OBJECT ORIENTED PROGRAMMING CONCEPTS

Object Oriented Programming follows bottom up approach in program design and emphasizes on safety and security of data..

FEATURES OF OBJECT ORIENTED PROGRAMMING:

Inheritance:

- Inheritance is the process of forming a new class from an existing class or base class. The base class is also known as parent class or super class.
- Derived class is also known as a child class or sub class. Inheritance helps in reusability of code , thus reducing the overall size of the program

Data Abstraction:

- It refers to the act of representing essential features without including the background details .Example : For driving , only accelerator, clutch and brake controls need to be learnt rather than working of engine and other details.

Data Encapsulation:

- It means wrapping up data and associated functions into one single unit called class..
- A class groups its members into three sections :public, private and protected, where private and protected members remain hidden from outside world and thereby helps in implementing data hiding.

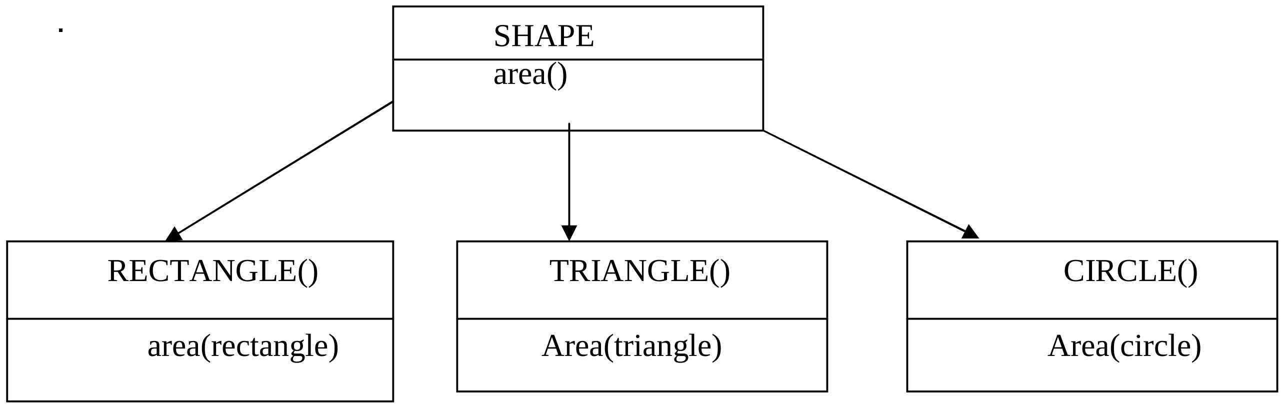
Modularity :

- The act of partitioning a complex program into simpler fragments called modules is called as modularity.
- It reduces the complexity to some degree and
- It creates a number of well defined boundaries within the program .

Polymorphism:

- **Poly** means many and **morphs** mean form, so polymorphism means one name multiple forms.
- It is the ability for a message or data to be processed in more than one form.
- C++ implements Polymorhism through Function Overloading , Operator overloading and Virtual functions .

.



Objects and Classes :

The major components of Object Oriented Programming are . **Classes & Objects**

A **Class i**s a group of similar objects . **Objects** share two characteristics: They all have *state* and *behavior*. For example : Dogs have state (name, color, breed, hungry) and behavior (barking, fetching, wagging tail). Bicycles also have state (current gear, current pedal cadence, current speed) and behavior (changing gear, applying brakes). Identifying the state and behavior for real-world objects is a great way to begin thinking in terms of object-oriented programming. These real-world observations all translate into the world of object-oriented programming.

Software objects are conceptually similar to real-world objects: they too consist of state and related behavior. An object stores its state in *fields* (variables in some programming languages) and exposes its behavior through functions

Classes in Programming :

- **It is a collection of variables, often of different types and its associated functions.**
- **Class just binds data and its associated functions under one unit there by enforcing encapsulation.**
- Classes define types of data structures and the functions that operate on those data structures.
- A class defines a blueprint for a data type.

Declaration/Definition :

A class definition starts with the keyword **class** followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations.

```
class  class_name {
                      access_specifier_1:
                      member1;
                      access_specifier_2:
                      member2;
                      ...
                      } object_names;
```

Where class_name is a valid identifier for the class, object_names is an optional list of names for objects of this class. The body of the declaration can contain members that can either be data or function declarations, and optionally access specifiers.

[Note: the default access specifier is private.

```
Example  : class Box  { int a;

           public:

              double length;   // Length of a box
              double breadth;  // Breadth of a box
              double height;   // Height of a box
};
```

Access specifiers in Classes:

Access specifiers are used to identify access rights for the data and member functions of the class. There are three main types of access specifiers in C++ programming language:

- private
- public
- protected

Member-Access Control

Type of Access	Meaning
Private	Class members declared as **private** can be used only by member functions and friends (classes or functions) of the class.
Protected	Class members declared as **protected** can be used by member functions and friends (classes or functions) of the class. Additionally, they can be used by classes derived from the class.
Public	Class members declared as **public** can be used by any function.

- **Importance of Access Specifiers**

Access control helps prevent you from using objects in ways they were not intended to be used. Thus it helps in implementing data hiding and data abstraction.

OBJECTS in C++:

Objects represent instances of a class. Objects are basic run time entities in an object oriented system.

Creating object / defining the object of a class:

The general syntax of defining the object of a class is:-

Class_name object_name;

In C++, a class variable is known as an object. The declaration of an object is similar to that of a variable of any data type. The members of a class are accessed or referenced using object of a class.

```
Box Box1;          // Declare Box1 of type Box
Box Box2;          // Declare Box2 of type Box
```

Both of the objects Box1 and Box2 will have their own copy of data members.

Accessing / calling members of a class ***All member of a class are private by default*.** Private member can be accessed only by the function of the class itself.Public member of a class can be accessed through any object of the class. They are accessed or called using object of that class with the help of dot operator (.).

The general syntax for accessing data member of a class is:-

Object_name.Data_member=value;

The general syntax for accessing member function of a class is:-

Object_name. Function_name (actual arguments);

The dot ('. ') used above is called the **<u>dot operator or class member access operator</u>**. The dot operator is used to connect the object and the member function. The private data of a class can be accessed only through the member function of that class.

<u>Class methods definitions (Defining the member functions)</u>

Member functions can be defined in two places:-

- ### Outside the class definition

The member functions of a class can be defined outside the class definitions. It is only declared inside the class but defined outside the class. The general form of member function definition outside the class definition is:

```
 Return_type Class_name:: function_name (argument list)
{
Function body
}
```

Where symbol :: is a scope resolution operator.

The scope resolution operator (::) specifies the class to which the member being declared belongs, granting exactly the same scope properties as if this function definition was directly included within the class definition

```
class sum
{
int A, B, Total;
public:
void getdata ();
void display ();
};
void sum:: getdata ()        // Function definition outside class definition Use of :: operator
{
cout<<" \n enter the value of A and B";
cin>>A>>B;
}
void sum:: display ()       // Function definition outside class definition Use of :: operator
{
Total =A+B;
cout<<"\n the sum of A and B="<<Total;
}
```

- **Inside the class definition**

The member function of a class can be declared and defined inside the class definition.

```
class sum
{
int A, B, Total;
public:
void getdata ()
{
cout< ”\n enter the value of A and B”;
cin>>A>>B;
}
void display ()
{
total = A+B;
cout<<”\n the sum of A and B=”<<total;
}
};
```

Differences between struct and classes in C++

In C++, a *structure* is a class defined with the `struct` keyword.Its members and base classes are public by default. A class defined with the `class` keyword has private members and base classes by default. This is the only difference between structs and classes in C++.

INLINE FUNCTIONS

- **Inline functions definition starts with keyword inline**
- **The compiler replaces the function call statement with the function code itself(expansion) and then compiles the entire code.**
- **They run little faster than normal functions as function calling overheads are saved.**
- **A function can be declared inline by placing the keyword inline before it.**

Example

```
inline  void Square (int a)

{ cout<<a*a;}

void main()

{.

  Square(4);         ──────►      { cout <<4*4;}

 Square(8)   ;        ──────►      { cout <<8*8; }

}
```

In place of function call , function body is substituted because Square () is inline function

Pass Object As An Argument

/*C++ PROGRAM TO PASS OBJECT AS AN ARGUMEMT. The program Adds the two heights given in feet and inches. */

```
#include< iostream.h>
#include< conio.h>
class height
{
int feet,inches;
public:
void getht(int f,int i)
{
feet=f;
inches=i;
}
void putheight()
{
cout< < "\nHeight is:"< < feet< < "feet\t"< < inches< < "inches"< < endl;
}
void sum(height a,height b)
{
height n;
n.feet = a.feet + b.feet;
n.inches = a.inches + b.inches;
if(n.inches ==12)
{
n.feet++;
n.inches = n.inches -12;
}
cout< < endl< < "Height is "< < n.feet< < " feet and "< < n.inches< < endl;
}};
void main()
{height h,d,a;
clrscr();
h.getht(6,5);
a.getht(2,7);
h.putheight();
a.putheight();
d.sum(h,a);
getch();
}
```

```
/**********OUTPUT***********
Height is:6feet 5inches
Height is:2feet 7inches
Height is 9 feet and 0
```

4 Marks Solved Problems :

Q 1**) Define a class TAXPAYER in C++ with following description :**
Private members :

- Name of type string
- PanNo of type string
- Taxabincm (Taxable income) of type float
- TotTax of type double
- A function CompTax( ) to calculate tax according to the following slab:

Taxable Income	**Tax%**
Up to 160000	0
>160000 and <=300000	5
>300000 and <=500000	10
>500000	15

Public members :

- A parameterized constructor to initialize all the members
- A function INTAX( ) to enter data for the tax payer and call function CompTax( ) to assign TotTax.
- A function OUTAX( ) to allow user to view the content of all the data members.

Ans.

```
class TAXPAYER
{
char Name[30],PanNo[30];
float Taxabincm;
double TotTax;
void CompTax()
{ if(Taxabincm >500000)
TotTax= Taxabincm*0.15;
else if(Taxabincm>300000)
TotTax= Taxabincm*0.1;
Else if(Taxabincm>160000)
TotTax= Taxabincm*0.05;
else
TotTax=0.0; }

public:
TAXPAYER(char nm[], char pan[], float tax, double tax) //parameterized constructor
{ strcpy(Name,nm);
strcpy(PanNo,pan);
Taxabincm=tax;
TotTax=ttax; }
void INTAX()
{ gets(Name);
cin>>PanNo>>Taxabincm;
CompTax(); }
void OUTAX()
{ cout<<Name<<'\n'<<PanNo<<'\n'<<Taxabincm<<'\n'<<TotTax<<endl; }
};
```

Q 2 : **Define a class HOTEL in C++ with the following description:**

Private Members

Rno	//Data Member to store Room No
Name	//Data Member to store customer Name
Tariff	//Data Member to store per day charge
NOD	//Data Member to store Number of days
CALC	//A function to calculate and return amount as NOD*Tariff and if the value of NOD*Tariff is more than 10000 then as 1.05*NOD*Tariff

Public Members:

- Checkin( ) //A function to enter the content RNo,Name, Tariff and NOD
- Checkout() //A function to display Rno, Name, Tariff, NOD and Amount (Amount to be displayed by calling function CALC( )

Solution :

```
#include<iostream.h>
class HOTEL
{       unsigned int Rno;
        char Name[25];
        unsigned int Tariff;
        unsigned int NOD;
        int CALC()
        {       int x;
                x=NOD*Tariff;
                if( x>10000)
                return(1.05*NOD*Tariff);
                else
                return(NOD*Tariff);
        }
        public:
        void  Checkin()
        {cin>>Rno>>Name>>Tariff>>NOD;}
        void Checkout()
        {cout<<Rno<<Name<<Tariff<<NOD<<CALC();}
        };
```

Q 3 Define a class Applicant in C++ with following description:
Private Members

- **A data member ANo (Admission Number) of type long**
- **A data member Name of type string**
- **A data member Agg(Aggregate Marks) of type float**
- **A data member Grade of type char**
- **A member function GradeMe( ) to find the Grade as per the Aggregate Marks obtained by a student. Equivalent Aggregate marks range and the respective Grades are shown as follows**

Aggregate Marks	**Grade**
> = 80	**A**
Less than 80 and > = 65	**B**
Less than 65 and > = 50	**C**
Less than 50	**D**

Public Members

- **A function Enter( ) to allow user to enter values for ANo, Name, Agg & call function GradeMe( ) to find the Grade**
- **A function Result ( ) to allow user to view the content of all the data members.**

Ans:
```
class Applicant
        {long  ANo;
        char Name[25];
        float Agg;
        char Grade;
        void GradeMe( )
        {       if (Agg > = 80)
                        Grade = ‘A’;
                else if (Agg >= 65 && Agg < 80 )
                        Grade = ‘B’;
                else if (Agg >= 50 && Agg < 65 )
                        Grade = ‘C;
                else
                        Grade = ‘D’;
        }
public:
        void Enter ( )
        {       cout <<”\n  Enter Admission No.      “;   cin>>ANo;
                cout <<”\n  Enter Name of  the Applicant     “;  cin.getline(Name,25);
                cout <<”\n  Enter Aggregate Marks obtained by the Candidate :“;  cin>>Agg;
                GradeMe( );
        }
        void Result( )
        {       cout <<”\n  Admission No.    “<<ANo;
                cout <<”\n  Name of  the Applicant   “;<<Name;
                cout<<”\n  Aggregate Marks obtained by the Candidate.      “ << Agg;
                cout<<\n    Grade Obtained  is    “ << Grade ;
        }
};
```

Q 4 Define a class ITEM in C++ with following description:
Private members:

- **Icode of type integer (Item Code)**
- **Item of type string (Item Name)**
- **Price of type Float (Price of each item)**
- **Qty of type integer (Quantity in stock)**
- **Discount of type float (Discount percentage on the item)**
- **A find function finddisc( ) to calculate discount as per the following rule:**

If Qty <=50	**discount is 0%**
If 50 < Qty <=100	**discount is 5%**
If Qty>100	**discount is 10%**

Public members :
A function Buy( ) to allow user to enter values for Icode, Item,Price, Qty and call function Finddisc ( ) to calculate the discount.
A function showall ( ) to allow user to view the content of all the data members.

```
Ans : class ITEM
{int Icode,Qty;
char item[20];
float price,discount;
void finddisc();
public:
void buy();
void showall();
};
void stock::finddisc()
{If (qty<=50)
Discount=0;
Else if (qty> 50 && qty <=100)
Discount=0.05*price;
Else if (qty>100)
Discount=0.10*price;
}
void stock::buy()
{cout<<"Item Code :";cin>>Icode;
cout<<"Name :";gets(Item);
cout<<"Price :";cin>>Price;
cout<<"Quantity :";cin>>Qty;
finddisc();
}
void TEST::DISPTEST()
{cout<<"Item Code :";cout<<Icode;
cout<<"Name :";cout<<Item;
cout<<"Price :";cout<<Price;
cout<<"Quantity :";cout<<Qty;
cout<<"Discount :";cout<<discount;
}
```

4 marks Practice Problems :

Q 1 Define a class **employee** with the following specifications : **4**

Private members of class employee

• empno	integer
• ename	20 characters
• basic, hra, da	float
• netpay	float
• calculate()	A function to calculate basic + hra + da with float return type

Public member function of class employee

- havedata() function to accept values for empno, sname, basic, hra, da and invoke calculate() to calculate netpay.
- dispdata() function to display all the data members on the screen.

Q2 Define a class **Student** with the following specifications : **4**

Private members :

• roll_no	integer
• name	20 characters
• class	8 characters
• marks[5]	integer
• percentage	float

- Calculate() a function that calculates overall percentage of marks and return the percentage of marks.

public members :
- Readmarks() a function that reads marks and invoke the Calculate function.
- Displaymarks() a function that prints the marks.

Q3 : Define a class **repor**t with the following specification : **4**

Private members :
- adno 4 digit admission number
- name 20 characters
- marks an array of 5 floating point values
- average average marks obtained
- getavg() to compute the average obtained in five subjects

Public members :
- readinfo() function to accept values for adno, name, marks, and invoke the function getavg().
- displayinfo() function to display all data members on the screen you should give function definitions.

Q4 Declare a class **myfolder** with the following specification : **4**

Private members of the class
- Filenames – an array of strings of size[10][25](to represent all the names of files inside myfolder)
- Availspace – long (to represent total number of bytes available in myfolder)
- Usedspace – long (to represent total number of bytes used in myfolder)

public members of the class

- Newfileentry() – A function to accept values of Filenames, Availspace and Usedspace fromuser
- Retavailspace() – A Fucntion that returns the value of total Kilobytes available (1 Kilobytes = 1024 bytes)
- Showfiles() – a function that displays the names of all the files in myfolder

2 Marks Practice Problems

1. What is relation between class and object?
2. What are inline functions? Give example
3. Difference between private & public access specifiers.
4. How class implements data-hiding & encapsulation?
5. What is the difference between structure and a class ?
6. How is inline function different from a normal function ?

CONSTRUCTORS AND DESTRUCTORS

CONSTRUCTORS :
A member function with the same as its class is called Constructor and it is used to initialize the object of that class with a legal initial value.
Example :

```
class Student
{
  int rollno;
float marks;
public:
student( )                //Constructor
{
rollno=0;
marks=0.0;
}
                        //other public members
};
```

TYPES OF CONSRUCTORS:

1. Default Constructor:
A constructor that accepts no parameter is called the Default Constructor. If you don't declare a constructor or a destructor, the compiler makes one for you. The default constructor and destructor take no arguments and do nothing.

2. Parameterized Constructors:
A constructor that accepts parameters for its invocation is known as parameterized Constructors , also called as Regular Constructors.

DESTRUCTORS:

- A destructor is also a member function whose name is the same as the class name but is preceded by tilde("~").It is automatically by the compiler when an object is destroyed. Destructors are usually used to deallocate memory and do other cleanup for a class object and its class members when the object is destroyed.
- A destructor is called for a class object when that object passes out of scope or is explicitly deleted.

Example :

```
class TEST
{   int Regno,Max,Min,Score;
Public:
 TEST( )                                // Default Constructor
 {                        }
 TEST (int Pregno,int Pscore)           //  Parameterized Constructor
{
Regno = Pregno ;Max=100;Max=100;Min=40;Score=Pscore;
}
~ TEST ( )                              // Destructor
{ Cout<<"TEST Over"<<endl;}
};
```

The following points apply to constructors and destructors:

- Constructors and destructors do not have return type, not even void nor can they return values.
- References and pointers cannot be used on constructors and destructors because their addresses cannot be taken.

- Constructors cannot be declared with the keyword virtual.
- Constructors and destructors cannot be declared static, const, or volatile.
- Unions cannot contain class objects that have constructors or destructors.
- The compiler automatically calls constructors when defining class objects and calls destructors when class objects go out of scope.
- Derived classes do not inherit constructors or destructors from their base classes, but they do call the constructor and destructor of base classes.
- The default destructor calls the destructors of the base class and members of the derived class.
- The destructors of base classes and members are called in the reverse order of the completion of their constructor:
- The destructor for a class object is called before destructors for members and bases are called.

Copy Constructor

- **A copy constructor is a special constructor in the C++ programming language used to create a new object as a copy of an existing object**.
- A copy constructor is a constructor of the form **classname(classname &).**The compiler will use the copy constructors whenever you initialize an instance using values of another instance of the same type.
- Copying of objects is achieved by the use of a copy constructor and a assignment operator.

Example :

```
class Sample{    int i, j;}
public:
Sample(int a, int b)        // constructor
{ i=a;j=b;}
Sample (Sample & s)      //copy constructor
{ j=s.j  ; i=s.j;
    Cout  <<"\n Copy constructor working \n";
}
void print (void)
{cout <<i<< j<< "\n";}
:
};
```

Note : ***The argument to a copy constructor is passed by reference, the reason being that when an argument is passed by value, a copy of it is constructed. But the copy constructor is creating a copy of the object for itself, thus ,it calls itself. Again the called copy constructor requires another copy so again it is called.in fact it calls itself again and again until the compiler runs out of the memory .so, in the copy constructor, the argument must be passed by reference.***

The following cases may result in a call to a copy constructor:

- **When an object is passed by value to a function**:
  The pass by value method requires a copy of the passed argument to be created for the function to operate upon .Thus to create the copy of the passed object, copy constructor is invoked
  If a function with the following prototype :
  void cpyfunc(Sample);   // Sample is a class
  then for the following function call
  cpyfunc(obj1);   // obj1 is an object of Sample type

the copy constructor would be invoked to create a copy of the obj1 object for use by cpyfunc().

- **When a function returns an object** :
  When an object is returned by a function the copy constructor is invoked

  Sample cpyfunc();   // Sample is a class and it is return type of cpyfunc()

  If func cpyfunc() is called by the following statement

  obj2 = cpyfunc();

  Then the copy constructor would be invoked to create a copy of the value returned by cpyfunc() and its value would be assigned to obj2. The copy constructor creates a temporary object to hold the return value of a function returning an object.

1 & 2 Marks Solved Problems :

Q1 :- Answer the questions after going through the following class.

```
class Exam
{char Subject[20] ;
        int Marks ;
public :
        Exam()                                    // Function 1
        {strcpy(Subject, “Computer” ) ; Marks = 0 ;}
        Exam(char P[ ])                           // Function 2
        {strcpy(Subject, P) ;
        Marks=0  ;
        }
      Exam(int M)                                 // Function 3
     {strcpy(Subject, “Computer”) ; Marks = M ;}
      Exam(char P[ ], int M)                      // Function 4
     {strcpy(Subject, P) ; Marks = M ;}
     };
```

a) Which feature of the Object Oriented Programming is demonstrated using Function 1, Function2, Function 3 and Function 4 in the above class Exam?

Ans:- Function Overloading (Constructor overloading)

b) Write statements in C++ that would execute Function 3 and Function 4 of class Exam.

Ans:- Exam a(10); and Exam b(“Comp”, 10);

Q2 Consider the following declaration :

```
class welcome
{public:
        welcome (int x, char ch);       // constructor with parameter
        welcome();                      // constructor without parameter
        void  compute();
private:
        int x;    char ch;
};
```

which of the following are valid statements

```
        welcome obj (33, ‘a9’);
        welcome obj1(50, ‘9’);
        welcome obj3();
        obj1= welcome (45, ‘T’);
        obj3= welcome;
```

Ans. Valid and invalid statements are

welcome obj (33, ‘a9’);	**valid**
welcome obj1(50, ‘9’);	**valid**
welcome obj3();	**invalid**
obj1= welcome (45, ‘T’);	**valid**
obj3= welcome;	**invalid**

2 Marks Practice Problems

Q1 What do you understand by constructor and destructor functions used in classes ? How are these functions different from other member functions ? 2

Q2 What do you understand by default constructor and copy constructor functions used in classes ? How are these functions different from normal constructors ? **2**

Q3 Given the following C++ code, answer the questions (i) & (ii). 2

```
class TestMeOut
{
public :
~TestMeOut() // Function 1
{ cout << "Leaving the examination hall " << endl; }
TestMeOut() // Function 2
{ cout << "Appearing for examination " << endl; }
void MyWork() // Function 3
{ cout << "Attempting Questions " << endl; }
};
```

(*i*) In Object Oriented Programming, what is Function 1 referred as and when does it get invoked / called ?
(*ii*) In Object Oriented Programming, what is Function 2 referred as and when does it get invoked / called ?

INHERITANCE

- **Inheritance is the process by which new classes called *derived* classes are created from existing classes called *base* classes**.
- The derived classes have all the features of the base class and the programmer can choose to add new features specific to the newly created derived class.
- The idea of inheritance implements the **is a** relationship. For example, mammal IS-A animal, dog IS-A mammal hence dog IS-A animal as well and so on.

Features or Advantages of Inheritance:

- ***Reusability of Code***
- ***Saves Time and Effort***
- ***Faster development, easier maintenance and easy to extend***
- ***Capable of expressing the inheritance relationship and its transitive nature which ensures closeness with real world problems .***

Base & Derived Classes:

A class can be derived from more than one classes, which means it can inherit data and functions from multiple base classes. A class derivation list names one or more base classes and has the form:

class derived-class: access-specifier base-class

Where access is one of **public, protected,** or **private**.

For example, if the *base* class is *MyClass* and the derived class is sample it is specified as:

```
class sample: public MyClass
```

The above makes sample have access to both *public* and *protected* variables of base class *MyClass*.

EXAMPLE OF SINGLE INHERITANCE

Consider a base class **Shape** and its derived class **Rectangle** as follows:

```
// Base class
class Shape
{
  public:
    void setWidth(int w)
    {
      width = w;
    }
    void setHeight(int h)
    {
      height = h;
    }
  protected:
    int width;
    int height;
};
```

```
// Derived class
class Rectangle: public Shape
{
  public:
    int getArea()
    {
      return (width * height);
    }
};
int main(void)
{
  Rectangle Rect;
   Rect.setWidth(5);
  Rect.setHeight(7);
  // Print the area of the object.
  cout << "Total area: " << Rect.getArea() << endl;

  return 0;

}
```

When the above code is compiled and executed, it produces following result:

```
Total area: 35
```

Access Control and Inheritance:

A derived class can access all the non-private members of its base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class.We can summarize the different access types according to who can access them in the following way:

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

A derived class inherits all base class methods with the following exceptions:

- Constructors, destructors and copy constructors of the base class.
- Overloaded operators of the base class.
- The friend functions of the base class.

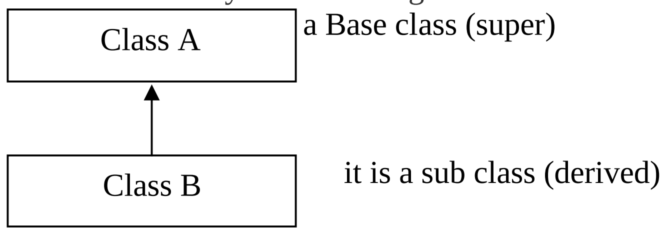
When deriving a class from a base class, the base class may be inherited through **public, protected** or **private** inheritance. We hardly use **protected** or **private** inheritance but **public** inheritance is commonly used. While using different type of inheritance, following rules are applied:

1. **Public Inheritance:** When deriving a class from a **public** base class, **public** members of the base class become **public** members of the derived class and **protected** members of the base class become **protected** members of the derived class. A base class's **private** members are never accessible directly from a derived class, but can be accessed through calls to the **public** and **protected** members of the base class.
2. **Protected Inheritance:** When deriving from a **protected** base class, **public** and **protected** members of the base class become **protected** members of the derived class.
3. **Private Inheritance:** When deriving from a **private** base class, **public** and **protected** members of the base class become **private** members of the derived Class.

TYPES OF INHERITANCE

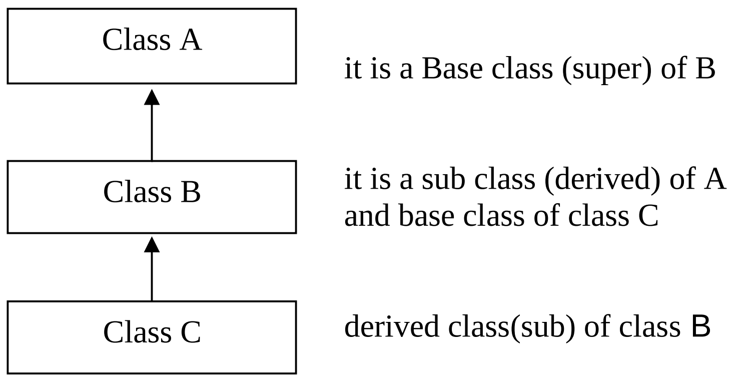
1. Single class Inheritance:

- Single inheritance is the one where you have a single base class and a single derived class.



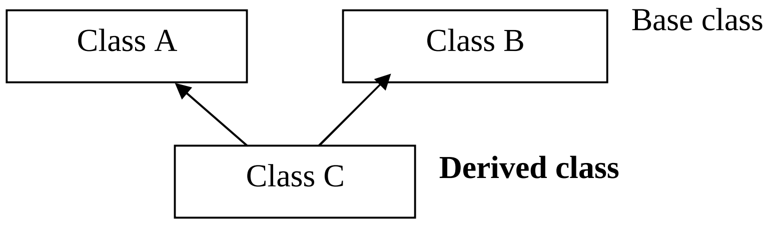
2. Multilevel Inheritance:

- In Multi level inheritance, a subclass inherits from a class that itself inherits from another class.


Class A
it is a Base class (super) of B
Class B
it is a sub class (derived) of A
and base class of class C
Class C
derived class(sub) of class B

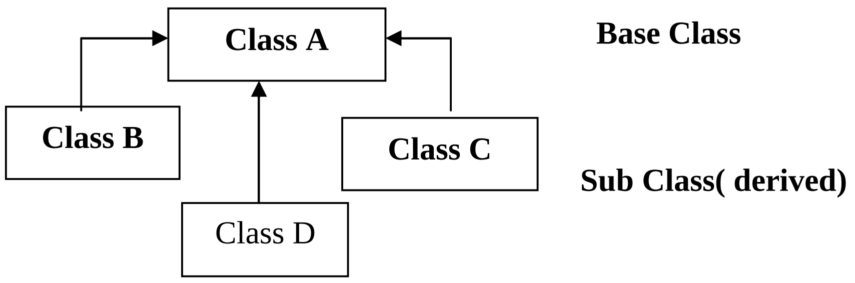
3. Multiple Inheritance:

- In Multiple inheritances, a derived class inherits from multiple base classes. It has properties of both the base classes.



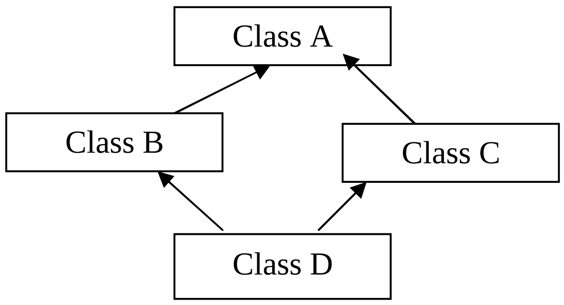
4. Hierarchical Inheritance:

- In hierarchial Inheritance, it's like an inverted tree. So multiple classes inherit from a single base class.



5. Hybrid Inheritance:

- It combines two or more forms of inheritance .In this type of inheritance, we can have mixture of number of inheritances but this can generate an error of using same name function from no of classes, which will bother the compiler to how to use the functions.
- Therefore, it will generate errors in the program. This has known as ambiguity or duplicity.
- Ambiguity problem can be solved by using **virtual base classes**



4 marks Solved Problems :

Q1. **Consider the following declarations and answer the questions given below :**

```
class WORLD
{
int H;
protected :
int S;
public :
void INPUT(int);
void OUTPUT();
};
class COUNTRY : private WORLD
{
int T;
protected :
int U;
public :
void INDATA( int, int)
void OUTDATA();
};
class STATE : public COUNTRY
{
int M;
public :
void DISPLAY (void);};
```

(*i*) Name the base class and derived class of the class COUNTRY.
(*ii*) Name the data member(s) that can be accessed from function DISPLAY().
(*iii*) Name the member function(s), which can be accessed from the objects of class STATE.
(*iv*) Is the member function OUTPUT() accessible by the objects of the class COUNTRY ?
Ans (*i*) Base class : WORLD
Derived class : STATE
(*ii*) M.
(*iii*) DISPLAY(), INDATA() and OUTDATA()
(*iv*) No

Q2. Consider the following declarations and answer the questions given below :

```
class living_being {
char name[20];
protected:
int jaws;
public:
void inputdata(char, int);
void outputdata();
}
class animal : protected living_being {
int tail;
protected:
int legs;
public:
void readdata(int, int);
void writedata();
};
class cow : private animal {
char horn_size;
public:
void fetchdata(char);
void displaydata();
};
```

(*i*) Name the base class and derived class of the class animal.
(*ii*) Name the data member(s) that can be accessed from function displaydata.
(*iii*) Name the data member(s) that can be accessed by an object of cow class.
(*iv*) Is the member function outputdata accessible to the objects of animal class.
Ans (*i*) Base class : living_being
Derived class : cow
(*ii*) horn_size, legs, jaws
(*iii*) fetchdata() and displaydata()
(*iv*) No

Q3. **Consider the following and answer the questions given below :**

```
class MNC
{
char Cname[25]; // Company name
protected :
char Hoffice[25]; // Head office
public :
MNC( );
char Country[25];
void EnterDate( );
void DisplayData( );
};
```

```
class Branch : public MNC
{
long NOE; // Number of employees
char Ctry[25]; // Country
protected:
void Association( );
public :
Branch( );
void Add( );
void Show( );
};
class Outlet : public Branch
{
char State[25];
public :
Outlet();
void Enter();
void Output();};
```

(*i*) Which class's constructor will be called first at the time of declaration of an object of class Outlet?
(*ii*) How many bytes an object belonging to class Outlet require ?
(*iii*) Name the member function(s), which are accessed from the object(s) of class Outlet.
(*iv*) Name the data member(s), which are accessible from the object(s) of class Branch.

Ans (*i*) class MNC
(*ii*) 129
(*iii*) void Enter(), void Output(), void Add(), void Show(), void EnterData(), void DisplayData().
(*iv*) char country[25]

Q4 Consider the following and answer the questions given below :

```
class CEO {
double Turnover;
protected :
int Noofcomp;
public :
CEO( );
void INPUT( );
void OUTPUT( );
};
class Director : public CEO {
int Noofemp;
public :
Director( );
void INDATA();
void OUTDATA( );
protected:
float Funda;
};
class Manager : public Director {
float Expense;
public :
Manager();
void DISPLAY(void);
};
```

(*i*) Which constructor will be called first at the time of declaration of an object of class Manager?
(*ii*) How many bytes will an object belonging to class Manager require ?
(*iii*) Name the member function(s), which are directly accessible from the object(s) of class Manager.
(*iv*) Is the member function OUTPUT() accessible by the objects of the class Director ?
Ans (*i*) CEO()
(*ii*) 16
(*iii*) DISPLAY(), INDATA(), OUTDATA(), INPUT(), OUTPUT()
(*iv*) Yes

4 marks Practice Problems:

Q1 :- Consider the following declarations and answer the questions given below:

```
class vehicle
        {int wheels;
        protected:
        int passenger;
        public:
        void inputdata( int, int);
        void outputdata();};
        class heavyvehicle: protected vehicle
        {int dieselpetrol;
        protected:
        int load;
        public:
        void readdata( int, int);
        void writedata();};
        class bus:private heavyvehicle
        {char marks[20];
        public:
        void fetchdata(char);
        void displaydata();};
```

(i) **Name the class and derived class of the class heavyvehicle.**
(ii) **Name the data members that can be accessed from function displaydata()**
(iii) **Name the data members that can be accessed by an object of bus class**
(iv) **Is the member function outputdata() accessible to the objects of heavyvehicle class.**

Q2:- Consider the following declarations and answer the questions given below:

```
        class book
        {
        char title[20];
        char author[20];
        int noof pages;
        public:
                void read();
                void show();};
        class textbook: private textbook
        {int noofchapters, noofassignments;
        protected:
        int standard;
        void readtextbook();
        void showtextbook();};
        class physicsbook: public textbook
        {char topic[20];
```

```
public:
void readphysicsbook();
void showphysicsbook();}
```

(i) Name the members, which can be accessed from the member functions of class physicsbook.
(ii) Name the members, which can be accessed by an object of Class textbook.
(iii) Name the members, which can be accessed by an object of Class physicsbook.
(iv) What will be the size of an object (in bytes) of class physicsbook.

Q3 : Answer the questions (i) to (iv) based on the following:

```
class CUSTOMER
{       int Cust_no;
        char Cust_Name[20];
        protected:
        void Register();
public:
        CUSTOMER( );
        void Status( );};
class SALESMAN
{       int Salesman_no;
        char Salesman_Name[20];
        protected:
          float Salary;
        public:
        SALESMAN( );
        void Enter( );
        void Show( );};
class  SHOP  :   private   CUSTOMER,  public  SALESMAN
{               char Voucher_No[10];
                char Sales_Date[8;
        public :
                SHOP( );
                void  Sales_Entry( );
                void  Sales_Detail( );};
```

(i) Write the names of data members, which are accessible from object belonging to class CUSTOMER.
(ii) Write the names of all the member functions which are accessible from object belonging to class SALESMAN.
(iii) Write the names of all the members which are accessible from member functions of class SHOP.
(iv) How many bytes will be required by an object belonging to class SHOP?

2marks Practice Problems:

1. What is access specifier ? What is its role ?
2. What are the types of inheritance ?
3. What is the significance of inheritance ?
4. What is the difference between private and public visibility modes?