19

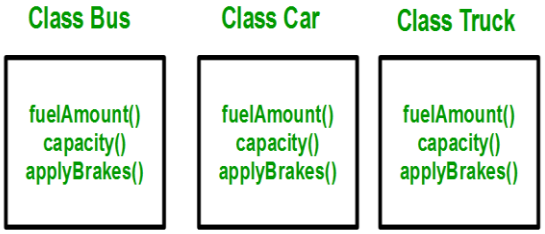
INHERITANCE- EXTENDING CLASSES

- **INHERITANCE:** The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important features of Object Oriented Programming.

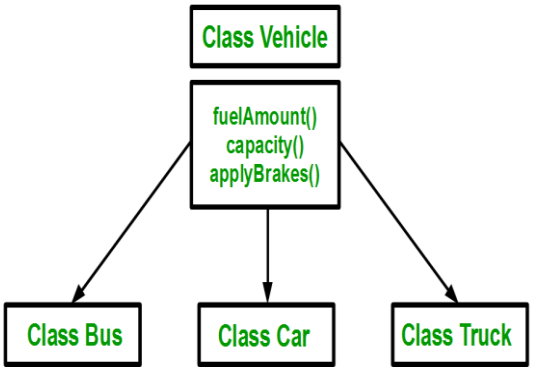
- **SUB CLASS:** The class that inherits properties from another class is called Sub class or Derived Class.

- **SUPER CLASS:** The class whose properties are inherited by sub class is called Base Class or Super class.

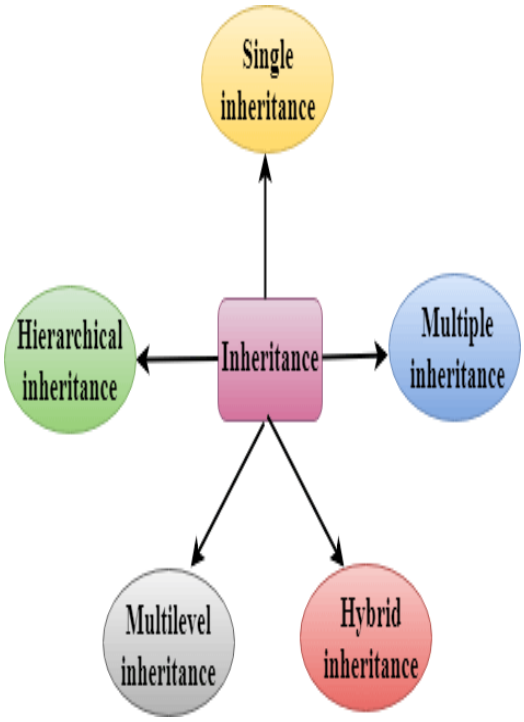
- **WHY AND WHEN TO USE INHERITANCE? :** Let's consider a group of vehicles named Bus, Car and Truck. The methods like **fuelAmount(), applyBrakes()** and **capacity()** will be the same for all three classes.



Using Inheritance:



- **TYPES OF INHERITANCE**:



- **IMPLEMENTING INHERITANCE**: **Syntax**:

Class subclass_name: access_mode base_class_name

{

//body of subclass

};

  - **Subclass_name –** It is the name of the sub class.
  - **Access_mode –** It is the mode in which you want to inherit this sub class such as **Private/Public/Protected**.

- **Base_class_name** - It is the name of the base class from which you want to inherit the sub class.

- **MODES OF INHERITANCE:**

Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

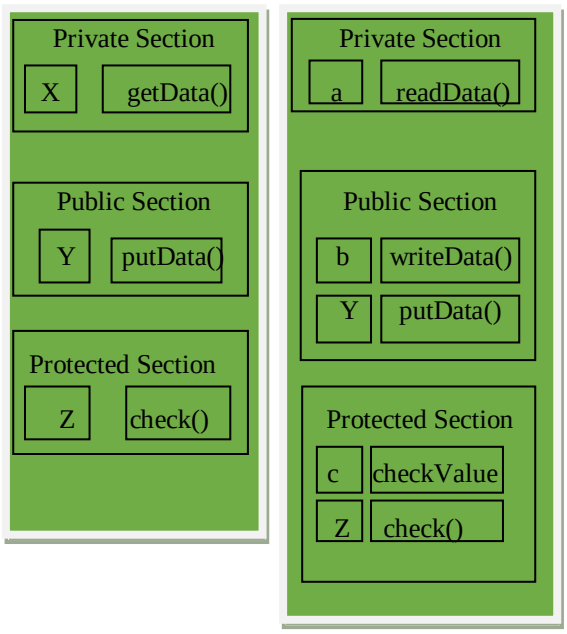
PROGRAM:

```
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
};
```

- **PUBLIC VISIBILITY MODE:**

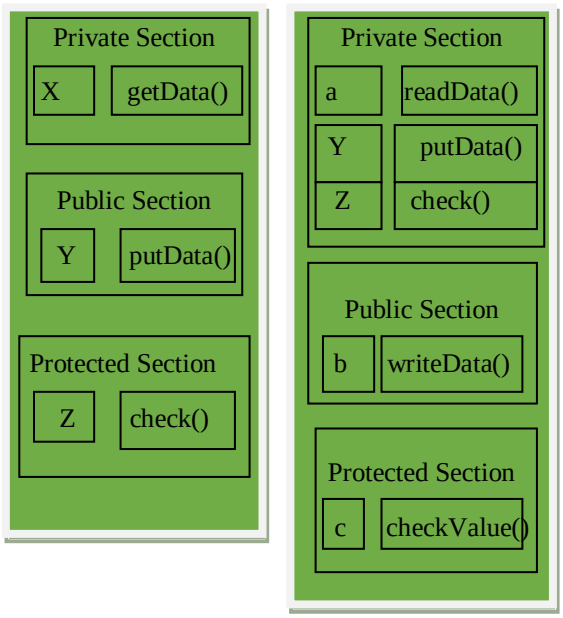
```
class student // base class
 { private :
     int x; // base class private members
void getdata ( );

public: //base class public  members
int y;
void putdata();

protected: //base class public  members
int z;
void check ( );

};

class marks : public student // class marks derived class
{
private :
 int a ;
void readdata ( );
public :
int b;
void writedata ( );
protected :
int c;
void checkvalue ( );
};
```

Class Student **Class marks**



- **PRIVATE VISIBILITY MODE:**

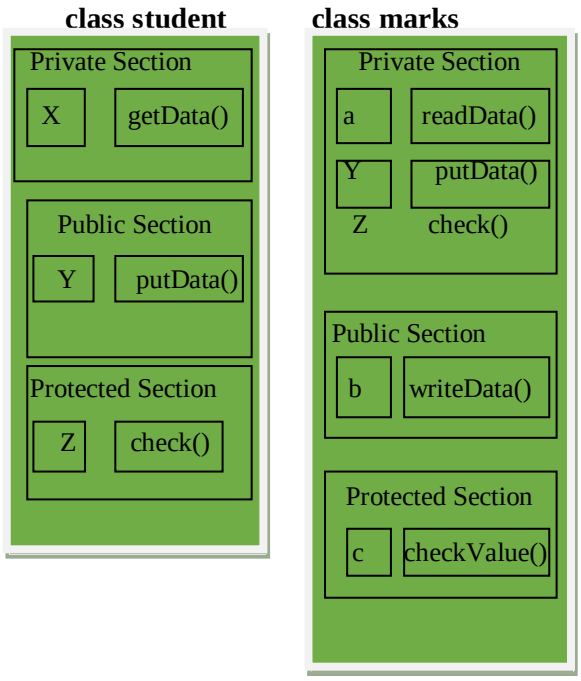
Class Student **Class marks**



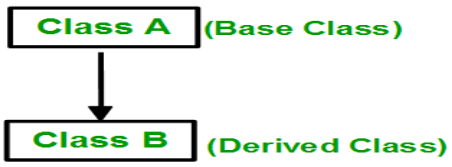
- **PROTECTED VISIBILITY MODE:**

A member declared as protected is accessible by the member functions of the class and its derived classes. It cannot be accessed by the member functions other than these classes.

class student **class marks**



- **SINGLE INHERITANCE:**



SYNTAX:

```
class subclass_name : access_mode
base_class
{
  //body of subclass
};
```

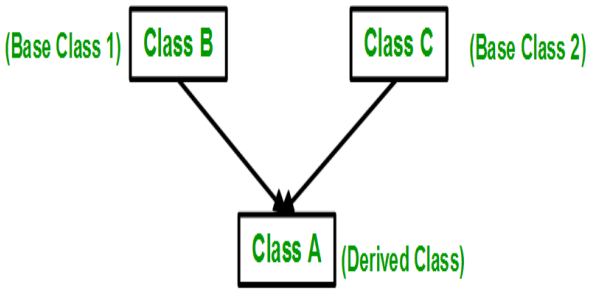
PROGRAM:

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

// sub class derived from a single base
classes
class Car: public Vehicle{


};

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

OUTPUT:

```
This is a Vehicle
```

- **MULTIPLE INHERITANCE:**



SYNTAX:

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

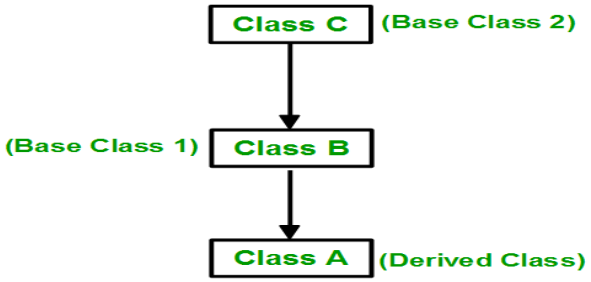
PROGRAM:

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

// first base class
class Vehicle {
  public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};

// second base class
class FourWheeler {
  public:
    FourWheeler()
    {
      cout << "This is a 4 wheeler Vehicle" <<
endl;
    }
};

// sub class derived from two base classes
class Car: public Vehicle, public
FourWheeler {

};

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

OUTPUT:

```
This is a Vehicle
This is a 4 wheeler Vehicle
```

- **MULTI-LEVEL INHERITANCE:**



PROGRAM:

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

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

// first sub_class derived from class vehicle
class fourWheeler: public Vehicle
{  public:
    fourWheeler()
    {
      cout<<"Objects with 4 wheels are
vehicles"<<endl;
    }
};
```

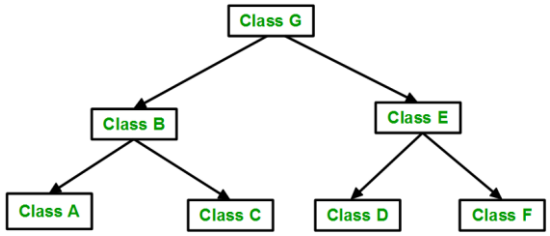
```
// sub class derived from the derived base
class fourWheeler
class Car: public fourWheeler{
  public:
    Car()
    {
      cout<<"Car has 4 Wheels"<<endl;
    }
};

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

OUTPUT:
This is a Vehicle

Objects with 4 wheels are vehicles

Car has 4 Wheels

- **HIERACHICAL INHERITANCE:**



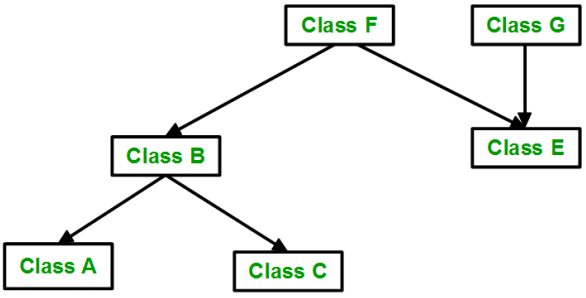
PROGRAM:

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

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

// first sub class
class Car: public Vehicle
{

};

// second sub class
class Bus: public Vehicle
{

};

// main function
int main()
{
   // creating object of sub class will
   // invoke the constructor of base class
   Car obj1;
   Bus obj2;
   return 0;
}
```

OUTPUT:
This is a Vehicle

This is a Vehicle

- **HYBRID INHERITANCE:**



PROGRAM:

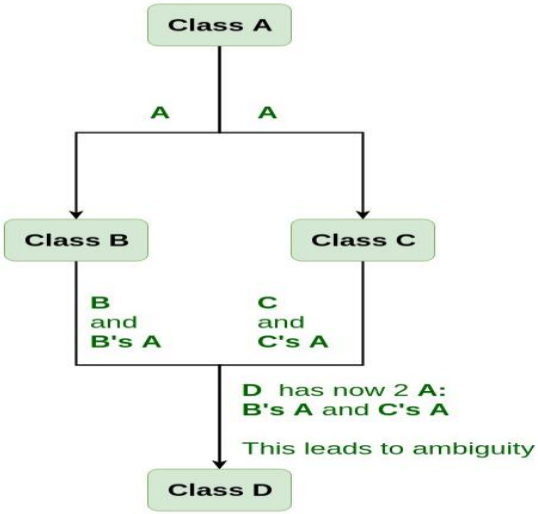
```
#include <iostream>
using namespace std;

// 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;
}
```

OUTPUT:
This is a Vehicle

Fare of Vehicle

- **VIRTUAL BASE CLASS:** Virtual base classes are used in virtual inheritance in a way of preventing multiple "instances" of a given class appearing in an inheritance hierarchy when using multiple inheritances.
  Let's consider the following situation.



CHECK YOURSELF

1. When the inheritance is private, the private methods in base class are _____ in the derived class (in C++).

   A) Inaccessible
   B) Accessible
   C) Protected
   D) Public

2. What is meant by multiple inheritance?

   A) Deriving a base class from derived class
   B) Deriving a derived class from base class
   C) Deriving a derived class from more than one base class
   D) None of the mentioned

3. Inheritance allow in C++ Program?

   A) Class Re-usability
   B) Creating a hierarchy of classes
   C) Extendibility
   D) All of the above

4. Can we pass parameters to base class constructor though derived class or derived class constructor?

   A) Yes
   B) No
   C) May Be
   D) Can't Say

5. In Multipath inheritance, in order to remove duplicate set of records in child class, we _____ .
A) Write Virtual function in parent classes
B) Write virtual functions is base class
C) Make base class as virtual base class
D) All of these

ANSWERS

Answers to Check Yourself:

1. A
2. C
3. D
4. A
5. C

STRETCH YOURSELF

1. Briefly explain about inheritance and types of inheritance.
2. What are the different accessibility modes ?
3. What do you mean by virtual base class and what is the use of it?