# Virtual Base Class:

\* Virtual base classes are used in "virtual inheritance" which is a way of preventing “multiple instances” of a given class appearing in an inheritance hierarchy when you're using multiple inheritances.

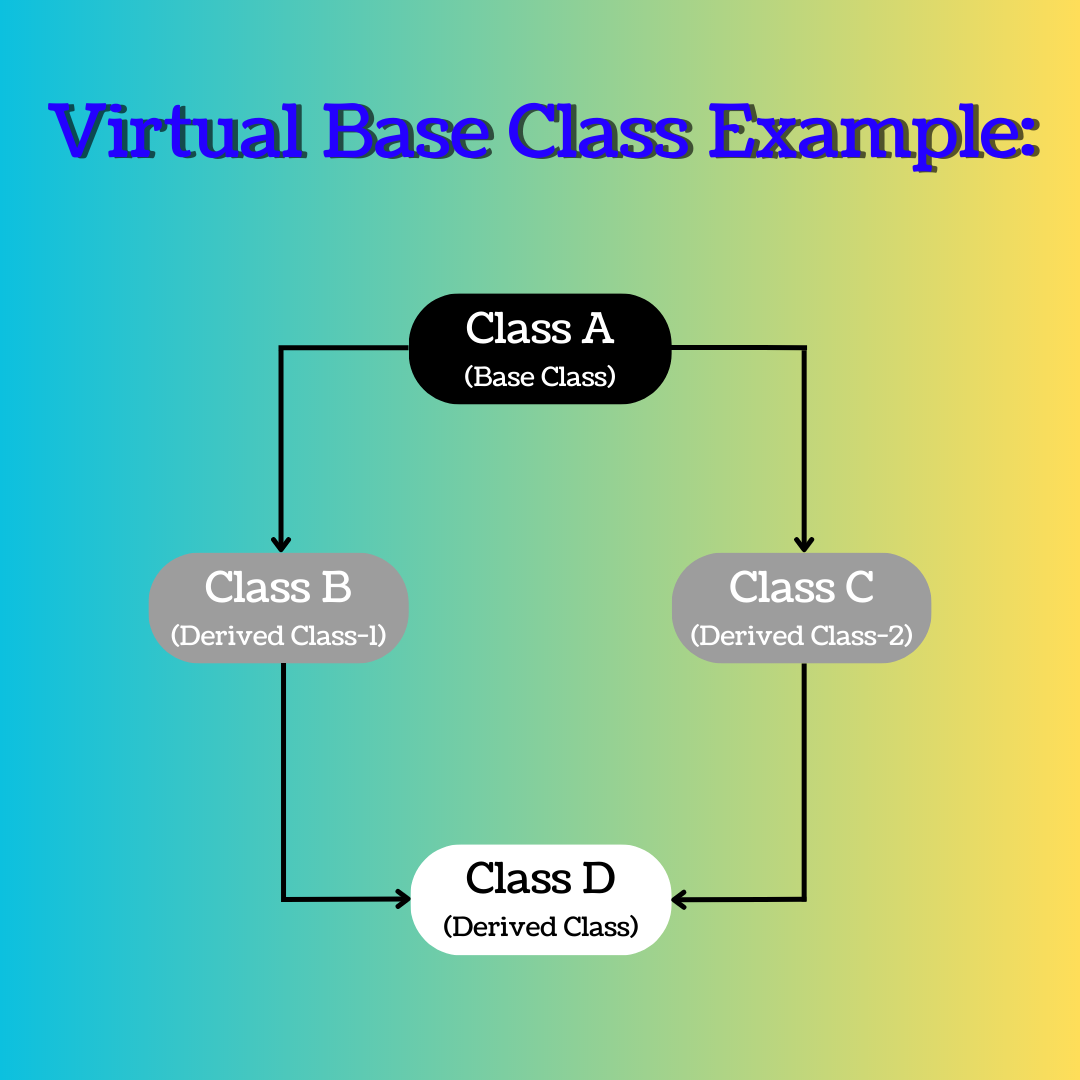
@ Let's understand in detail with an example that why do we need a "Virtual Base Class".

# Example of "Virtual Base Class":

\* Now, let's say that we have a class A, and it is inherited by two classes, Class B and Class C.

\* And now these two classes, class B and class C are inherited by class D.

\* The "Inheritance Diagram" for the above scenario looks like this:



\* Now in this scenario, class A have some inheritable members in it, let's say those are represented with small "a".

\* When class B and C inherited class A. Both the classes got a copy of these inheritable members of class A.

\* Now class B and C both have inheritable members of class A.

\* But when class D is inherited from class B and class C. It gets all the inheritable members of class A 2 times, one time it gets all inheritable members of class A from class B and one time from class C.

\* Now, because class D have 2 instances of inheritable members of class A, that's why this scenario leads to ambiguity error whenever we try to call or use any members of class A using the object of class D.

\* Because whenever we try to do this, the compiler gets confused that there are 2 instances of this thing in class D which you’re trying to use, and I don't know that which instance I should use out of these 2 instances, both are exactly similar. There is "inexactness" in your call, you should do something so that only one instance of inheritable members of class A exists in class D.

A diagram of a class

Description automatically generated

\* Now here comes the role of "Virtual Base Class" to solve this type of ambiguity in an inheritance hierarchy.

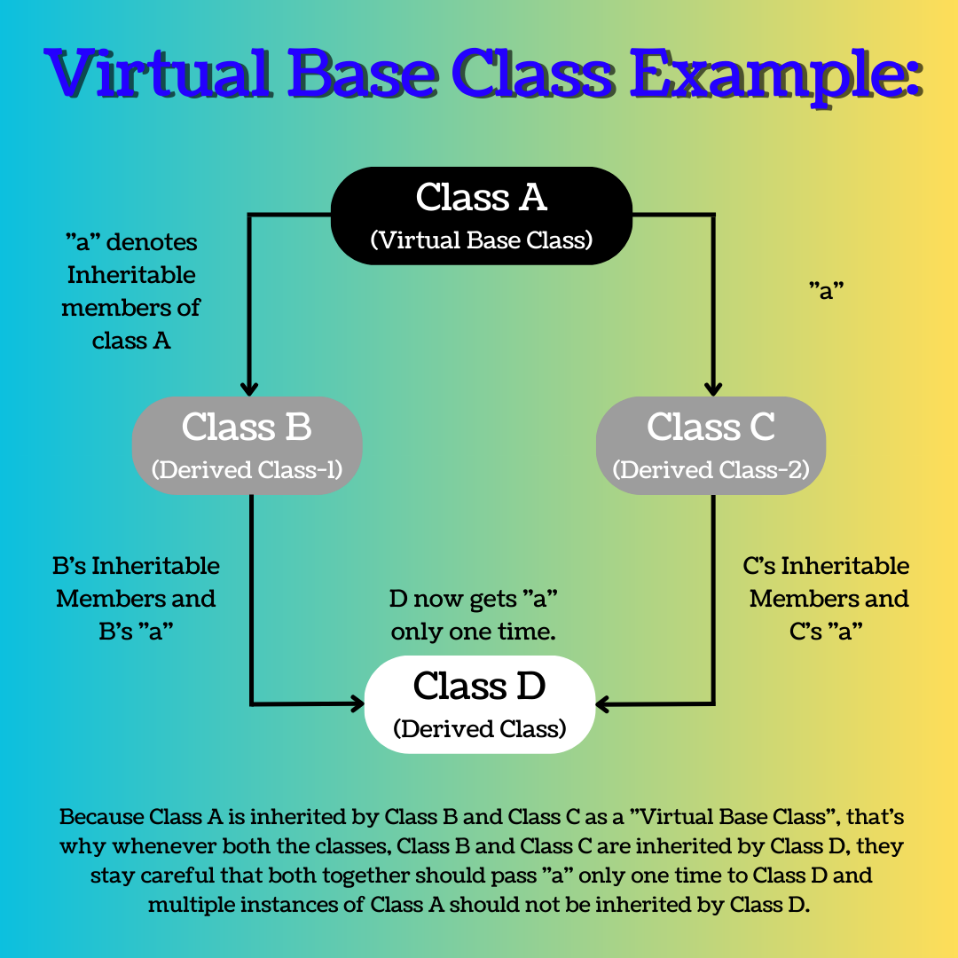
\* The "Base Class" here is "class A", whose multiple instances are present in "class D" and this is leading to ambiguity error. So, to prevent this from happening.

\* We inherit "class A" as “Virtual Base Class”. What is this "Virtual Base Class”??

\* Inheriting any class as "Virtual Base Class" means that only one copy of that base class's members is inherited by second-level derivatives (a.k.a. grandchild derived classes).

\* This means that only one copy of "class A's inheritable members will be passed by class B and class C to the class D".

\* Simply when class B and class C are inheriting class A, they inherit "class A" as a "Virtual Base Class" and this means that class B and class C both are informed by the class A, that if any class in future is derived from both of you then you guys have to make sure that only one copy of "My Inheritable Members (this is being said by class A)" is passed to that grandchild class of mine (means the class D here is the grandchild and it should only get one copy of "Inheritable Members of class A").



# Syntax for "Virtual Base Class"

// So, to resolve this ambiguity, when class A is inherited by both class B and class C, it is declared as “Virtual Base Class” by placing a keyword "virtual" like this:

// Syntax 1:

class B : virtual public A{

};

// Syntax 2:

class C : public virtual A{

};

// Note: "Virtual" keyword can be written before or after the public (means the "visibility mode" in which the class is getting inherited) and it won't change anything, the code will run smoothly in both the cases, both the syntaxes are "Valid".

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

# Example Code for "Virtual Base Class" in C++:

// This code will cause an ambiguity error

#include<iostream>

using namespace std;

class A{

public:

int a = 100;

void greet(){

cout << "Hehehe" << endl;

}

};

class B : public A{

public:

int b = 200;

};

class C : public A{

public:

int c = 300;

};

class D : public B, public C{

public:

int d = 400;

};

int main(){

D obj;

cout << obj.a << endl;

obj.greet();

// Above 2 lines of code will give ambiguity error and we have studied earlier that why they're giving ambiguity error

cout << obj.b << endl;

cout << obj.c << endl;

cout << obj.d << endl;

return 0;

}

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

// In this code we have solved the ambiguity error by using concept of "Virtual Inheritance" or you can say "Virtual Base Class"

#include<iostream>

using namespace std;

class A{

public:

int a = 100;

void greet(){

cout << "Hehehe" << endl;

}

};

class B : virtual public A{

public:

int b = 200;

};

class C : virtual public A{

public:

int c = 300;

};

class D : public B, public C{

public:

int d = 400;

};

int main(){

D obj;

cout << obj.a << endl;

obj.greet();

// Above 2 lines of code will give ambiguity error and we have studied earlier that why they're giving ambiguity error

cout << obj.b << endl;

cout << obj.c << endl;

cout << obj.d << endl;

return 0;

}

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

@ References Cited:

https://www.geeksforgeeks.org/virtual-base-class-in-c/

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_