S. S Jain Subodh P.G. (Autonomous) College

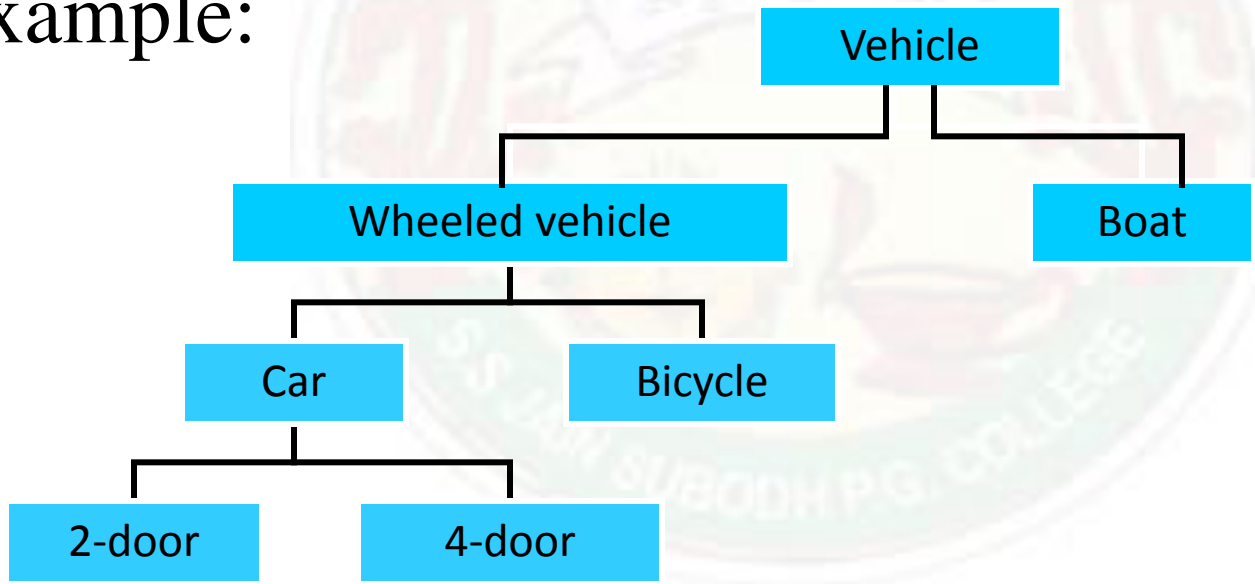
SUBJECT - Object Oriented Programming Concept

TITLE - Inheritance

Created By:
Shalu J. Rajawat

Inheritance is the process by which a class can be derived from a base class with all features of a base class and some of its own. This increased code reusability.

Example:



C++ and inheritance

- The language mechanism by which one class acquires the properties (data and operations) of another class
- Base Class (or superclass): the class being inherited from
- Derived Class (or subclass): the class that inherits

Syntax:
class derived-class: access-specifier base-class

```
#include <iostream.h>
// 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;
}
```

Advantages of inheritance

- When a class inherits from another class, there are **three** benefits:
- (1) You can *reuse* the methods and data of the existing class

  (2) You can *extend* the existing class by adding new data and new methods

  (3) You can *modify* the existing class by overloading its methods with your own implementations

Inheritance and accessibility

- A class inherits the *behavior* of another class and enhances it in some way
- Inheritance *does not* mean inheriting access to another class' private members

Access Control and Inheritance

Access	public	protected	private
Same classes	Yes	Yes	Yes
Derived classes	Yes	Yes	No
Outside classes	Yes	No	No

Types of base classes:

There is three types of base clsses in Inheritance of C++:

1. Public Inheritance
2. Protected Inheritance
3. Private Inheritance

Base class members working in child classes:

Inheritance	Public Members	Protected Members
Public inheritance	public	protected
Protected inheritance	protected	protected
Private inheritance	private	private

Rules for building a class hierarchy

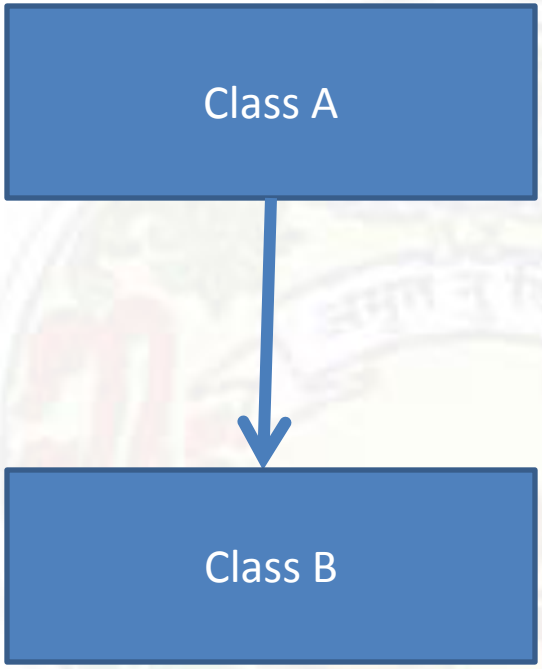
- Derived classes are **special cases** of base classes
- A derived class **can also serve** as a base class for new classes.
- There is no limit on the **depth of inheritance** allowed in C++ (as far as it is within the limits of your compiler)
- It is possible for a class to be a base class for **more than one** derived class

Types of Inheritance:

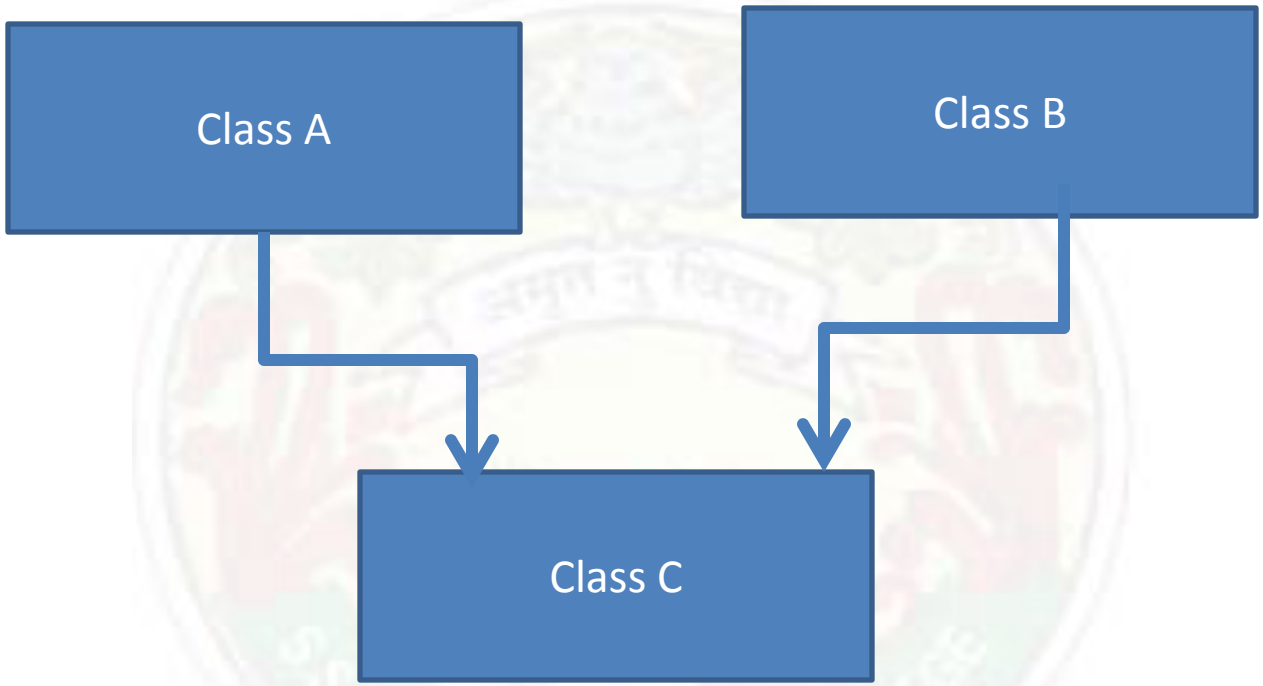
There is five types of inheritance allowed in c++:

1. Single Inheritance
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance

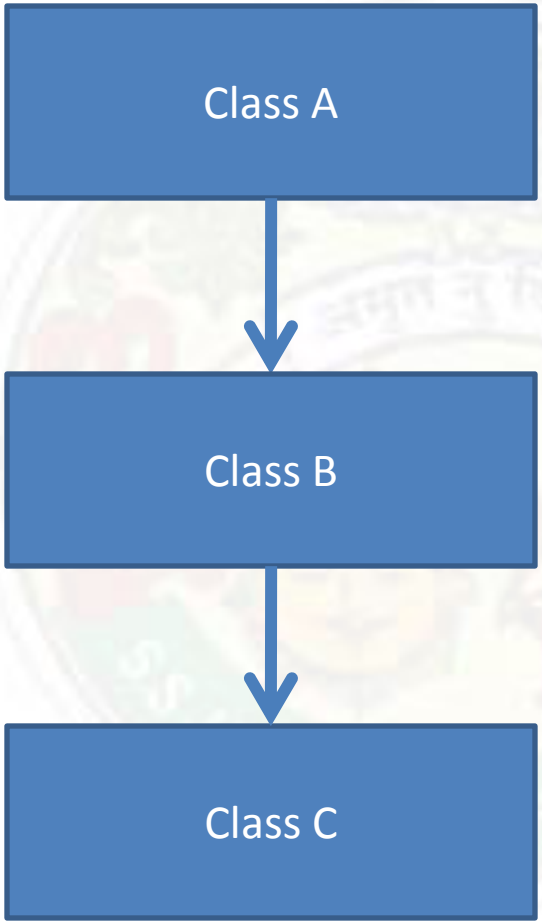
Single Inheritance:



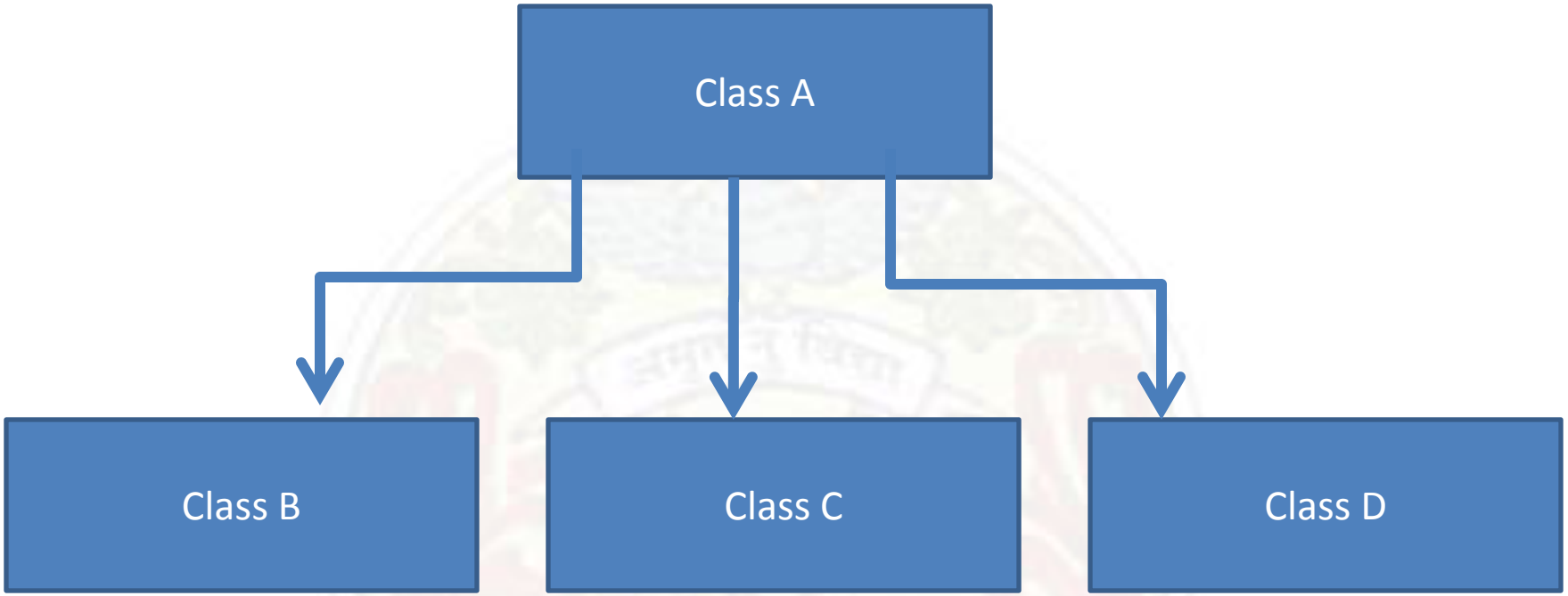
Multiple Inheritance



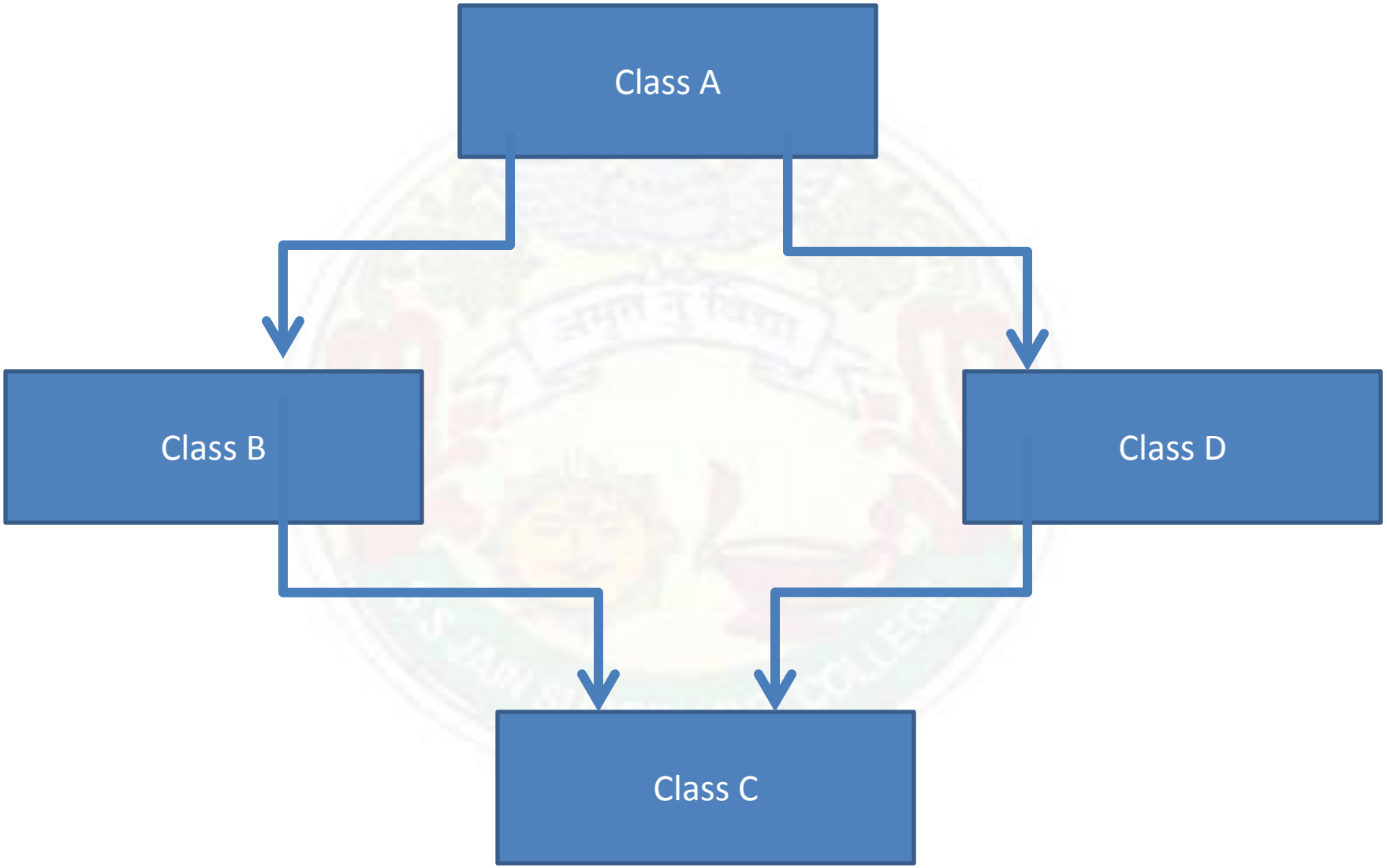
Multilevel Inheritance:



Hierarchical Inheritance



Hybrid Inheritance



PROGRAM:

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

class student
{
   protected:
      int rno,m1,m2;
   public:
              void get()
             {
                        cout<<"Enter the Roll no :";
                        cin>>rno;
                        cout<<"Enter the two marks   :";
                        cin>>m1>>m2;
             }
};
```

```
class sports
{
    protected:
       int sm;                    // sm = Sports mark
    public:
                 void getsm()
                {
                   cout<<"\nEnter the sports mark :";
                   cin>>sm;

                }
};
```

```
class statement:public student,public sports{
    int tot,avg;
    public:
    void display() {
                tot=(m1+m2+sm);
                avg=tot/3;
                cout<<"\n\n\tRoll No : "<<rno<<"\n\tTotal : "<<tot;
                cout<<"\n\tAverage    : "<<avg;
              }
};
void main(){
              clrscr();
              statement obj;
              obj.get();
              obj.getsm();
              obj.display();
   getch();
}
```

Output:

```
Enter the Roll no: 100
 Enter two marks
90
80
Enter the Sports Mark: 90
Roll No: 100
Total    : 260
Average: 86.66
```

Constructor and Destructor in Derived classes:

In Inheritance, Destructors are executed in reverse order of constructor execution. The destructor are executed when an object goes out of scope.
To know the execution of constructor and Destructors:

A Program to show sequence of execution of constructor and destructor in multiple Inheritance:

```
class Base {
        public:   Base ( ) {
                        cout << "Inside Base constructor" << endl;
                  }
                  ~Base ( ) {
                        cout << "Inside Base destructor" << endl;
                  }
};
 class Derived : public Base {
        public:    Derived ( ) {
                      cout << "Inside Derived constructor"<< endl;
                   }
                   ~Derived ( ) {
                      cout << "Inside Derived destructor" << endl;
                   }
};
 void main( ) {
        Derived x;
}
```

OUTPUT:

Inside Base constructor
Inside Derived constructor
Inside Derived destructor
Inside Base destructor

THANKS