CHAPTER 8

OBJECTS AND CLASSES

After completing this lesson, you will be able to:

- Define class and object
- Know the members of class
- Understand and access specifies
- Know the concept of data hiding
- Define constructor and destructor
- Declare objects to access
- Understand the concept of the following with daily life examples

8.1 INTRODUCTION

Q1. What is the main purpose of C++ Programming?

Ans:

The main purpose of C++ programming is to add object orientation to the C++ programming language. Classes are the central feature of C++ that supports object-oriented programming and are often called user-defined types. A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package. The data and functions within a class are called members of the class.

8.1.1. CLASS AND OBJECT

Q2. Explain the concept of class in C++

Ans:

Classes

The classes are the most important feature of C++ that leads to Object Oriented Programming.

Class

Class is a user defined data type which holds its own data members and member functions. These can be accessed and used by creating object.

Object

An instance of a class is called object

Data Members and Members Functions

The variables inside class definition are called as data members and the functions are called member functions. A class consists of both attributes of real world objects and functions. Functions perform operations on these attributes. The attributes are called data members and the operations are called members functions.

Format of Class

The general format of class is

```
Class class_name
{
Access_specifier_1:     //Body of the class
Member1:
Access_specifier_2:
Member2,
}
```

Here class_name is a valid identifier for the class, which is followed by the body of the class that is enclosed in pair of braces.

Body of the Class

The body of the class consist of

1. Data members
2. Member functions

Access Specifier

Data members and member functions are used within the access specifiers

Types of Access Specifier

1) Private
2) Public
3) Protected

Program

Consider the following example to understand the concept of class

```
//Class declaration example

Class CRectangle

{

Private:              //declaration of data members

Int x, y :

Public,

Void set_values (int a, int b):       //declaration of member function

Int area ( ):                    //declaration of member function

};
```

Explanation

The above statements declare a class named CRectangle. This class contains four members

1. Two data members of type int (member x and member y) with private access

2. Two member functions with public access set_values ( ) and area ( )

Q3. Explain the concept of Object in C++

Ans:

Object

Objects are instances of class which holds the data variables declared in class and the member functions work on these class objects. In other words, a variable of type class is called object.

Instance of a Class

In C++ when we declare a variable of type class we call it instantiating the class and the variable itself is called an instance or object of that class.

Importance of object in OOP

Object is of great importance in OOP, because a class cannot be used without creating its object

Syntax to create an Object

The general syntax for creating an object of a class is given below

Class_name Object_name:

Example

For the class CRectangle the object can be created as follows

CRectangle R1; **// R1 is the object of class CRectangle**

We can also create more than one object in a single statement like

CRectaangle R1, R2, R3

Here R1, R2, R3 are the objects of the same class **CRectangle** that share the same data member a member functions. The definition of a class does not occupy any memory. It

only defines what the class looks like in user to use a class a variable of that class type must be declared. When an object is created then memory is on aside for all the data members and member functions of that class.

Program

Consider the following program to implement the class **CRectangle:**

```
// Object creation in Class
#include <iostream.h>
#include <conio.h>
Class CRectangle
{
Private:
Int X, Y:
Public:
Void set_values (int a, int b)
{
X= a;
Y= b;
}
Int area ()
{
Cout <<"area of the rectangle=" <<(X*Y);
}
};
```

```
CRectangle R1;
R1. Set_values (44, 22)        // call to set_values function
R1.area ();                    // call to area function
Getch();
Return 0;
}
```

In this program, the member function set-values and area are accessed by R1.set_values (44, 22) and R.area().

Output of the Program

Area of the rectangle = 968

8.1.2 MEMBER OF A CLASS

Q4. Describe the members of a class

Answer

Class has two members

i. Data members
ii. Member function

1. **Data member**

   The attributes of a class are called data members. Mostly, data members are declared under the private access Specifier to make them private to the class in which they are used

   Data members can also be used under public access specifier.

Explanation

The following lines of code are taken from the program discussed above to explain data members. For each objet of a class, a separate copy of the data members is created in memory

```
Class CRectangle
{
Private:
Int x;
Int y:
Public:
};
```

2. **Member function**

   The functions declared or defined inside the body of the class are called member functions. These member functions are usually written under the public access specifier to operate on the data members of the class. Member functions can also be written in the private area of the class but the practice is to write them in the public area

Program

```
// example of a class with its members
#include <iostream>
#include <conio.h>
Class date
{
Public
Int Month:
Int day:
Int Year:
```

```
Void SetDate (int nDay, int nMonth, int nYear)
{
Day= nDay:
Month = nMonth;
Year = nYear;
}
Void showDate ()
{
Cout<<"Day of birth:"<Day<<end1;
Cout<<"Month of birth"<<month<<end1;
Cout<<"Year of birth"<<year;
}
};
Int main ()
{
Data d1;
D1. SetDate (21. 07. 2010);          // call to showDate function
Getch();
Return 0;
}
```

Output of the Program

Day of birth : 21

Month of birth: 07

Year of birth: 2010

8.1.3 ACCESS SPECIFIERS

Q5. What is meant by access specifier? Explain most commonly used access specifier in C++ with the help of programs.

Answer

Access Specifier

Access specifiers in C++ define how the members of the class can be accessed. Access specifiers determine which member of a class is accessible in which part of the class/program.

Access Specifiers in C++

The most commonly used access specifiers in C++ are given below

i. Private access specifier
ii. Public access specifier

1. **Private access specifier**

   Private access specifier tells the computer that the members defined in a class by preceding this specifier are accessible only within the class and its friend function private members are not accessible outside the class.

Program

A detailed program explaining the use of private members within and outside the class i.e. in main() function is given here under.

```
// private access specifier example

#include <iostream.h>

#include <conio.h>

Class date

{

Private
```

```
Int Day:

Int Month;

Int Year:

Public

Void SetDate (int nDay, int nMonth, int nYear)

Day = nDay;

Month = nMonth;

Year = nYear;

Cout<<"Today's date is"<<Day<<month<<"/<<year;

Int main (:)

{

Date cDate;

// cDate. Day = 12;

// cDate. Month = 05;

// cDate. Year = 2010;

cDate SetDate (12, 05, 2010)        // valid

getch();

return 0;

}
```

2. **Public Access Specifier**

   Public members are accessible from anywhere where the object is visible i.e. within the class in the derived classes and in the main function

Program

Consider the following program to demonstrate the visibility of public members of a class

```
// public access specifier example 1
#include <iostream>
#include <conio.h>
Class PublicTest
{
Public:
Int x = 10:                    // Public data member
Void showPublic ()          // public data member
{
Cout<<"x=" << x;
}
};
Int main ()
{
Public Test PT1:             // PT1 is an object to class PublicTest
PT1. x:                       // access private data members
PT1. ShowPublic ();
Getch()
Return 0:
}
```

Explanation

In the above program, the class Public Test is defined that comprise of one data members X and one member function showPublic in its public part. The data member 'X' is accessed within the member function showPublic and outside the class i.e. in main() function, without generating any error message. Similarly the member function showPublic is also accessed in main()

Program

Here is an example of a class that uses all access specifiers

```
// private and public access specifier program

#include <iostream.h>

#include <conio.h>

Class Access

{

Int A;          // private by default

Void Get A()          // private by default

{

Cout<<"I am a private member accessible only through public member function"

<<End1;

}

Public:

Int B;          // public

Void GetB()

{

Get A ():

Cout<<"I am B in public"<<end1;
```

```
}
};
Int main ()
{
Access cAccess;    // cAccess is an object of the class Access
// cAccess. A = 2;    // WRONG because A is private data member
// cAccess. getA ();      // WRONG because GetA() is private member function
aAccess, B = 10          // Okay because B is public data member
cAccess. GetB ();        // Okay because GetB () is public member function
getch();
return 0;
}
```

Output of the Program

I am private member accessible only through public member function

I am B in public

8.1.4 DATA HANDLING/ENCAPSULTION

Q6. What is meant by data hiding/encapsulation? Explain.

Answer

Data Hinding/ Encapsulation

Data hiding is one of C++ features in which the members of a class are protected against illegal access from outside the class. In C++ we have the facility of hiding data using different access levels in classes

i. Private
ii. Protected
iii. Pubic

It is also called Encapsulation

Explanation

Private data members and member functions can only be accessed by the members of the same class defining them. They cannot be accessed from outside the class. Similarly, only the class defining them and its derived classes can be access protected members of a class. By using friend function the private and protected members of a class can be also accessed.

Levels of Hiding Members of a Class

The following table shows the level of hiding members of a class

Access	Public	Protected	Private
Access of members (data, functions) in the same class	Yes	Yes	Yes
Access of members (data, functions) in the derived class	Yes	Yes	No
Access of members (data,	Yes	No	No

functions) from outside the class			

8.1.5 CONSTRUCTOR AND DESTRUCTOR

Constructors and Destructors

Constructors and destructors are special member functions within the class with the same name as that of the class.

Q7. Explain the term "constructor" with the help of program.

Ans:

Constructor

A constructor is a special type of member function that initializes an object automatically when it is created. Compiler identified a given member function is a constructor by its name and the return type.

Rules for Naming Constructors

Unlike functions constructors have specific rulers/features for how they must be named

1) Constructors have the same name as that of the class
2) Constructors have no return type (not even void)
3) Constructor is always public

Program

Consider the following simple program to explain the concept of constructor in classes

```
//Constructor Example 1

#include <namespace.h>
```

```
#include <conio.h>
Class contest
{
Public:
ConstTest ( ) //Constructor
{
Cout < < "I am constructor";
}
Int main ( )
ConstTest CT;     // CT is an object to the class ConstTest
Getch ( );
Return 0;
}
```

Output of the program

I am constructor

Explanation

In the above example, a class ConstTest is defined which having the constructor ConstTest ( ) in its public part. In main ( ) program there is no explicit call to this constructor and it is automatically called when the object CT of this class is created.

Program

Consider another program to implement the class CRectangle including a constructor

```
//Constructor Example 2
#include <iostream.h>
```

```
#include <conio.h>
Class CRectangle
{
Int width , height :
Public:
CRectangle (int a, int b) //Constructor
{
Width=a:
Height=6;
}
Int area ( ) \
{
CRectangle r1 (13, 14):
CRectangle r2 (15, 16):
Cout < < "Area of Rectangle 1:" < < r1. Area ( ) < < end1:
Cout < < "Area of Rectangle 2:" < < r2. Area ( ) < < end1:
Getch ( )
Return 0:
}
```

Output of the program

```
Area of rectangle 1:182
Area of rectangle 2:240
```

Q8. Explain the type of constructor with the help of program

Ans:

Types of constructors

Following are the two types of constructors

1) Default constructor/ implicit constructor
2) User-defined constructor/ explicit constructor

1. Default constructor

A constructor without any arguments or with default values for every argument is treated as default/implicit constructor. If we do not declare any constructions in a class definition. The computer assumes the class to have a default constructor with no arguments. Therefore, after declaring a class likes

```
//default/implicit constructor

Class DConstructor

{

Public;

Int a, b, c ;

Void multiply (int n, int m )

{

A=n;

B=m;

C=a*b;

}

};
```

The compiler assumes that the class DConstructor has a default constructor, so the object of this class can be declared by specifying no arguments as given below

DConstructor DC :

Here DC is an object to the class DConstructor with no arguments. Default constructors are also called implicit constructors.

2. User-defined constructor/explicit constructor

When user defines constructors in class for their own purpose especially for initialization of variables then such types of constructions are called user defined constructors. When constructors is declared for a class, the compiler no longer provides an implicit default constructor. So the object should be declared according to the constructor prototypes that have been defined for the class

Program

The following program defines user-defined constructor

```
//user-defined constructor example

#include <iostream.h>

#include <conio.h>

Class CExample

{

Public :

Int a, b , c ;

Cexample (int n, int m)

{

A=n;
```

```
B=m;
}
Int multiply ( )
{
C=a*b;
Return c;
}
};
Int main ( )
{
CExample C0bj (50, 10);
Int result C0bj. Multiply ( );
Cout < < "Multiplication Results is :" < < result,
Getch ( ) ;
Return 0 ;
}
```

Output of the program

Multiplication result is 500

Explanation

Here a constructor CExample with two parameters of type int has been declared that initializes private variables a, and b. the object declaration is done as follows

CExample C0bj (5,10);

Q9. What is meant by constructor overloading? Describe with the help of program

Ans:

Constructor Overloading

Using more than one constructor in the same class is called constructor overloading.

Explanation

Like any other function, a constructor can be also be overloaded with more than one function that have the same name but different types or number of parameters. For an overloaded function, the compiler calls the function whose parameter(s) match the arguments used in the function call. In the case of constructor overloading the constructor is executed whose parameter(s) match(es) the arguments passed on their object declaration.

Program

Consider the following program

```
//construction overloading
#include <iostream.h>
#include <conio.h>
Class CRectangle
{
Int width, length;
Public :
CRectangle ( )
{
Width=5;
Length=15;
```

```
}
CRectangle (int a, int b)
{
Width=a;
Length=b;
}
Int area ( )
{
Return (width*length);
}
};
Int main ()
CRectangle r1 (5,14) ;
CRectangle r2;
Cout < < "Area of first rectangle :" < < r1. Area ( ) < < end1;
Cout < < "Area of second rectangle:"r2. Area ( ) < < end1;
Getch ( )
Return 0;
}
```

Output of the program

Area of first rectangle 70

Area of second rectangle 75

Explanation

In the above example the first object r1 passes the arguments (5,14) to the second constructor and thus the output generated is 70. The second object r2 calls the non-parameterized constructor of the class and initializes the variables width with 5 and length with 15 and thus the result calculated is 75

Q10. Explain the term destructor with the help of program

Ans:

Destructors

Destructors are special member functions that are executed when an object is destroyed. Destructor fulfils the opposite functionality of constructor and thus de-allocates the memory already allocated to an object during its creation. They are called automatically when objects are destroyed. They are denoted by a title symbol-

Features of destructor

Destructors have the following specific features

1. Destructors has the same names as that of the class preceded by a title symbol ~
2. Destructor cannot take arguments
3. Destructor has no return type

Note: the second feature given above, implies that only one destructor may exist per class, as there is no way to overload destructors since they cannot be differentiated from each other based on arguments.

Program

Consider the following program to explain the concept of destructor

```
//constructors and destructors example

#include <iostream.h>
```

```
#include <conio.h>
Class A
{
Public :
A ( )          //constructor function
Cout < <"I am constructor " < < end1'
}
-A ( )         //destructor function
{
Cout < < "I am destructor";
}
};
Int main ( )
{
A a1;
Getch ( );
Return 0;
}
```

Output of the program

I am constructor

I am destructor

Explanation

In this example ~A( ) is a destructor when the program runs and the object a1 is created. Constructor A( ) is called displaying the message "I am constructor" But when the control goes out of the program and the object is destroyed by ~A( ). A message "I am destructor" is displayed to verify that the destructor is executed.

8.1.6 DECLARATION OF OBJECTS FOR ACCESSING MEMBERS OF A CLASS

Q11. Describe the method of declaration of objects

Ans:

Normally, the execution of each program written in C++ program language starts from the main ( ) function. As we know that class is one of the main building blocks of OOP languages. These classes are written outside the main ( ) body, therefore they must be brought into the scope of main program. For the purpose objects are declared by the help of which class members are executed.

Program

The following shows how objects are declared

```
//Class objects declaration
#include <iostream.h>
#include <conio.h>
Class B
{
Public:
Private:
};
Int main ( )
```

```
B b1, b2, b3 :      //objects declaration
Getch ( ) :
Return 0:
}
```

Q12. Describe the method to access data members.

Ans:

Accessing data members

To access members of an object, whether data members or members functions dot operator (.) is used with the member name as shown here under

Object.member_name:

Program

Consider the following program

```
//Accessing class data members
#include <iostream.h>
#include <conio.h>
Class B
{
Private :
Int x,          //defines private data member
Public
Int y,          //defines public data member
},
```

```
Int main ()
{
B b1:
//b1.x: (invalid: private members are not accessible outside the class)
B1.y: //access public data member
Getch ( )
Return 0;
}
```

Explanation

The above program access the public data member y using the statement b1.y, where b1 is the object of the class B.

Q13. Describe the method of accessing members of a class

Ans:

Accessing member function

Member functions of a class can be accessed by the same way as data members are accessed i.e by the use of dot operator (.) in concatenation with the member function name as given below

Object.member function_name;

Program

Consider the following program

```
//Accessing class data members
#include <iostream.h>
#include <conio.h>
```

```
Class B
{
Private :
Int x,          //defines private data member
Public
Int y,          //defines public data member
Void member function ( )        //defines member function
{
Int x=10,
Int y=20;
Cout < <"Sum of x and y in member function is: " < < (x+y);
}
};
Int main  ()
{
B b1;
b1.member function ( );             //access member function
Getch ( )
Return 0;
}
```

Output of the program

Sum of x and y in member function is 30

8.1.7 INHERITANCE AND POLYMORPHSM

Q14. Introduce the concept of inheritance and polymorphism in C++

Ans:

Object Oriented Program (OOP) languages have the features of code reusability and polymorphism. Code reusability is the key feature among all other features of OOP which can be achieved by the use of inheritance. Similarly, C++ has the ability to use same thing for multiple tasks.

Q15. Explain the concept of inheritance in C++ with the help of program

Ans:

Inheritance

A key feature of C++ classes in which new classes are created from existing classes is called inheritance. Inheritance uses the concept of parent and child class.

Parent class/base class

A parent class is the class from which others classes are derived. It is also called base class.

Child class/sub class

A sub class is a class which inherits features from the base class. A sub-class is also called child class or derived class

Examples of inheritance

A few examples of inheritance from daily life are given below

1. Consider polygon as base class which has a series of child classes that describe polygons i.e square and pentagon. They have certain common properties, such as both can be described by means of only two sides and angles

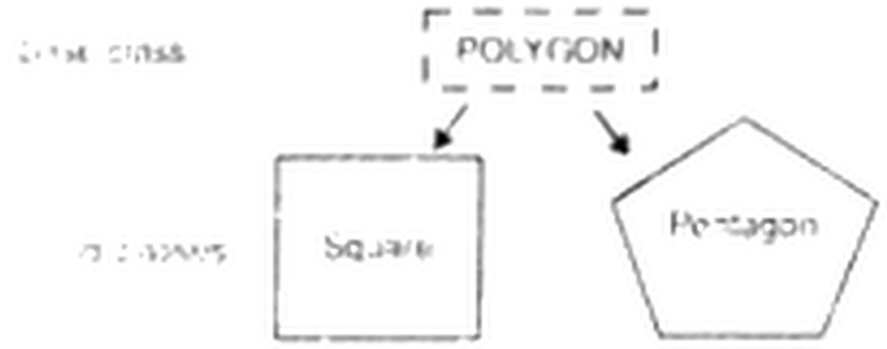

Figure 8.1: Inheritance of Polygon Class

2. We can also represent patient class into indoor and outdoor patients in which both have some common feature like name, father name, age, address and disease but indoor patient have ward no and bed no as its unique features and the outdoor patient have next date of visit as its unique feature. It can be represented as follows

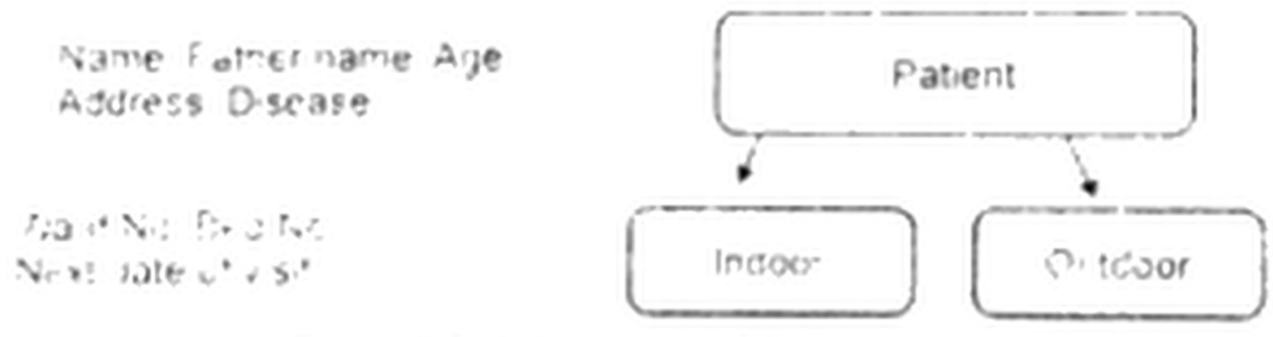

Figure 8.2: Inheritance of Patient Class

Syntax of using inheritance in classes

```
//syntax of using inheritance in classes
Class A                    //base class
{
}
Class B: public A      //class B is derived from class A
{
};
```

Program – inheritance of classes

A complete C++ program for the above sketch is given here under

```
//inheritance of classes
```

```
#include <iostream.h>
#include <conio.h>
Class A          //base class
{
Public
Void showa ( )
{
Cout < < "I am in base class A" < < end1;
}
};
Class B public A //class B is derived from class A
{
Public
Void showb( )
{
Cout < < "I am in derived class B"< < end1;
}
};
Int main ( )
{
B b1, //object of the derived class
B1.showa( );        //object of B inheriting function of base class b1
B1.showb( );
```

```
Getch( ):
Return 0:
}
```

Output of the program

```
I am in base class A
I am in derived class B
```

Explanation

In this example we have two classes A and B. Class B is publically derived from Class A and has therefore right to access the member function of Class A. In main ( ) we have created only object b1 for class B and have called the member functions showa( ) of base class A and showb( ) of class B that have produced the output as shown above

Note: in inheritance, sometimes a class is derived from more than one parent class the phenomenon is called multiple inheritance.

Q16. Explain the concept of polymorphism. Explain the methods of polymorphism in C++ with the help of programs.

Ans:

Polymorphism

Polymorphism is the ability to use an operator or function in multiple ways.
Polymorphism gives different meanings or functionality to the operators or functions.
Poly refers to many signifies that there are many uses of these operators and functions. It is the use of a single function or operator in many ways.

Methods to perform polymorphism

In C++ one of the following concepts can be achieve polymorphism

1. Function name overloading
2. Operator overloading
3. Virtual functions

1. **Function overloading**

   Function overloading is the concept of using same function name for related but slightly different purposes in a single program

Examples

Consider the following simple examples to understand the concept of polymorphism with respect to operator overloading

6+10;

The above statement refers to integer addition with the help of + operator. The same + operator can also be used for addition of two floating point numbers or two strings (concentration) as shown here under

7.15+3.78

"Computer " + "Training"

Polymorphism is a powerful feature of every OOP language especially on C++

2. **Operator Overloading**

   A single operator behaves differently in different contexts such as integer and float addition and strings concatenator. This concept is known as operator overloading.

3. **Virtual Function**

   A virtual function or virtual method is a function or method whose behavior can be overridden within an inheriting class by a function with the same signature. This concept is a very important part of the polymorphism portion of Object Oriented Program OOP.

DAILY LIFE EXAMPLES OF INHERITANCE AND POLYMORPHISM

Q17. Describe the daily life examples of inheritance and polymorphism.

Ans:

Inheritance

Acquiring some of the common qualities from parents (for instance eyes like mother, hair like father etc) is called inheritance.

A new model car having some inherited features from the old model like break system and navigation system etc

Polymorphism

It means multiple possible states for a single property. For example, a person can be a student as well as a friend to another person. Also a person can be an engineer as well as a teacher.

Key Points

- A class consists of attributes called data members and function called member function
- A variable of type class is called object. In C++ when variables of type class are declared they creates objects
- The attributes of a real world objects are called data members
- The functions declared or defined inside the body of the class are called member function
- Access specifiers determine that which member of a class is accessible by which member of the class/program
- Private access specifiers tells the compiler that the members defined in a class by preceding this specifier are accessible only within the class and its friend function

- Public members are accessible from anywhere where the object is visible
- Data hiding is the concept in which the members of a class are protected against illegal access from the outside the class
- A constructor is a special kind of class member function that is executed when an object of the class is instantiated
- Destructors are special member functions having the same name as that of the class with a title symbol ~ and executed when an object is destroyed
- To access members of a class dot operator (.) is used with the member name
- The phenomenon of creating new classes from existing classes is called inheritance
- Polymorphism is the ability to use an operator or function in multiple ways