Lesson 12: Encapsulation

12.1. Introduction

In our previous lesson, we learnt about abstraction, one of the four principles of object oriented programming. Today, you will learn about encapsulation. Encapsulation is simply the process of forming objects.

12.2. Lesson objectives

By the end of this lesson, you will be able to:

- Understand the basic principle of encapsulation
- Describe the benefits and limitations of encapsulation
- Describe ways of implementing encapsulation

12.3. Lesson outline

This lesson is structured as follows:

12.4. Encapsulation in C++

Encapsulation is a feature of object oriented programming that binds together data and operations into units called objects.

An encapsulated object is often referred to as an abstract data type (ADT). By encapsulating object we create containers that can be accessed through a controlled manner. C++ supports the properties of encapsulation through the creation of classes. After encapsulation we can define member function called accessors to access the members.

12.5. Advantages of encapsulation

- It makes it easier to maintain the program.
- It makes it easier to understand the program. It improves the security of the code.

12.6. Disadvantages of encapsulation

- Encapsulation requires that the programmer clearly understands the access specifiers when coding accessor functions.
- Increased challenges when integrating functions/methods
- Increased challenges when maintaining the consistency in naming variables and functions.

12.7. Implementation of encapsulation

Encapsulation is supported in C++ through the use of the access modifiers such as public, protected and private which are placed in the declaration of the class. Anything in the class placed after the public keyword is accessible to all the users of the class; elements placed after the protected keyword are accessible only to the methods of the class or classes derived from that class; elements placed after the private keyword are accessible only to the methods of the class.

12.8. Structure versus class

To appreciate encapsulation, consider t two programs below that uses structure and class respectively. Notice classes support encapsulation.

```
#include<iostream>
 using namespace std;
struct rectangle //a structure to store variables
{
        int length; //public variable
         int width;  //public variable
         int area(int, int) //public function defined
             {
                 return (length*width);
             }
};
void main()
{
 rectangle rect; // instantiate object of type rectangle
cout << "Enter length of the rectangle:" << endl;
cin >> rect.length;
cout << "Enter length of the rectangle:" << endl;
         cin >> rect.width;
        cout << "The area of the rectangle is:" << rect.area(rect.length,rect.width)<< endl;
}
```

From this program we note that the structure has three members: two data members (length and width) and one member function(area). One object **rect** of type rectangle is instantiated. The members of the structure were accessed using dot notation.

Now consider the program below that uses class called rectangle.

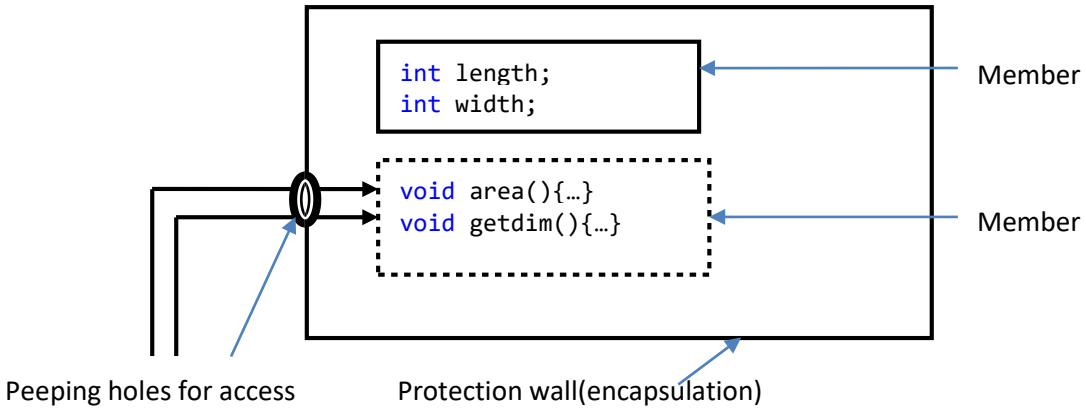
```
#include<iostream> using
namespace std; class rectangle
//a structure to store
variables
{
private:
```

```
int length; //public
variable    int
width;  //public
variable public:
        void area() //public function
        {
                cout<<length*width<<endl;
        }
        void getdim()
        {
                cout << "Enter length of the rectangle:" << endl;
                cin >> length;
                cout << "Enter length of the rectangle:" << endl;
                cin >> width;
        }
};
void main()
{
        rectangle rect; // instantiate object of type rectangle
        rect.getdim(); //pass message to the
        cout << "The area of the rectangle is:";
        rect.area();
}
```

From this program we note that the class has four members: two data members (length and width) and two member function(area and getdim). One object **rect** of type rectangle is instantiated. The members of the structure were accessed using dot notation but through methods.
Differences noted:

- By default structure members are public and so can be accessed from anywhere in the program however, class members are private by default.
- There is introduction of access specifiers for a class while in structure no access specifiers were indicated.
- The private members in the class could not be accessed directly. Public member functions of the same class were used to access the private members (controlled access).

This example demonstrates the importance of encapsulation in c++. This encapsulation can be summarized using the diagram below:



From this diagram representing our class, we notice that all the members (member variables and member functions) are enclosed in a protection wall (encapsulation). The peeping hole is providing a controlled access. From the main function in the program it is now possible to access public members of the class by passing a message (invoking member functions).Notice public members of a class can be accessed from anywhere in the program. Since the member functions **area()** and **getdim()** belong to the same class as the private member variables, it is possible for the member functions to access the member variable.

12.9. Revision questions

[a] Describe how encapsulation is implemented in C++

[b] Describe the differences between a structure and class. Explain how a class ensures encapsulation.

[c] Describe any two disadvantages of encapsulation.

12.10. Summary

In this lesson, you have learnt about encapsulation. You have learnt that by encapsulating an object we create containers that can be accessed through a controlled manner. C++ supports the properties of encapsulation through the creation of classes. The controlled access is made possible by use of accessor functions. We also learnt that encapsulation has both draws and drawbacks.

12.11. Suggested reading

[1]. Sams Teach Yourself C++ in 24 hours by Jesse Liberty and Rogers Cadenhead.

[2]. Object oriented programming using c++ by Joyce Farrell 4th.Edition

[3]. Object oriented programming with C++ by E Balagurusamy 3rd ed; publisher: Tata Mcraw Hill

[4]. Object-oriented programming with c++ by Sourav Sahay.