**Lab 02 – *Classes and Data Abstraction***

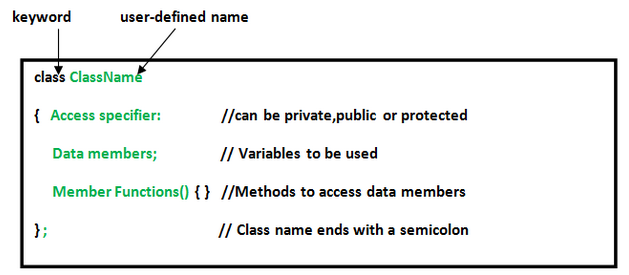
1. **Objective:**

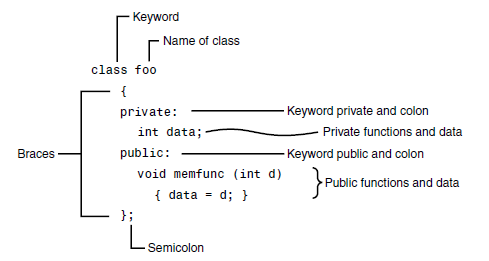
Objective of this lab is to understand the importance of classes and construction of objects using classes.

1. **Outcomes:** 
   1. The student will be able to declare classes and objects.
   2. The student will be able to declare member functions and member variables of a class.
   3. He will understand the importance and use of constructor.
   4. He will understand the use of destructor in class.
2. **Introduction** 
   1. **Class & Object:**

The fundamental idea behind object-oriented languages is to combine into a single unit both data and the functions that operate on that data. Such a unit is called an object. A class serves as a plan, or blueprint. It specifies what data and what functions will be included in objects of that class. An object is often called an “instance” of a class **Syntax:**

Classes are generally declared using the keyword class, with the following format:





**3.2 Data Abstraction:**

Abstraction is the process of recognizing and focusing on important characteristics of a situation or object and leaving/filtering out the un-wanted characteristics of that situation or object. For example a person will be viewed differently by a doctor and an employer.

* A doctor sees the person as patient. Thus he is interested in name, height, weight, age, blood group, previous or existing diseases etc of a person
* An employer sees a person as an employee. Therefore employer is interested in name, age, health, degree of study, work experience etc of a person.
  1. **Member Functions and Variables:**

Member variables represent the characteristics of the object and member functions represent the behavior of the object. For example length & width are the member variables of class Rectangle and set\_values(int,int), area() are the member functions.

* 1. **Constructors:**

It is a special function that is automatically executed when an object of that class is created. It has no return type and has the same name as that of the class. It is normally defined in classes to initialize data members.

**Syntax:**

class\_name( )

{

// Constructor body

}

* 1. **Destructors:**

It is a special function that is automatically executed when an object of that class is destroyed. It has no return type and has the same name as that of the class preceded by tild (~) character. Unlike constructors, destructors cannot take arguments.

**Syntax:**

~ class\_name ()

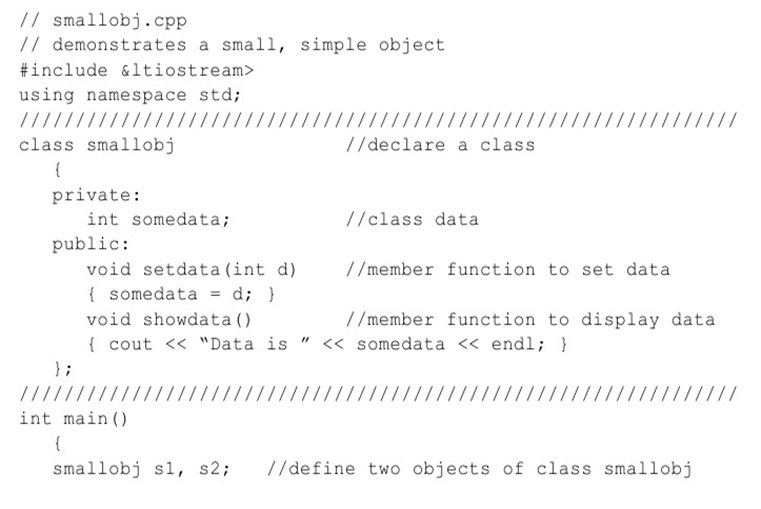
{

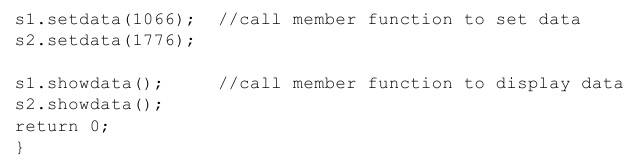
// Destructor body

}

1. **Examples**

**Basic:**

****



## 4.1 The following example shows the declaration of class Rectangle. It also demonstrates how member functions can be defined both inside and outside the class and how objects of class rectangle can be used to access its member functions.

#include <iostream>

**class** Rectangle

{

private:

**int** length, width; public:

void set\_values**(int**,**int**); //set the values of length and width

**int** area ()

{

return (length\*width);

}

};

**void** Rectangle::set\_values (**int** l, **int** w)

{

length = l; width = w;

}

**int** main ()

{

**Rectangle** rect;

rect.set\_values (3,4); //pass values to set length and width cout<< "area =" <<rect.area( );

return 0;

}

**OUTPUT:**

area= 12

**4.2 This example demonstrates how constructors and destructors work**

**class** counter

{

**private:**

int count;

**public:**

counter()

{

count=0;

cout<<"I am a constructor"<<endl;

}

~counter()

{ cout<<"I am a destructor"<<endl; }

};

**int** main( )

{

counter c;

}

1. **Lab Tasks** 
   1. Write a class that displays a simple message “I am object no. \_\_”, on the screen whenever an object of that class is created.
   2. Write a program to calculate the number of objects created and destroyed for the counter class.
   3. Create a class that imitates part of the functionality of the basic data type ‘int’, call the class **Int**. The only data in this class is an integer variable. Include member functions to initialize an **Int** to 0, initialize it to an ‘int’ value, to display it, and to add two **Int** values. Write a program that exercises this class by creating one uninitialized and two initialized **Int** values, adding the two initialized **Int** values and placing the response in uninitialized value and then displaying the result.
2. **Post Lab Tasks:** 
   1. Create a class named time, the data members are hours, minutes and seconds. Write a function to read the data members supplied by the user, write a function to display the data members in standard (24) hour and also in (12) hour format.
   2. Write a class marks with three data members to store three marks. Write three member functions, set\_marks() to input marks, sum() to calculate and return the sum and avg() to calculate and return average marks. Write a program that exercises this class by creating its objects and displaying results.