Virtual Constructors and Destructors

The virtual method works when we have a base class pointer to a derived class object.

The constructor must be non-virtual because when a constructor of a class is executed, there is no virtual table in the memory, which means no virtual pointer defined yet. So, the constructor should never be virtual. But virtual destructor is possible.

Virtual Constructor

The creation of a virtual constructor is not possible because of the reasons given below:

- There is no virtual memory table present while calling the constructor. So, the construction of a virtual constructor is not possible.
- Because the object is not created, virtual construction is impossible.

The following program will throw an error because it tries to call a virtual constructor.

```
#include <iostream>
using namespace std;
class Base {
  public:
  virtual Base()                    // Virtual Constructor
  { cout << "Base created" << endl; }
  void show()
  {  cout << "Show the Base class object" << endl; }
};
class Derived: public Base{
  public:
  Derived()
  {cout << "Derived created" << endl; }
  void show()
  {cout << "Show the Derived class object " << endl;}
};
int main()
{
  Base* ptr;
  Derived d;
  ptr = &d;
  ptr->show() }
```

Virtual Destructor

We cannot delete a derived class object using a base class pointer that has a non-virtual destructor. To delete the derived class object using the base class pointer, the base class must contain a virtual destructor.

```
// CPP program without virtual destructor
#include <iostream>
using namespace std;
class Base {
  public:
    Base()
    { cout << "Base created\n"; }
    ~Base()
    { cout<< "Base destroyed\n"; }
};

class Derived: public Base {
  public:
    Derived()
     { cout << "Derived created\n"; }
    ~Derived()
       { cout << "Derived destroyed\n"; }
};
int main()
{
  Base *b = new Derived;
  delete b;
  getchar();
  return 0;
}
```

Output:

```
Base created
Derived created
Base destroyed
```

In the given example, both Base class and Derived class destructors are called. We are using a virtual destructor for Base Class. This will first call the Derived class destructor, and then the destructor of the Base class is called.

```
// CPP program virtual destructor
#include <iostream>
using namespace std;
class Base
{
   public:
   virtual ~Base()
   {
      cout << "Base Destroyed\n";
   }
};
class Derived:public Base
{
   public:
   ~Derived()
   {
      cout<< "Derived Destroyed\n";
   }
};
int main()
{
   Base* b = new Derived;     // Upcasting
   delete b;
}
```

Output:

```
Derived Destroyed
Base Destroyed
```

1. **Is it possible to have a virtual constructor and virtual destructor?**
   We can't have virtual constructors in our program but we can use a virtual destructor. A virtual destructor is used to destroy an object through a base class pointer by calling the derived destructors appropriately.
2. **Why are there no virtual constructors?**
   A constructor can not be virtual because when the constructor of a class is executed, there is no virtual table in the memory, which means no virtual pointer defined yet. Hence the constructor should always be non-virtual.
3. **Why are virtual destructors used?**
   Virtual destructors in C++ are used to avoid memory leaks, especially when your class contains unmanaged code, i.e., pointers or object handles to files, databases, or other external objects