C++ INTERLUDE 1 C++ CLASSES



Basic Class

```
Initializer list ce indicator
                                                     "clude "PlainBox.h"
#ifndef _PLAIN_BOX
                             Pre-processor
#define _PLAIN_BOX
                                directives
                                                       inBox::PlainBox()
                                                                                        Default Constructor
                                                        : item(0.0)
                                                       } // end default constructor
class PlainBox
                          Parameters are
private:
                       passed by constant
                                                                                                 Parameterized
                                                                                     m)
  double
          item;
                                                       item = 0.0;
                              reference
                                                                                                   Constructor
                                                     } // end default constructor
public:
  PlainBox();
                                                     void PlainBox::setItem(const double& theItem)
  PlainBox(const double& theltem);
                                                       item = theltem;
  void setItem(const dubied the tem);
                                                     } // end setItem
  double getItem() cor st:
                                                                                                 Method
};
                                                     double PlainBox::getItem() const
                                                                                           Implementations
             Accessor methods are
                                                       return item;
                  declared const
#endif
                                                     } // end getItem
                            PlainBox.h
                                                       PlainBox.cpp
```

Basic Class

```
#ifndef _PLAIN_BOX
#define _PLAIN_BOX
class PlainBox
private:
   double
            item;
public:
   PlainBox();
   PlainBox(const double& theltem);
   void setItem(const double& theItem);
   double getItem() const;
};
#endif
```

PlainBox.h

```
Client Code
#include "Plai
                double dish = 8.5;
                PlainBox firstBox(dish);
                std::cout << firstBox.getItem() << std::endl;</pre>
PlainBox::Plai
  : item(0.0)
  } // end defa
                double bowl = 4.0;
                PlainBox anotherBox = PlainBox(bowl);
PlainBox::Plai
                anotherBox.setItem(dish);
  : item(thelte
                std::cout << anotherBox.getItem() << std::endl;</pre>
  } // end cor
          Bayyaatitam(const double& theitem)
```

```
dou (const return item;
)/firstBox

anotherBox
```

PlainBox.cpp



Templates

```
#ifndef _PLAIN_BOX
#define _PLAIN_BOX
   typeslatekoulds ItemType>
class PlainBox
                                                                                                                                                                                              template allows
private:
                                                                                                                                                                                                      client to decide
                    demalype item;
                                                                                                                                                                                                                                                                       type
public:
                    PlainBox();
                    PlainBox(const them by Beta ettelte);;
                     void settlem(const themally peta eltelte);;;
                    the until property that the control is the until property that the control is the until property that the unitil property the unitil property that the unitil property the unitil property that the unitil property that the unitil property that the unitil property that the unitil property the unitil property that the unitil prope
#include "PlainBox.cpp"
#endif
                                                                                                                                                                                                                                                 PlainBox.h
```

```
#include "PlainBox.h"
template<class ItemType>
PlainBox<ItemType>::PlainBox()
  } // end default constructor
template<class ItemTvpe>
PlainBox<ItemType>::Pla nBox(const ItemType& theItem)
 : itéten(t (télééten););
  3 // end constructor
template<class ItemType>
void i lainBox<item i ype>::setitei n(const ItemType& theItem)
item = theltem;
} // end setItem
template<class ItemType>
Item Type FiainBox<Item Type>::getItem() const
 return item;
} // end getItem
   PlainBox.cpp
```

Templates

```
#ifndef_PLAIN_BOX
#includes the property of th
```

```
#define _PLAIN_BOX
template<class ItemType>
class PlainBox
private:
  ItemType item;
public:
  PlainBox();
   PlainBox(const ItemType& theItem);
  void setItem(const ItemType& theItem);
  ItemType getItem() const;
};
#include "PlainBox.cpp"
#endif
```

PlainBox.h

```
double dish = 8.5;
        PlainBox<double> firstBox(dish);
emp
        std::cout << firstBox.getItem() << std::endl;</pre>
        string animal = "Dog";
        PlainBox<string> anotherBox = PlainBox<string>(animal);
        anotherBox.setItem("Cat");
        std::cout << anotherBox.getItem() << std::endl;</pre>
template<class ItemType>
    Plain Box Sitom Tyngoriset Item (rean st dtem Type & the Item)
 item
                       Туре≥нgetItem()
 return item;
 firstBox
```

PlainBox.cpp

C++ INTERLUDE 1 C++ CLASSES



TOYBOX CLASS

Derived Class

```
#ifndef _TOYBOX
#define _TOYBOX
#include "PlainBox.h"
enum Color {BLACK, RED, BLUE, GREEN, YELLOW};
template<class ItemType>
class ToyBox : public PlainBox<ItemType>
private:
   Color boxColor;
public:
   ToyBox();
   ToyBox(const Color& theColor);
   ToyBox(const ItemType& theItem,
                    const Color& theColor);
   Color getColor() const;
}; // end ToyBox
#include "ToyBox.cpp"
#endif
```

Cliant Code

ToyBox.h ToyBox.cpp

```
#include "ToyBox.h" // optional
                                     Base class default
template<class ItemType>
ToyBox<ItemType>::ToyBox()
                                   constructor is called
  : boxColor(BLACK)
{ } // end default constructor
                                            implicitly
template<class ItemType>
ToyBox<ItemType>::ToyBox(const Color& theColor)
   : boxColor(theColor)
{ } // end constructor
template<class ItemType>
ToyBox<ItemType>::ToyBox(const ItemType& theItem,
                                     const Color& theColor)
   : PlainBox<ItemType>(theItem), boxColor(theColor)
{ } // end constructor
template<class ItemType>
                              Base class
Color ToyBox<ItemType>
                       constructor must be
  return boxColor;
                       first in the initializer
} // end getColor
```

list

Pearson

BOXINTERFACE CLASS

Abstract Base Class

```
#ifndef _BOXINTERFACE
#define _BOXINTERFACE
template <class ItemType>
class BoxInterface
public:
  virtual void setItem(const ItemType& theItem) = 0;
  virtual ItemType getItem() const = 0;
 virtual ~BoxInterface() { }
}; // end BoxInterface
#endif
                            BoxInterface.h
```

```
#ifndef _PLAIN_BOX
#define _PLAIN_BOX
#include "BoxInterface.h"
template<class ItemType>
class PlainBox : public BoxInterface<ItemType>
private:
   ItemType item;
public:
   PlainBox();
   PlainBox(const ItemType& theItem);
  virtual void setItem(const ItemType& theItem);
  virtual ItemType getItem() const;
#include "PlainBox.cpp"
                               PlainBox.h
#endif
```