Virtual Copy Constructor in C++

Before digging deep into the topics lets brush up all the related terms.

A **copy constructor** is a special type of constructor that is used to create an object that is an exact copy of the object that is passed.

A virtual function is a member function that is declared in the parent class and is redefined (overridden) in a child class that inherits the parent class.

With the use of a virtual copy constructor, the programmer will be able to create an object without knowing the exact data type of the object.

In C++ programming language, copy Constructor is used to creating an object copied from another. but if you want the program to decide at the runtime about the type of object created i.e. that object type is defined at runtime, not at compile-time and is based on some input provided by the user for a certain condition. in this situation, We need a copy constructor with some special powers to do this thing. So so to do this virtual copy constructor is declared that offers the cloning of objects in real-time.

Let's take an example, Suppose we have a figure who is the area is to be found out using the program. but the type up of an object is defined as real-time that it can be a square rectangle or a circle. So we will use a virtual copy constructor that will copy the objects based on the type that the user inputted.

For virtual constructors to work properly there are two methods that are defined over the base class. they are −

```
clone()
create()
```

Copy constructor uses the virtual clone method whereas the virtual create method is used by the default constructors for creating a virtual constructor.

Example

```
#include <iostream>
using namespace std;
class figure{
   public:
   figure() { }
   virtual
   ~figure() { }
   virtual void ChangeAttributes() = 0;
   static figure *Create(int id);
   virtual figure *Clone() = 0;
};
class square : public figure{
   public:
   square(){
```

```
      cout << "square created" << endl;
   }
   square(const square& rhs) { }
   ~square() { }
   void ChangeAttributes(){
      int a;
      cout<<"The side of square";
      cin>>a;
      cout<<"Area of square is "<<a*a;
   }
   figure *Clone(){
      return new square(*this);
   }
};
class circle : public figure{
   public:
   circle(){
      cout << "circle created" << endl;
   }
   circle(const circle& rhs) { }
   ~circle() { }
   void ChangeAttributes(){
      int r;
      cout << "enter the radius of the circle ";
      cin>>r;
      cout<<"the area of circle is "<<((3.14)*r*r);
   }
   figure *Clone(){
      return new circle(*this);
   }
};
class rectangle : public figure{
   public:
   rectangle(){
      cout << "rectangle created" << endl;
   }
   rectangle(const rectangle& rhs) { }
   ~rectangle() { }
   void ChangeAttributes(){
      int a ,b;
      cout<<"The dimensions of rectangle ";
      cin>>a>>b;
      cout<<"Area of rectangle is "<<a*b;
   }
   figure*Clone(){
      return new rectangle(*this);
   }
};
figure *figure::Create(int id){
```

```
   if( id == 1 ){
      return new square;
   }
   else if( id == 2 ){
      return new circle;
   }
   else{
      return new rectangle;
   }
}
class User{
   public:
   User() : figures(0){
      int input;
      cout << "Enter ID (1, 2 or 3): ";
      cin >> input;
      while( (input != 1) && (input != 2) && (input != 3) ){
         cout << "Enter ID (1, 2 or 3 only): ";
         cin >> input;
      }
      figures = figure::Create(input);
   }
   ~User(){
      if( figures ){
         delete figures;
         figures = 0;
      }
   }
   void Action(){
      figure *newfigure = figures->Clone();
      newfigure->ChangeAttributes();
      delete newfigure;
   }
   private:
   figure *figures;
};
int main(){
   User *user = new User();
   user->Action();
   delete user;
}
```

Output

```
Enter ID (1, 2 or 3): 2
circle created
enter the radius of the circle R 3
the area of circle is 28.26
```