**Nama : Sofia zalsabilah**

**Nim : 236270031**

**Prodi : Sistem informasi A**

**Apa itu OOP?**

Pemrograman Berorientasi Objek (OOP) adalah cara untuk menulis program dengan menggunakan objek. Objek ini menggabungkan data dan fungsi yang berhubungan. OOP membuat kode lebih terorganisir, mudah dipahami, dan dapat digunakan kembali.

Konsep Dasar OOP

- Kelas (Class) Kelas adalah cetak biru untuk membuat objek. Kelas mendefinisikan atribut dan metode yang dimiliki oleh objek.
- Objek (Object) Objek adalah instansi dari kelas. Setiap objek dapat memiliki data yang berbeda, meskipun berasal dari kelas yang sama.
- Encapsulation (Enkapsulasi) Proses menyembunyikan detail internal objek dan hanya memperlihatkan informasi yang diperlukan. Ini melibatkan penggunaan akses modifier seperti public, protected, dan private.

**Variabel**

Variabel adalah tempat untuk menyimpan data. Dalam OOP, variabel yang ada di dalam kelas disebut atribut.

**Metode**

Metode adalah fungsi yang didefinisikan dalam kelas. Metode ini biasanya digunakan untuk melakukan aksi pada objek atau untuk mengakses dan mengubah data dalam objek.

- Contoh Kode: Kelas Person

  Berikut adalah contoh kode yang menggambarkan kelas Person:

```python
class Person:
    # Konstruktor untuk inisialisasi atribut
    def __init__(self, name, umur):
        self.__name = name        # Atribut private
        self.__umur = umur        # Atribut private

    # Getter untuk nama
    def get_name(self):
        return self.__name

    # Setter untuk nama
    def set_name(self, name):
        self.__name = name

    # Getter untuk umur
    def get_umur(self):
        return self.__umur

    # Setter untuk umur
    def set_umur(self, umur):
        if umur > 0:
            self.__umur = umur
        else:
            print("Umur harus positif!")
```

```
    def display_info(self):
        print(f"Nama: {self.get_name()}, Umur: {self.get_umur()}")
```

**Penjelasan Kode**

- **Konstruktor __init__:** Digunakan untuk menginisialisasi atribut name dan umur.
- **Getter dan Setter:** Metode get_name, set_name, get_umur, dan set_umur digunakan untuk mengakses dan mengubah nilai atribut. Penggunaan atribut private (__name, __umur) melindungi data agar tidak diubah sembarangan.
- **Metode display_info:** Menampilkan informasi tentang objek termasuk nama, dan umur

**Inheritance (Pewarisan)**

Inheritance adalah konsep di mana kelas baru (subclass) mewarisi atribut dan metode dari kelas yang sudah ada (superclass).

**Contoh Kode: Kelas Student**

```
# Kelas anak (subclass)
class Student(Person):
    def __init__(self, name, umur, student_id):
        super().__init__(name, umur)  # Memanggil konstruktor kelas induk
        self.__student_id = student_id  # Atribut private

    # Getter untuk ID mahasiswa
    def get_student_id(self):
        return self.__student_id

    def display_info(self):
        super().display_info()  # Memanggil method dari kelas induk
        print(f"ID Mahasiswa: {self.get_student_id()}")
```

**Penjelasan Kode Student**

- Kelas Student mewarisi atribut dan metode dari kelas Person.
- Menggunakan super() untuk memanggil konstruktor dan metode dari kelas induk.

**Contoh Penggunaan**

```
# Fungsi untuk membuat mahasiswa
def create_students():
    students = []

    # Menambahkan mahasiswa dengan ID unik
    students.append(Student("Salsabila", 20, "2362400"))
    students.append(Student("Rizky", 21, "2362500"))
    students.append(Student("zulkifli", 22, "2362600"))
    students.append(Student("sofia", 20, "2362700"))

    return students

# Membuat daftar mahasiswa
student_list = create_students()

# Menampilkan informasi semua mahasiswa
```

```
for student in student_list:
    student.display_info()
```

**Kesimpulan**

Dengan menggunakan OOP, kita dapat membuat kode yang lebih terstruktur dan efisien. Konsep seperti encapsulation dan inheritance membantu dalam mengorganisir data dan fungsi dengan cara yang lebih logis.