DSA
Lab Manual

Topic: Inheritance

Session: Spring 2023

Faculty of Information Technology

UCP Lahore Pakistan

Objectives:

To write a program that implements an important concept of OOP: Inheritance, Aggregation and Templates. Students are already familiar with these concepts; the purpose is revision and brining the flow of programming back.

Inheritance

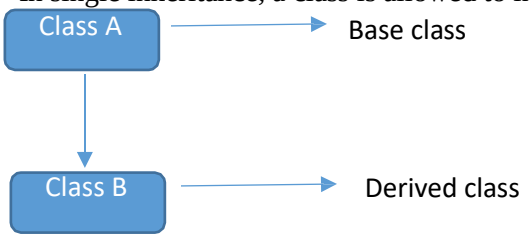
Inheritance is a feature of object oriented programming which allows classes to inherit common properties from other classes. For example, if there is a class such as 'vehicle', other classes like 'car', 'bike', can inherit common properties from the 'vehicle' class. This property helps you get rid of redundant code thereby reducing the overall size of the code.

Types of Inheritance

- Single Inheritance
- Multiple Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Single Inheritance

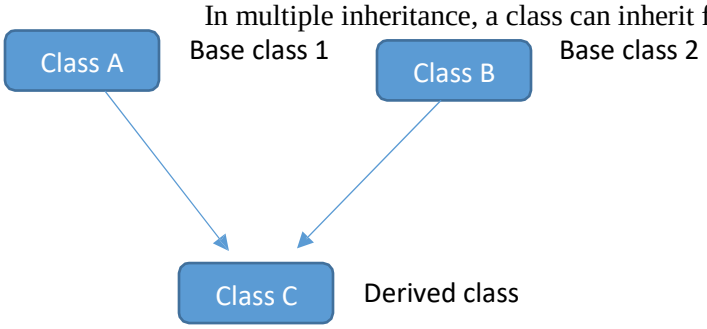
In single inheritance, a class is allowed to inherit from only one class.




```
Class DerivedClass_name : access_specifier Base_Class{};
```

Multiple Inheritance

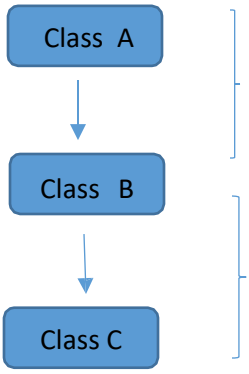
In multiple inheritance, a class can inherit from more than one class.




```
Class DerivedClass_name : access_specifier Base_Class1, access_specifier Base_Class2, ....{};
```

Multi-level inheritance

In multi-level inheritance, a derived class is created from another derived class.



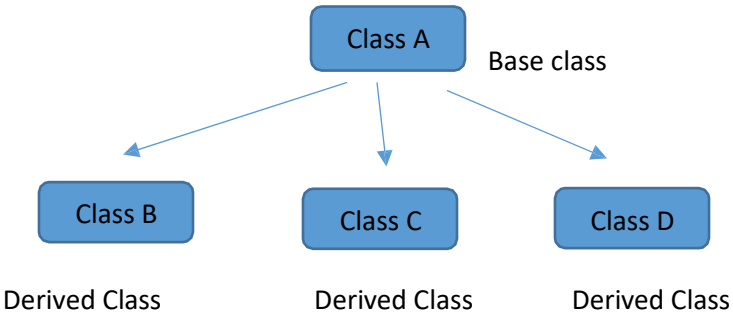
Class B is derived from class A

Class C is derived from Class B

```
class A {
members of Class A
};
class B: public A {
members of Class B
};
class C: public B {
members of Class C
};
```

Hierarchical inheritance

In Hierarchical inheritance, more than one sub-class is inherited from a single base class.

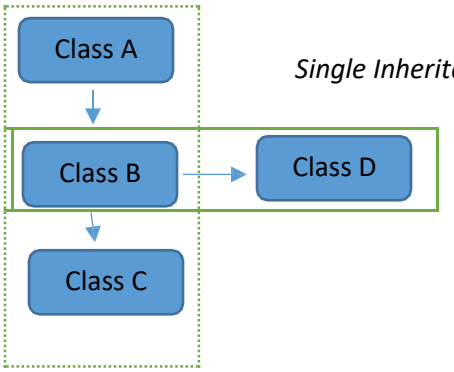



```
class A {
members of Class A
};
class B: public A {
members of Class B
};
class C: public A {
members of Class C
}; class
D: public A {
members of Class D
};
```

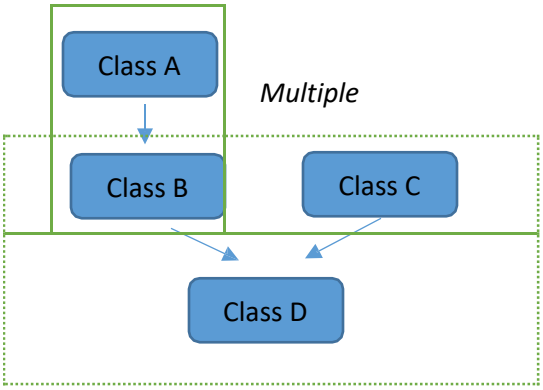
Hybrid Inheritance

Hybrid Inheritance is implemented by combining more than one type of inheritance.

Multi-level Inheritance



Single Inheritance




```
class A
{
Members of class A
};
class B: public A   // class B derived
from class A
{
Members of class B
};
class C: public B   // class C derived
from class B
{
Members of class C
};
class D: public B   // class D derived
from class B
{
Members of class D
};
```

```
class A
{
Members of class A
};
class B: public A // class B
derived from a single class A --
follows single inheritance
{
Members of class B
};
class C
{
Members of Class C
};
class D: public B, public
C
// class D derived from two
classes, class B and class C --
follows multiple inheritance
{
Members of class D
};
```

Aggregation
Aggregation is a type of association that is used to represent the "HAS-A" relationship between two objects.
Syntax:
Aggregation is a way to represent "HAS-A" relation between the objects of 2 individual classes. It is a subtype of association type of relation but more restrictive.

In the below syntax, the class2 represents the class that is a container class for other class1 that is contained in the object of the class2. Here each object of class2 holds a reference pointer to the object of the class1.

```
Class Class1{
//instance variables
//instance methods
}
class2{ Class1*
class1;
}
```

Templates
A feature, which allows us to write generic programs which makes a program shorter and manageable and allows a single class to work with different data types.
Types of templates:

- **Function Templates**

```
template<class type>ret-type func-name(parameter list)
          {
        //body of the function
           }
```

- **Class Templates**

```
template<class Ttype>
class class_name
{
 //class body;
}
```

Task 1

Using inheritance hierarchy used below, create four class and attributes and function as provided below.

1. # Sign shows Protected and – sign show private
2. Calculate Salary is used to calculate salary on different criteria defined as follows. Calculate Salary in Employee class just return 0.

Manager	**Salary**
General	Random (1000 to 10000)
Assistant	Random (20000 to 30000)
Marketing	Random (50000 to 80000)

Developer	**Salary**
Web	Random (100000 to 200000)
Mobile	Random (150000 to 200000)
Game	Random (100000 to 300000)

Task 2

Each derived class should have a function checkType() that will check the type of vehicle and set the Vehicles's typeOfVehcile based on the stored information about number of wheels (attribute). All types are based on number of wheels.

Bike:

Attributes:

- height : (double)
- selfStart : (bool)
- discBrake : (bool)
- numberOfBikes : static int //This is to store the number of available bikes

Car:

Attributes:

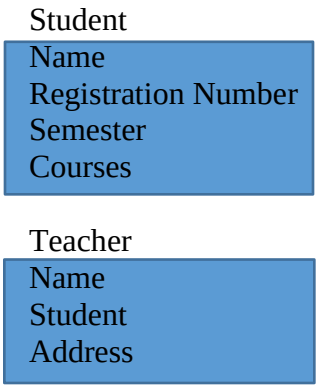
- noOfDoors : (int)
- trasmission : (character pointer) eg: automatic or manual
- noOfSeats : (int)
- numberOfCars : static int //This is to store the number of available cars

For each class above provide:

- Parameterized constructor with default arguments and Copy Constructor (use base initializer list with both constructors)
- Destructor (with no memory leakage)
- Getters/Setters for all private attributes (with no memory leakage and no returning of original memory handler)
- Assignment Operator (with no memory leakage)
- Don't provide function where you can overload operators, therefore must provide the following operators (only in child classes):
- cin>> and cout<< operators (Do not use friend keyword)

Task 3

Illustrate the aggregation relation between "student" and "Teacher" classes using C++ program.



Hint: Teacher is a container class and student is a class whose object is contained within the container class object.

Task 4

Get an array of size 7 from user, find the sum and arrange it in ascending order using function template.

Task 5

Make a menu-based program for basic arithmetic operation using class template

Press 1 for addition Press

2 for subtraction Press 3

for multiplication Press 4

for division

Note: Program should perform arithmetic functions for both data type: integer and float.