Subject: Re: Vector of object: cast to inherited class
Posted by Xemuth on Sat, 14 Sep 2019 11:41:07 GMT
View Forum Message <> Reply to Message

Hello Novo, Thanks for you respons.

I have take a look at Array documentation and it say :
Quote:Array can be also used to store polymorphic elements - stored elements could be derived from T. To store such elements, you pass a pointer to an element previously created on the heap. Still, Array takes over ownership of such element (it e.g. deletes it when appropriate). You can also use this method to create an Array of elements that do not have either pick, deep copy constructor nor default constructor.

"To store such elements, you pass a pointer to an element previously created on the heap"
It mean I must use an Array<A*> ? Their is no way to do something like the exemple bellow without having to work with ptr ?

Thanks in advance, have good day.

```
#include <Core/Core.h>

using namespace Upp;

class A {
 public:
 A(){}
 virtual ~A(){}
 virtual void Hello(){
  Cout() << "Hello from A" <<"\n";
 }
};

class B :public A{
 public:
 B(){}
 ~B(){}
 void Hello(){
  Cout() << "Hello from B" <<"\n";
 }
};

CONSOLE_APP_MAIN
{
 Array<A> myArray;
 B b;
 static_cast<B&>(myArray.Add(b)).Hello(); //"Hello from A"
 ((B&)myArray.Add(b)).Hello(); //"Hello from A"
 //Looking for "Hello from B" without using Array<A*>
 Array<A*> myArray2;
 myArray2.Add(&b)->Hello(); //Working properly but using ptr
}
```