```
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:

In single inheritance, a class is allowed to inherit from only one class.
i.e. one sub class is inherited by one base class only.

Syntax:

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

```
// C++ program to explain
// Single inheritance
#include <iostream>
using namespace std;
```

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

// sub class derived from two 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: 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.

Syntax:

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

Here, the number of base classes will be separated by a comma (‘, ‘) and access mode for every base class must be specified.

```
// C++ program to explain
// multiple inheritance
#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
```

Please visit this link to learn multiple inheritance in details.

Multilevel Inheritance: In this type of inheritance, a derived class is created from another derived class.

```
// C++ program to implement
// Multilevel Inheritance
#include <iostream>
using namespace std;

// base class
class Vehicle
{
  public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};
class fourWheeler: public Vehicle
{  public:
    fourWheeler()
    {
      cout<<"Objects with 4 wheels are vehicles"<<endl;
    }
};
// sub class derived from two base classes
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
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.
filter_none
edit
play_arrow

brightness_4

```
// C++ program to implement
// Hierarchical Inheritance
#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 (Virtual) Inheritance: Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.
Below image shows the combination of hierarchical and multiple inheritance:
filter_none
edit
play_arrow

brightness_4
```
// C++ program for Hybrid Inheritance

#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
```