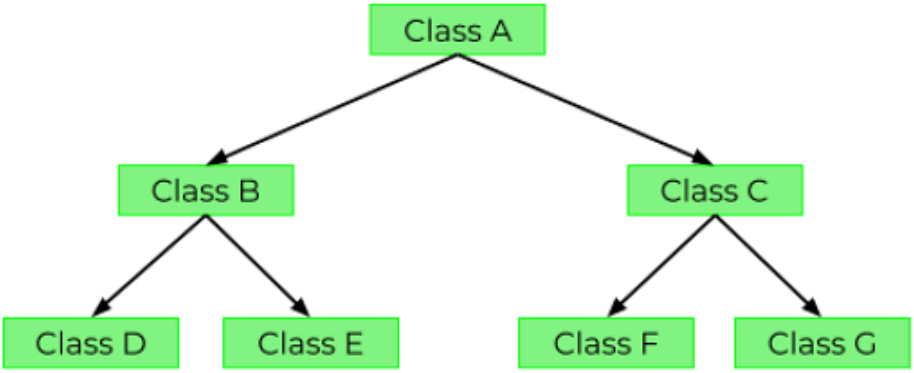
Class Hierarchies-Public and Private Inheritance

Class Hierarchies :

Hierarchical Inheritance in C++ **refers to the type of inheritance that has a hierarchical structure of classes**. A single base class can have multiple derived classes, and other subclasses can further inherit these derived classes, forming a hierarchy of classes.



C++ Hierarchical Inheritance Syntax

```
class A // base class
{
   .............
};
class B : access_specifier A // derived class from A
{
   ..........
 } ;
 class C : access_specifier A // derived class from A
 {
   ..........
 } ;
 class D : access_specifier A // derived class from A
 {
    ..........
 } ;
```

C++ Hierarchical Inheritance Example

```
// hierarchial inheritance.cpp
#include <iostream>
using namespace std;

class A //single base class
{
   public:
            int x, y;
            void getdata()
            {
               cout << "\nEnter value of x and y:\n"; cin >> x >> y;
            }
};
```

```
class B : public A //B is derived from class base
{
   public:
            void product()
            {
               cout << "\nProduct= " << x * y;
            }
};
class C : public A //C is also derived from class base
{
   public:
            void sum()
            {
      cout << "\nSum= " << x + y;
            }
};
int main()
{
   B obj1;         //object of derived class B
   C obj2;         //object of derived class C
   obj1.getdata();
   obj1.product();
   obj2.getdata();
   obj2.sum();
   return 0;
}  //end of program
```

Output

```
Enter value of x and y:
2
3
Product= 6
Enter value of x and y:
2
3
Sum= 5
```

Public and Private Inheritance:

public, **protected,** and **private** inheritance have the following features:

- **public inheritance** makes public members of the base class public in the derived class, and the protected members of the base class remain protected in the derived class.
- **protected inheritance** makes the public and protected members of the base class protected in the derived class.
- **private inheritance** makes the public and protected members of the base class private in the derived class.

Example 1: C++ public Inheritance

```
// C++ program to demonstrate the working of public inheritance

#include <iostream>
using namespace std;

class Base {
  private:
    int pvt = 1;

  protected:
    int prot = 2;

  public:
    int pub = 3;

    // function to access private member
    int getPVT() {
      return pvt;
    }
};

class PublicDerived : public Base {
  public:
    // function to access protected member from Base
    int getProt() {
      return prot;
    }
};

int main() {
  PublicDerived object1;
  cout << "Private = " << object1.getPVT() << endl;
  cout << "Protected = " << object1.getProt() << endl;
  cout << "Public = " << object1.pub << endl;
  return 0;
}
```

Run Code

Output

```
Private = 1
Protected = 2
Public = 3
```

Example 3: C++ private Inheritance

```
// C++ program to demonstrate the working of private inheritance
```

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

class Base {
  private:
    int pvt = 1;

  protected:
    int prot = 2;

  public:
    int pub = 3;

    // function to access private member
    int getPVT() {
      return pvt;
    }
};

class PrivateDerived : private Base {
  public:
    // function to access protected member from Base
    int getProt() {
      return prot;
    }

    // function to access private member
    int getPub() {
      return pub;
    }
};

int main() {
  PrivateDerived object1;
  cout << "Private cannot be accessed." << endl;
  cout << "Protected = " << object1.getProt() << endl;
  cout << "Public = " << object1.getPub() << endl;
  return 0;
}
```

Run Code

Output

Private cannot be accessed.
Protected = 2
Public = 3