Durvesh N. Patil 12/10/2021

2019BTEEN00035 T.Y. B.Tech (Batch: E4)

Object Oriented Programming

Exp. 7 : Inheritance in OOP

(a) Inheritance Part - 1 :

Using a library to create the ~~the~~ derived classes, public inheritance, accessing the base class members, constructors & destructors in derived and base class.

→

Library :

(a) publication.h

```
using namespace std;

class Publication
{
  protected:
        float price;
        string name;
  public:
        Publication ()
        {
            cout<<"Base class constructor is called"<<endl;
        }
        ~Publication()
        {
            cout<<" Base class destructor is called "<<endl;
        }
        void setData();
        void putData();
};
```

ⓑ publication.cpp

```
#include<iostream>
#include <string>
# include "publication.h"
using namespace std;

void  Publication :: setData()
{
   cout <<"\nEnter name of publication:" ;
   cin >> name;
   cout << "\nEnter price:";
   cin >> price;
}

void  Publication :: putData()
{
    cout <<"\nPublication Name:" << name << endl;
    cout <<"\nPublication Price: " << price << endl;
}
```

Now, create book.cpp as console mode & from parameters → Linker → add the required library.

ⓒ book.cpp

```
# include <iostream>
# include < string>
# include "publication.h"

using namespace std;
```

```
class Book : public Publication
{
  public:
        Book ( )
        {
            cout <<"\n Constructor of derived class is called ;
        }

        ~Book ( )
        {
            cout <<"\n Destructor of derived class is called ;
        }

        string book_name ;
        int  total_pages ;

        void setBookData ( )
        {
            cout <<"\n Enter book name: " ;
            cin >> book_name ;
            cout <<"\n Enter total pages: " ;
            cin >> total_pages ;
        }

        void putBookData ( )
        {
            cout <<"\n Book Name: " << book_name ;
            cout << "\n Total pages: " << total_pages ;
        }
 };
```

```
int  main()
{
     Book  b;
     b.setData();
     b.setBookData();
     b.putData();
     b.putBookData();

     return 0;
}
```

<u>Output</u>:

```
Constructor of base class is called
Constructor of derived class is called.
Enter name of publication: McGraw Hill
Enter price: 500
Enter book name: Indian Economy
Enter total pages: 680
Publication Name : McGraw Hill
Publication Price : 500
Book Name: Indian Economy
Total Pages: 680
Destructor of derived class is called
Destructor of base class is called.
```

M T W T F S S
Page No.: YOUVA
Date:

⊙ Inheritance Part - 2 :

ⓐ Private Inheritance

```
#include <iostream>
using namespace std;

class Base
{
  private:
        int pvt = 1;
  protected:
        int prot = 2;
  public :
        int pub = 3;
        int getPvt()
        {
            return pvt;
        }
};

class Derived : private Base
{
  public :
        int getProt()
        {
            return prot;
        }

        int getPub()
        {
            return pub;
        }
};
```

```
int main()
{
   Derived obj;
   cout << "Protected: " << obj.getProt() << endl;
   cout << "Public: " << obj.getPub() << endl;

   return 0;
}
```

Output : Protected = 2
         Public = 3

(b) Multiple Inheritance

class can inherit from more than one classes.

```
#include <iostream>
using namespace std;

class Vehicle
{
   public:
      Vehicle()
      {
         cout << "This is Vehicle"<< endl;
      }
};


class FourWheeler
{
   public:
      FourWheeler()
      {
       cout << "This is a 4 wheeler vehicle << endl;
      }
};
```

Scanned with CamScanner

```
class Car : public Vehicle, public fourWheeler
{
    Public:
        Car ()
        {
          cout << "This is car" << endl;
        }
};

int main()
{
    Car obj;
    return 0;
}
```

Output : This is a vehicle
         This is a 4 wheeler Vehicle
         This is a car.

© Hierarchical Inheritance

More than one derived class is created from a single base class

```
#include <iostream>
using namespace std;

class Vehicle
{
  public:
     Vehicle()
     {
       cout << "This is a vehicle" << endl;
     }
};
```

```
class Car : public Vehicle
{
};

class Bus : public Vehicle
{
};

int main()
{
    Car c;
    Bus b;
    return 0;
}
```

Output : This is a vehicle
         This is a vehicle

ⓓ Hybrid Inheritance :

It is implemented by combining more than one type of inheritance.

```
#include <iostream>
using namespace std;

class Vehicle
{
   public :
       Vehicle()
       {
          cout << "This is a vehicle " << endl;
       }
};
```

```
class fare
{
  public:
      fare ()
      {
         cout <<"fare of vehicle" <<endl;
      }
};

class Car : public Vehicle
{
};

class Bus : public Vehicle, public fare
{

};

int main()
{
    Bus b;
    return 0;
}
```

output : This is a Vehicle
             fare of Vehicle.

ⓔ Ambiguity in inheritance :

Ambiguity caused due to multipath inheritance. This problem is solved by making the common base class as virtual base while declaring classes during inheritance.

```
#include <iostream>
using namespace std;

class A
{
  public :
        void show()
        {
          cout << "A" << endl;
        }
};
class B
{
  public :
        void show()
        {
          cout << "B" << endl;
        }

};

class C : public A, public B
{
};

int main()
{
    C obj ;
    obj. A :: show();
    obj. B :: show();
    return 0;
}
```

<u>Output</u> :

```
A
B
```

⊙ Questions :

1. Which is the correct syntax of inheritance ?
⇒ class derived_classname : access_mode base_classname
{ /* define class body */ };

2. While inheriting a class, if no access mode is specified then which among the following is true ?
⇒ It gets inherited privately by default.

3. If a derived class object is created, which constructor is called first ?
⇒ Base class constructor.

4. The private members of the base class are visible in derived class but are not accessible directly.
⇒ True.

5. If base class has constructor with arguments, then it is mandatory for the derived class to have constructor and pass the arguments to base class constructor.

6. Inheritance allows in C++ Program ?
⇒ All :
i) class Re-usability
ii) creating hierarchy of classes
iii) extendibility.