# Chapter -3
# CLASSES AND OBJECTS

## Q1. What is class? What is its need?
## Classes

A class is a way to bind the data describing an entity and its associated functions together. In C++ , class makes a data type that is used to create objects of this type.

**Need for classes**

Classes are needed to represent real-world entities that not only have data type properties (Characteristics) but also have associated operations (their behaviour)

## Q2 . Differentiate between local and global class with help of example.

**Ans. 1. Global Class**

A class is said to be global it its declaration occurs outside the bodies of all functions in a program.

**2. Local class**

A class is said to be local if its occurs inside the function body

```
# include<iostream.h>
class x                          //x is a global class
{
members variable functions of class;
};

x ob1;                           //global object

int main ( )
{
class num                        // num is local class
        {
        public:
        int a=10;
        void num( )              // constructor (same name of class)
        {
        cout<<num;
        };

        x ob2;                           //local object of class x
        num n1;                          // n1is a class object
        n1.num( );
        }                                // closing of main ()
```

Q3. What do you mean by nested class ?Explain.

**Ans. Nested class**

Declaration of one class in to another class is called is called nested of a class.

```
#include<iostream.h>
```

```
            #include<conio.h>       // for clrscr()

            class score
            {
            int a;              //private member variable

            class batesman          // nested class
            {
            int b;

            public:
            int c;
            void total( )
            {
            cin>>b>>c;
            int sum = b+c;
            }
            batesman( )
            {
            c=10;
            }
            };                  //  end of class batesman
            batesman obj2;          // object of class batesman

            void second( void)
            {
            cout<<score :: second()<<endl;
            cout<<"A =" <<a
            }
                    score( )
                    {
                    a= 25;
                    }
            };      //end of class score;
    void main()
    {
            score ab;
            batesman bc;
            score :: batesman cd;
            ab.second();
            bc.total();
            cd.total();
    }
```

**Q4.** **What are the advantages and disadvantages of inline functions?**

**Ans.** The main advantages of inline functions is that they save on the overheads of a function call as the function is not invoked, rather its code is replaced in the program.

The major disadvantage of inline functions is that with more function calls, more memory is wasted as for every function call, the same function code is inserted in the program. Repeated occurrences of same function code waste memory space.

**Q5.** **What do you understand by member function?**

**Ans** Member function have full access privilege to both the public and private members of the class. These are defined within the class scope that is they are not visible outside the scope of class.

But not member functions are visible outside the class (ordinary functions)

**Q6.** **What are static class members?**

**Ans** A class can have static data members as well as static member functions The static data members are the class variables that are common for all the objects of the class. Only one copy of static data members is maintained which is shared by all the objects of the class. They are visible only with in the class

**Q7.** **Identify the errors in the following code fragment:**

```
class ab
{
        int x;
        static int ctr;
Public:
        Void init(void)         {
        x=ctr=0;                }
static void print(void)         {
        cout<<ctr<<x;           }       };
```

**Ans** In the above code fragment, a static member function is trying to access a non static member that is ‘x’ which is the error

**The correct code for static member function is**

```
        static void print(void)
        {
        cout<<ctr;              }
```

**Q8.** **What is the significance of access specifiers in a class?**

**Ans** A class provides three access specifiers namely private, public and protected

(a) A data member declared as private or protected remains hidden from outside world and it can only be accessed by the member functions of the class.

(b) A data member declared as public is made available to the outside world. That is, it can be accessed by any function, any expression in the program but only by using an object of the same class type.

These access labels enforce data hiding and abstraction also.

**Q9 .** **Write the output of the following code**

```
#include<iostream.h>
class counter                   {
private:
        unsigned int count;
public:
        counter()               // constructor
        {
        count=0;        }

        void inc_count( )       // increment in count variable
        {
                count ++;              }

                int get_count( )       {
                return count;          }
                                       };
        void main()
        {
        counter c1,c2;
        cout<< " \n value for c1 is:" <<c1.get_count( );
         cout<< " \n value for c2 is:" <<c2.get_count( );
        c1.inc_count();
        c2.inc_count();
        c2.inc_count();
        cout<< "\t c1=" <<c1.get_count();
        cout<< "\t c2=" <<c2.get_count();              }
```

**Ans**  **The output will be**

```
c1=0          c2=0          c1=1          c2=2
```

# Chapter – 4
# CONSTRUCTOR AND DESTRUCTORS

## Q1. What is Constructor ?
## Ans. Constructors:

A constructor is a special member function of a class that is automatically called, when an object is created of that class. **A member function with the same name as its class is called constructor.**

## Q2. What is destructor ? What is its need ?
## Ans. Destructor:

A destructor is also a member function whose name is the same as the class name but is **preceded by tilde ('~')**

A destructor takes no arguments and no return types can be specified for it. It is called automatically by the compiler when an object is destroyed. A destructor cleans up the storage ( memory area of the object) that is no longer accessible.

### Need for Destructors

Allocated resources (like constructor) must be de-allocated before the object is destroyed. It works as a de-allocating and releasing memory area and **it perform our works as a clean up tasks.** Therefore, a destructor is equally useful as a constructor is.

```
class stud
 {
        stud()                          //constructor
        {
        cout << " welcome"
```