MATERI AJAR

Pemrograman Berorientasi Objek

Materi Encapsulation

1. **Pengertian Enkapsulasi**

   Encapsulation adalah salah satu konsep fundamental dalam pemrograman berorientasi objek (OOP) yang menggabungkan data dan metode yang beroperasi pada data tersebut ke dalam satu unit, seperti kelas. Konsep ini membatasi akses langsung ke beberapa komponen objek, sehingga mencegah modifikasi data secara tidak sengaja dan memastikan integritas data.

2. **Tujuan Encapsulation**
   - Perlindungan Data: Mencegah akses dan modifikasi data secara langsung oleh kode di luar kelas, sehingga menjaga konsistensi dan validitas data.
   - Modularitas: Memungkinkan pemrogram untuk memisahkan kode menjadi unit-unit yang dapat dikelola dan dipahami secara terpisah.
   - Pemeliharaan: Memudahkan pemeliharaan kode dengan memungkinkan perubahan internal tanpa mempengaruhi bagian lain dari program yang menggunakan kelas tersebut.

3. **Konsep Dasar Encapsulation**

   Encapsulation melibatkan penggabungan data (atribut) dan metode yang beroperasi pada data tersebut ke dalam satu unit, seperti kelas. Akses ke data ini dikendalikan melalui metode khusus, sering disebut sebagai "getter" dan "setter", yang memungkinkan kontrol lebih besar atas bagaimana data diakses atau dimodifikasi.

4. ***Access Modifiers***

   Dalam OOP, access modifiers digunakan untuk menentukan tingkat aksesibilitas komponen kelas:
   - Private: Anggota hanya dapat diakses dalam kelas itu sendiri.
   - Protected: Anggota dapat diakses dalam kelas itu sendiri dan oleh subclass.
   - Public: Anggota dapat diakses dari kode di luar kelas.

5. **Manfaat Encapsulation**
   - Keamanan Data: Mencegah data dari modifikasi yang tidak sah atau tidak disengaja.
   - Kontrol: Memberikan kontrol penuh atas bagaimana data diakses atau dimodifikasi.
   - Fleksibilitas: Memungkinkan perubahan internal tanpa mempengaruhi kode eksternal yang bergantung pada kelas tersebut.
   - Pemeliharaan Kode: Meningkatkan kemampuan pemeliharaan dengan memisahkan antarmuka publik dari implementasi internal.

6. Contoh Implementasi

```
public class Siswa
{
    private string nama; // Atribut private, tidak bisa diakses langsung dari luar kelas

    // Getter untuk mengambil nilai nama
    public string GetNama()
    {
        return nama;
    }

    // Setter untuk mengubah nilai nama
    public void SetNama(string value)
    {
        nama = value;
    }
}

class Program
{
    static void Main()
    {
        Siswa siswa1 = new Siswa(); // Membuat objek Siswa
        siswa1.SetNama("Budi"); // Mengatur nilai nama menggunakan setter
        Console.WriteLine("Nama Siswa: " + siswa1.GetNama()); // Mengambil dan menampilkan
    }
}
```

Penjelasan:

- Atribut nama dibuat private → Tidak bisa diakses langsung dari luar class Siswa.
- Metode GetNama() dan SetNama() dibuat public → Untuk mengakses dan mengubah data secara aman.
- Di dalam Main() → Kita membuat objek siswa1, lalu menggunakan setter untuk mengisi nama dan getter untuk mengambil nilainya.

Referensi

Wilson, K. (2022). Object-Oriented Programming. In: The Absolute Beginner's Guide to Python Programming. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-8716-3_9