
Subject: Re: why no 'Ctrl* Ctrl::Clone() const = 0' (virtual constructor)

Posted by [kohait00](#) on Thu, 09 Sep 2010 09:54:08 GMT

[View Forum Message](#) <> [Reply to Message](#)

SOLUTION FOUND: by providing a copy interface with base class info

```
//this one works by providing additional information about the common base class
```

```
#if 1
```

```
//copyable interface defining the common base class C, without implementing Copy
```

```
template<class C>
```

```
class CopyableC
```

```
{
```

```
public:
```

```
    virtual ~CopyableC() {}
```

```
    virtual CopyableC* Copy() const = 0;
```

```
    virtual const C& GetC() const = 0;
```

```
    virtual C& GetC() = 0;
```

```
    operator const C&() const {return GetC(); }
```

```
    operator C&() {return GetC(); }
```

```
};
```

```
//provides the implementation of Copy, which is used by PolyDeepNew
```

```
//and implements the base class accessors.
```

```
//T is the derived type, i.e. EditInt, C is common base class, i.e. Ctrl
```

```
//PolyCopyingC<EditInt, Ctrl> a;
```

```
//CopyableC<Ctrl>* p = a.Copy();
```

```
//p->GetC().SizePos();
```

```
template<class T, class C>
```

```
class PolyCopyingC : public PolyDeepCopyNew<PolyCopyingC<T,C>, T>, public CopyableC<C>
```

```
{
```

```
public:
```

```
    virtual PolyCopyingC* Copy() const { return new PolyCopyingC(); }
```

```
    virtual const C& GetC() const { return *this; }
```

```
    virtual C& GetC() { return *this; }
```

```
};
```

```
#endif
```

```
///
```

```
PolyCopyingC<EditInt, Ctrl> abc; //provide common base class info
```

```
CopyableC<Ctrl>* pabc = abc.Copy();
```

```
CopyableC<Ctrl>* pabc2 = pabc->Copy();
```

```
pabc->GetC().SizePos();  
Add(*pabc);  
Add(pabc2->GetC().HSizePos().TopPos(0,100));
```
