}) atau keyword. Sebagai gantinya, menggunakan spasi (white space) untuk memisahkan bolok-blok kode. Phyton lebih mudah dibaca dibandingkan dengan bahasa pemrograman lain seperti Java atau C++.

Konsep Dasar Dalam Object Oriented Programming (OOP)

Konsep dasar dalam object oriented programming pada phyton terdiri dari abstraksi, enkapulasi, inheritane, dan polymorphism.

1. Abstraksi

   Pada abstraksi, user dapat mengetahui apa yang objek lakukan, namun tidak mengetahui mekanisme dibelakangnya. Abstraksi dibuat dengan menampilkan atribut yang esensial dan menyembunyikan detail-detail yang tidak penting dari user dengan tujuan untuk mengurangi kompleksitas.

```python
from abc import ABC,abstractmethod
# abstract class
class Matkul(ABC):
    @abstractmethod
    def subject(self):
        pass
class Maths(Matkul):
    def subject(self):
        print("Mata kuliah matematika")
class Physics(Matkul):
    def subject(self):
        print("Mata kuliah fisika")
maths=Maths()
maths.subject()
physics=Physics()
physics.subject()
```

   Output:

```
Mata kuliah matematika
Mata kuliah fisika
```

2. Enkapulasi

   Enkapulasi adalah method yang dipakai untuk mengatur struktur kelas dengan menyembunyikan alur kerja kelas tersebut. Enkapulasi menyembunyikan property dan method dari suatu kelas agar hanya property dan method tertentu saja yang dapat diakses dari luar kelas. Untuk membatasi hak akses terhadap struktur kelas, terdapat 3 jenis access modifier, yaitu public access, protected access, dan private access.

```python
class Student:
    def __init__(self, name, age, nim):
        self.name = name
        self.__age = age #private variable
        self._nim = nim #protected variable

    # getter method
    def get__age(self):
        return self.__age

    # setter method
    def set__age(self, age):
        self.__age = age

    # getter method
    def get_nim (self):
        return self._nim

    # setter method
    def set_nim (self, nim):
        return self._nim

stud = Student('Budi', 14, 121140000)

# retrieving age using getter
print('Name:', stud.name)
print('Age: ', stud.get__age())
print("NIM: ", stud.get_nim())

# changing age using setter
stud.set__age(16)
stud.set_nim(1211140060)

# retrieving age using getter
print("Updated student's profile")
print('Name:', stud.name)
print('Age: ', stud.get__age())
print("NIM: ", stud.get_nim())
```

Output:

```
Age:  14
NIM:  121140000
Updated student's profile
Name: Budi
Age:  16
NIM:  121140000
```

3. Inheritance

   Inheritance memungkinkan sebuah class parent untuk menurunkan fungsi dan atribut kepada kelas lain. Kelas yang diturunkan tersebut adalah class child. Sementara itu, class child juga bisa memiliki atribut baru yang berbeda dengan class parent.

```python
# parent class
class Student:
    # attributes and method of the parent class
    def __init__(self, nama, nim) -> None:
        self.nama = nama
        self.nim = nim

# child class
class FirstYear(Student):
    def year(self):
        return "2020"

class SecondYear(Student):
    def year(self):
        return "2021"

# make object
p1 = FirstYear("Andi", 121140080)
p2 = SecondYear("Budi", 121140070)

# access attribute and methods from child class
print(p1.nama)
print(p1.nim)
print(p1.year())
print(p2.nama)
print(p2.nim)
print(p2.year())
```

Output:

```
Andi
121140080
2020
Budi
121140070
2021
```

4. Polymorphism

   Pada polymorphism, program yang dibuat dapat memiliki identical interfaces dengan eksekusi yang berbeda. Singkatnya, polymorphism adalah class turunan dari class parent, di mana pada class turunan, terdapat struktur yang berbeda.

```python
# parent class
class Student:
    # attributes and method of the parent class
    def __init__(self, nama) -> None:
        self.nama = nama

    def year (self):
        pass

# child class
class FirstYear(Student):
    def year(self):
        return "2020"

class SecondYear(Student):
    def year(self):
        return "2021"

def person_year(Student):
    print(Student.year())

# make object
p1 = FirstYear("Andi")
p2 = SecondYear("Budi")

# access attribute and methods from child class
person_year(p1)
person_year(p2)
```