Lesson 10: Classes and objects

10.1. Introduction

In our previous lesson, you learnt about function overloading and overriding. Although this lesson was a continuation of lesson 8 about functions, we noted classes and objects were mentioned. In this lesson, you will learn about classes and objects in details. After the end of this lesson, you may refer to our discussions about function overloading and overriding again.

10.2. Lesson objectives

By the end of this lesson the learner will be able to

- Identify class from a given case/scenario
- Understand and use class, object, object instance and message.
- Understand and use keyword public and private.
- Describe different types of member functions
- Design and implement a class

10.3. Lesson structure

This lesson is organized as follows:

10.4. Declaring a class

C++ allows the programmer to define their own data types and thus it is considered to be extensible language. The syntax for declaring a class is as follows:

```
Class class_name
{
Access
specifier(visibility):
Data member(s);
```

```
Member function(s);
};
```

Class: The keyword for declaring a class.
Class_name: Any valid identifier for class e.g. book.
Access specifier: This is a keyword that defines the visibility of the class member that follows e.g. private, public or protected.
Data members: These are class members that define the state of the class. They are also referred to as attributes/variables.
Member functions: These are class members that define the behavior of the class. They are also referred to as operations or methods.
Remark: Binding together of data members and member functions is referred to as encapsulation.

10.5. Data members

They are also referred to as attributes/variables. The data members are usually declared as private.
The data members can be accessed using a dot operator. They are also referred to as attributes/variables.

```
Private:
Type 1 variable1;
Type2 variable 2;
……..
Type n variable n;
```

Type is any valid data type in c++ e.g. int, float, double, bool, char e.t.c.
The variable is any valid identifier e.g. BookNo, BookTitle.

10.6. Member functions

They are also referred to as operations or methods. These are functions that usually manipulate the data members. A member function can be declared as follows:

```
Type
function_name(argument
list);
```

Example

```
void area( float a, float l);
```

Every member function declared must be defined. The function body is added to describe what the function does. A function can be defined inside or outside the class declaration.

[a]. Defining a member function inside class declaration

```
Type function_name(argument list)

{ Statement(s);

}
```

The function is defined inside the class declaration and treated as an inline function. This means only small functions can be defined inside the class declaration.

[b]. Defining a member function outside class declaration

```
Type class_name:: function_name(argument list) {
Statement(s); }
```

The member functions are declared inside the class declaration. Each member function is defined (given function header and body). The scope resolution operator (::) must be used to give identity label.

10.7. Classifying the member functions

The roles of member functions can be placed into four categories:

[a]. Inspector functions (accessor functions): These are member functions that return information about an object state (data members). They can display some or all of objects attributes.

Example

```
getData();
displayinfo();
```

[b]. Mutator functions (implementers): These are member functions which change an objects state or attributes. They can obtain values from input device and assign those values to an objects data members. They can also calculate a data member based on values in other data members.

```
setData();

computerValue();
```

[c]. Auxiliary functions (facilitators): These functions perform the same action or service

e.g. sorting data or searching for data.

```
sortAscending();
findLowestValue();
```

[d]. Manager functions: These are functions used to create or destroy objects. They perform initialization and clean up. These are constructors and destructors.

[i]. Constructors: This is a member function (manager function) used to initialize the objects of its class. Its name is the same as the name of the class. It is invoked whenever an object of its associated class is created. All constructors are public. They are invoked automatically. Constructors do not have return types. They cannot be inherited.

```
Class A
{
Private:
……..;
……..;
Public:
A();
};
```

[ii]. Destructors: This is a member function (manager function) used to delete objects created by the constructor. Its name is the same as the name of the class but the name is preceded by a tilde (~).All destructors are public. They are invoked automatically. Destructors do not have return types and cannot be inherited.

```
Class A
{
Private:
……..;
……..;
Public:
~A();
```

```
};
```

10.8. Sample class program

This program uses a class called rectangle. Its members are length and width which are both private. It uses public member functions get dimensions and calculate Area to get dimensions and calculate the area respectively. Other member functions are the construct and destructor.

```
#include<iostream> using namespace std;
class rectangle
{
private:
        float length;      float width; public:
        void getdimensions();    void
calculateArea();            rectangle();
        ~rectangle();
};
rectangle::rectangle()
{
        cout<<" Initializing the object..."<<endl;
}
rectangle::~rectangle()
{
        cout<<" Deleting the object..."<<endl;
}
void rectangle::getdimensions()
{
        cout<<"Enter the length:"<<endl;          cin>>length;
        cout<<"Enter the width:"<<endl;
        cin>>width;
}
void rectangle::calculateArea()
{
        cout<<"The area of the rectangle is:"<<length*width<<endl;;
}
void main()
{
rectangle fig; fig.getdimensions();
fig.calculateArea();
}
```

10.9. Revision questions

[a]. Explain the difference between member function and data member.
[b]. Differentiate between destructor and
constructor [c]. Differentiate between private
and protected data member [d]. Describe
various ways of classifying member functions.

10.10. Summary

In this lesson you have learnt how to perform object oriented analysis, object oriented design and object oriented implementation. As a result you have learnt about the two important principles of object oriented programming. Firstly, encapsulation, which is the principle, used in binding data members and member functions into units called objects. Secondly, data abstraction (information hiding) which is the principle used to hide unnecessary details about the data members (data abstraction) by making them private and using member functions at the interface without concern of how they are implemented(functional abstraction).

10.11. Suggested reading.

[1]. Sams teach yourself c++ in 24 hours by Jesse Liberty and Rogers Cadenhead.
[2]. Object oriented programming in c++ by Joyce Farrel.