NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

CL 217 – Object Oriented Programing

Lab 06

Outline

- Inheritance
- Types of Inheritance with Respect to Base Class Access Control
- Examples
- Exercise

Object Relationship

- Object oriented programming generally support 4 types of relationships that are:

Inheritance

- Is-a relationship

Composition

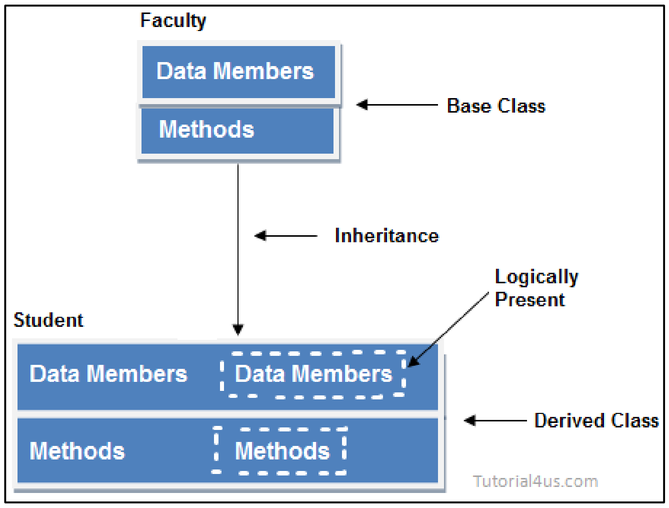
- Part-of relationship

Aggregation and Association

- has-a relationship

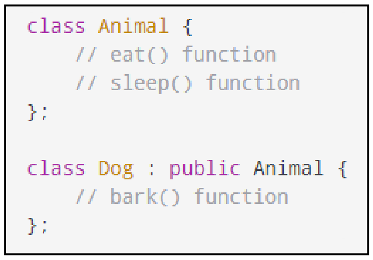
Inheritance

- The technique of deriving a new class from an old one is called inheritance
- Capability of a class to derive properties and characteristics from another class.
- The extended (or child) class contains all the features of its base (or parent) class, and may additionally have some unique features of its own.



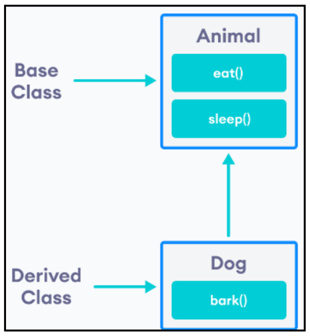
Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.
Super Class: The class whose properties are inherited by sub class is called Base Class or Super class

For example



```
class Animal {
    // eat() function
    // sleep() function
};

class Dog : public Animal {
    // bark() function
};
```



Program as Example

```
#include <bits/stdc++.h>
using namespace std;
 //Base class
class Parent
{    public:
     int id_p;
};
 // Sub class inheriting from Base
      Class(Parent)
class Child : public Parent
{
        public:
     int id_c; };
```

```
//main function
int main()
   {
          Child obj1;
           // An object of class child has all data members
        // and member functions of class parent
        obj1.id_c = 7;
        obj1.id_p = 91;
        cout << "Child id is " <<  obj1.id_c << endl;
        cout << "Parent id is " <<  obj1.id_p << endl;
              return 0; }
```

Program as Example

```
#include <iostream>
using namespace std;
class Animal {
 public:
   void eat()
  { cout << "I can eat!" << endl; }
void sleep()
  { cout << "I can sleep!" << endl; } };
```

```
// derived class
class Dog : public Animal {
 public:
       void bark()
       { cout << "I can bark!
Woof    woof!!" << endl; }
};
```

```
int main() {
// Create object of the Dog class
 Dog  dog1;
// Calling members of the base
class dog1.eat();
dog1.sleep();
// Calling member of the derived
class
```

Output

```
I can eat!
I can sleep!
I can bark! Woof woof!!
```

Implementing inheritance in C++

- Syntax:
  class subclass_name : access_mode base_class_name
  {//body of subclass};

Mode of Inheritance

- **Public mode**: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.
- **Protected mode**: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.
  Private mode: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class

```
class A
{
public:
    int x;
protected:
    int y;
private:
    int z;
};
```

```
class B : public A
{
    // x is public
    // y is protected
    // z is not accessible
from B
};
```

```
class C : protected A
{
    // x is protected
    // y is protected
    // z is not accessible from C
};
```

```
class D : private A    //
'private' is default for classes
{
    // x is private
    // y is private
    // z is not accessible from D
};
```

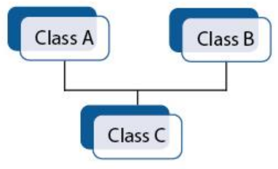
Single Inheritance

```
class Person
{
            char name[100],gender[10];
            int age;
public:
            void getdata(){
                        cout<<"Name: ";
                        cin>>name;
                        cout<<"Age: ";
                        cin>>age;
                        cout<<"Gender: ";
                        cin>>gender;}
void display(){
            cout<<"Name: "<<name<<endl;
            cout<<"Age: "<<age<<endl;
            cout<<"Gender:
"<<gender<<endl;
}};
class Employee: public Person
{
            char company[100];
            float salary;
public:
```

```
            void getdata()
            {
            Person::getdata();
            cout<<"Name of Company: ";
            cin>>company;
            cout<<" Salary: Rs.";
            cin>>salary;
}
void display()
{
Person::display();
            cout<<"Nameof
Company:"<<company<<endl;
cout<<"Salary: Rs."<<salary<<endl;
}};
int main(){
            Employee emp;
            cout<<"Enter data"<<endl;
            emp.getdata();
            cout<<endl<<"Displaying
data"<<endl;
            emp.display();
R           eturn 0;}
```

Multiple Inheritance

- Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one sub class is inherited from more than one base classes



Multiple Inheritance

- **Syntax:**

class subclass_name : access_mode base_class1, access_mode base_class2, ...{//body of subclass};

```
class stud {
  protected:
    int roll, m1, m2;
  public:
      void get()
      { cout << "Enter the Roll No.: ";
      cin >> roll;
      cout << "Enter the two highest marks: ";
      cin >> m1 >> m2; } };
class extracurriculam {
    protected:
      int xm;
     public:
      void getsm()
     { cout << "\nEnter the mark for Extra
Curriculam Activities: "; cin >> xm; } }
```

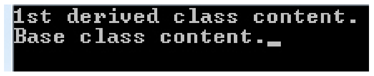
```
class output : public stud, public
extracurriculam {    int tot, avg;
public:
      void display() {
           tot = (m1 + m2 + xm);
           avg = tot / 3;
           cout << "\n\n\tRoll No : "
 << roll <<  "\n\tTotal : " << tot;
           cout << "\n\tAverage : " << avg; } };
int main() {
     output O;
     O.get();
     O.getsm();
     O.display(); }
```

Multilevel Inheritance

- In this type of inheritance, a derived class is created from another derived class.

```
#include <iostream>
using namespace std;
class base {
  public:
    void display1()
    { cout << "\nBase class content."; } };
 class derived : public base {
   public:
    void display2()
    { cout << "1st derived class content."; }
};
```

```
class derived2 : public derived
{ void display3()
{ cout << "\n2nd Derived class content."; }
};
int main()
{ derived2 D;
   //D.display3();
   D.display2();
   D.display1(); }
```



```
1st derived class content.
Base class content._
```

Hierarchical Inheritance

- In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.

```
include <iostream>
#include <string.h>
using namespace std;
class member {
    char gender[10];
    int age;
public:
    void get()
    {
        cout << "Age: "; cin >> age;
        cout << "Gender: "; cin >>
gender;
    }
    void disp()
    {
        cout << "Age: " << age <<
endl;
        cout << "Gender: " <<
gender << endl;
    }};
```

```
class stud : public member {
    char level[20];
public:
    void getdata()
    {
        member::get();
        cout << "Class: "; cin >>
level;
    }
    void disp2()
    {
        member::disp();
        cout << "Level: " << level
<< endl;
    }
};
```

```
class staff : public member {
    float salary;
public:
    void getdata()
    {
        member::get();
        cout << "Salary: Rs."; cin
>> salary;
    }
    void disp3()
    {
        member::disp();
        cout << "Salary: Rs." <<
salary << endl;
```

```
int main()
{
    member M;
    staff S;
    stud s;
    cout << "Student" << endl;
    cout << "Enter data" << endl;
    s.getdata();
    cout << endl
        << "Displaying data" << endl;
    s.disp();
    cout << endl
        << "Staff Data" << endl;
    cout << "Enter data" << endl;
    S.getdata();
    cout << endl
        << "Displaying data" << endl;
    S.disp();
}
```

```
Student
Enter data
Age: 12
Gender: Female
Class: 10
Displaying data
Age: 12
Gender: Female
```

Output

```
Staff Data
Enter data
Age: 12
Gender: male
Salary: Rs.100000
Displaying data
Age: 12
Gender: male
```

Hybrid (Virtual) Inheritance

- Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.

```
// base class
class Vehicle
{
  public:
    Vehicle()
    {
      cout << "This is a Vehicle"
<< endl;
    }
};
//base class
class Fare
{
    public:
    Fare()
    {
        cout<<"Fare of Vehicle\n";
    } };
```

```
// first sub class
class Car: public Vehicle
{
};
// second sub class
class Bus: public Vehicle,
public Fare
{
};
```

```
// main function
int main()
{
    // creating object of sub class
will
    // invoke the constructor of
base class
    Bus obj2;
    return 0;
}
```

```
This is a Vehicle
Fare of Vehicle
```

Order of constructor and Destructor call

```
#include <iostream>
using namespace std;
// base class
class Parent
{
    public:
    // base class
constructor
    Parent()
    {
        cout << "Inside
base class" << endl;
    }
```

```
// sub class
class Child : public
Parent {
    public:
    //sub class constructor
    Child()
    { cout << "Inside sub
class" << endl;
    }
};
```

```
// main function
int main() {

    // creating object of sub
class
    Child obj;
    return 0;
}
```

Output

```
Inside base class
Inside sub class
```

Concept: Calling parameterized constructor of base class in derived class constructor!

- To call the parameterized constructor of base class when derived class’s parameterized constructor is called, you have to explicitly specify the base class’s parameterized constructor in derived class

Concept: Calling parameterized constructor of base class in derived class constructor!

Program as Example

```
class Base
{
   int x;
   public:
   // parameterized constructor
   Base(int i)
   {
      x = i;
      cout << "Base Parameterized
Constructor\n";
   }
};
```

```
class Derived : public Base{
   int y;
   public:
   // parameterized constructor
   Derived(int j):Base(j)
   {
      y = j;
      cout << "Derived Parameterized
Constructor\n";
   }
};
```

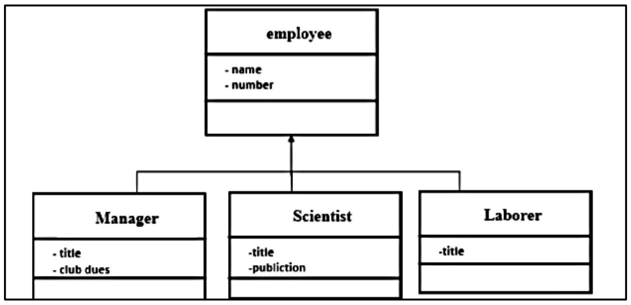
```
int main()
{
   Derived d(10) ;
}
```

Output

```
Base Parameterized Constructor
Derived Parameterized Constructor
```

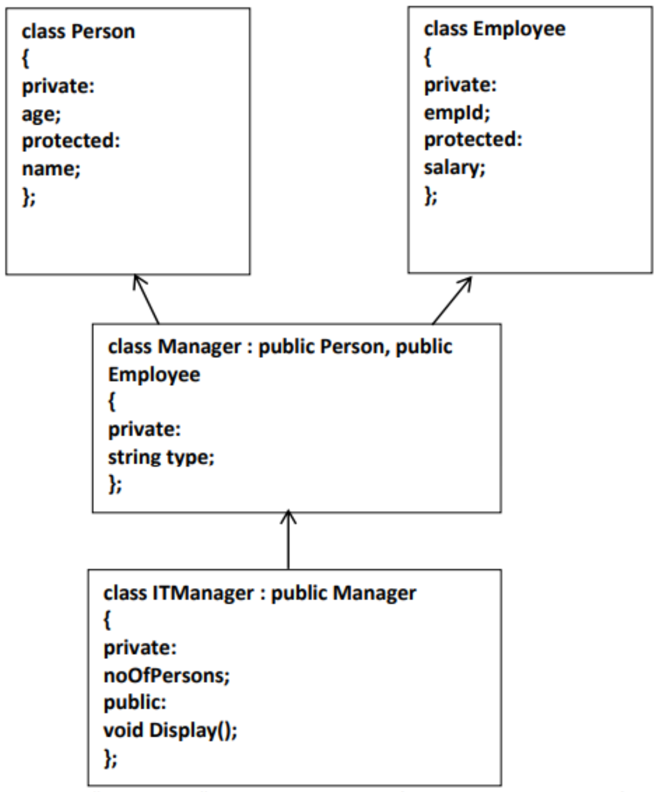
Activity

1. Make a class named Fruit with a data member to calculate the number of fruits in a basket. Create two other class named Apples and Mangoes to calculate the number of apples and mangoes in the basket. Print the number of fruits of each type and the total number of fruits in the basket.
2. Write a C++ program to define a base class Item (item-no, name, price). Derive a class Discounted-Item (discount-percent). A customer purchases 'n' items. Display the item-wise bill and total amount using appropriate format.
3. In the database only three kinds of employees are represented. Managers manage, scientists perform research to develop better widgets, and laborers operate the dangerous widget-stamping presses.
4. The database stores a name and an employee identification number for all employees, no matter what their category is. However, for managers, it also stores their titles and golf club dues. For scientists, it stores the number of scholarly articles they have published and their title. Laborers store the title only. You must start with a base class employee. This class handles the employee’s last name and employee number. From this class three other classes are derived: manager, scientist, and laborer. All three classes contain additional information about these categories of employee, and member functions to handle this information as shown in figure.



5. The Display() function in "ITManager" should be capable of displaying the values of all the data members declared

- in the scenario (age,name,empId,salary,type,noOfPersons) without being able to alter thevalues.
- The "int main()" function should contain only three program statements which are as follows:
- In the first statement, create object of "ITManager" and pass the values for all the datamembers:
- ITManager obj(age,name,empId,salary,type,noOfPersons);
- In the second statement, call the Display () function.
- In the third statement, return 0.



```
class Person
{
private:
age;
protected:
name;
};
```

```
class Employee
{
private:
empId;
protected:
salary;
};
```

```
class Manager : public Person, public
Employee
{
private:
string type;
};
```

```
class ITManager : public Manager
{
private:
noOfPersons;
public:
void Display();
};
```