Dr. D. Y. Patil Pratishthan's

Institute for Advanced Computing and Software Development

Sub- c++
Day5

Friend Function

- Non-member functions of a class will not have access to the private data of another class.
- There could be situations where we want two classes to share some functions and the data members.
- In that case, we can make the function a friend of these classes, and that will enable the function to access the private and protected data members of the classes.
- A friend function is a function that is specified outside a class but has the ability to access the class members' protected and private data.

- A friend can be a member's function, function template, or function, or a class or class template, in which case the entire class and all of its members are friends.

Features

- A friend function does not fall within the scope of the class for which it was declared as a friend. Hence, functionality is not limited to one class.
- The friend function can be a member of another class or a function that is outside the scope of the class.
- A friend function can be declared in the private or public part of a class without changing its meaning.
- Friend functions are not called using objects of the class because they are not within the class's scope.

- Without the help of any object, the friend function can be invoked like a normal member function.
- Friend functions can use objects of the class as arguments.
- A friend function cannot explicitly access member names directly. Every member name has to use the object's name and dot operator .
- Syntax

```
class className{

   // Other Declarations

   friend returnType functionName(arg list);

};
```

Case 1: Outsider function as Friend of class

Case 2 : member function of one class will be friend of another class

case 3: one whole class will be friend of other class

```
 class className1{
    // Other Declarations
    int functionName1(); // member function of
 className1
 };
class className2
{
   // Other Declarations
   friend int className1::functionName1();
    //The functionName1() is a friend of
className2
};
```

Friend Class

- A friend class can have access to the data members and functions of another class in which it is declared as a friend.
- They are used in situations where we want a certain class to have access to another class’s private and protected members.
- Classes declared as friends to any another class will have all the member functions become friend functions to the friend class.
- Friend functions are used to work as a link between the classes.

- Syntax of friend class:

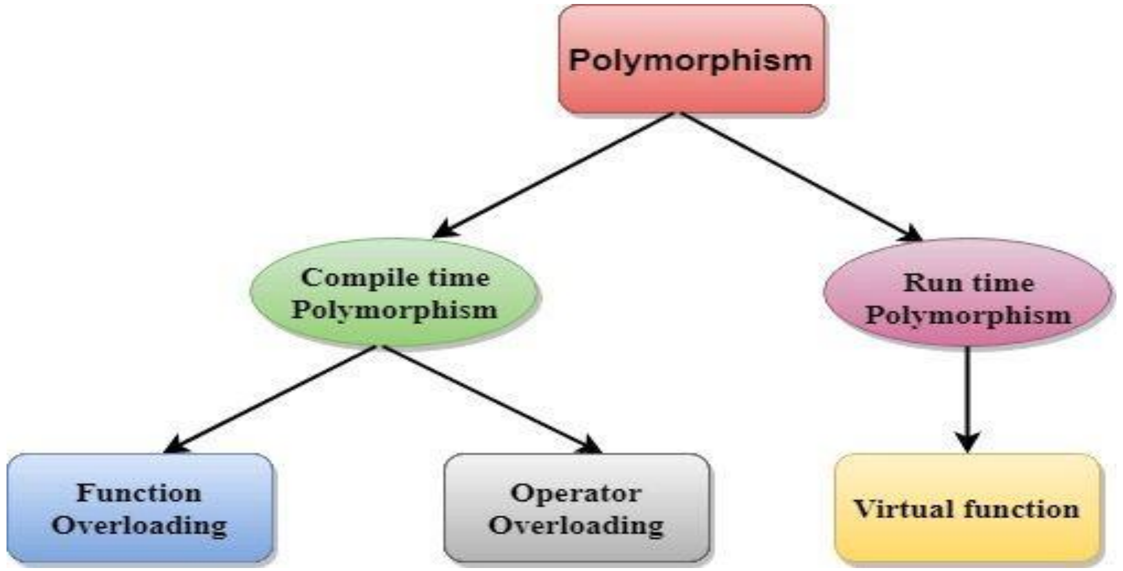
```
class S; //forward declaration

class P{
  // Other Declarations
  friend class S;
};

class S{
  // Declarations
};
```

Polymorphism

The term "Polymorphism" is the combination of "poly" + "morphs" which means many forms. It is a greek word. In object-oriented programming, we use 3 main concepts: inheritance, encapsulation, and polymorphism.



Operator Overloading

The mechanism of giving special meaning to an operator is known as operator overloading.

For example, we can overload an operator ‘+’ in a class like string to concatenate two strings by just using +.

Implementation of Operator overloading:

1. Member function: It is in the scope of the class in which it is declared.

2.Friend function: It is a non-member function of a class with permission to access both private and protected members.

Operator Overloading

Using Friend function

Program to overload + operator using friend function

- At least one of the operands in overloaded operators must be user-defined, which means we cannot overload the minus operator to work with one integer and one double. However, you could overload the minus operator to work with an integer and a mystring.
- It is not possible to change the number of operands of an operator supports.
- All operators keep their default precedence and associations (what they use for), which cannot be changed.
- Only built-in operators can be overloaded.

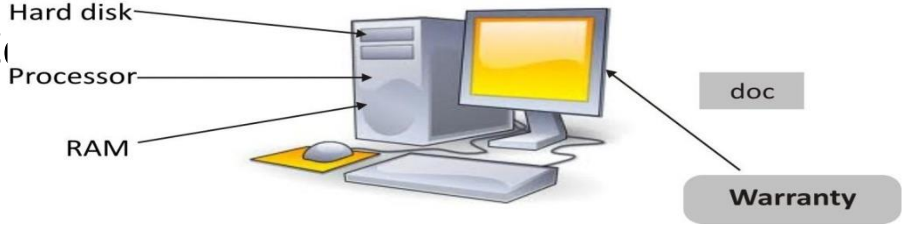
Syntax

```
Ret_Type operator Symbol(parametrs)
{

}
```

Containment

- One object may contain another as a part of its attribute
  - Document contains sentences which contain words.
  - Computer system has a hard disk, processor, RAM, mouse, monitor, etc.
- Containment need not be physical
  - E.g. C



Containment:

- We can create an object of one class into another and that object will be a member of the class.
- This type of relationship between classes is known as **containership** or **has_a** relationship as one class contain the object of another class.
- Containment represents ‘has a’ or ‘is a part of’ relationship.
- Containment relationship depicts how an object may be a part of another object.
  - For example,

    Engine, wheels, etc. are a part of a car.

    Pan card is required to open a bank account
- Container relationship brings reusability of code.
  - For example,

    Engine is also used in an airplane.
    Pan card is also used in I-T returns.

Class `cEmployee`

```
class cDate {
private:

   int mDay, mMon, mYear;
public:
   cDate();
   cDate(int, int, int);
   void display();
   . . .
   void setDay(int);
};
```

```
// container class
class Employee
{
protected: // accessible in derived classes
  int mId;
  int mBasicSal;
  Date mBdate;  // contained object
public :
  Employee(){   // no-argument c'tor
    mId = 0;
    mBasicSal = 0;
  }
  cEmployee(int, int,  int, int, int);
  void display();
};
```

Constructor for the Class `cEmployee`

```
Employee::Employee(int i, int sal,

int d, int m, int y){
    mId = i;
    mBasicSal = sal;

    mBdate = Date(d, m, y);
}
```

Uses constructor conversion function to convert built-in type to class-type

```
int main()
{
  Employee e1( 1, 10000,17,01,1970);
  return 0;
}
```

Calls parameterized constructor

Inheritance

- One of the key-concepts of object-oriented approach.
- Allows creation of hierarchical classification.
- In C++, inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviors which are defined in other class.
- In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

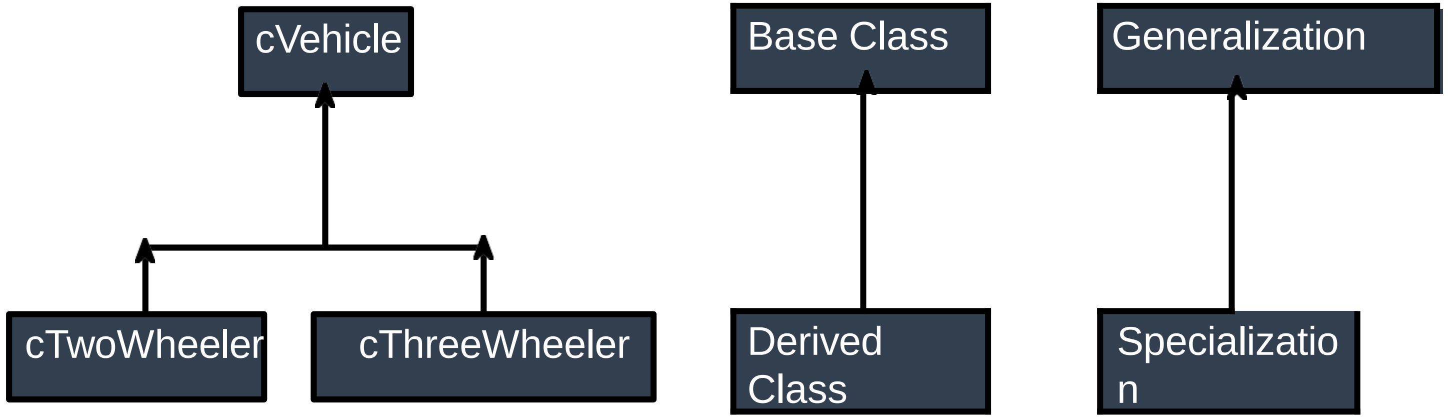
- Advantages of Inheritance
    - Reusability
    - Extensibility

Containment Vs Inheritance

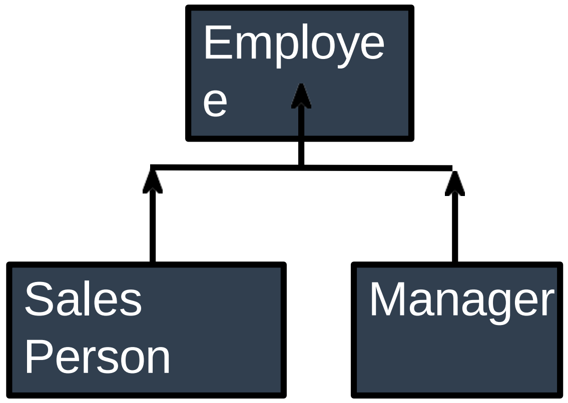
- Containment is used:
  - When the features of an existing class are wanted inside a new class, but not its interface.
    - Computer system has a hard disk.
    - Car has an engine, chassis, steering wheel.
- Inheritance is used:
  - When it is necessary that the new type has to be the same type as the base class.
    - Computer system is an electronic device.
    - Car is a vehicle.

Base and Derived Class

Derived class inherits all data members and methods of its base class.



Base and Derived Classes - Example



- This is ‘is a’ kind of hierarchy.
- More than one class can inherit attributes from a single base class.
- A derived class can be a base class to another class.

- **C++ supports five types of inheritance:**
- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance
- Hybrid inheritance

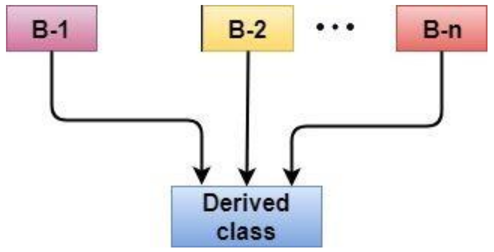
Single inheritance is defined as the inheritance in which a derived class is inherited from the only one base class.

Class B---> Class A

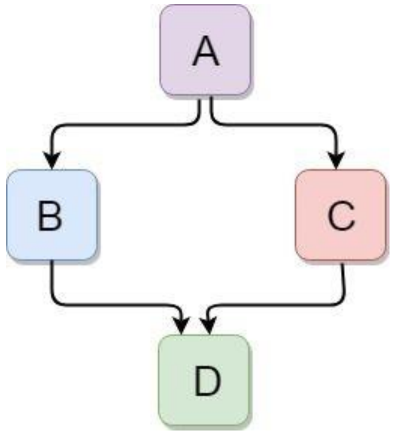
Multilevel inheritance is a process of deriving a class from another derived class.

Class C—>Class B—>Class A

Multiple inheritance is the process of deriving a new class that inherits the attributes from two or more classes.



- Hybrid inheritance is a combination of more than one type of inheritance.



- Hierarchical inheritance is defined as the process of deriving more than one class from a base class.