5

IT 2022: Object Oriented Programming

Encapsulation

Sandali Goonatilleke

Department of Computer Engineering

Module Content

- Introduction to Object-Oriented Programming in C++
- Classes & Objects
- Constructors & Destructors
- Class Abstraction
- Encapsulation
- Composition
- Inheritance
- Polymorphism

Overview

- What is Encapsulation
- How Encapsulation is achieved in a class
- Advantages of Encapsulation
- Abstraction vs Encapsulation

Encapsulation - Real Life Example

Sales Section

Accounts Section

Finance Section

Encapsulation – Real Life Example

- In a company there are different sections like the accounts section, finance section, sales section etc.
- Each section handles different task
- The finance section handles all the financial transactions and the sales section handles all the sales related activities
- In case an official from finance section needs all the data about sales in a particular month

Encapsulation – Real Life Example

- In this case, he is not allowed to directly access the data of sales section
- He will first have to contact some other officer in the sales section and then request him to give the particular data
- Here the data of sales section and the employees that can manipulate them are **wrapped under a single name "sales section"**

Encapsulation

- In normal terms **Encapsulation** is defined as wrapping up of data and information under a single unit

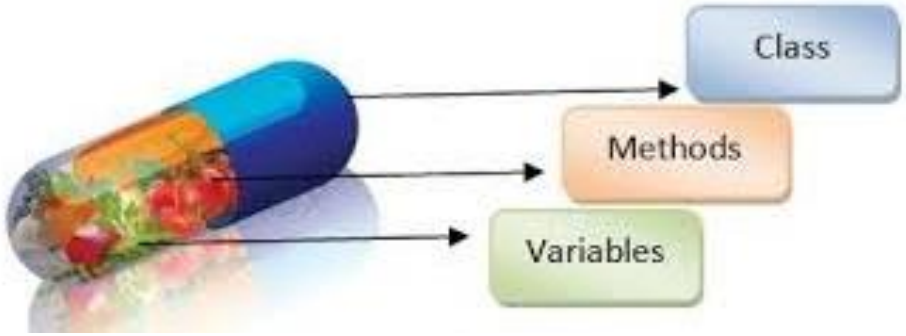
- In Object Oriented Programming, Encapsulation is defined as **binding together the data and the functions that manipulates them**

Encapsulation

- A concept can be used to bundle **data members** and **functions** inside a single class
- A process of wrapping similar code in one place
- Helps to keep data members and functions safe from outside interference and misuse
- Encapsulation led to the important OOP concept of **data hiding**
- Gives maintainability, flexibility and extensibility to the code

Encapsulation

- C++ supports the properties of encapsulation and data hiding through the creation of user-defined types, called **classes**



Role of Access Specifiers in Encapsulation

- Access specifiers plays an important role in implementing encapsulation in C++
- The process of implementing encapsulation can be sub-divided into two steps:

1. The data members should be labeled as private using the **private** access specifiers

2. The member function which manipulates the data members should be labeled as public using the **public** access specifier

Example

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

class Employee {
  private:
    // Private attribute
    int salary;
```

Example

```
public:
  // Setter
  void setSalary(int s) {
    salary = s;
  }
  // Getter
  int getSalary() {
    return salary;
  }
};
```

Example

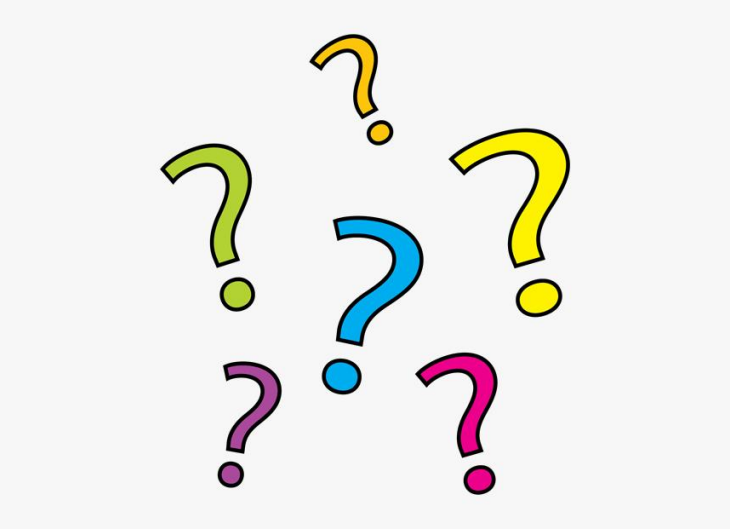
```
int main() {
   Employee myObj;
   myObj.setSalary(50000);
   cout << myObj.getSalary();
   return 0;
}
```

Advantages of Encapsulation

- The main advantage of using of encapsulation is to secure the data from other methods
- Encapsulated classes reduce complexity
- Encapsulated classes are easier to change

Abstraction Vs Encapsulation

Exercise: Compare and contrast Data Abstraction and Encapsulation



Abstraction Vs Encapsulation

Abstraction	Encapsulation
Hide the things at design level	Hide things at implementation level
Process of hiding the unwanted details and exposing only the essential features of a particular object or concept	A way of wrapping up data and methods into a single unit. Encapsulation puts the data safe from the outside world by binding the data and codes into single unit

Abstraction Vs Encapsulation

Abstraction	Encapsulation
Focus on what the object does instead of how it does	Hiding the internal details of how an object works
Means to show *What* part of functionality	Means to hide the *How* part of the functionality
Supported using interface and abstract class	Supported using access modifiers e.g. public, private and protected

Thank You!