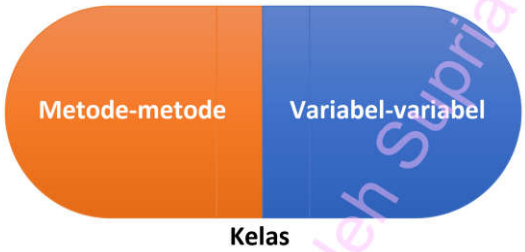
3.2. Enkapsulasi

Dalam istilah normal, Enkapsulasi didefinisikan sebagai membungkus data dan informasi di bawah satu unit. Dalam Pemrograman Berorientasi Objek, Enkapsulasi didefinisikan sebagai mengikat bersama data dan fungsi yang memanipulasinya. Pertimbangkan contoh enkapsulasi kehidupan nyata, di perusahaan, ada bagian yang berbeda seperti bagian akun, bagian keuangan, bagian penjualan, dll. Bagian keuangan menangani semua transaksi keuangan dan menyimpan catatan semua data yang terkait dengan keuangan. Demikian pula, bagian penjualan menangani semua aktivitas terkait penjualan dan menyimpan catatan semua penjualan. Sekarang mungkin timbul situasi ketika karena alasan tertentu seorang pejabat dari bagian keuangan membutuhkan semua data tentang penjualan pada bulan tertentu. Dalam hal ini, ia tidak diizinkan untuk langsung mengakses data bagian penjualan. Dia pertama-tama harus menghubungi beberapa petugas lain di bagian penjualan dan kemudian memintanya untuk memberikan data tertentu. Inilah yang dimaksud dengan enkapsulasi. Di sini data bagian penjualan dan karyawan yang dapat memanipulasi mereka dibungkus dengan satu nama "bagian penjualan".



Gambar 1. Enkapsulasi di C++

Enkapsulasi juga mengarah pada abstraksi data atau penyembunyian data. Menggunakan enkapsulasi juga menyembunyikan data. Dalam contoh di atas, data dari salah satu bagian seperti penjualan, keuangan, atau akun disembunyikan dari bagian lain.

3.2.1. Pengapsulan pada C++

Pengapsulan atau Enkapsulasi dalam C++ diartikan sebagai pembungkusan data dan informasi dalam satu kesatuan. Dalam Pemrograman Berorientasi Objek, Enkapsulasi didefinisikan sebagai pengikatan data dan fungsi yang memanipulasinya.

Perhatikan contoh enkapsulasi di kehidupan nyata, di sebuah perusahaan, ada bagian yang berbeda seperti bagian akun, bagian keuangan, bagian penjualan, dll. Sekarang,

- Bagian keuangan menangani semua transaksi keuangan dan mencatat semua data yang berkaitan dengan keuangan.
- Demikian pula, bagian penjualan menangani semua aktivitas yang berhubungan dengan penjualan dan menyimpan catatan semua penjualan.

Sekarang mungkin timbul situasi ketika karena alasan tertentu seorang pejabat dari bagian keuangan membutuhkan semua data tentang penjualan pada bulan tertentu.

Dalam hal ini, dia tidak diperbolehkan mengakses langsung data bagian penjualan. Pertama-tama dia harus menghubungi petugas lain di bagian penjualan dan kemudian memintanya untuk memberikan data tertentu.

Inilah yang dimaksud dengan Enkapsulasi. Di sini data bagian penjualan dan karyawan yang dapat memanipulasinya digabungkan dalam satu nama “bagian penjualan”.

Dua properti penting Enkapsulasi

- Perlindungan Data: Enkapsulasi melindungi keadaan internal suatu objek dengan menjaga kerahasiaan anggota datanya. Akses dan modifikasi anggota data ini dibatasi pada metode dengan hak akses publik yang ada dalam kelas, sehingga memastikan manipulasi data terkontrol dan aman. Metode adalah fungsi dalam pada kontek pemrograman tersetruktur, sedangkan hak akses secara bawaan adalah privat, terlindungi, dan publik.
- Penyembunyian Informasi: Enkapsulasi menyembunyikan detail implementasi internal suatu kelas dari kode eksternal. Hanya antarmuka publik yang dapat diakses, memberikan abstraksi dan menyederhanakan penggunaan kelas sekaligus memungkinkan implementasi internal dimodifikasi tanpa memengaruhi kode eksternal.

Misalnya jika kita memberi input, dan output harus setengah dari input

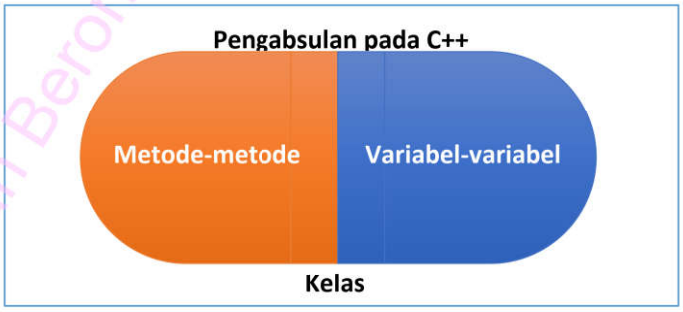
```
#include <iostream>
using namespace std;

class sementara{
      int a;
      int b;

      public:
      int penyelesaian(int input){
            a=input;
            b=a/2;
            return b;
      }
};

int main() {
      int n;
      cin>>n;
      sementara setengah;
      int ans=setengah.penyelesaian(n);
      cout<<ans<<endl;
}
```

3.2.2. Fitur Enkapsulasi



Gambar 2

Di bawah ini adalah fitur-fitur enkapsulasi:

- Kita tidak dapat mengakses fungsi apa pun dari kelas secara langsung. Kita memerlukan objek untuk mengakses fungsi tersebut yang menggunakan variabel anggota kelas tersebut.

- Fungsi yang kita buat di dalam kelas harus menggunakan variabel anggota saja, baru kemudian disebut enkapsulasi.
- Jika kita tidak membuat fungsi di dalam kelas yang menggunakan variabel anggota kelas tersebut maka kita tidak menyebutnya enkapsulasi.
- Enkapsulasi meningkatkan keterbacaan, pemeliharaan, dan keamanan dengan mengelompokkan data dan metode bersama-sama.
- Ini membantu untuk mengontrol modifikasi anggota data.

Enkapsulasi juga mengarah pada abstraksi data. Penggunaan enkapsulasi juga menyembunyikan data, seperti pada contoh di atas, data bagian seperti penjualan, keuangan, atau akun disembunyikan dari bagian lainnya.

Simple Example of C++:

```
#include <iostream>
#include <string>

using namespace std;

class Orang {
	private:
		string nama;
		int umur;
	public:
		Orang(string nama, int umur) {
		this->nama = nama;
		this->umur = umur;
	}
	void beriNama(string nama) {
		this->nama = nama;
	}
	string dapatkanName() {
		return nama;
	}
	void beriUmur(int umur) {
		this->umur = umur;
	}
	int dapatkanUmur() {
		return umur;
	}
};

int main() {
	Orang Orang("John Doe", 30);

	cout << "Nama: " << Orang.dapatkanName() << endl;
	cout << "Umur: " << Orang.dapatkanUmur() << endl;

	Orang.beriNama("Jane Doe");
	Orang.beriUmur(32);

	cout << "Nama: " << Orang.dapatkanName() << endl;
	cout << "Umur: " << Orang.dapatkanUmur() << endl;

	return 0;
}
```

Hasil eksekusi:
Nama: John Doe
Umur: 30
Nama: Jane Doe
Umur: 32

Di C++, enkapsulasi dapat diimplementasikan menggunakan kelas dan pengubah akses.

Contoh:

```
// Program C++ untuk mencoba
// Encapsulation
#include <iostream>
using namespace std;

class Enkapsulasi {
      private:
      // Data disembunyikan dari dunia luar
      int x;

      public:
      // Fungsi untuk menetapkan nilai
      // variabel x
      void beriNilai(int a) { x = a; }

      // Berfungsi untuk mengembalikan nilai
      // variabel x
      int dapatkan() { return x; }
};

// Kode pengubah
int main()
{
      Enkapsulasi obj;
      obj.beriNilai(5);
      cout << obj.dapatkan();
      return 0;
}
```

Hasil eksekusi:
5

Penjelasan: Pada program di atas, variabel x dijadikan privat. Variabel ini dapat diakses dan dimanipulasi hanya menggunakan fungsi dapatkan() dan beriNilai() yang ada di dalam kelas. Jadi kita dapat mengatakan bahwa di sini, variabel x dan fungsi dapatkan() dan beriNilai() terikat bersama yang tidak lain hanyalah enkapsulasi.

```
#include <iostream>
using namespace std;

// mendeklarasikan kelas
class Lingkaran {
      // pengubah akses
      private:
            // Anggota data
            float luas;
            float jari_jari;

      public:
            void dapatkanJari_jari()
            {
                  cout << "Masukkan jari_jari\n";
                  cin >> jari_jari;
            }
            void cariLuas()
            {
                  luas = 3.14 * jari_jari * jari_jari;
                  cout << "Luas Lingkaran=" << luas;
            }
};

int main()
{
      //Membuat obyek dari kelas lingkaran
      Lingkaran ling;
      ling.dapatkanJari_jari(); // panggil fungsi
      ling.cariLuas(); // panggil fungsi
}
```

Hasil eksekusi:
Masukkan jari_jari
5
Luas Lingkaran=78.5

3.2.3. Peran Penentu Akses dalam Enkapsulasi

Peran Penentu Akses dalam Enkapsulasi

Penentu akses memfasilitasi Penyembunyian Data dalam program C++ dengan membatasi akses ke fungsi anggota kelas dan anggota data. Ada tiga jenis penentu akses di C++:

- private: Penentu akses pribadi berarti fungsi anggota atau anggota data hanya dapat diakses oleh fungsi anggota lain dari kelas yang sama.
- protected: Penentu akses yang dilindungi berarti bahwa fungsi anggota atau anggota data dapat diakses oleh fungsi anggota lain dari kelas yang sama atau oleh kelas turunan.
- publik: Penentu akses publik berarti fungsi anggota atau anggota data dapat diakses dengan kode apa pun.

Secara asal atau bawaan, semua anggota data dan fungsi anggota suatu kelas dijadikan privat oleh kompiler.

Hal-hal yang Perlu Dipertimbangkan

Seperti yang telah kita lihat pada contoh di atas, penentu akses memainkan peran penting dalam mengimplementasikan enkapsulasi di C++. Proses penerapan enkapsulasi dapat dibagi lagi menjadi dua langkah:

1. Membuat kelas untuk merangkum semua data dan metode ke dalam satu unit.
2. Menyembunyikan data yang relevan menggunakan penentu akses.