Classes and Objects

- Placing data and functions (that work upon data) together into a single entity is the central idea in object-oriented programming.
- C++ programmers use structures to exclusively hold data and classes to hold both data and functions.
  Example1:

```
#include <iostream>

using namespace std;
class Rectangle
{
    private:
        int len,br;
    public:
        void getdata()
        {
            cout<<endl<<"Enter the length and the breadth";
            cin>>len>>br;
        }
        void setdata(int l, int b)
        {
            len = l;
            br =bl
        }
        void displaydata()
        {
            cout<<endl<<"Length="<<len;
            cout<<endl<<"Breadth="<<br;
        }
        void area()
        {
            int a ;
            a=len*br;
            cout<<endl<<"Area="<<a;
        }
};

int main()
{
    Rectangle R1, R2;      // Define two objects of class Reactangle ().
    R1.setdata(10,20);
    R1.displaydata();
    R1.area();
    R2.getdata();
    R2.displaydata();
    R2.area();

    return 0;
}
```

Objects:

- Objects is an instance of a class, and the process of creating an object is called instantiation.

```
Rectangle R1; // R1 is an instance of class Rectangle.
```

- In example 1, the *private* and the *public*, are used to achieved the *data hiding*. This *data hiding* is used to conceal the data within a class.

- Usually, the data within a class is private and the functions are public.

Constructor:

- A constructor is a special member function.
- It is used to initialize the values of the data members in a class.
- It executes automatically whenever an object is created.
- Constructor has the same name as the Class it belongs to.
- No return type is used for the constructors.
- When no constructor is present in a class the compiler builds an implicit constructor. Note that once we declare a one-argument constructor it is necessary to define the implicit constructor.
  Example2:

```
#include<iostream>
using namespace std;
class A
{
    private:
        int i;
    public:
        void getdata()
        {
            cout<<endl<<"Enter any integer";
            cin>>i;
        }

        void setdata(int j)
        {
            i=j;
        }
        A()  // Zero Argument Constructor
        {

        }
        A(int j) // One argument Constructor
        {
            i=j;
        }
        void display()
        {
            cout<<endl<<"Value of i="<<i<<endl;
```

```
        }
};
int main()
{
    A a1(100),a2,a3;
    a1.display();
    a2.setdata(200)
    a2.display();
    a3.getdata();
    a3.display();
    return 0
}
```

Destructor:

- When an object is destroyed a special function called destructor automatically gets called.
- A destructor has the same name as the constructor but preceded by a tilde.
- When the control goes out of the *main()* the destructor gets executed to destruct the created objects

Example3:

```
#include<iostream>
using namespace std;
class A
{
    private:
        int i;
    public:

        A()  // Zero Argument Constructor
        {
            cout<<endl<<"Inside the Constructor"<<endl;
        }
        ~A() // One argument Constructor
        {
            cout<<endl<<"Inside the Destructor"<<endl;
        }

};
int main()
{
    A a;
    return 0
}
```