Marwadi
education foundation

Prof. Meghnesh Jayswal

UNIT 5
Inheritance

Objective of this Unit

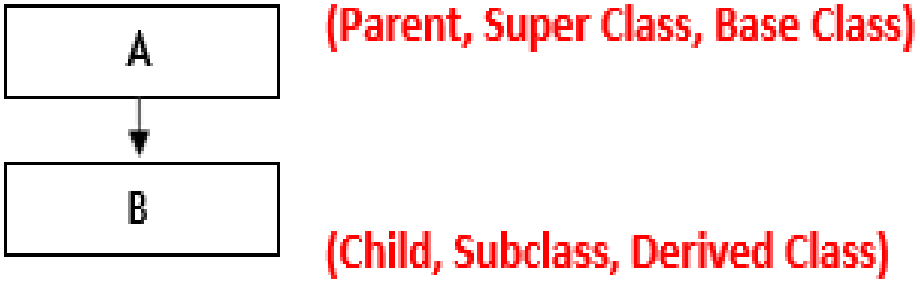
- Introduction to Inheritance
- Derived Class,
- Levels of Inheritance
- Hybrid Inheritance
- Abstract class
- Virtual base class

Introduction to Inheritance

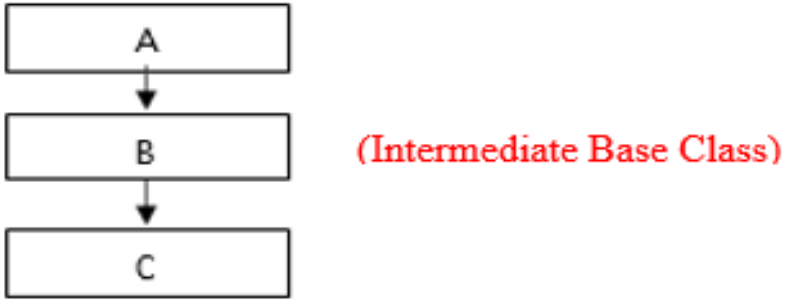
- Inheritance is **the process,** by which class can acquire(take) the **properties and methods of another class.**
- The mechanism of deriving a new class from an old class is called inheritance.
- The new class is called **derived** class and old class is called **base class.**
- The derived class may have all the features of the base class and the programmer can add new features to the derived class.

- **Types of Inheritance**
- Single Inheritance
- Multilevel Inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

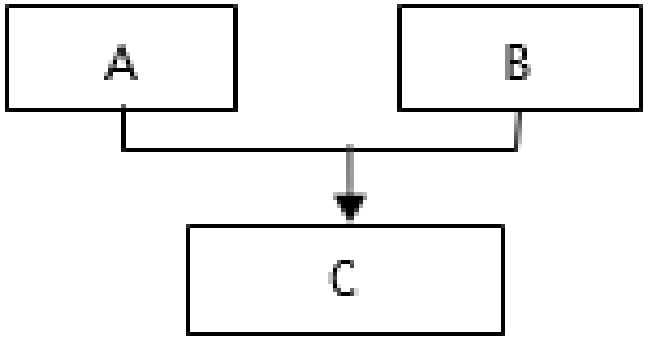
- **Single Inheritance**
- If a class is derived from a single class then it is called single inheritance.
- Class B is derived from class A



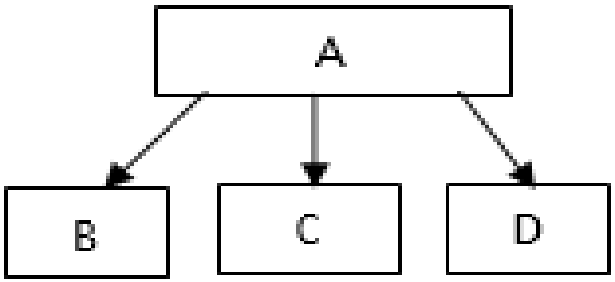
- **Multilevel Inheritance**
- A class is derived from a class which is derived from another class then it is called multilevel inheritance
- Here, class *C* is derived from class *B* and class *B* is derived from class *A*, so it is called multilevel inheritance.



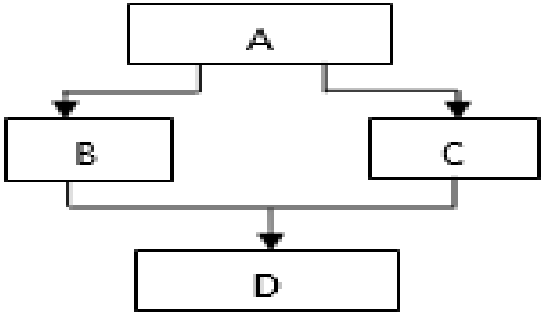
- **Multiple Inheritance**
- If a class is derived from more than one class then it is called multiple inheritance.
- Here, class C is derived from two classes, class A and class B



- **Hierarchical Inheritance**
- If one or more classes are derived from one class then it is called hierarchical inheritance.
- Here, class B, class C and class D are derived from class A.



- **Hybrid Inheritance**
- It is a combination of any above inheritance types. That is either multiple or multilevel or hierarchical or any other combination.
- Here, class *B* and class *C* are derived from class *A* and class *D* is derived from class *B* and class *C*.
- Class *A*, class *B* and class *C* is example of Hierarchical Inheritance and class *B*, class *C* and class *D* is example of Multiple Inheritance so this hybrid inheritance is combination of Hierarchical and Multiple Inheritance.



- **Example in document**

Derived Class

- A class can be derived from more than one classes, which means it can inherit data and functions from multiple base classes.
- **Syntax**: **class derived-class: access-specifier base-class**

```
// 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;
}
```

Abstract class

- A class from which we would never want to create objects is called an **abstract class.** Such a class exist only as a parent for the derived classes.
- Anybody who tries to create an object from such a base class would be generate an error by compiler and give complain that you are trying to create object of the abstract class.
- Abstract classes are essential to providing an abstraction to the code to make it reusable and extendable.

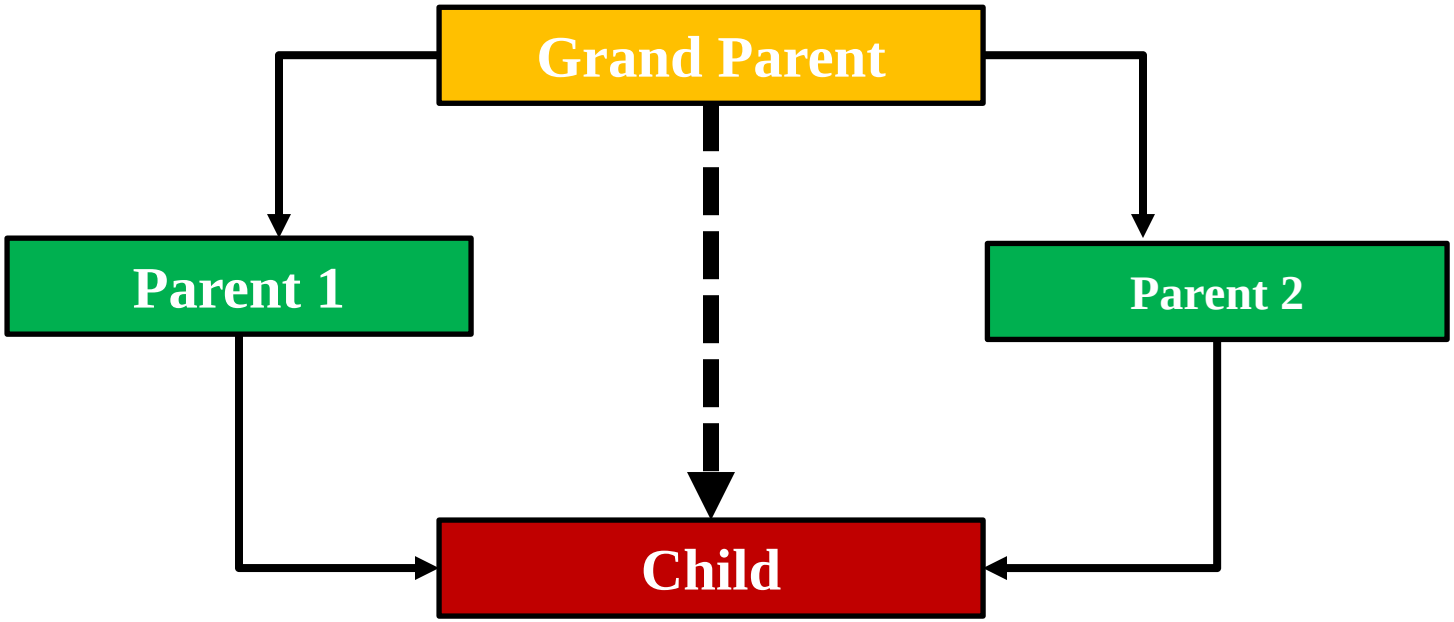
- **Example:** A Vehicle parent class with **Truck** and **Motorbike** inheriting from it is an abstraction that easily allows more vehicles to be added. However, even though all vehicles have wheels, not all vehicles have the same number of wheels – this is where a pure virtual function is needed.

Example:

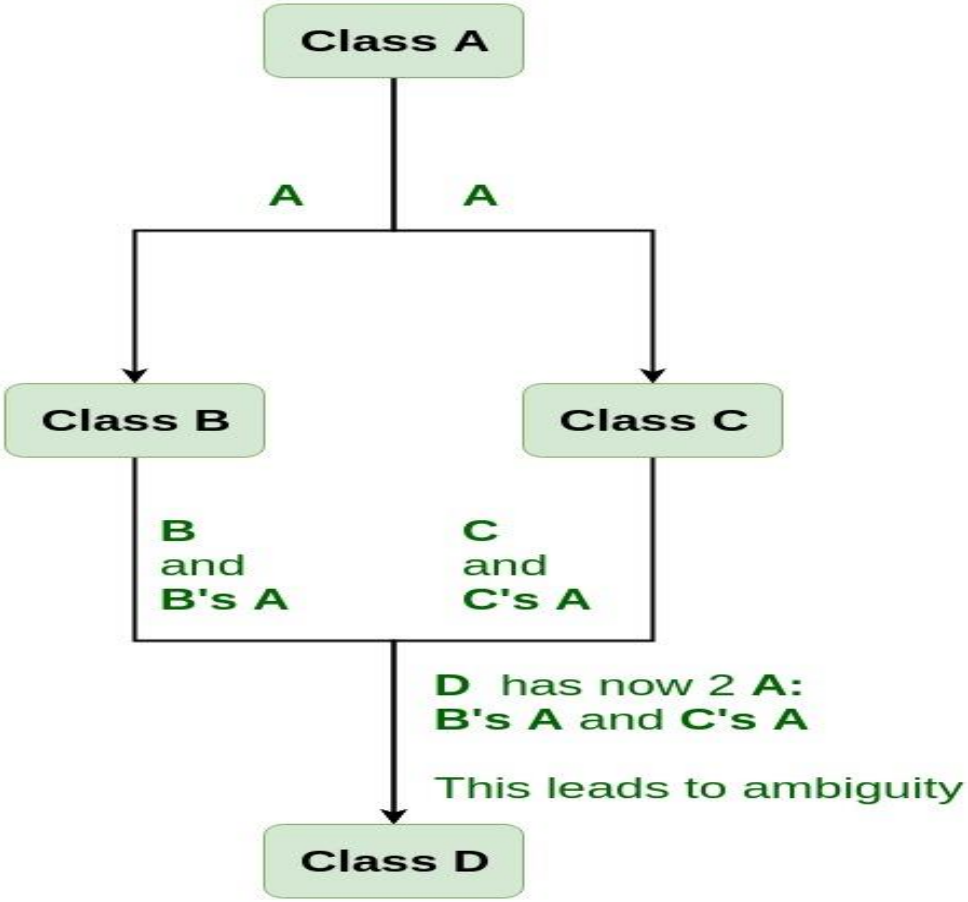
```
#include <iostream.h>
#include <conio.h>
abstract class A
{
      void disp()
      {
            cout<<"abstract class";
      }
};
class B:public A
{
      public:
      void display()
      {
            cout<<"derived class";
      }
};
void main()
{
      B bb;
      clrscr();
      bb.disp();
      bb.display();
      getch();
}
```

Virtual base class

- Consider a situation where all the three kinds of inheritance , namely multilevel, multiple and hierarchical inheritance are involved. This situation shown below:



- **Need for Virtual Base Classes:**
- Consider the situation where we have one class **A** .This class is **A** is inherited by two other classes **B** and **C**. Both these class are inherited into another in a new class **D** as shown in figure below.



- As we can see from the figure that data members/function of class **A** are inherited twice to class **D**. One through class **B** and second through class **C**. When any data / function member of class **A** is accessed by an object of class **D**, **ambiguity(Difficulty) arises as to which data/function member would be called?** One inherited through **B** or the other inherited through **C**. **This confuses compiler and it displays error.**

- **How to resolve this issue?**
- To resolve this ambiguity when class **A** is inherited in both class **B** and class **C**, it is declared as **virtual base class** by placing a keyword **virtual** as :
- **Syntax for Virtual Base Classes:**

Syntax 1:

```
class B : virtual public A
{
};
```

Syntax 2:

```
class C : public virtual A
{
};
```

Thank You