CECS2222 Computer Programming II
C. Talavera
Programa que define una caja como una instancia de la clase *PlainBox* que esta definida como clase base y las clases derivadas *ToyBox* y *MagicBox*. La clase base contiene funciones virtuales y se le anade una clase Pura Abstracte de nombre BoxInterface.

```
#include<iostream>
#include<string>
using namespace::std;
#include "PlainBox.h"
#include "Car.h"
#include "ToyBox.h"
#include "MagicBox.h"
#include  "BoxInterface.h"

void myColor(int color);
int main(){

	BoxInterface<string> *myItem1 = new PlainBox <string> ("Candies");
	BoxInterface<string> *myItem2 = new ToyBox <string>("puzzel",YELLOW);
	BoxInterface<string> *myItem3 = new MagicBox <string>("Pens");

	cout << myItem1->getItem() << endl;
	cout << myItem2->getItem() << endl;
	cout << myItem3->getItem() << endl;

	delete  myItem1;
	delete  myItem2;
	delete  myItem3;

	system("pause");
	return 0;
}


/*PlainBox constructor executing
PlainBox constructor executing
MagicBox contructor executing
Candies
puzzel
Pens
PlainBox destructor executing
ToyBox destructor executing
PlainBox destructor executing
MagicBox destructor executing
PlainBox destructor executing
Press any key to continue . . .*/
/*
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#ifndef _BOX_INTERFACE_
#define _BOX_INTERFACE_
template < class ItemType>
	class BoxInterface{
	public:
		virtual ~BoxInterface() {}
		virtual void setItem(const ItemType &theItem) = 0;
		virtual ItemType getItem() const = 0;
	};
#endif
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#ifndef _PLAINT_BOX
#define _PLAINT_BOX
#include "BoxInterface.h"
template<class ItemType>
class PlainBox: public BoxInterface<ItemType>{
private:
	ItemType item;
public:
	PlainBox();
	PlainBox(const ItemType &theItem);
	virtual~PlainBox();
    virtual void setItem(const ItemType &theItem);
	virtual ItemType getItem() const;
};

template<class ItemType>
PlainBox<ItemType>::PlainBox(){
	cout << "PlainBox constructor executing\n";
}

template<class ItemType>
PlainBox<ItemType>::PlainBox(const ItemType &theItem){
	setItem(theItem);
}

template<class ItemType>
PlainBox<ItemType>::~PlainBox(){
	cout << "PlainBox destructor executing\n";
}

template<class ItemType>
void PlainBox<ItemType>::setItem(const ItemType &theItem){
	item = theItem;
}

template<class ItemType>
ItemType PlainBox<ItemType>::getItem() const{
	return item;
}

#endif
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#ifndef _MAGIC_BOX
#define _MAGIC_BOX
#include "PlainBox.h"
```

```
template < class ItemType >
class MagicBox :public PlainBox<ItemType>{
private:
	bool firstItemStored;
public:
	MagicBox();
	MagicBox(const ItemType &theItem);
	~MagicBox();
	void setItem(const ItemType &theItem);


};

template<class ItemType>
MagicBox<ItemType>::MagicBox() :PlainBox<ItemType>(){
	firstItemStored = false;
	cout << "MagicBox constructor executing\n";
}

template<class ItemType>
MagicBox<ItemType>::MagicBox(const ItemType &theItem) : PlainBox<ItemType>(){
	PlainBox<ItemType> ::setItem(theItem);
	firstItemStored = false;
	cout << "MagicBox contructor executing\n";
}

template<class ItemType>
MagicBox<ItemType>::~MagicBox(){
	cout << "MagicBox destructor executing\n";
}

template<class ItemType>
void MagicBox<ItemType>::setItem(const ItemType &theItem){
	if (!firstItemStored){
		PlainBox<ItemType> ::setItem(theItem);
		firstItemStored = true;
	}
}

#endif
```