Object Oriented programming:

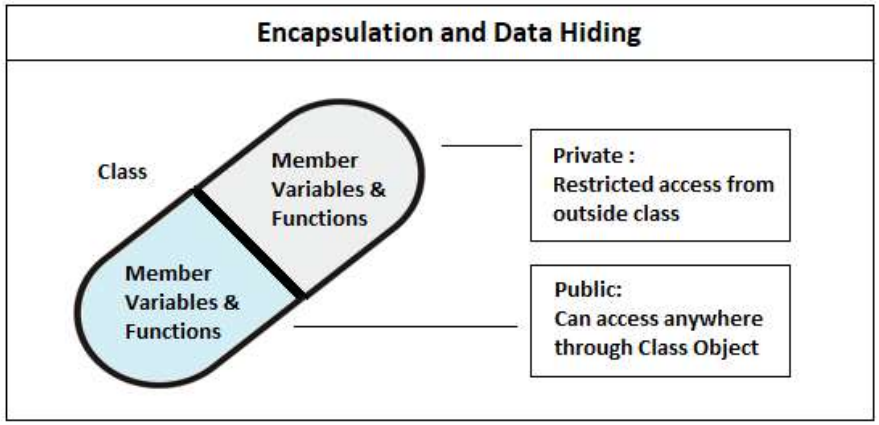
Encapsulation and Data Hiding

Course Code: CSC1102 &1103 Course Title: Introduction to Programming

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**10**	**Week No:**	**8(1X1.5)**	**Semester:**	
Lecturer:	*Name & email*				

What is Encapsulation



Encapsulation is the method of combining the data and functions inside a class. Thus the data gets hidden from being accessed directly from outside the class. This is similar to a capsule where several medicines are kept inside the capsule thus hiding them from being directly consumed from outside. All the members of a class are **private** by default, thus preventing them from being accessed from outside the class.

Data Hiding

- Data hiding is a technique especially practiced in object-oriented programming (OOP).Data hiding is hiding the details of internal data members of an object.
- Data hiding is also known as Information hiding.
- Sometimes Data Hiding includes Encapsulation. Thus Data Hiding is heavily related to Abstraction and Encapsulation.
- Data Hiding is the one most important OOP mechanism. Which is hide the details of the class from outside of the class.
- The Class used by only a limited set of variables and functions, others are hidden by the class.

An example

A Washing Machine and It's Power Button:

What is the function that power button does? Switches the machine on (obviously). But did you ever imagined the inside mechanism. Doesn't matter unless it's functioning well. That's encapsulation. The object is wrapped and inner details are hidden. Only thing is that object can be called and used.

Why Encapsulation

- Encapsulation is necessary to keep the details about an object hidden from the users of that object.
- Details of an object are stored in its data members. This is the reason we make all the member variables of a class private and most of the member functions public.
- Member variables are made private so that these cannot be directly accessed from outside the class (to hide the details of any object of that class like how the data about the object is implemented)
- Most member functions are made public to allow the users to access the data members through those functions.

Advantages

- The main advantage of using of encapsulation is to secure the data from other methods, when we make a data private then these data only use within the class, but these data not accessible outside the class.
- The major benefit of data encapsulation is the security of the data. It protects the data from unauthorized users that we inferred from the above stated real-real problem.
- We can apply the concept of data encapsulation in the marketing and finance sector where there is a high demand for security and restricted access of data to various departments.
- Encapsulation helps us in binding the data(instance variables) and the member functions(that work on the instance variables) of a class.
- Encapsulation is also useful in hiding the data(instance variables) of a class from an illegal direct access.
- Encapsulation also helps us to make a flexible code which is easy to change and maintain.

Encapsulation

How Encapsulation is achieved in a class:

- Make all the data members private.
- Create public setter and getter functions for each data member in such a way that the set function set the value of data member and get function get the value of data member.

Role of Access modifiers:

- Access specifiers plays an important role in implementing encapsulation in C++. The process of implementing encapsulation can be sub-divided into two steps:
- The data members should be labeled as private using the private access specifiers.
- The member function which manipulates the data members should be labeled as public using the public access specifier.


```cpp
#include<iostream>
using namespace std;
class Square
{
public:
    int Num;
    void Get(int n)
    {
        Num=n;
    }
    void Display()
    {
        cout << "Square Is:" << Num*Num<<endl;
    }
};

int main()
{
    Square Obj;
    Obj.Get(5);
    Obj.Display();
    Obj.Num=100;
    int a=Obj.Num;
    Obj.Get(a);
    Obj.Display();
}
```

```
Square Is:25
Square Is:10000
```

No encapsulation here.
Why???


```cpp
#include<iostream>
using namespace std;
class Square
{
private:
int Num;
public:

    void Get(int n)
    {
        Num=n;
    }
    void Display()
    {
         cout << "Square Is:" << Num*Num<<endl;
    }
};

int main()
{
    Square Obj;
    Obj.Get(5); Obj.Display();
    Obj.Num=100;
    int a=Obj.Num;
    Obj.Get(a); Obj.Display();
}
```

ERROR!!!!!

```
prog2.cpp: In function 'int main()':
prog2.cpp:24:6: error: 'int Square::Num' is private within this context
  Obj.Num=100;
      ^~~
prog2.cpp:6:5: note: declared private here
 int Num;
     ^~~
prog2.cpp:25:12: error: 'int Square::Num' is private within this context
  int a=Obj.Num;
            ^~~
prog2.cpp:6:5: note: declared private here
 int Num;
     ^~~
```

A program to demonstrate Encapsulation

```
#include<iostream>
using namespace std;
class Square
{
private:
    int Num;
public:
    void Get()
    {
        cout << "Enter Number:";
        cin>>Num;
    }
    void Display()
    {
        cout << "Square Is:" << Num*Num;
    }
};

int main()
{
    Square Obj;
    Obj.Get();
    Obj.Display();
}
```

- In this example, the variable "***Num***" is private. Hence this variable can be accessed only by the members of the same class and is not accessible anywhere else. Hence outside the classes will be unable to access this variable Which is called data hiding.
- At the same time, "***Square***" class contains two other methods namely "***Get***" and "***Display***" which has public members. Here "***Get***" method just prints the value while "***Display***" method prints the square of the value in the variable "***Num***".
- Here the class "***Square***" implements Data Encapsulation concept by capsulation of the value in the variable "***Num***" and thus allowing the user only to perform a restricted set of operations on the hidden variable "***Num***".

Another Example of Data Abstraction


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

class Adder{
   public:
 // interface to outside world
      void addNum(int number) {
          total += number;
      }
 // interface to outside world
      int getTotal() {
          return total;
      };
    private:
// hidden data from outside world
      int total = 2;
};
```

```
int main( )
{
   Adder a;

   a.addNum(10);
   a.addNum(20);
   a.addNum(30);

   cout << "Total "
        << a.getTotal();

   return 0;

}
```

Data Encapsulation prevents users of the classes from unintentionally altering a critical piece of data or gaining access to a sensitive data item. This programming strategy is called encapsulation or data hiding

Output

```
Total 62
```

Thank You