BCA SEMESTER-II

Object Oriented Programming using C++

Paper Code: BCA CC203

Unit :4

# Inheritance in C++

Nimisha Manan

Asst. Prof.

Deptt. of Computer Science

Patna Women's College

## What is Inheritance ?

- Inheritance is one of the most significant features of Object-Oriented Programming using C++.

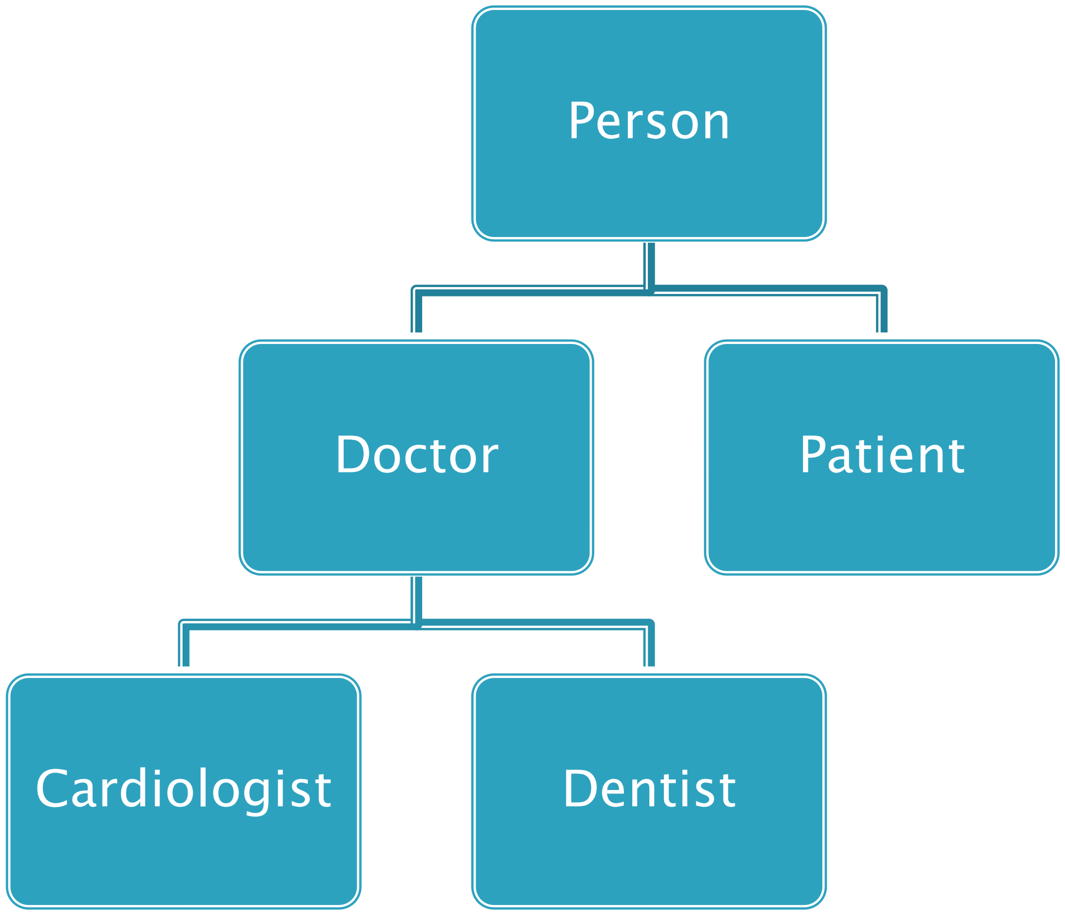
- Inheritance is a process of creating a new class from an existing class.

## Base Class & Derived Class

- **Base Class :-** The existing (old) class is called the Base class
- **Derived Class :-** The new class is called the Derived class or Sub class. The Derived class inherits the data and member functions of the Base class along with its new specialized data and functions.
- The derived class is the specialized class for the base class.

## The idea of Inheritance implements the is-a relationship.

For example, Cardiologist -IS A Doctor, Doctor IS-A Person, hence Cardiologist IS-A Person as well and so on.



# Advantages of Inheritance in C++

- Code Reusability :-. The Derived class can reuse all the functionalities of the Base class. So there is no need to redefine the members in the Derived class.
- Due to code reusability, less coding is required in the derived class.
- Enhanced Reliability:- The code of base class is already tested and debugged.
- Inheritance which makes it easier to create and maintain an application.
- The Derived classes can follow a standard interface.
- Code redundancy is minimized
- Inheritance supports code extensibility.

# Defining Derived Classes

A class can be derived from one or many classes. Thus the derived class can inherit data and functions from its base classes.

```
class derived_class_name : [access_specifier] base_class_name
{
   ----
   ----
}
```

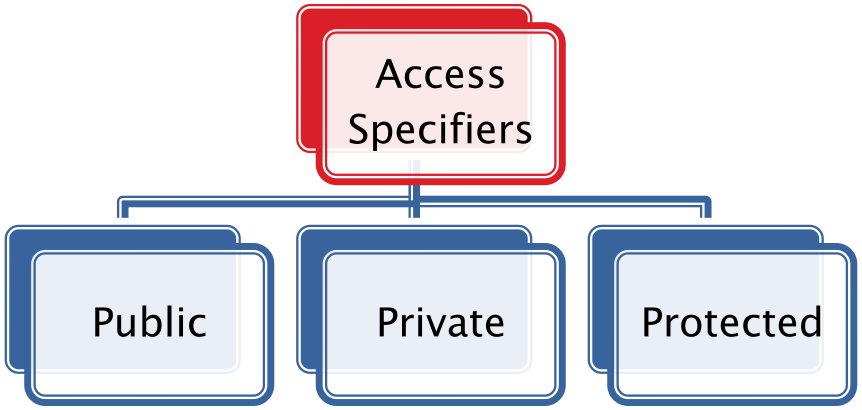
- **access_specifier** also known as **visibility mode** is optional and can be either **public, protected,** or **private.** If the access-specifier is not used, then it is private by default.

- **derived_class_name** is derived from the **base_class_name** which is the name of a previously defined class.

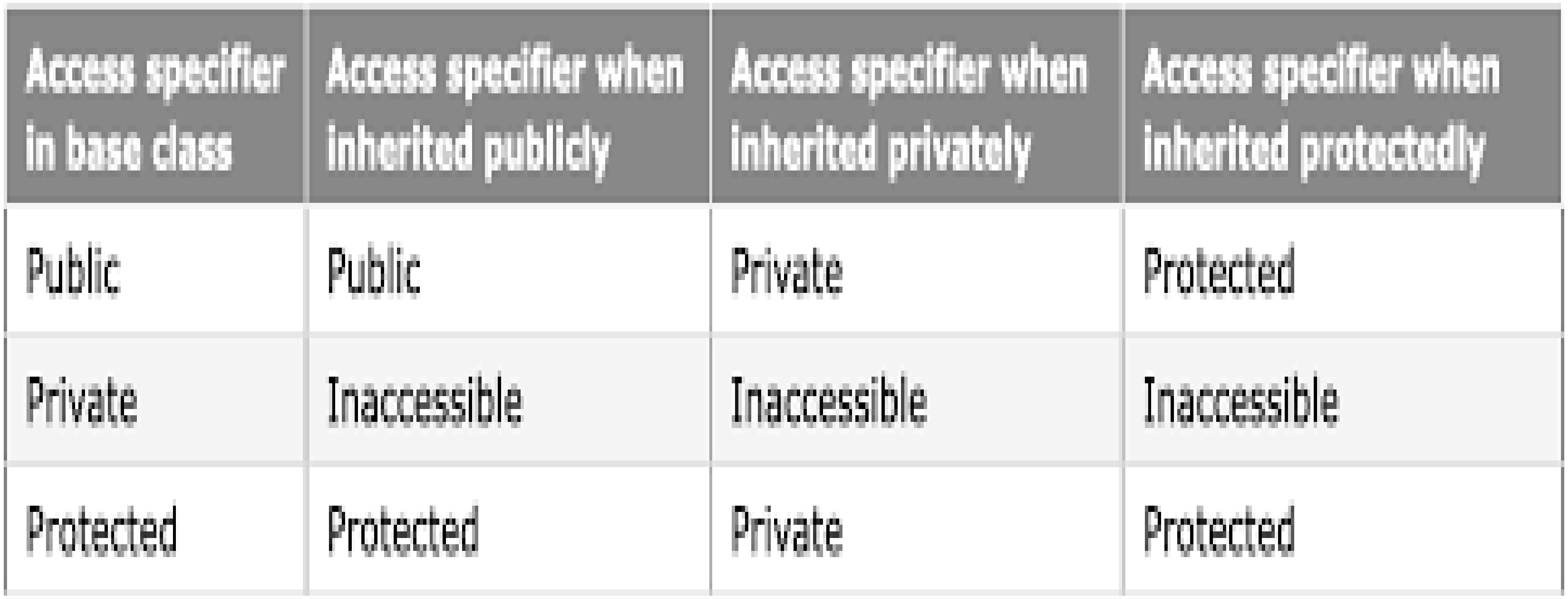
# Access Specifier

- **Access specifiers** define how the members (attributes and methods) of a class can be accessed.

- **public** – members are accessible from outside the class
- **private** – members cannot be accessed (or viewed) from outside the class
- **protected** – members cannot be accessed from outside the class, however, they can be accessed in inherited classes.



## Visibility modes of Inherited Members



| Access specifier in base class | Access specifier when inherited publicly | Access specifier when inherited privately | Access specifier when inherited protectedly |
|---|---|---|---|
| Public | Public | Private | Protected |
| Private | Inaccessible | Inaccessible | Inaccessible |
| Protected | Protected | Private | Protected |

```
#include<iostream.h>
class Doctor
{ public:
    Doctor()
    { cout<<"Hello, I am a Doctor"<<endl; }
 char HospitalName[20]="Sahyog Hospital";
};
  class Cardiologist: public Doctor      //This class inherits Doctor class
    {  public:
       Cardiologist()
           { cout<<"I am a Cardiologist "<<endl;  }      }
         char specialization[20]="Cardiology";
         char name[10]="Rakesh"; };
int main()
{  Cardiologist obj;
   cout<<"Name: "<<obj.name<<endl;
   cout<<"Hospital Name: "<<obj.HospitalName<<endl;
   cout<<"Specialization: "<<obj.specialization<<endl;
   return 0;    }
```

## Output:

```
Hello, I am a Doctor
I am a Cardiologist
Name: Rakesh
Hospital Name: Sahyog Hospital
Specialization: Cardiology
```

# THANK YOU