

C++ Software Transactional memory

Zoltan Fuzesi

C00197361 IT Carlow

Supervisor : Joe Kehoe

Contents

1	OSTM C++ Software Transactional Memory	1
1.1	Object Based Software Transactional Memory.	1
1.1.1	Brief. Download the zip file from the provided link in the web-site, that contains the libostm.so, TM.h, TX.h, OSTM.h files.	1
1.1.2	Step 1: Download the archive file.	1
1.1.3	Step 2: Unzip in the target destination.	1
1.1.4	Step 3: Copy the shared library (libostm.so) to the operating system folder where the other shared library are stored.	1
1.1.5	Step 4: Achieve the required class hierarchy between the OSTM library and your own class structure.	1
1.1.6	Step 5: Create an executable file as you linking together the TM.h, TX.h, OSTM.h files with your own files.	1
1.1.7	Step 6: Now your application use transactional environment, that guarantees the consistency between object transactions.	1
1.1.8	Step 7: Run the application.	1
2	README	1
3	Hierarchical Index	2
3.1	Class Hierarchy	2
4	Class Index	2
4.1	Class List	2
5	File Index	3
5.1	File List	3

6	Class Documentation	4
6.1	AIB Class Reference	4
6.1.1	Detailed Description	7
6.1.2	Constructor & Destructor Documentation	7
6.1.3	Member Function Documentation	9
6.2	BANK Class Reference	16
6.2.1	Detailed Description	18
6.2.2	Constructor & Destructor Documentation	18
6.2.3	Member Function Documentation	20
6.3	BOA Class Reference	23
6.3.1	Detailed Description	26
6.3.2	Constructor & Destructor Documentation	26
6.3.3	Member Function Documentation	27
6.4	BOI Class Reference	35
6.4.1	Detailed Description	38
6.4.2	Constructor & Destructor Documentation	38
6.4.3	Member Function Documentation	39
6.5	CARLOW_W Class Reference	47
6.5.1	Detailed Description	50
6.5.2	Constructor & Destructor Documentation	50
6.5.3	Member Function Documentation	52
6.6	CARPHONE_WAREHOUSE Class Reference	60
6.6.1	Detailed Description	63
6.6.2	Constructor & Destructor Documentation	63
6.6.3	Member Function Documentation	65
6.7	DUNDALK_W Class Reference	73
6.7.1	Detailed Description	76
6.7.2	Constructor & Destructor Documentation	76
6.7.3	Member Function Documentation	78
6.8	KILKENNY_W Class Reference	86

6.8.1	Detailed Description	89
6.8.2	Constructor & Destructor Documentation	89
6.8.3	Member Function Documentation	91
6.9	SLIGO_W Class Reference	99
6.9.1	Detailed Description	102
6.9.2	Constructor & Destructor Documentation	102
6.9.3	Member Function Documentation	104
6.10	SWBPLC Class Reference	113
6.10.1	Detailed Description	116
6.10.2	Constructor & Destructor Documentation	116
6.10.3	Member Function Documentation	117
6.11	TALLAGH_W Class Reference	125
6.11.1	Detailed Description	128
6.11.2	Constructor & Destructor Documentation	128
6.11.3	Member Function Documentation	130
6.12	ULSTER Class Reference	139
6.12.1	Detailed Description	142
6.12.2	Constructor & Destructor Documentation	142
6.12.3	Member Function Documentation	143
6.13	UNBL Class Reference	151
6.13.1	Detailed Description	154
6.13.2	Constructor & Destructor Documentation	154
6.13.3	Member Function Documentation	155
6.14	WAREHOUSE Class Reference	163
6.14.1	Detailed Description	165
6.14.2	Constructor & Destructor Documentation	165
6.14.3	Member Function Documentation	167

7 File Documentation	172
7.1 AIB.cpp File Reference	172
7.2 AIB.cpp	173
7.3 AIB.h File Reference	174
7.4 AIB.h	175
7.5 BANK.cpp File Reference	176
7.6 BANK.cpp	176
7.7 BANK.h File Reference	177
7.8 BANK.h	178
7.9 BOA.cpp File Reference	179
7.10 BOA.cpp	180
7.11 BOA.h File Reference	181
7.12 BOA.h	182
7.13 BOI.cpp File Reference	183
7.14 BOI.cpp	184
7.15 BOI.h File Reference	185
7.16 BOI.h	186
7.17 CARLOW_W.cpp File Reference	187
7.18 CARLOW_W.cpp	188
7.19 CARLOW_W.h File Reference	189
7.20 CARLOW_W.h	190
7.21 CARPHONE_WAREHOUSE.cpp File Reference	191
7.22 CARPHONE_WAREHOUSE.cpp	192
7.23 CARPHONE_WAREHOUSE.h File Reference	193
7.24 CARPHONE_WAREHOUSE.h	194
7.25 DUNDALK_W.cpp File Reference	195
7.26 DUNDALK_W.cpp	196
7.27 DUNDALK_W.h File Reference	197
7.28 DUNDALK_W.h	198
7.29 KILKENNY_W.cpp File Reference	199

7.30 KILKENNY_W.cpp	200
7.31 KILKENNY_W.h File Reference	201
7.32 KILKENNY_W.h	202
7.33 main.cpp File Reference	203
7.33.1 Function Documentation	204
7.34 main.cpp	220
7.35 OSTM.cpp File Reference	221
7.36 OSTM.cpp	222
7.37 OSTM.h File Reference	223
7.37.1 Function Documentation	223
7.38 OSTM.h	225
7.39 README.md File Reference	225
7.40 README.md	226
7.41 SLIGO_W.cpp File Reference	226
7.42 SLIGO_W.cpp	227
7.43 SLIGO_W.h File Reference	228
7.44 SLIGO_W.h	229
7.45 SWBPLC.cpp File Reference	230
7.46 SWBPLC.cpp	231
7.47 SWBPLC.h File Reference	232
7.48 SWBPLC.h	233
7.49 TALLAGH_W.cpp File Reference	234
7.50 TALLAGH_W.cpp	235
7.51 TALLAGH_W.h File Reference	236
7.52 TALLAGH_W.h	237
7.53 TM.cpp File Reference	238
7.54 TM.cpp	239
7.55 TM.h File Reference	240
7.55.1 Function Documentation	240
7.56 TM.h	242

7.57 TX.cpp File Reference	242
7.58 TX.cpp	243
7.59 TX.h File Reference	244
7.59.1 Function Documentation	244
7.60 TX.h	247
7.61 ULSTER.cpp File Reference	248
7.62 ULSTER.cpp	249
7.63 ULSTER.h File Reference	250
7.64 ULSTER.h	251
7.65 UNBL.cpp File Reference	252
7.66 UNBL.cpp	253
7.67 UNBL.h File Reference	254
7.68 UNBL.h	255
7.69 WAREHOUSE.cpp File Reference	256
7.70 WAREHOUSE.cpp	256
7.71 WAREHOUSE.h File Reference	257
7.72 WAREHOUSE.h	258

1 OSTM C++ Software Transactional Memory

1.1 Object Based Software Transactional Memory.

OSTM is a polymorphic solution to store and manage shared memory spaces within c++ programming context. You can store and managed any kind of object in transactional environment as a shared and protected memory space.

1.1.1 Brief. Download the zip file from the provided link in the web-site, that contains the libostm.so, TM.h, TX.h, OSTM.h files.

Unzip the archive file to the desired destination possibly where in you program is stored.

1.1.2 Step 1: Download the archive file.

1.1.3 Step 2: Unzip in the target destination.

1.1.4 Step 3: Copy the shared library (libostm.so) to the operating system folder where the other shared library are stored.

It will be different destination folder on different platforms. (Linux, Windows, Mac OS) [More Information](#)

1.1.5 Step 4: Achieve the required class hierarchy between the OSTM library and your own class structure.

Details and instruction of class hierarchy requirements can be found on the web-site. www.serversite.info/ostm

1.1.6 Step 5: Create an executable file as you linking together the TM.h, TX.h, OSTM.h files with your own files.

1.1.7 Step 6: Now your application use transactional environment, that guarantees the consistency between object transactions.

1.1.8 Step 7: Run the application.

2 README

C++ Software Transactional Memory (STM)

This documentation includes all the project specific files that required to build the STM library and the client code to use the library. The client code is demonstrate the usage of the STM API (Application Programming Interface). The STM library is a object based implementation, where the client need to inherite from the library on order to achieve the polymorphic Object Oriented Programming (OOP) behaviour.

The client application use a middle class to declare the child (Classes inherite from [BANK](#)) specific behaviour as a virtual methods. Whit this implementation the client application need to casting back the OSTM object to [BANK](#) object to use the child class implemented specific behaviours.

3 Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

OSTM

BANK	16
AIB	4
BOA	23
BOI	35
SWBPLC	113
ULSTER	139
UNBL	151
WAREHOUSE	163
CARLOW_W	47
CARPHONE_WAREHOUSE	60
DUNDALK_W	73

KILKENNY_W	86
SLIGO_W	99
TALLAGH_W	125

4 Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AIB	4
BANK	16
BOA	23
BOI	35
CARLOW_W	47
CARPHONE_WAREHOUSE	60
DUNDALK_W	73
KILKENNY_W	86
SLIGO_W	99
SWBPLC	113
TALLAGH_W	125
ULSTER	139
UNBL	151
WAREHOUSE	163

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

AIB.cpp	172
AIB.h	174
BANK.cpp	176
BANK.h	177
BOA.cpp	179

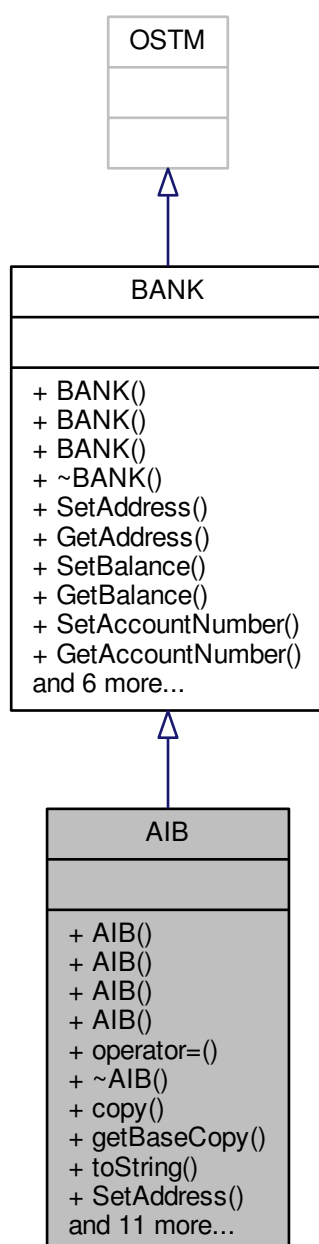
BOA.h	181
BOI.cpp	183
BOI.h	185
CARLOW_W.cpp	187
CARLOW_W.h	189
CARPHONE_WAREHOUSE.cpp	191
CARPHONE_WAREHOUSE.h	193
DUNDALK_W.cpp	195
DUNDALK_W.h	197
KILKENNY_W.cpp	199
KILKENNY_W.h	201
main.cpp	203
OSTM.cpp	221
OSTM.h	223
SLIGO_W.cpp	226
SLIGO_W.h	228
SWBPLC.cpp	230
SWBPLC.h	232
TALLAGH_W.cpp	234
TALLAGH_W.h	236
TM.cpp	238
TM.h	240
TX.cpp	242
TX.h	244
ULSTER.cpp	248
ULSTER.h	250
UNBL.cpp	252
UNBL.h	254
WAREHOUSE.cpp	256
WAREHOUSE.h	257

6 Class Documentation

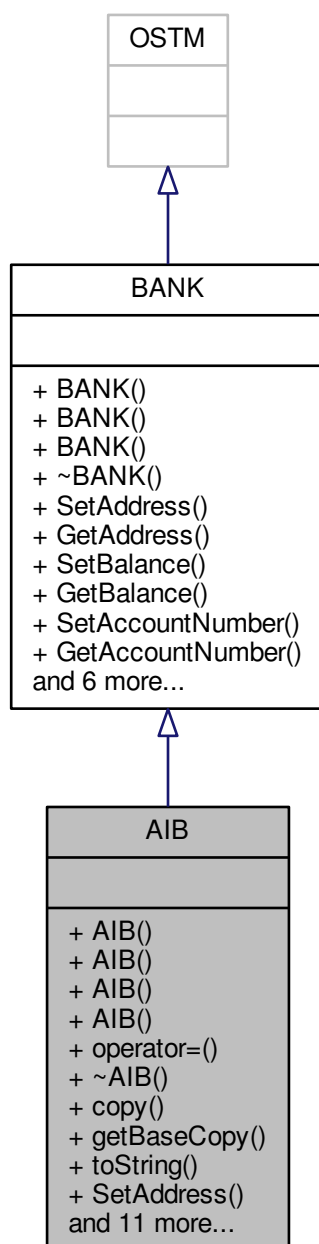
6.1 AIB Class Reference

```
#include <AIB.h>
```

Inheritance diagram for AIB:



Collaboration diagram for AIB:



Public Member Functions

- [AIB \(\)](#)
- [AIB](#) (int accountNumber, double balance, std::string firstName, std::string lastName, std::string address)
- [AIB](#) (std::shared_ptr< [BANK](#) > obj, int _version, int _unique_id)
- [AIB](#) (const [AIB](#) &orig)
- [AIB operator=](#) (const [AIB](#) &orig)

- virtual `~AIB ()`
- virtual void `copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)`
copy function, make deep copy of the object/pointer
- virtual `std::shared_ptr< OSTM > getBaseCopy (std::shared_ptr< OSTM > object)`
getBaseCopy function, make deep copy of the object/pointer and Return a new std::shared_ptr<BANK> type object
- virtual void `toString ()`
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void `SetAddress (std::string address)`
- virtual `std::string GetAddress () const`
- virtual void `SetBalance (double balance)`
- virtual `double GetBalance () const`
- virtual void `SetAccountNumber (int accountNumber)`
- virtual `int GetAccountNumber () const`
- virtual void `SetLastName (std::string lastName)`
- virtual `std::string GetLastName () const`
- virtual void `SetFirstName (std::string firstName)`
- virtual `std::string GetFirstName () const`
- virtual void `SetFullname (std::string fullname)`
- virtual `std::string GetFullname () const`

6.1.1 Detailed Description

Inherit from [BANK](#)

Definition at line 18 of file [AIB.h](#).

6.1.2 Constructor & Destructor Documentation

6.1.2.1 AIB::AIB () [inline]

Constructor

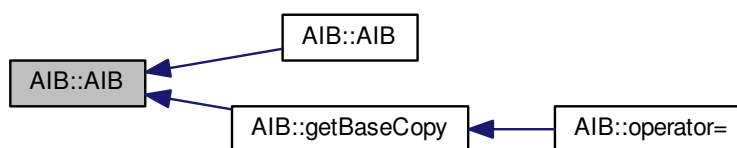
Definition at line 23 of file [AIB.h](#).

Referenced by [AIB\(\)](#), and [getBaseCopy\(\)](#).

```

00023         : BANK()
00024     {
00025         this->accountNumber = 0;
00026         this->balance = 50;
00027         this->firstName = "Joe";
00028         this->lastName = "Blog";
00029         this->address = "High street, Carlow";
00030         this->fullname = firstName + " " + lastName;
00031     };
00032 
```

Here is the caller graph for this function:



6.1.2.2 AIB::AIB (int *accountNumber*, double *balance*, std::string *firstName*, std::string *lastName*, std::string *address*) [inline]

Custom constructor

Definition at line 36 of file [AIB.h](#).

```
00036                                     :
00037     BANK()
00038     {
00039         this->accountNumber = accountNumber;
00039         this->balance = balance;
00040         this->firstName = firstName;
00041         this->lastName = lastName;
00042         this->address = address;
00043         this->fullname = firstName + " " + lastName;
00044     };
```

6.1.2.3 AIB::AIB (std::shared_ptr< BANK > *obj*, int *_version*, int *_unique_id*) [inline]

Custom constructor, used by the library for deep copying

Definition at line 48 of file [AIB.h](#).

References [AIB\(\)](#).

```
00048                                     : BANK(_version, _unique_id)
00049     {
00050
00051         this->accountNumber = obj->GetAccountNumber();
00052         this->balance = obj->GetBalance();
00053         this->firstName = obj->GetFirstName();
00054         this->lastName = obj->GetLastName();
00055         this->address = obj->GetAddress();
00056         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00057
00058     };
```

Here is the call graph for this function:



6.1.2.4 AIB::AIB (const AIB & *orig*)

Copy constructor

Definition at line 14 of file [AIB.cpp](#).

```
00014     {
00015 }
```

6.1.2.5 AIB::~~AIB () [virtual]

de-constructor

Definition at line 17 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#).

```
00017     {
00018 }
```

Here is the caller graph for this function:



6.1.3 Member Function Documentation

6.1.3.1 void AIB::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>

Definition at line 37 of file [AIB.cpp](#).

References [SetAccountNumber\(\)](#).

Referenced by [operator=\(\)](#).

```
00037     {
00038
00039         std::shared_ptr<AIB> objTO = std::dynamic_pointer_cast<AIB>(to);
00040         std::shared_ptr<AIB> objFROM = std::dynamic_pointer_cast<AIB>(from);
00041         objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00042         objTO->Set_Version(objFROM->Get_Version());
00043         objTO->SetAccountNumber(objFROM->GetAccountNumber());
00044         objTO->SetBalance(objFROM->GetBalance());
00045     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.2 `int AIB::GetAccountNumber () const [virtual]`

Implements [BANK](#).

Definition at line 81 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00081                                     {  
00082     return accountNumber;  
00083 }
```

Here is the caller graph for this function:



6.1.3.3 `std::string AIB::GetAddress () const [virtual]`

Implements [BANK](#).

Definition at line 65 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#).

```
00065                                     {
00066     return address;
00067 }
```

Here is the caller graph for this function:



6.1.3.4 `double AIB::GetBalance () const [virtual]`

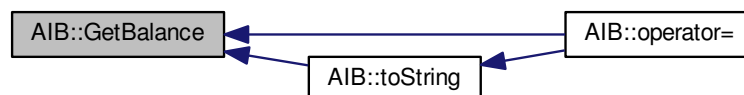
Implements [BANK](#).

Definition at line 73 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00073                                     {
00074     return balance;
00075 }
```

Here is the caller graph for this function:



6.1.3.5 `std::shared_ptr< OSTM > AIB::getBaseCopy (std::shared_ptr< OSTM > object) [virtual]`

`getBaseCopy` function, make deep copy of the object/pointer and Return a new `std::shared_ptr<BANK>` type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a <code>std::shared_ptr<BANK></code> return type

Definition at line 24 of file [AIB.cpp](#).

References [AIB\(\)](#).

Referenced by [operator=\(\)](#).

```

00025 {
00026
00027     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00028     std::shared_ptr<BANK> obj(new AIB(objTO, object->Get_Version(), object->Get_Unique_ID()));
00029     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00030     return ostm_obj;
00031 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.6 `std::string AIB::GetFirstName () const` `[virtual]`

Implements [BANK](#).

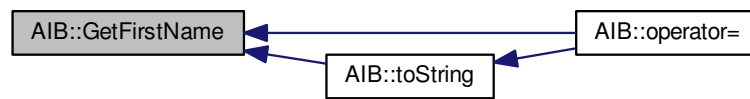
Definition at line 97 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00097                                     {
00098     return firstName;
00099 }
```

Here is the caller graph for this function:



6.1.3.7 `std::string AIB::GetFullName () const [virtual]`

Implements [BANK](#).

Definition at line 105 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#).

```
00105         {  
00106     return fullname;  
00107 }
```

Here is the caller graph for this function:



6.1.3.8 `std::string AIB::GetLastName () const [virtual]`

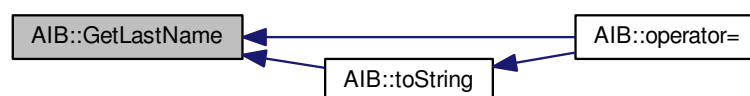
Implements [BANK](#).

Definition at line 89 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00089         {  
00090     return lastName;  
00091 }
```

Here is the caller graph for this function:



6.1.3.9 AIB AIB::operator= (const AIB & orig) [inline]

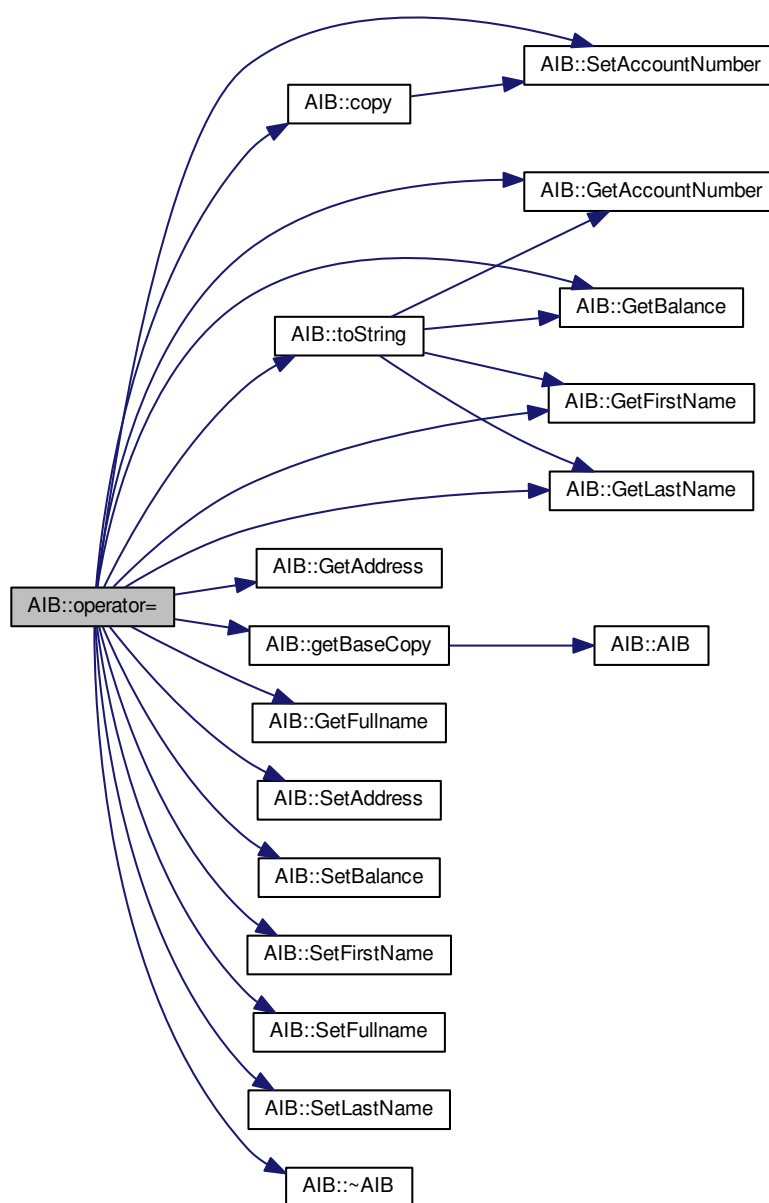
Operator

Definition at line 66 of file AIB.h.

References [copy\(\)](#), [GetAccountNumber\(\)](#), [GetAddress\(\)](#), [GetBalance\(\)](#), [getBaseCopy\(\)](#), [GetFirstName\(\)](#), [GetFullname\(\)](#), [GetLastName\(\)](#), [SetAccountNumber\(\)](#), [SetAddress\(\)](#), [SetBalance\(\)](#), [SetFirstName\(\)](#), [SetFullname\(\)](#), [SetLastName\(\)](#), [toString\(\)](#), and [~AIB\(\)](#).

```
00066 {};
```

Here is the call graph for this function:



6.1.3.10 void AIB::SetAccountNumber (int *accountNumber*) [virtual]

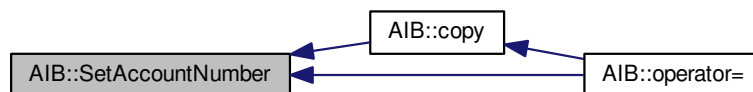
Implements [BANK](#).

Definition at line 77 of file [AIB.cpp](#).

Referenced by [copy\(\)](#), and [operator=\(\)](#).

```
00077                                     {  
00078     this->accountNumber = accountNumber;  
00079 }
```

Here is the caller graph for this function:

**6.1.3.11** void AIB::SetAddress (std::string *address*) [virtual]

Implements [BANK](#).

Definition at line 61 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#).

```
00061                                     {  
00062     this->address = address;  
00063 }
```

Here is the caller graph for this function:



6.1.3.12 void AIB::SetBalance (double *balance*) [virtual]

Implements [BANK](#).

Definition at line 69 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#).

```
00069                                     {  
00070     this->balance = balance;  
00071 }
```

Here is the caller graph for this function:



6.1.3.13 void AIB::SetFirstName (std::string *firstName*) [virtual]

Implements [BANK](#).

Definition at line 93 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#).

```
00093                                     {  
00094     this->firstName = firstName;  
00095 }
```

Here is the caller graph for this function:



6.1.3.14 void AIB::SetFullName (std::string *fullname*) [virtual]

Implements [BANK](#).

Definition at line 101 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#).

```
00101                                     {  
00102     this->fullname = fullname;  
00103 }
```

Here is the caller graph for this function:

**6.1.3.15** void AIB::SetLastName (std::string *lastName*) [virtual]

Implements [BANK](#).

Definition at line 85 of file [AIB.cpp](#).

Referenced by [operator=\(\)](#).

```
00085                                     {  
00086     this->lastName = lastName;  
00087 }
```

Here is the caller graph for this function:



6.1.3.16 void AIB::toString() [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

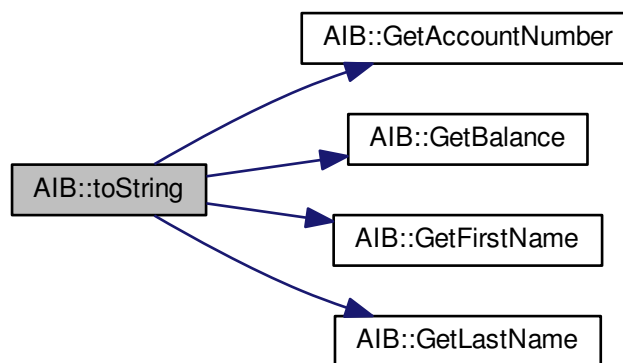
Definition at line 56 of file [AIB.cpp](#).

References [GetAccountNumber\(\)](#), [GetBalance\(\)](#), [GetFirstName\(\)](#), and [GetLastName\(\)](#).

Referenced by [operator=\(\)](#).

```
00057 {
00058     std::cout << "\nAIB BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " << this->
    GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
    " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
    number : " << this->Get_Version() << std::endl;
00059 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



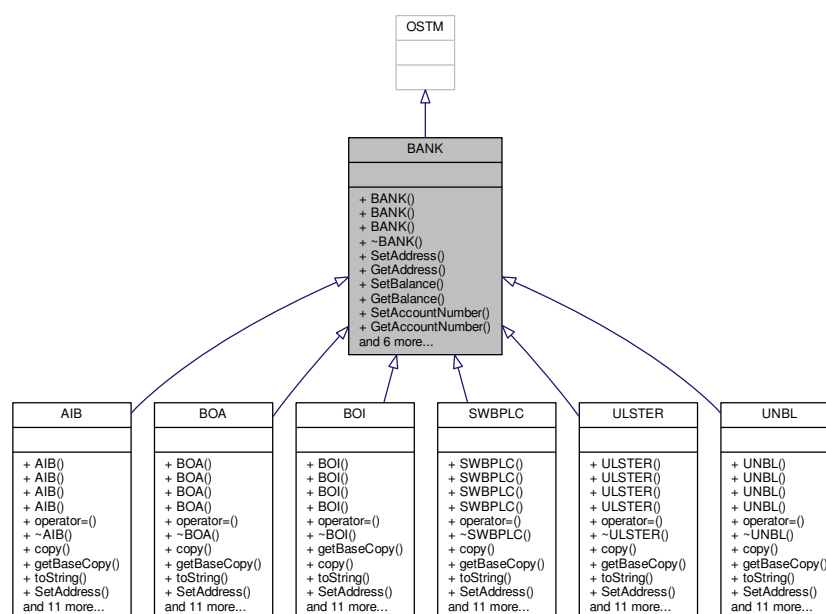
The documentation for this class was generated from the following files:

- [AIB.h](#)
- [AIB.cpp](#)

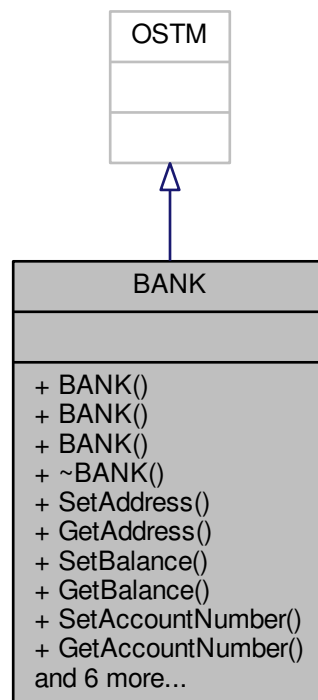
6.2 BANK Class Reference

```
#include <BANK.h>
```

Inheritance diagram for BANK:



Collaboration diagram for BANK:



Public Member Functions

- [BANK](#) ()
- [BANK](#) (int _version, int _unique_id)
- [BANK](#) (const [BANK](#) &orig)
- virtual [~BANK](#) ()
- virtual void [SetAddress](#) (std::string address)=0
- virtual std::string [GetAddress](#) () const =0
- virtual void [SetBalance](#) (double balance)=0
- virtual double [GetBalance](#) () const =0
- virtual void [SetAccountNumber](#) (int accountNumber)=0
- virtual int [GetAccountNumber](#) () const =0
- virtual void [SetLastName](#) (std::string lastName)=0
- virtual std::string [GetLastName](#) () const =0
- virtual void [SetFirstName](#) (std::string firstName)=0
- virtual std::string [GetFirstName](#) () const =0
- virtual void [SetFullname](#) (std::string fullname)=0
- virtual std::string [GetFullname](#) () const =0

6.2.1 Detailed Description

[BANK](#) inherit from the OSTM library. It is declares the common functions in the child classes as a virtual function.

Definition at line 16 of file [BANK.h](#).

6.2.2 Constructor & Destructor Documentation

6.2.2.1 BANK::BANK() [inline]

Constructor

Definition at line 23 of file [BANK.h](#).

Referenced by [BANK\(\)](#).

```
00023         : OSTM() {  
00024  
00025     };
```

Here is the caller graph for this function:



6.2.2.2 BANK::BANK(int _version, int _unique_id) [inline]

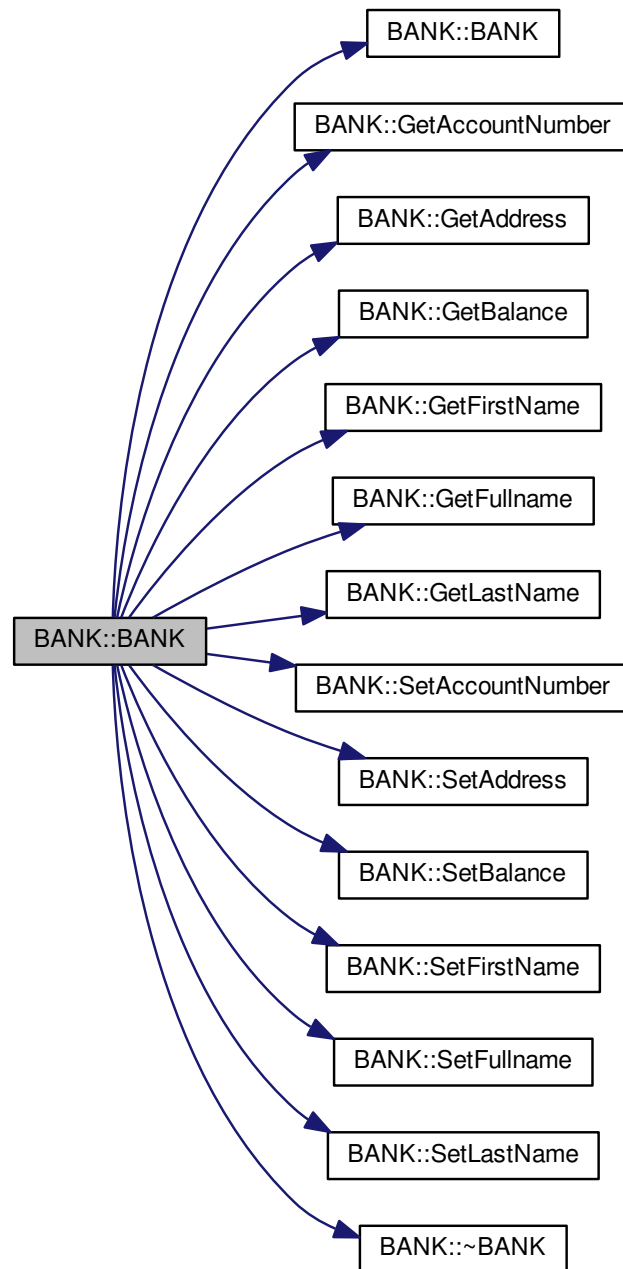
Custom Constructor

Definition at line 29 of file [BANK.h](#).

References [BANK\(\)](#), [GetAccountNumber\(\)](#), [GetAddress\(\)](#), [GetBalance\(\)](#), [GetFirstName\(\)](#), [GetFullname\(\)](#), [GetLastName\(\)](#), [SetAccountNumber\(\)](#), [SetAddress\(\)](#), [SetBalance\(\)](#), [SetFirstName\(\)](#), [SetFullname\(\)](#), [SetLastName\(\)](#), and [~BANK\(\)](#).

```
00029         : OSTM(_version, _unique_id){  
00030  
00031     };
```

Here is the call graph for this function:



6.2.2.3 BANK::BANK (const BANK & orig)

Copy constructor

Definition at line 11 of file [BANK.cpp](#).

```

00011         {
00012     }
  
```

6.2.2.4 BANK::~~BANK () [virtual]

de-constructor

Definition at line 14 of file [BANK.cpp](#).

Referenced by [BANK\(\)](#).

```
00014         {  
00015     }
```

Here is the caller graph for this function:



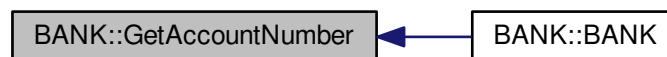
6.2.3 Member Function Documentation

6.2.3.1 virtual int BANK::GetAccountNumber () const [pure virtual]

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:



6.2.3.2 virtual std::string BANK::GetAddress () const [pure virtual]

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:



6.2.3.3 virtual double `BANK::GetBalance () const` [pure virtual]

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:



6.2.3.4 virtual std::string `BANK::GetFirstName () const` [pure virtual]

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:



6.2.3.5 virtual std::string `BANK::GetFullname () const` [pure virtual]

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:



6.2.3.6 `virtual std::string BANK::GetLastName () const [pure virtual]`

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:

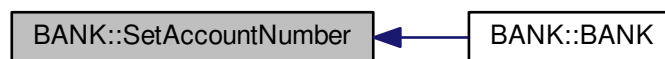


6.2.3.7 `virtual void BANK::SetAccountNumber (int accountNumber) [pure virtual]`

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:



6.2.3.8 `virtual void BANK::SetAddress (std::string address) [pure virtual]`

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:

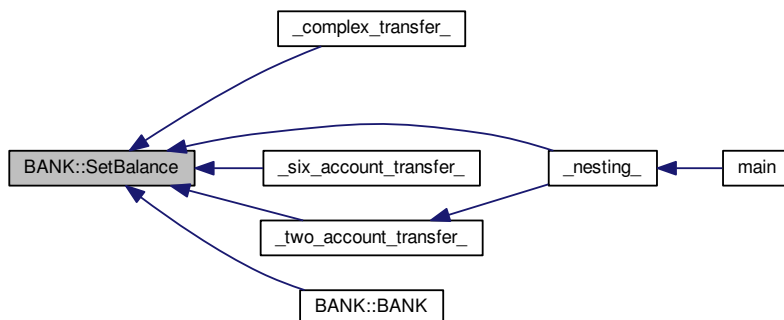


6.2.3.9 virtual void BANK::SetBalance (double *balance*) [pure virtual]

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [_complex_transfer_\(\)](#), [_nesting_\(\)](#), [_six_account_transfer_\(\)](#), [_two_account_transfer_\(\)](#), and [BANK::K\(\)](#).

Here is the caller graph for this function:



6.2.3.10 virtual void BANK::SetFirstName (std::string *firstName*) [pure virtual]

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:

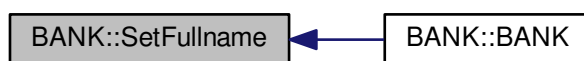


6.2.3.11 virtual void BANK::SetFullname (std::string *fullname*) [pure virtual]

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:



6.2.3.12 `virtual void BANK::SetLastName (std::string lastName) [pure virtual]`

Implemented in [AIB](#), [BOA](#), [BOI](#), [SWBPLC](#), [ULSTER](#), and [UNBL](#).

Referenced by [BANK\(\)](#).

Here is the caller graph for this function:



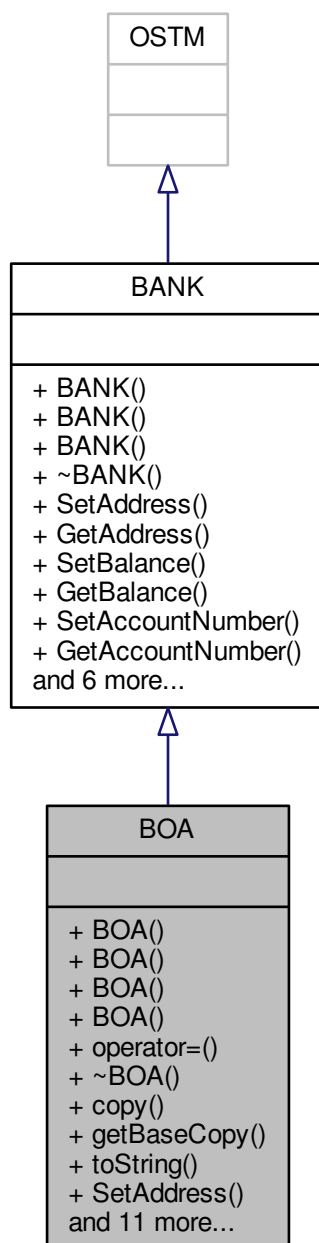
The documentation for this class was generated from the following files:

- [BANK.h](#)
- [BANK.cpp](#)

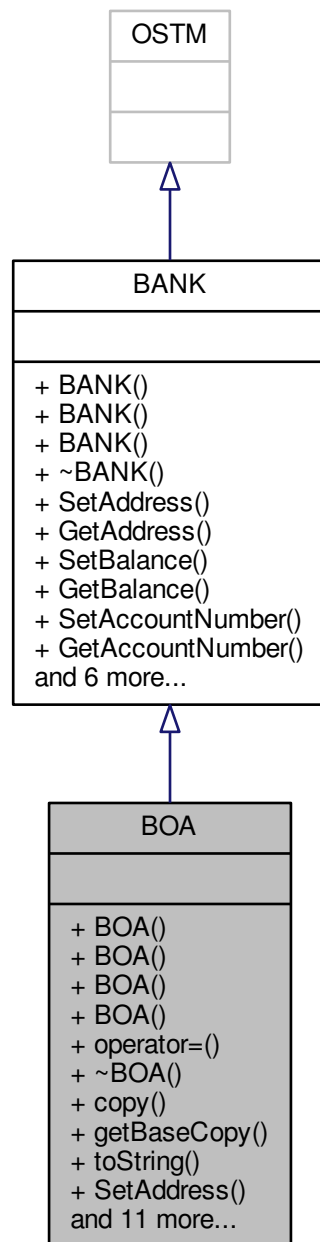
6.3 BOA Class Reference

```
#include <BOA.h>
```

Inheritance diagram for BOA:



Collaboration diagram for BOA:



Public Member Functions

- [BOA](#) ()
- [BOA](#) (int accountNumber, double balance, std::string firstName, std::string lastName, std::string address)
- [BOA](#) (std::shared_ptr< [BANK](#) > obj, int _version, int _unique_id)
- [BOA](#) (const [BOA](#) &orig)
- [BOA operator=](#) (const [BOA](#) &orig)

- virtual `~BOA ()`
- virtual void `copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)`
copy function, make deep copy of the object/pointer
- virtual `std::shared_ptr< OSTM > getBaseCopy (std::shared_ptr< OSTM > object)`
getBaseCopy function, make deep copy of the object/pointer and Return a new std::shared_ptr< BANK> type object
- virtual void `toString ()`
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void `SetAddress (std::string address)`
- virtual `std::string GetAddress () const`
- virtual void `SetBalance (double balance)`
- virtual double `GetBalance () const`
- virtual void `SetAccountNumber (int accountNumber)`
- virtual int `GetAccountNumber () const`
- virtual void `SetLastName (std::string lastName)`
- virtual `std::string GetLastName () const`
- virtual void `SetFirstName (std::string firstName)`
- virtual `std::string GetFirstName () const`
- virtual void `SetFullname (std::string fullname)`
- virtual `std::string GetFullname () const`

6.3.1 Detailed Description

Inherit from [BANK](#)

Definition at line 18 of file [BOA.h](#).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `BOA::BOA () [inline]`

Constructor

Definition at line 24 of file [BOA.h](#).

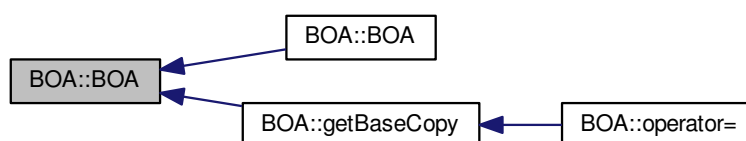
Referenced by [BOA\(\)](#), and [getBaseCopy\(\)](#).

```

00024         : BANK() {
00025             this->accountNumber = 0;
00026             this->balance = 50;
00027             this->firstName = "Joe";
00028             this->lastName = "Blog";
00029             this->address = "High street, Carlow";
00030             this->fullname = firstName + " " + lastName;
00031     };

```

Here is the caller graph for this function:



6.3.2.2 BOA::BOA (int *accountNumber*, double *balance*, std::string *firstName*, std::string *lastName*, std::string *address*) [inline]

Custom constructor

Definition at line 35 of file [BOA.h](#).

```

00035                                     :
00036     BANK() {
00037         this->accountNumber = accountNumber;
00037         this->balance = balance;
00038         this->firstName = firstName;
00039         this->lastName = lastName;
00040         this->address = address;
00041         this->fullname = firstName + " " + lastName;
00042     };

```

6.3.2.3 BOA::BOA (std::shared_ptr< BANK > *obj*, int *_version*, int *_unique_id*) [inline]

Custom constructor, used by the library for deep copying

Definition at line 46 of file [BOA.h](#).

References [BOA\(\)](#).

```

00046                                     : BANK(_version, _unique_id) {
00047
00048         this->accountNumber = obj->GetAccountNumber();
00049         this->balance = obj->GetBalance();
00050         this->firstName = obj->GetFirstName();
00051         this->lastName = obj->GetLastName();
00052         this->address = obj->GetAddress();
00053         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00054     };

```

Here is the call graph for this function:



6.3.2.4 BOA::BOA (const BOA & *orig*)

Copy constructor

Definition at line 12 of file [BOA.cpp](#).

```

00012     {
00013 }

```

6.3.2.5 BOA::~BOA () [virtual]

de-constructor

Definition at line 15 of file BOA.cpp.

Referenced by [operator=\(\)](#).

```
00015         {
00016     }
```

Here is the caller graph for this function:



6.3.3 Member Function Documentation

6.3.3.1 void BOA::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>

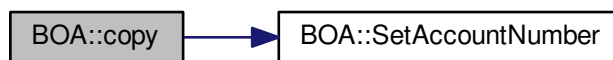
Definition at line 34 of file BOA.cpp.

References [SetAccountNumber\(\)](#).

Referenced by [operator=\(\)](#).

```
00034                                     {
00035
00036     std::shared_ptr<BOA> objTO = std::dynamic_pointer_cast<BOA>(to);
00037     std::shared_ptr<BOA> objFROM = std::dynamic_pointer_cast<BOA>(from);
00038     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00039     objTO->Set_Version(objFROM->Get_Version());
00040     objTO->SetAccountNumber(objFROM->GetAccountNumber());
00041     objTO->SetBalance(objFROM->GetBalance());
00042
00043 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.3.2 `int BOA::GetAccountNumber () const [virtual]`

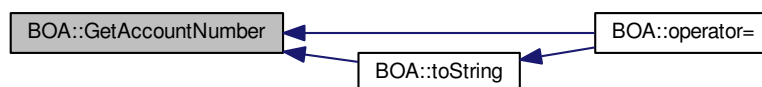
Implements [BANK](#).

Definition at line 80 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00080                                     {  
00081     return accountNumber;  
00082 }
```

Here is the caller graph for this function:



6.3.3.3 `std::string BOA::GetAddress () const [virtual]`

Implements [BANK](#).

Definition at line 64 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#).

```
00064                                     {
00065     return address;
00066 }
```

Here is the caller graph for this function:



6.3.3.4 `double BOA::GetBalance () const [virtual]`

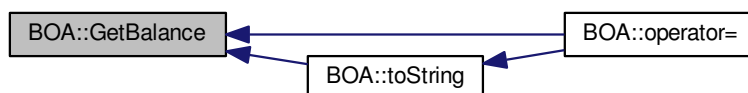
Implements [BANK](#).

Definition at line 72 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00072                                     {
00073     return balance;
00074 }
```

Here is the caller graph for this function:



6.3.3.5 `std::shared_ptr<OSTM> BOA::getBaseCopy (std::shared_ptr<OSTM> object) [virtual]`

`getBaseCopy` function, make deep copy of the object/pointer and Return a new `std::shared_ptr<BANK>` type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a <code>std::shared_ptr<BANK></code> return type

Definition at line 22 of file [BOA.cpp](#).

References [BOA\(\)](#).

Referenced by [operator=\(\)](#).

```

00023 {
00024     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00025     std::shared_ptr<BANK> obj(new BOA(objTO, object->Get_Version(), object->Get_Unique_ID()));
00026     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00027     return ostm_obj;
00028 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.3.6 `std::string BOA::GetFirstName () const` [virtual]

Implements [BANK](#).

Definition at line 96 of file [BOA.cpp](#).

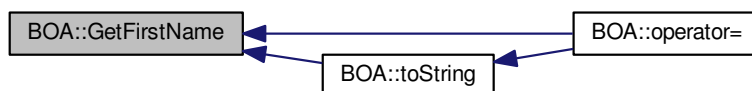
Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00096                                     {
00097     return firstName;
00098 }

```

Here is the caller graph for this function:



6.3.3.7 `std::string BOA::GetFullName () const [virtual]`

Implements [BANK](#).

Definition at line 104 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#).

```

00104         {
00105     return fullname;
00106 }
  
```

Here is the caller graph for this function:



6.3.3.8 `std::string BOA::GetLastName () const [virtual]`

Implements [BANK](#).

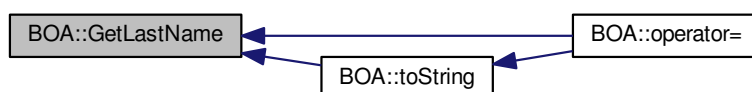
Definition at line 88 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00088         {
00089     return lastName;
00090 }
  
```

Here is the caller graph for this function:



6.3.3.9 BOA BOA::operator= (const BOA & orig) [inline]

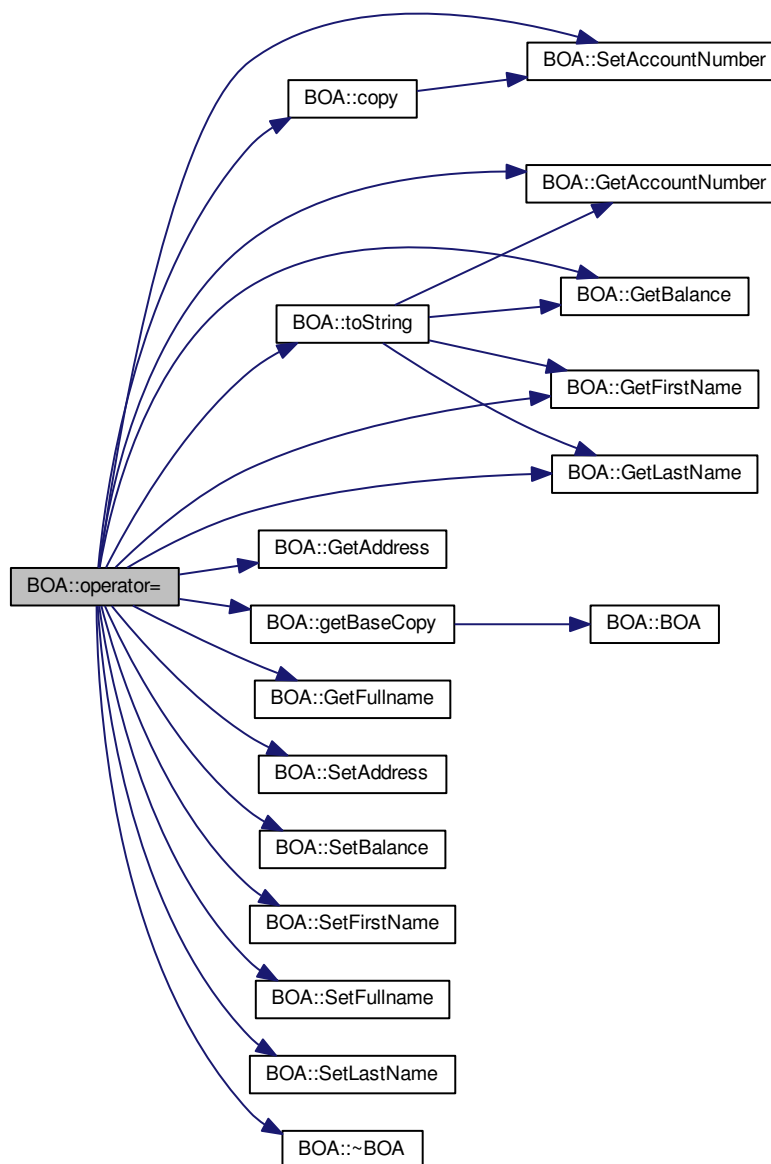
Operator

Definition at line 64 of file BOA.h.

References [copy\(\)](#), [GetAccountNumber\(\)](#), [GetAddress\(\)](#), [GetBalance\(\)](#), [getBaseCopy\(\)](#), [GetFirstName\(\)](#), [GetFullname\(\)](#), [GetLastName\(\)](#), [SetAccountNumber\(\)](#), [SetAddress\(\)](#), [SetBalance\(\)](#), [SetFirstName\(\)](#), [SetFullname\(\)](#), [SetLastName\(\)](#), [toString\(\)](#), and [~BOA\(\)](#).

```
00064                                     {
00065     };
```

Here is the call graph for this function:



6.3.3.10 void BOA::SetAccountNumber (int *accountNumber*) [virtual]

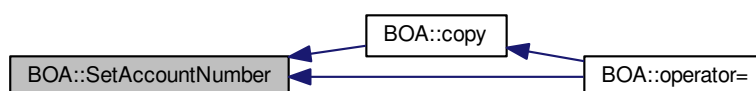
Implements [BANK](#).

Definition at line 76 of file [BOA.cpp](#).

Referenced by [copy\(\)](#), and [operator=\(\)](#).

```
00076                                     {  
00077     this->accountNumber = accountNumber;  
00078 }
```

Here is the caller graph for this function:



6.3.3.11 void BOA::SetAddress (std::string *address*) [virtual]

Implements [BANK](#).

Definition at line 60 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#).

```
00060                                     {  
00061     this->address = address;  
00062 }
```

Here is the caller graph for this function:



6.3.3.12 void BOA::SetBalance (double *balance*) [virtual]

Implements [BANK](#).

Definition at line 68 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#).

```
00068                                     {  
00069     this->balance = balance;  
00070 }
```

Here is the caller graph for this function:

**6.3.3.13** void BOA::SetFirstName (std::string *firstName*) [virtual]

Implements [BANK](#).

Definition at line 92 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#).

```
00092                                     {  
00093     this->firstName = firstName;  
00094 }
```

Here is the caller graph for this function:



6.3.3.14 void BOA::SetFullname (std::string *fullname*) [virtual]

Implements [BANK](#).

Definition at line 100 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#).

```
00100                                     {  
00101     this->fullname = fullname;  
00102 }
```

Here is the caller graph for this function:



6.3.3.15 void BOA::SetLastName (std::string *lastName*) [virtual]

Implements [BANK](#).

Definition at line 84 of file [BOA.cpp](#).

Referenced by [operator=\(\)](#).

```
00084                                     {  
00085     this->lastName = lastName;  
00086 }
```

Here is the caller graph for this function:



6.3.3.16 void BOA::toString () [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

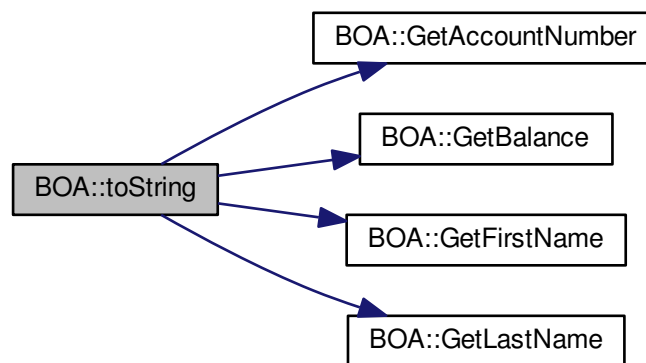
Definition at line 54 of file [BOA.cpp](#).

References [GetAccountNumber\(\)](#), [GetBalance\(\)](#), [GetFirstName\(\)](#), and [GetLastName\(\)](#).

Referenced by [operator=\(\)](#).

```
00055 {
00056     // std::cout << "\nUnique ID : " << this->GetUniqueID() << "\nInt value : " << this->GetV_int() <<
        "\nDouble value : " << this->GetV_double() << "\nFloat value : " << this->GetV_float() << "\nString value : " <<
        this->GetV_string() << "\nVersion number : " << this->GetVersion() << "\nLoad Counter : "<<
        this->GetLoadCounter() << "\nWrite Counter : "<< this->GetWriteCounter() << std::endl;
00057     std::cout << "\nBOA BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " << this->
        GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
        " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
        number : " << this->Get_Version() << std::endl;
00058 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



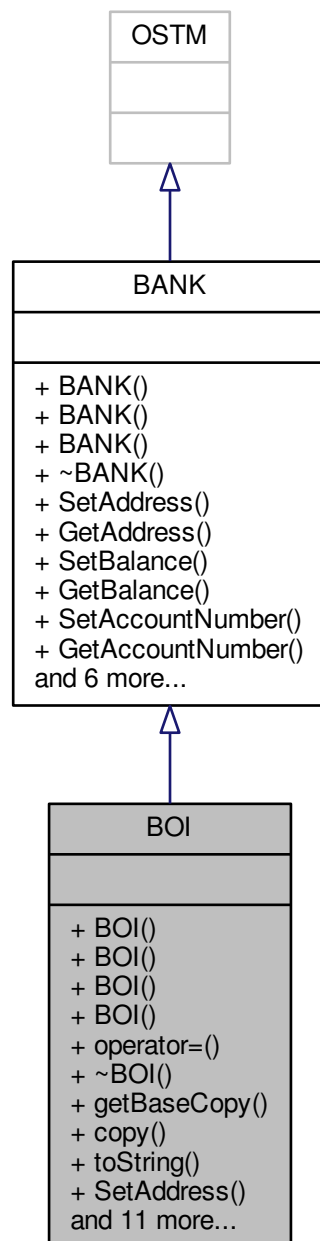
The documentation for this class was generated from the following files:

- [BOA.h](#)
- [BOA.cpp](#)

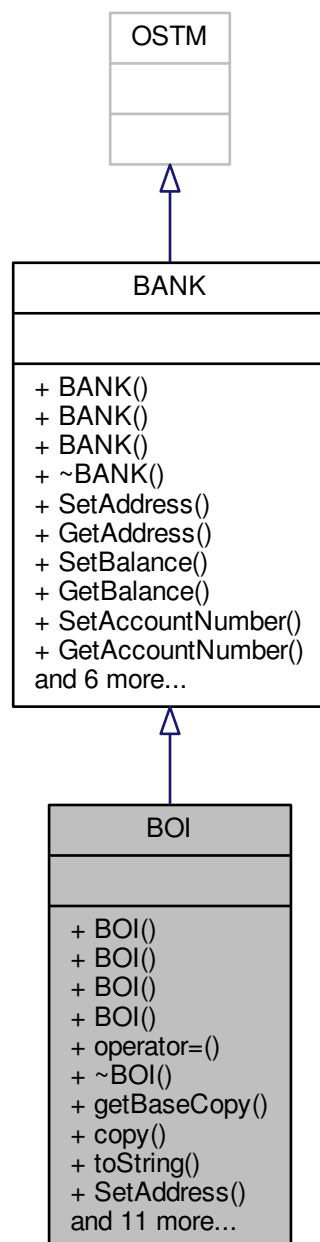
6.4 BOI Class Reference

```
#include <BOI.h>
```

Inheritance diagram for BOI:



Collaboration diagram for BOI:



Public Member Functions

- `BOI ()`
- `BOI (int accountNumber, double balance, std::string firstName, std::string lastName, std::string address)`
- `BOI (std::shared_ptr< BOI > obj, int _version, int _unique_id)`
- `BOI (const BOI &orig)`
- `BOI operator= (const BOI &orig)`

- virtual `~BOI ()`
- virtual `std::shared_ptr< OSTM > getBaseCopy (std::shared_ptr< OSTM > object)`
getBaseCopy function, make deep copy of the object/pointer and Return a new BANK type object*
- virtual void `copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)`
copy function, make deep copy of the object/pointer
- virtual void `toString ()`
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void `SetAddress (std::string address)`
- virtual `std::string GetAddress () const`
- virtual void `SetBalance (double balance)`
- virtual `double GetBalance () const`
- virtual void `SetAccountNumber (int accountNumber)`
- virtual `int GetAccountNumber () const`
- virtual void `SetLastName (std::string lastName)`
- virtual `std::string GetLastName () const`
- virtual void `SetFirstName (std::string firstName)`
- virtual `std::string GetFirstName () const`
- virtual void `SetFullname (std::string fullname)`
- virtual `std::string GetFullname () const`

6.4.1 Detailed Description

Inherit from [BANK](#)

Definition at line 19 of file [BOI.h](#).

6.4.2 Constructor & Destructor Documentation

6.4.2.1 `BOI::BOI () [inline]`

Constructor

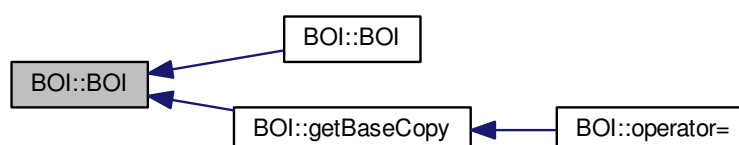
Definition at line 24 of file [BOI.h](#).

Referenced by [BOI\(\)](#), and [getBaseCopy\(\)](#).

```

00024         : BANK()
00025     {
00026         this->accountNumber = 0;
00027         this->balance = 50;
00028         this->firstName = "Joe";
00029         this->lastName = "Blog";
00030         this->address = "High street, Carlow";
00031         this->fullname = firstName + " " + lastName;
00032     }
00033 
```

Here is the caller graph for this function:



6.4.2.2 BOI::BOI(int *accountNumber*, double *balance*, std::string *firstName*, std::string *lastName*, std::string *address*) [inline]

Custom constructor

Definition at line 37 of file [BOI.h](#).

```

00037                                     :
00038     BANK()
00039     {
00040         this->accountNumber = accountNumber;
00041         this->balance = balance;
00042         this->firstName = firstName;
00043         this->lastName = lastName;
00044         this->address = address;
00045         this->fullname = firstName + " " + lastName;
00046     };

```

6.4.2.3 BOI::BOI(std::shared_ptr< BOI > *obj*, int *_version*, int *_unique_id*) [inline]

Custom constructor, used by the library for deep copying

Definition at line 49 of file [BOI.h](#).

References [BOI\(\)](#).

```

00049                                     : BANK(_version, _unique_id)
00050     {
00051         this->accountNumber = obj->GetAccountNumber();
00052         this->balance = obj->GetBalance();
00053         this->firstName = obj->GetFirstName();
00054         this->lastName = obj->GetLastName();
00055         this->address = obj->GetAddress();
00056         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00057     };

```

Here is the call graph for this function:



6.4.2.4 BOI::BOI(const BOI & *orig*)

Copy constructor

Definition at line 15 of file [BOI.cpp](#).

```

00015     {
00016 }

```

6.4.2.5 BOI::~BOI() [virtual]

de-constructor

Definition at line 12 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#).

```
00012         {
00013     }
```

Here is the caller graph for this function:



6.4.3 Member Function Documentation

6.4.3.1 void BOI::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>

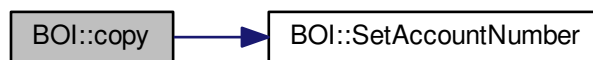
Definition at line 35 of file [BOI.cpp](#).

References [SetAccountNumber\(\)](#).

Referenced by [operator=\(\)](#).

```
00035                                     {
00036
00037     std::shared_ptr<BOI> objTO = std::dynamic_pointer_cast<BOI>(to);
00038     std::shared_ptr<BOI> objFROM = std::dynamic_pointer_cast<BOI>(from);
00039     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00040     objTO->Set_Version(objFROM->Get_Version());
00041     objTO->SetAccountNumber(objFROM->GetAccountNumber());
00042     objTO->SetBalance(objFROM->GetBalance());
00043 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.3.2 int BOI::GetAccountNumber () const [virtual]

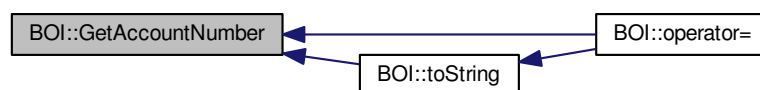
Implements [BANK](#).

Definition at line 78 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00078                                     {  
00079     return accountNumber;  
00080 }
```

Here is the caller graph for this function:



6.4.3.3 `std::string BOI::GetAddress () const [virtual]`

Implements [BANK](#).

Definition at line 62 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#).

```
00062                                     {
00063     return address;
00064 }
```

Here is the caller graph for this function:



6.4.3.4 `double BOI::GetBalance () const [virtual]`

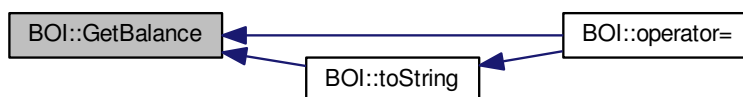
Implements [BANK](#).

Definition at line 70 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00070                                     {
00071     return balance;
00072 }
```

Here is the caller graph for this function:



6.4.3.5 `std::shared_ptr< OSTM > BOI::getBaseCopy (std::shared_ptr< OSTM > object) [virtual]`

`getBaseCopy` function, make deep copy of the object/pointer and Return a new `BANK*` type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a BANK* return type

Definition at line 22 of file [BOI.cpp](#).

References [BOI\(\)](#).

Referenced by [operator=\(\)](#).

```

00023 {
00024
00025     std::shared_ptr<BOI> objTO = std::dynamic_pointer_cast<BOI>(object);
00026     std::shared_ptr<BOI> obj(new BOI(objTO,object->Get_Version(),object->Get_Unique_ID()));
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00028     return ostm_obj;
00029 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.3.6 `std::string BOI::GetFirstName () const` `[virtual]`

Implements [BANK](#).

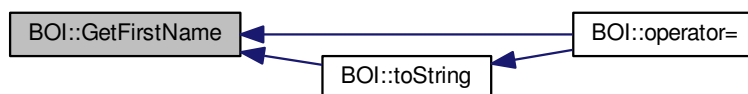
Definition at line 94 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00094                                     {
00095     return firstName;
00096 }
```

Here is the caller graph for this function:



6.4.3.7 `std::string BOl::GetFullname () const [virtual]`

Implements [BANK](#).

Definition at line 102 of file [BOl.cpp](#).

Referenced by [operator=\(\)](#).

```

00102         {
00103     return fullname;
00104 }
  
```

Here is the caller graph for this function:



6.4.3.8 `std::string BOl::GetLastName () const [virtual]`

Implements [BANK](#).

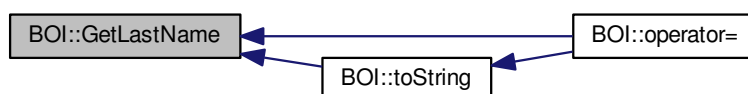
Definition at line 86 of file [BOl.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00086         {
00087     return lastName;
00088 }
  
```

Here is the caller graph for this function:



6.4.3.9 BOI BOI::operator= (const BOI & orig) [inline]

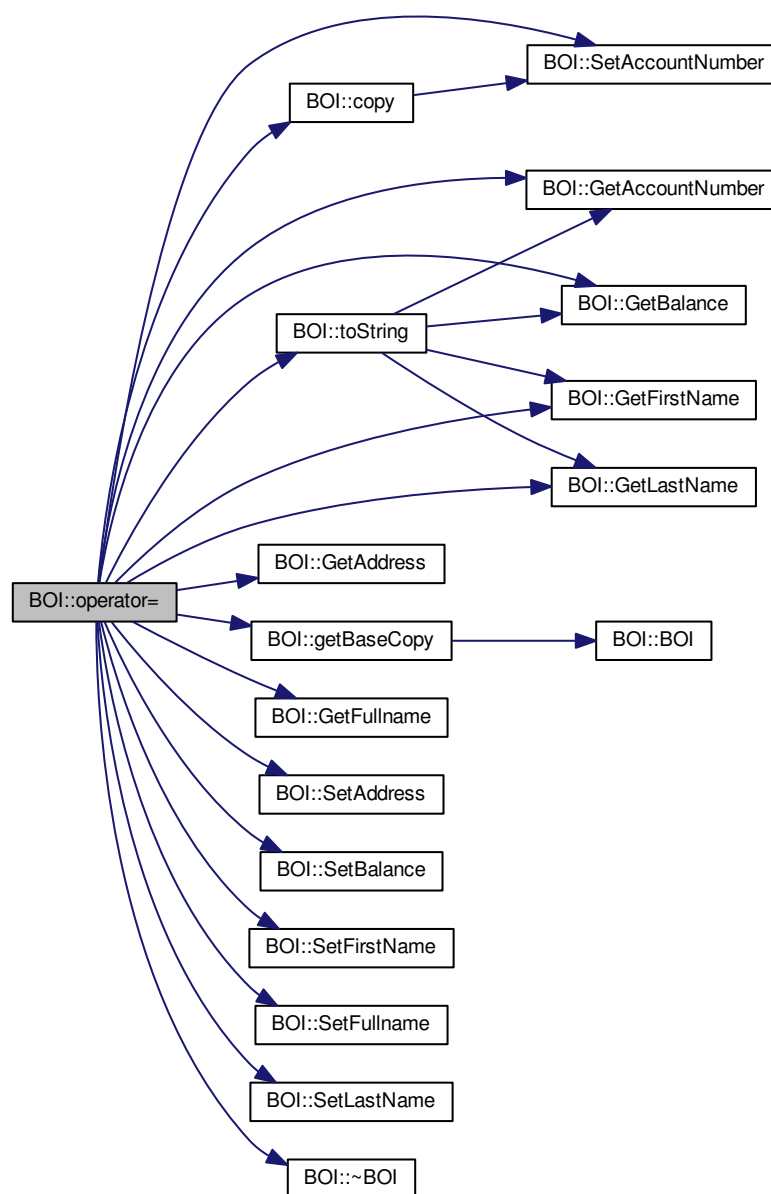
Operator

Definition at line 65 of file BOI.h.

References [copy\(\)](#), [GetAccountNumber\(\)](#), [GetAddress\(\)](#), [GetBalance\(\)](#), [getBaseCopy\(\)](#), [GetFirstName\(\)](#), [GetFullname\(\)](#), [GetLastName\(\)](#), [SetAccountNumber\(\)](#), [SetAddress\(\)](#), [SetBalance\(\)](#), [SetFirstName\(\)](#), [SetFullname\(\)](#), [SetLastName\(\)](#), [toString\(\)](#), and [~BOI\(\)](#).

```
00065 {};
```

Here is the call graph for this function:



6.4.3.10 void BOI::SetAccountNumber (int *accountNumber*) [virtual]

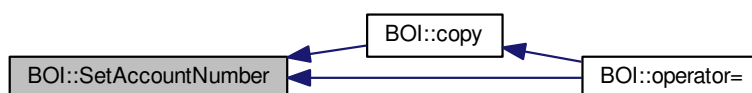
Implements [BANK](#).

Definition at line 74 of file [BOI.cpp](#).

Referenced by [copy\(\)](#), and [operator=\(\)](#).

```
00074                                     {  
00075     this->accountNumber = accountNumber;  
00076 }
```

Here is the caller graph for this function:



6.4.3.11 void BOI::SetAddress (std::string *address*) [virtual]

Implements [BANK](#).

Definition at line 58 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#).

```
00058                                     {  
00059     this->address = address;  
00060 }
```

Here is the caller graph for this function:



6.4.3.12 void BOI::SetBalance (double *balance*) [virtual]

Implements [BANK](#).

Definition at line 66 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#).

```
00066                                     {  
00067     this->balance = balance;  
00068 }
```

Here is the caller graph for this function:

**6.4.3.13** void BOI::SetFirstName (std::string *firstName*) [virtual]

Implements [BANK](#).

Definition at line 90 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#).

```
00090                                     {  
00091     this->firstName = firstName;  
00092 }
```

Here is the caller graph for this function:



6.4.3.14 void BOI::SetFullname (std::string *fullname*) [virtual]

Implements [BANK](#).

Definition at line 98 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#).

```
00098                                     {  
00099     this->fullname = fullname;  
00100 }
```

Here is the caller graph for this function:



6.4.3.15 void BOI::SetLastName (std::string *lastName*) [virtual]

Implements [BANK](#).

Definition at line 82 of file [BOI.cpp](#).

Referenced by [operator=\(\)](#).

```
00082                                     {  
00083     this->lastName = lastName;  
00084 }
```

Here is the caller graph for this function:



6.4.3.16 void BOI::toString() [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

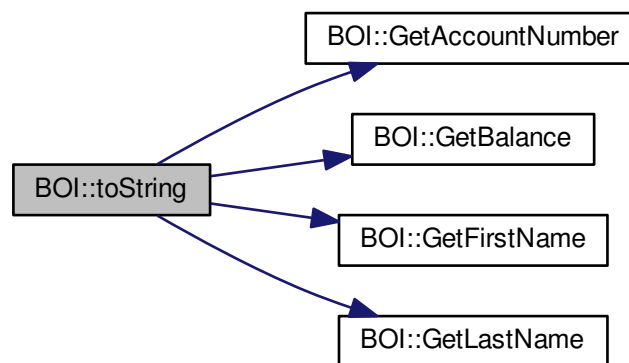
Definition at line 54 of file [BOI.cpp](#).

References [GetAccountNumber\(\)](#), [GetBalance\(\)](#), [GetFirstName\(\)](#), and [GetLastName\(\)](#).

Referenced by [operator=\(\)](#).

```
00055 {
00056     std::cout << "\nBOI BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " << this->
    GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
    " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
    number : " << this->Get_Version() << std::endl;
00057 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



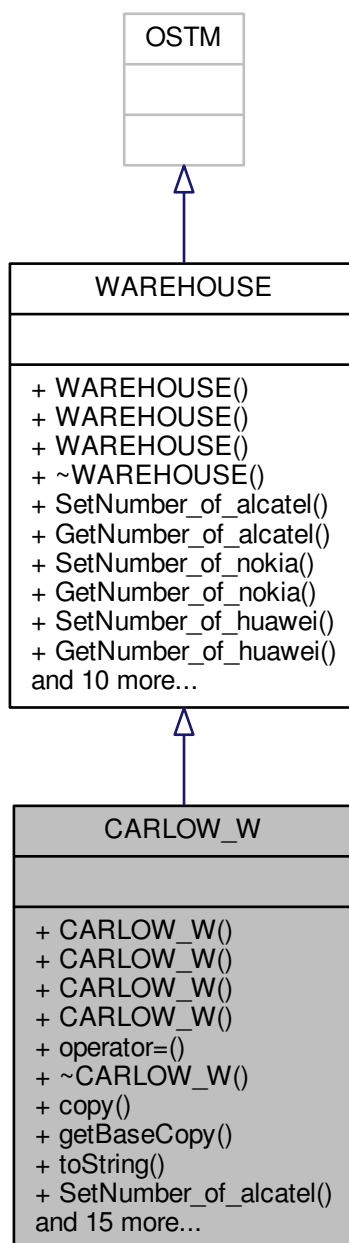
The documentation for this class was generated from the following files:

- [BOI.h](#)
- [BOI.cpp](#)

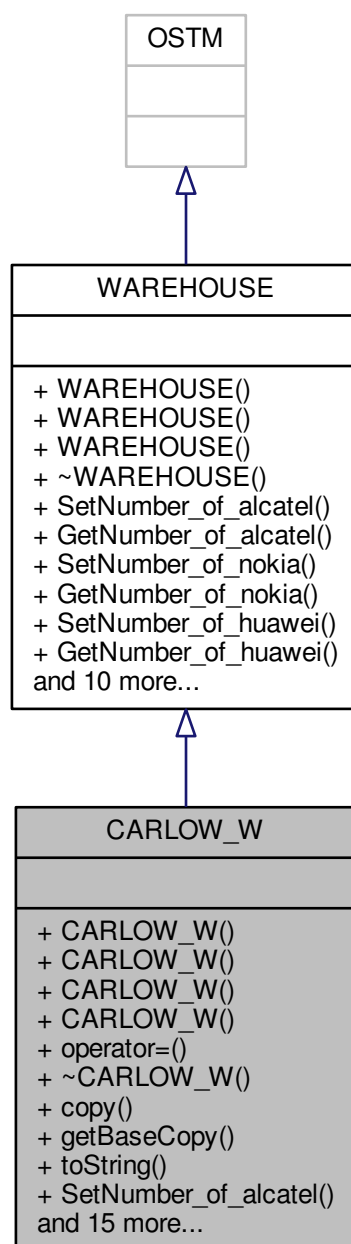
6.5 CARLOW_W Class Reference

```
#include <CARLOW_W.h>
```

Inheritance diagram for CARLOW_W:



Collaboration diagram for CARLOW_W:



Public Member Functions

- [CARLOW_W](#) ()
- [CARLOW_W](#) (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel)
- [CARLOW_W](#) (std::shared_ptr< [WAREHOUSE](#) > obj, int _version, int _unique_id)
- [CARLOW_W](#) (const [CARLOW_W](#) &orig)

- [CARLOW_W operator=](#) (const [CARLOW_W](#) &orig)
- virtual [~CARLOW_W](#) ()
- virtual void [copy](#) (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)
copy function, make deep copy of the object/pointer
- virtual std::shared_ptr< OSTM > [getBaseCopy](#) (std::shared_ptr< OSTM > object)
getBaseCopy function, make deep copy of the object/pointer and Return a new BANK type object*
- virtual void [toString](#) ()
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void [SetNumber_of_alcatel](#) (int _number_of_alcatel)
- virtual int [GetNumber_of_alcatel](#) ()
- virtual void [SetNumber_of_nokia](#) (int _number_of_nokia)
- virtual int [GetNumber_of_nokia](#) ()
- virtual void [SetNumber_of_huawei](#) (int _number_of_huawei)
- virtual int [GetNumber_of_huawei](#) ()
- virtual void [SetNumber_of_sony](#) (int _number_of_sony)
- virtual int [GetNumber_of_sony](#) ()
- virtual void [SetNumber_of_samsung](#) (int _number_of_samsung)
- virtual int [GetNumber_of_samsung](#) ()
- virtual void [SetNumber_of_iphones](#) (int _number_of_iphones)
- virtual int [GetNumber_of_iphones](#) ()
- virtual void [SetShop_name](#) (std::string _shop_name)
- virtual std::string [GetShop_name](#) ()
- virtual void [SetShop_address](#) (std::string _shop_address)
- virtual std::string [GetShop_address](#) ()

6.5.1 Detailed Description

Inherit from [WAREHOUSE](#)

Definition at line 19 of file [CARLOW_W.h](#).

6.5.2 Constructor & Destructor Documentation

6.5.2.1 [CARLOW_W::CARLOW_W](#) () [inline]

Constructor

Definition at line 24 of file [CARLOW_W.h](#).

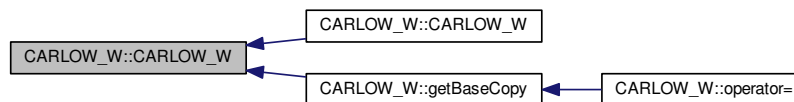
Referenced by [CARLOW_W\(\)](#), and [getBaseCopy\(\)](#).

```

00024         : WAREHOUSE() {
00025
00026         this->_shop_address = "Carlow potato street";
00027         this->_shop_name = "CARLOW C_WAREHOUSE";
00028         this->_number_of_iphones = 200;
00029         this->_number_of_samsung = 200;
00030         this->_number_of_sony = 200;
00031         this->_number_of_huawei = 200;
00032         this->_number_of_nokia = 200;
00033         this->_number_of_alcatel = 200;
00034     };

```


Here is the caller graph for this function:



6.5.2.2 CARLOW_W::CARLOW_W (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel) [inline]

Custom constructor

Definition at line 38 of file [CARLOW_W.h](#).

```

00038
                                : WAREHOUSE () {
00039     /*
00040      * copy over values
00041      */
00042     this->_shop_address = address;
00043     this->_shop_name = shop_name;
00044     this->_number_of_iphones = iphone;
00045     this->_number_of_samsung = samsung;
00046     this->_number_of_sony = sony;
00047     this->_number_of_huawei = huawei;
00048     this->_number_of_nokia = nokia;
00049     this->_number_of_alcatel = alcatel;
00050
00051 };
  
```

6.5.2.3 CARLOW_W::CARLOW_W (std::shared_ptr< WAREHOUSE > obj, int_version, int_unique_id) [inline]

Custom constructor, used by the library for deep copying

Definition at line 55 of file [CARLOW_W.h](#).

References [CARLOW_W\(\)](#).

```

00055
                                :
00056     WAREHOUSE(_version, _unique_id) {
00057     /*
00058      * copy over values
00059      */
00059     this->_shop_address = obj->GetShop_address();
00060     this->_shop_name = obj->GetShop_name();
00061     this->_number_of_iphones = obj->GetNumber_of_iphones();
00062     this->_number_of_samsung = obj->GetNumber_of_samsung();
00063     this->_number_of_sony = obj->GetNumber_of_sony();
00064     this->_number_of_huawei = obj->GetNumber_of_huawei();
00065     this->_number_of_nokia = obj->GetNumber_of_nokia();
00066     this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067 };
  
```

Here is the call graph for this function:



6.5.2.4 CARLOW_W::CARLOW_W (const CARLOW_W & orig)

Copy constructor

Definition at line 17 of file [CARLOW_W.cpp](#).

```
00017                                     {
00018 }
```

6.5.2.5 CARLOW_W::~~CARLOW_W () [virtual]

de-constructor

Definition at line 14 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00014                                     {
00015 }
```

Here is the caller graph for this function:



6.5.3 Member Function Documentation

6.5.3.1 void CARLOW_W::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>

Definition at line 37 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00037                                     {
00038
00039     std::shared_ptr<CARLOW_W> objTO = std::dynamic_pointer_cast<CARLOW_W>(to);
00040     std::shared_ptr<CARLOW_W> objFROM = std::dynamic_pointer_cast<CARLOW_W>(from);
00041     objTO->_shop_address = objFROM->GetShop_address();
```

```

00042     objTO->_shop_name = objFROM->GetShop_name();
00043     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00044     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00045     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00046     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00047     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00048     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00049     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00050     objTO->Set_Version(objFROM->Get_Version());
00051
00052
00053 }

```

Here is the caller graph for this function:



6.5.3.2 `std::shared_ptr< OSTM > CARLOW_W::getBaseCopy (std::shared_ptr< OSTM > object)` [virtual]

getBaseCopy function, make deep copy of the object/pointer and Return a new BANK* type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a BANK* return type

Definition at line 24 of file [CARLOW_W.cpp](#).

References [CARLOW_W\(\)](#).

Referenced by [operator=\(\)](#).

```

00025 {
00026
00027     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00028     std::shared_ptr<WAREHOUSE> obj(new CARLOW_W(objTO, object->Get_Version(),object->Get_Unique_ID(
00029 ))) ;
00029     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00030     return ostm_obj;
00031 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.3 `int CARLOW_W::GetNumber_of_alcatel () [virtual]`

Implements [WAREHOUSE](#).

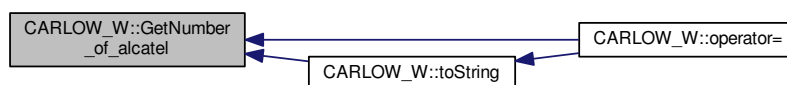
Definition at line 75 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00075 {
00076     return _number_of_alcatel;
00077 }
  
```

Here is the caller graph for this function:



6.5.3.4 `int CARLOW_W::GetNumber_of_huawei () [virtual]`

Implements [WAREHOUSE](#).

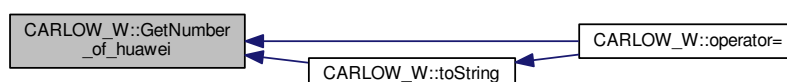
Definition at line 91 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00091 {
00092     return _number_of_huawei;
00093 }
  
```

Here is the caller graph for this function:



6.5.3.5 int CARLOW_W::GetNumber_of_iphones () [virtual]

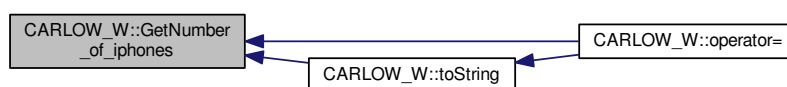
Implements [WAREHOUSE](#).

Definition at line 115 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00115 {  
00116     return _number_of_iphones;  
00117 }
```

Here is the caller graph for this function:



6.5.3.6 int CARLOW_W::GetNumber_of_nokia () [virtual]

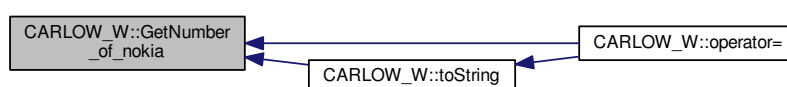
Implements [WAREHOUSE](#).

Definition at line 83 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00083 {  
00084     return _number_of_nokia;  
00085 }
```

Here is the caller graph for this function:



6.5.3.7 int CARLOW_W::GetNumber_of_samsung () [virtual]

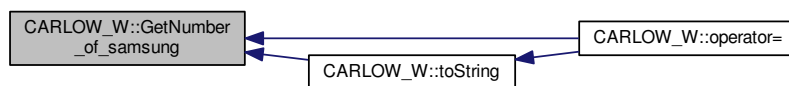
Implements [WAREHOUSE](#).

Definition at line 107 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00107 {  
00108     return _number_of_samsung;  
00109 }
```

Here is the caller graph for this function:



6.5.3.8 int CARLOW_W::GetNumber_of_sony () [virtual]

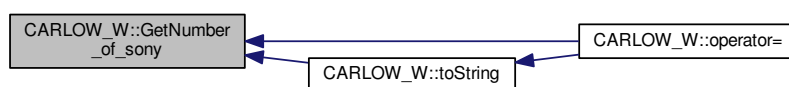
Implements [WAREHOUSE](#).

Definition at line 99 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00099 {  
00100     return _number_of_sony;  
00101 }
```

Here is the caller graph for this function:



6.5.3.9 std::string CARLOW_W::GetShop_address () [virtual]

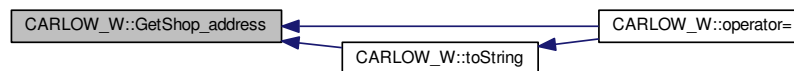
Implements [WAREHOUSE](#).

Definition at line 131 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00131                                     {  
00132     return _shop_address;  
00133 }
```

Here is the caller graph for this function:



6.5.3.10 std::string CARLOW_W::GetShop_name () [virtual]

Implements [WAREHOUSE](#).

Definition at line 123 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00123                                     {  
00124     return _shop_name;  
00125 }
```

Here is the caller graph for this function:



6.5.3.11 CARLOW_W CARLOW_W::operator= (const CARLOW_W & orig) [inline]

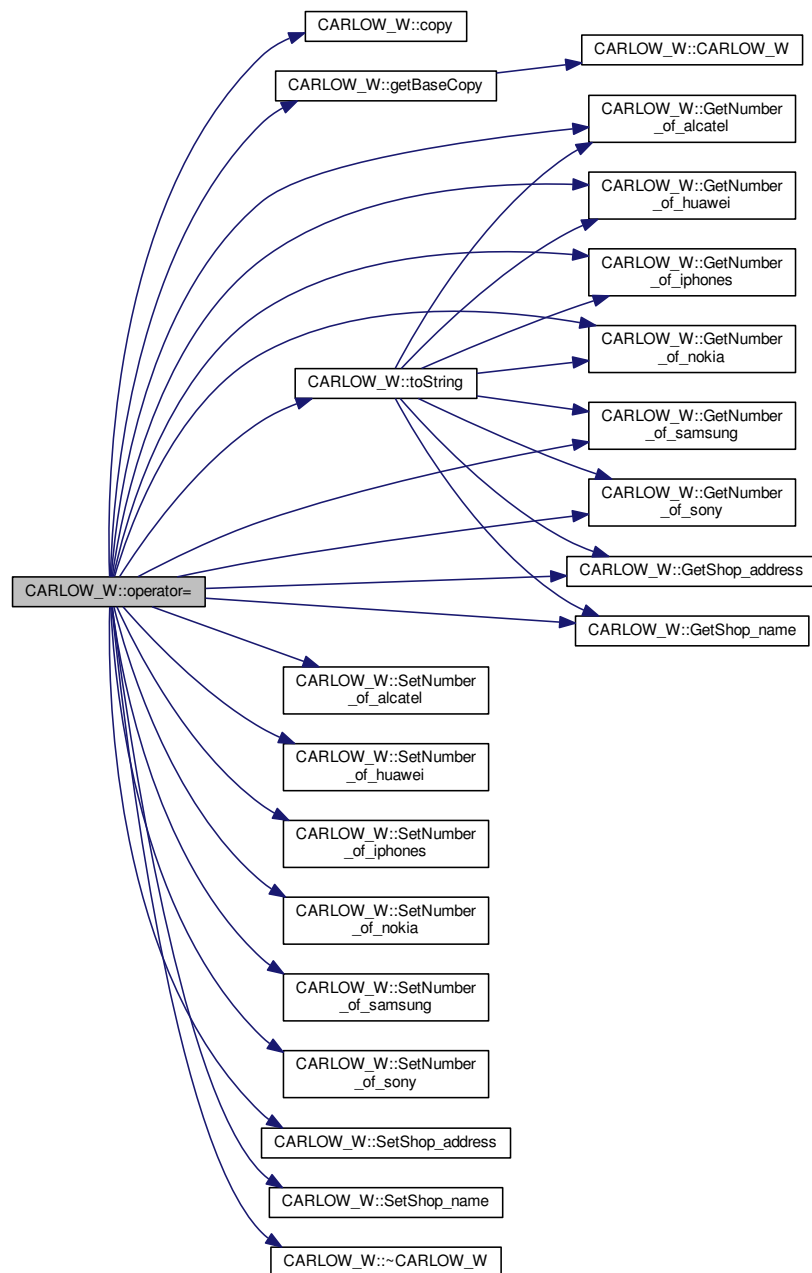
Operator

Definition at line 75 of file CARLOW_W.h.

References [copy\(\)](#), [getBaseCopy\(\)](#), [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), [GetShop_name\(\)](#), [SetNumber_of_alcatel\(\)](#), [SetNumber_of_huawei\(\)](#), [SetNumber_of_iphones\(\)](#), [SetNumber_of_nokia\(\)](#), [SetNumber_of_samsung\(\)](#), [SetNumber_of_sony\(\)](#), [SetShop_address\(\)](#), [SetShop_name\(\)](#), [toString\(\)](#), and [~CARLOW_W\(\)](#).

```
00075 {};
```

Here is the call graph for this function:



6.5.3.12 void CARLOW_W::SetNumber_of_alcatel (int *_number_of_alcatel*) [virtual]

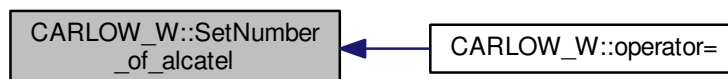
Implements [WAREHOUSE](#).

Definition at line 71 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00071                                     {  
00072     this->_number_of_alcatel = _number_of_alcatel;  
00073 }
```

Here is the caller graph for this function:



6.5.3.13 void CARLOW_W::SetNumber_of_huawei (int *_number_of_huawei*) [virtual]

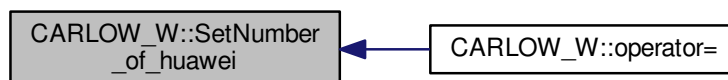
Implements [WAREHOUSE](#).

Definition at line 87 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00087                                     {  
00088     this->_number_of_huawei = _number_of_huawei;  
00089 }
```

Here is the caller graph for this function:



6.5.3.14 void CARLOW_W::SetNumber_of_iphones (int *_number_of_iphones*) [virtual]

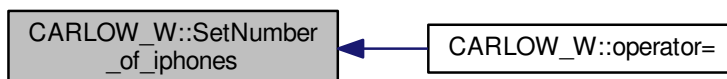
Implements [WAREHOUSE](#).

Definition at line 111 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00111                                     {  
00112     this->_number_of_iphones = _number_of_iphones;  
00113 }
```

Here is the caller graph for this function:



6.5.3.15 void CARLOW_W::SetNumber_of_nokia (int *_number_of_nokia*) [virtual]

Implements [WAREHOUSE](#).

Definition at line 79 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00079                                     {  
00080     this->_number_of_nokia = _number_of_nokia;  
00081 }
```

Here is the caller graph for this function:



6.5.3.16 void CARLOW_W::SetNumber_of_samsung (int *_number_of_samsung*) [virtual]

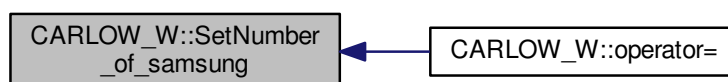
Implements [WAREHOUSE](#).

Definition at line 103 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00103                                     {  
00104     this->_number_of_samsung = _number_of_samsung;  
00105 }
```

Here is the caller graph for this function:



6.5.3.17 void CARLOW_W::SetNumber_of_sony (int *_number_of_sony*) [virtual]

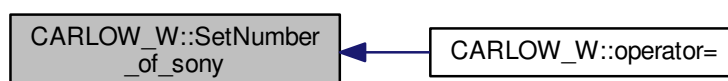
Implements [WAREHOUSE](#).

Definition at line 95 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00095                                     {  
00096     this->_number_of_sony = _number_of_sony;  
00097 }
```

Here is the caller graph for this function:



6.5.3.18 void CARLOW_W::SetShop_address (std::string_shop_address) [virtual]

Implements [WAREHOUSE](#).

Definition at line 127 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00127                                     {  
00128     this->_shop_address = _shop_address;  
00129 }
```

Here is the caller graph for this function:



6.5.3.19 void CARLOW_W::SetShop_name (std::string_shop_name) [virtual]

Implements [WAREHOUSE](#).

Definition at line 119 of file [CARLOW_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00119                                     {  
00120     this->_shop_name = _shop_name;  
00121 }
```

Here is the caller graph for this function:



6.5.3.20 void CARLOW_W::toString () [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

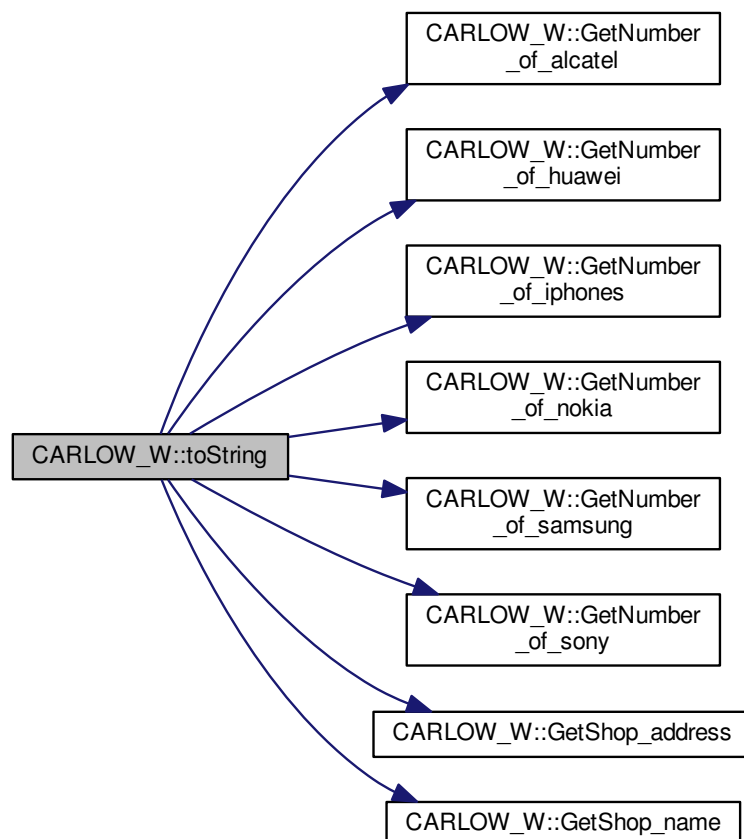
Definition at line 64 of file [CARLOW_W.cpp](#).

References [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), and [GetShop_name\(\)](#).

Referenced by [operator=\(\)](#).

```
00065 {
00066     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
    \nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
    GetShop_address() << "\nNo. Iphones : " << this->
    GetNumber_of_iphones() << "\nNo. Samsung : " << this->
    GetNumber_of_samsung() << "\nNo. Sony : " << this->
    GetNumber_of_sony() << "\nNo. Huawei : " << this->
    GetNumber_of_huawei() << "\nNo. Nokia : " << this->
    GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
    GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00067 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



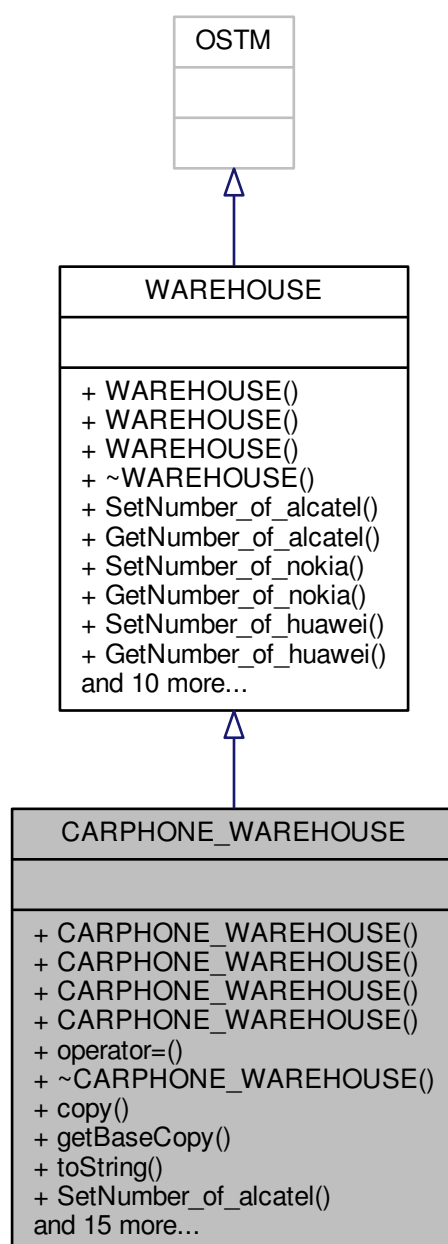
The documentation for this class was generated from the following files:

- [CARLOW_W.h](#)
- [CARLOW_W.cpp](#)

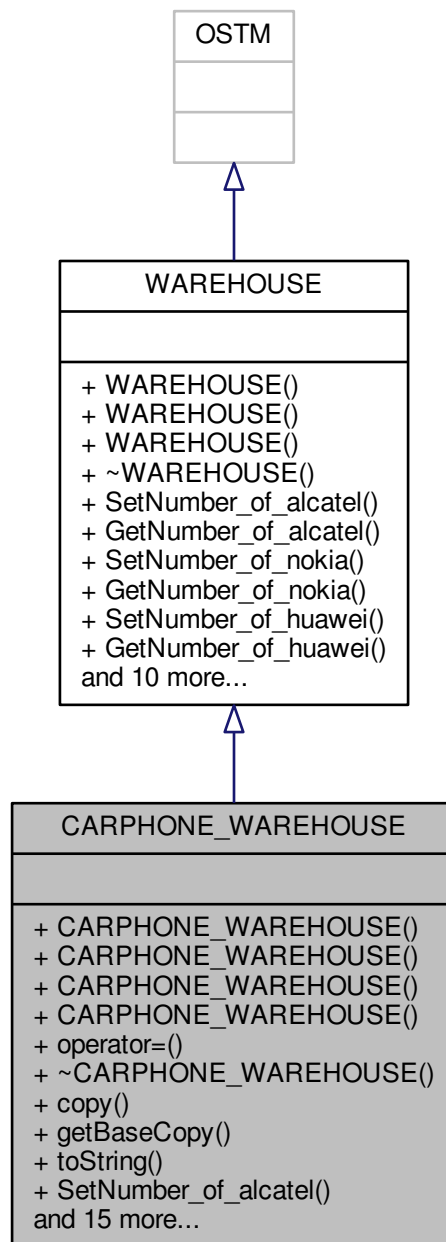
6.6 CARPHONE_WAREHOUSE Class Reference

```
#include <CARPHONE_WAREHOUSE.h>
```

Inheritance diagram for CARPHONE_WAREHOUSE:



Collaboration diagram for CARPHONE_WAREHOUSE:



Public Member Functions

- [CARPHONE_WAREHOUSE](#) ()
- [CARPHONE_WAREHOUSE](#) (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel)
- [CARPHONE_WAREHOUSE](#) (std::shared_ptr< [WAREHOUSE](#) > obj, int _version, int _unique_id)
- [CARPHONE_WAREHOUSE](#) (const [CARPHONE_WAREHOUSE](#) &orig)

- [CARPHONE_WAREHOUSE operator=](#) (const [CARPHONE_WAREHOUSE](#) &orig)
- virtual [~CARPHONE_WAREHOUSE](#) ()
- virtual void [copy](#) (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)
copy function, make deep copy of the object/pointer
- virtual std::shared_ptr< OSTM > [getBaseCopy](#) (std::shared_ptr< OSTM > object)
getBaseCopy function, make deep copy of the object/pointer and Return a new BANK type object*
- virtual void [toString](#) ()
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void [SetNumber_of_alcatel](#) (int _number_of_alcatel)
- virtual int [GetNumber_of_alcatel](#) ()
- virtual void [SetNumber_of_nokia](#) (int _number_of_nokia)
- virtual int [GetNumber_of_nokia](#) ()
- virtual void [SetNumber_of_huawei](#) (int _number_of_huawei)
- virtual int [GetNumber_of_huawei](#) ()
- virtual void [SetNumber_of_sony](#) (int _number_of_sony)
- virtual int [GetNumber_of_sony](#) ()
- virtual void [SetNumber_of_samsung](#) (int _number_of_samsung)
- virtual int [GetNumber_of_samsung](#) ()
- virtual void [SetNumber_of_iphones](#) (int _number_of_iphones)
- virtual int [GetNumber_of_iphones](#) ()
- virtual void [SetShop_name](#) (std::string _shop_name)
- virtual std::string [GetShop_name](#) ()
- virtual void [SetShop_address](#) (std::string _shop_address)
- virtual std::string [GetShop_address](#) ()

6.6.1 Detailed Description

Inherit from [WAREHOUSE](#)

Definition at line 19 of file [CARPHONE_WAREHOUSE.h](#).

6.6.2 Constructor & Destructor Documentation

6.6.2.1 [CARPHONE_WAREHOUSE::CARPHONE_WAREHOUSE \(\)](#) [inline]

Constructor

Definition at line 24 of file [CARPHONE_WAREHOUSE.h](#).

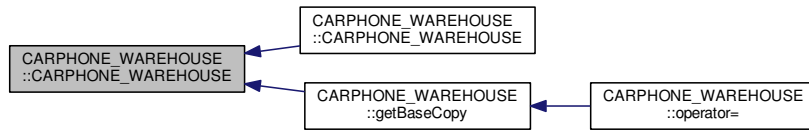
Referenced by [CARPHONE_WAREHOUSE\(\)](#), and [getBaseCopy\(\)](#).

```

00024             : WAREHOUSE () {
00025
00026             this->_shop_address = "DUBLIN XII";
00027             this->_shop_name = "DISTRIBUTION CENTER";
00028             this->_number_of_iphones = 10000;
00029             this->_number_of_samsung = 10000;
00030             this->_number_of_sony = 10000;
00031             this->_number_of_huawei = 10000;
00032             this->_number_of_nokia = 10000;
00033             this->_number_of_alcatel = 10000;
00034         };

```

Here is the caller graph for this function:



6.6.2.2 CARPHONE_WAREHOUSE::CARPHONE_WAREHOUSE (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel) [inline]

Custom constructor

Definition at line 38 of file [CARPHONE_WAREHOUSE.h](#).

```

00038                                     : WAREHOUSE() {
00039     /*
00040     * copy over values
00041     */
00042     this->_shop_address = address;
00043     this->_shop_name = shop_name;
00044     this->_number_of_iphones = iphone;
00045     this->_number_of_samsung = samsung;
00046     this->_number_of_sony = sony;
00047     this->_number_of_huawei = huawei;
00048     this->_number_of_nokia = nokia;
00049     this->_number_of_alcatel = alcatel;
00050
00051 };
  
```

6.6.2.3 CARPHONE_WAREHOUSE::CARPHONE_WAREHOUSE (std::shared_ptr< WAREHOUSE > obj, int _version, int _unique_id) [inline]

Custom constructor, used by the library for deep copying

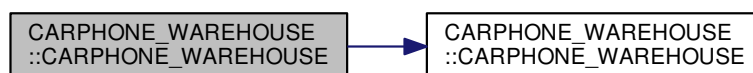
Definition at line 55 of file [CARPHONE_WAREHOUSE.h](#).

References [CARPHONE_WAREHOUSE\(\)](#).

```

00055                                     :
00055     WAREHOUSE(_version, _unique_id) {
00056     /*
00057     * copy over values
00058     */
00059     this->_shop_address = obj->GetShop_address();
00060     this->_shop_name = obj->GetShop_name();
00061     this->_number_of_iphones = obj->GetNumber_of_iphones();
00062     this->_number_of_samsung = obj->GetNumber_of_samsung();
00063     this->_number_of_sony = obj->GetNumber_of_sony();
00064     this->_number_of_huawei = obj->GetNumber_of_huawei();
00065     this->_number_of_nokia = obj->GetNumber_of_nokia();
00066     this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067 }
  
```

Here is the call graph for this function:



6.6.2.4 CARPHONE_WAREHOUSE::CARPHONE_WAREHOUSE (const CARPHONE_WAREHOUSE & orig)

Copy constructor

Definition at line 11 of file [CARPHONE_WAREHOUSE.cpp](#).

```
00011                                     {
00012 }
```

6.6.2.5 CARPHONE_WAREHOUSE::~~CARPHONE_WAREHOUSE () [virtual]

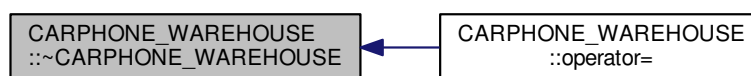
de-constructor

Definition at line 14 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

```
00014                                     {
00015 }
```

Here is the caller graph for this function:



6.6.3 Member Function Documentation

6.6.3.1 void CARPHONE_WAREHOUSE::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>

Definition at line 34 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

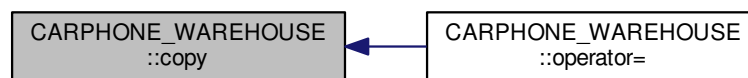
```
00034                                     {
00035
00036     std::shared_ptr<CARPHONE_WAREHOUSE> objTO = std::dynamic_pointer_cast<
```

```

CARPHONE_WAREHOUSE>(to);
00037     std::shared_ptr<CARPHONE_WAREHOUSE> objFROM = std::dynamic_pointer_cast<
CARPHONE_WAREHOUSE>(from);
00038     objTO->_shop_address = objFROM->GetShop_address();
00039     objTO->_shop_name = objFROM->GetShop_name();
00040     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00041     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00042     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00043     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00044     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00045     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00046     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00047     objTO->Set_Version(objFROM->Get_Version());
00048
00049 }

```

Here is the caller graph for this function:



6.6.3.2 `std::shared_ptr< OSTM > CARPHONE_WAREHOUSE::getBaseCopy (std::shared_ptr< OSTM > object)`
[virtual]

getBaseCopy function, make deep copy of the object/pointer and Return a new BANK* type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a BANK* return type

Definition at line 21 of file [CARPHONE_WAREHOUSE.cpp](#).

References [CARPHONE_WAREHOUSE\(\)](#).

Referenced by [operator=\(\)](#).

```

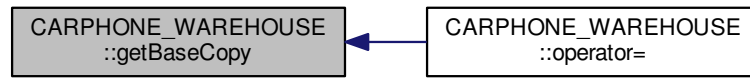
00022 {
00023
00024     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00025     std::shared_ptr<WAREHOUSE> obj(new CARPHONE_WAREHOUSE(objTO, object->Get_Version(),
object->Get_Unique_ID()));
00026     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00027     return ostm_obj;
00028 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.3.3 `int CARPHONE_WAREHOUSE::GetNumber_of_alcatel() [virtual]`

Implements [WAREHOUSE](#).

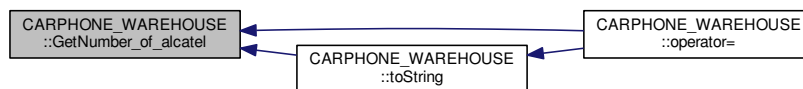
Definition at line 71 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00071                                     {
00072     return _number_of_alcatel;
00073 }
```

Here is the caller graph for this function:



6.6.3.4 `int CARPHONE_WAREHOUSE::GetNumber_of_huawei() [virtual]`

Implements [WAREHOUSE](#).

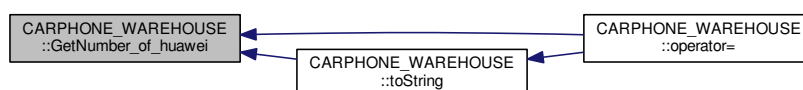
Definition at line 87 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00087                                     {
00088     return _number_of_huawei;
00089 }
```

Here is the caller graph for this function:



6.6.3.5 int CARPHONE_WAREHOUSE::GetNumber_of_iphones () [virtual]

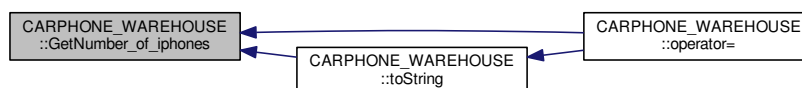
Implements [WAREHOUSE](#).

Definition at line 111 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00111                                     {  
00112     return _number_of_iphones;  
00113 }
```

Here is the caller graph for this function:



6.6.3.6 int CARPHONE_WAREHOUSE::GetNumber_of_nokia () [virtual]

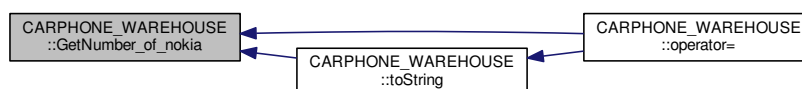
Implements [WAREHOUSE](#).

Definition at line 79 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00079                                     {  
00080     return _number_of_nokia;  
00081 }
```

Here is the caller graph for this function:



6.6.3.7 int CARPHONE_WAREHOUSE::GetNumber_of_samsung () [virtual]

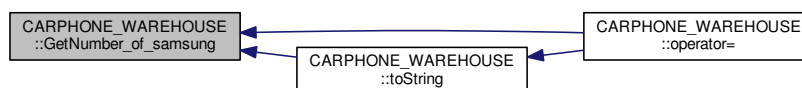
Implements [WAREHOUSE](#).

Definition at line 103 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00103                                     {  
00104     return _number_of_samsung;  
00105 }
```

Here is the caller graph for this function:



6.6.3.8 int CARPHONE_WAREHOUSE::GetNumber_of_sony () [virtual]

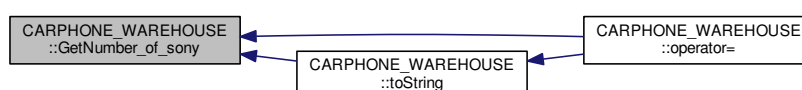
Implements [WAREHOUSE](#).

Definition at line 95 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00095                                     {  
00096     return _number_of_sony;  
00097 }
```

Here is the caller graph for this function:



6.6.3.9 `std::string CARPHONE_WAREHOUSE::GetShop_address () [virtual]`

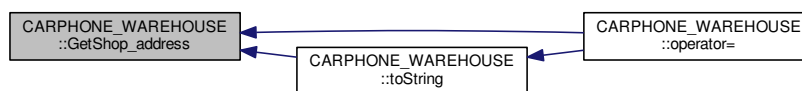
Implements [WAREHOUSE](#).

Definition at line 127 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00127                                     {  
00128     return _shop_address;  
00129 }
```

Here is the caller graph for this function:



6.6.3.10 `std::string CARPHONE_WAREHOUSE::GetShop_name () [virtual]`

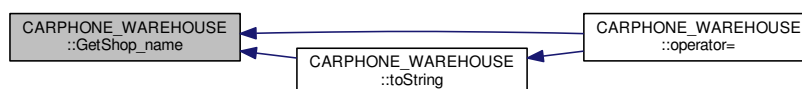
Implements [WAREHOUSE](#).

Definition at line 119 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00119                                     {  
00120     return _shop_name;  
00121 }
```

Here is the caller graph for this function:



6.6.3.11 **CARPHONE_WAREHOUSE** **CARPHONE_WAREHOUSE::operator=** (const **CARPHONE_WAREHOUSE** & *orig*) [inline]

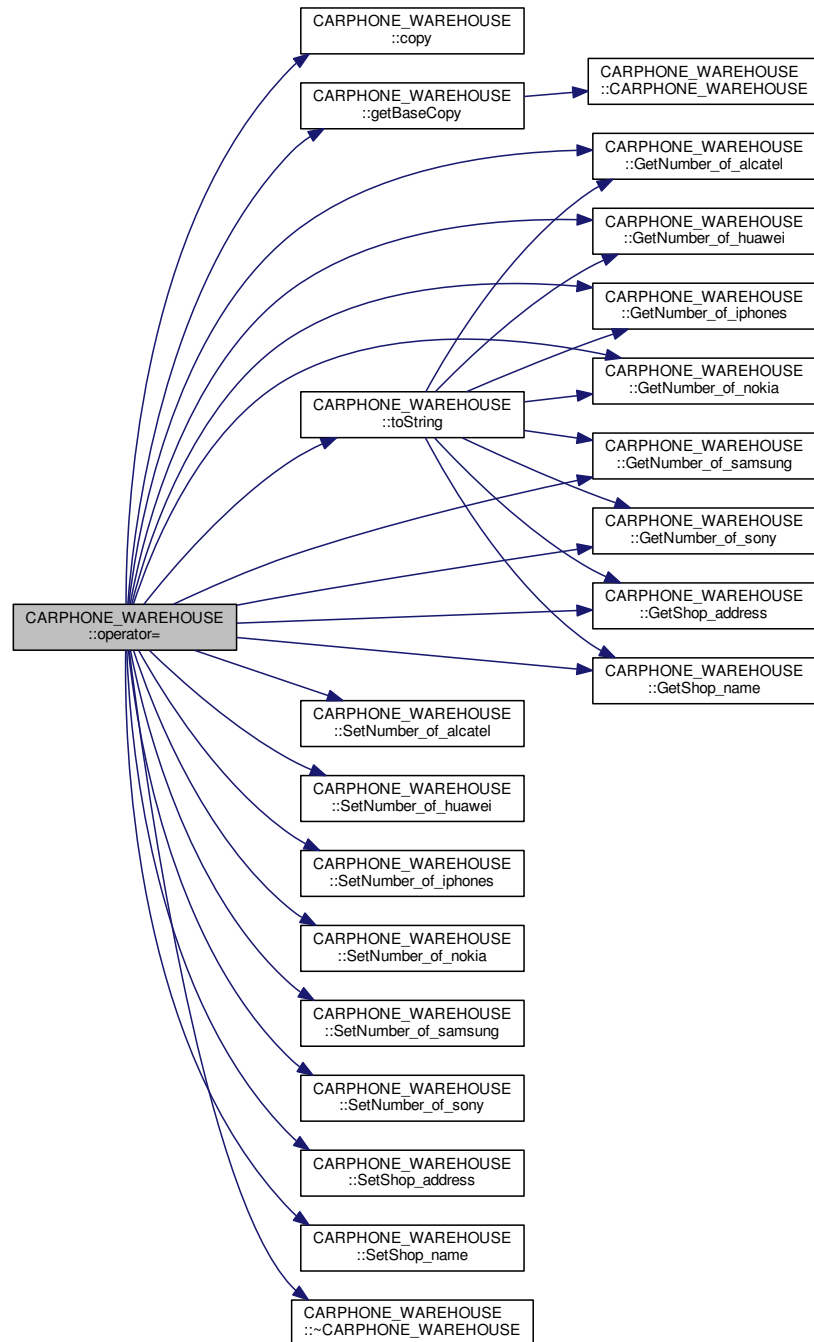
Operator

Definition at line 75 of file [CARPHONE_WAREHOUSE.h](#).

References [copy\(\)](#), [getBaseCopy\(\)](#), [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), [GetShop_name\(\)](#), [SetNumber_of_alcatel\(\)](#), [SetNumber_of_huawei\(\)](#), [SetNumber_of_iphones\(\)](#), [SetNumber_of_nokia\(\)](#), [SetNumber_of_samsung\(\)](#), [SetNumber_of_sony\(\)](#), [SetShop_address\(\)](#), [SetShop_name\(\)](#), [toString\(\)](#), and [~CARPHONE_WAREHOUSE\(\)](#).

00075 {};

Here is the call graph for this function:



6.6.3.12 `void CARPHONE_WAREHOUSE::SetNumber_of_alcatel (int number_of_alcatel)` [virtual]

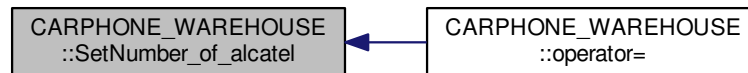
Implements [WAREHOUSE](#).

Definition at line 67 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

```
00067                                     {  
00068     this->_number_of_alcatel = _number_of_alcatel;  
00069 }
```

Here is the caller graph for this function:



6.6.3.13 void CARPHONE_WAREHOUSE::SetNumber_of_huawei (int _number_of_huawei) [virtual]

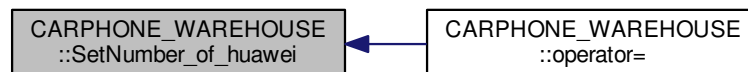
Implements [WAREHOUSE](#).

Definition at line 83 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

```
00083                                     {  
00084     this->_number_of_huawei = _number_of_huawei;  
00085 }
```

Here is the caller graph for this function:



6.6.3.14 void CARPHONE_WAREHOUSE::SetNumber_of_iphones (int _number_of_iphones) [virtual]

Implements [WAREHOUSE](#).

Definition at line 107 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

```
00107                                     {  
00108     this->_number_of_iphones = _number_of_iphones;  
00109 }
```

Here is the caller graph for this function:



6.6.3.15 void CARPHONE_WAREHOUSE::SetNumber_of_nokia (int _number_of_nokia) [virtual]

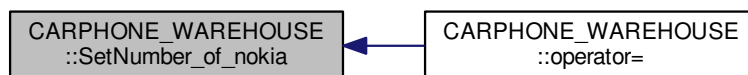
Implements [WAREHOUSE](#).

Definition at line 75 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

```
00075                                     {  
00076     this->_number_of_nokia = _number_of_nokia;  
00077 }
```

Here is the caller graph for this function:



6.6.3.16 void CARPHONE_WAREHOUSE::SetNumber_of_samsung (int _number_of_samsung) [virtual]

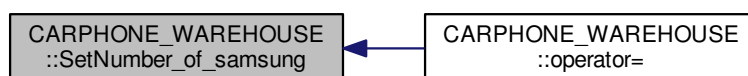
Implements [WAREHOUSE](#).

Definition at line 99 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

```
00099                                     {  
00100     this->_number_of_samsung = _number_of_samsung;  
00101 }
```

Here is the caller graph for this function:



6.6.3.17 void CARPHONE_WAREHOUSE::SetNumber_of_sony (int *_number_of_sony*) [virtual]

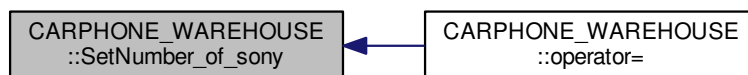
Implements [WAREHOUSE](#).

Definition at line 91 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

```
00091                                     {  
00092     this->_number_of_sony = _number_of_sony;  
00093 }
```

Here is the caller graph for this function:



6.6.3.18 void CARPHONE_WAREHOUSE::SetShop_address (std::string *_shop_address*) [virtual]

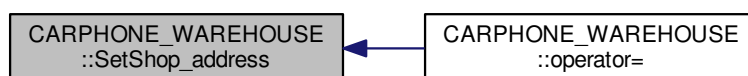
Implements [WAREHOUSE](#).

Definition at line 123 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

```
00123                                     {  
00124     this->_shop_address = _shop_address;  
00125 }
```

Here is the caller graph for this function:



6.6.3.19 void CARPHONE_WAREHOUSE::SetShop_name (std::string _shop_name) [virtual]

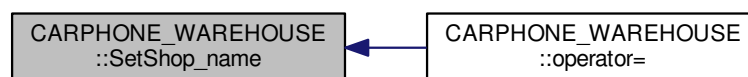
Implements [WAREHOUSE](#).

Definition at line 115 of file [CARPHONE_WAREHOUSE.cpp](#).

Referenced by [operator=\(\)](#).

```
00115                                     {
00116     this->_shop_name = _shop_name;
00117 }
```

Here is the caller graph for this function:



6.6.3.20 void CARPHONE_WAREHOUSE::toString () [virtual]

`_cast`, is use to cast bak the `std::shared_ptr<OSTM>` to the required type

`toString` function, displays the object values in formatted way

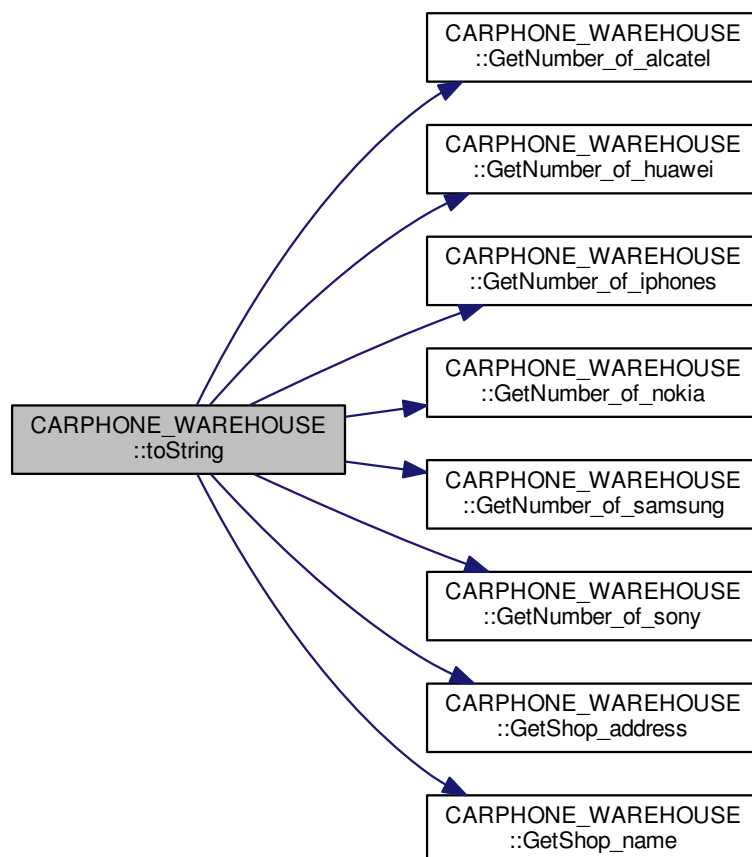
Definition at line 60 of file [CARPHONE_WAREHOUSE.cpp](#).

References [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), and [GetShop_name\(\)](#).

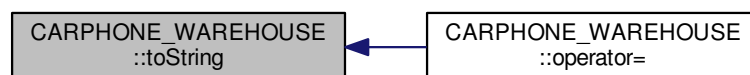
Referenced by [operator=\(\)](#).

```
00061 {
00062     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
    \nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
    GetShop_address() << "\nNo. Iphones : " << this->
    GetNumber_of_iphones() << "\nNo. Samsung : " << this->
    GetNumber_of_samsung() << "\nNo. Sony : " << this->
    GetNumber_of_sony() << "\nNo. Huawei : " << this->
    GetNumber_of_huawei() << "\nNo. Nokia : " << this->
    GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
    GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00063 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



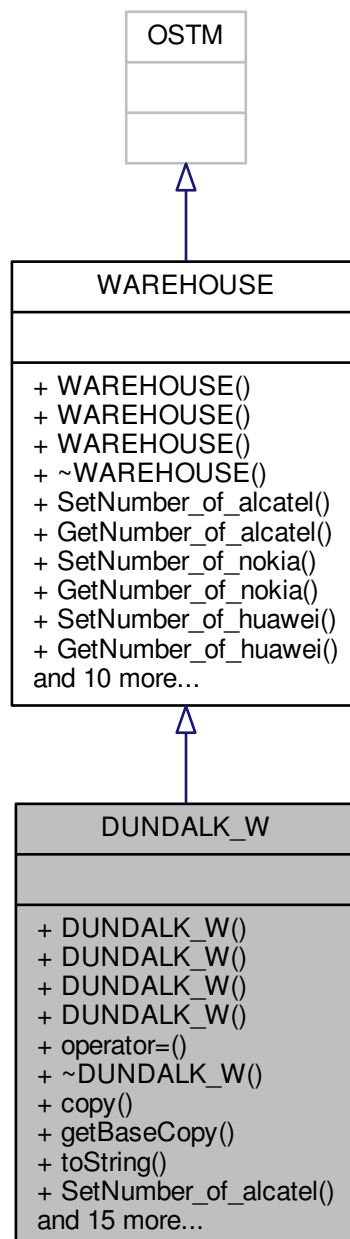
The documentation for this class was generated from the following files:

- [CARPHONE_WAREHOUSE.h](#)
- [CARPHONE_WAREHOUSE.cpp](#)

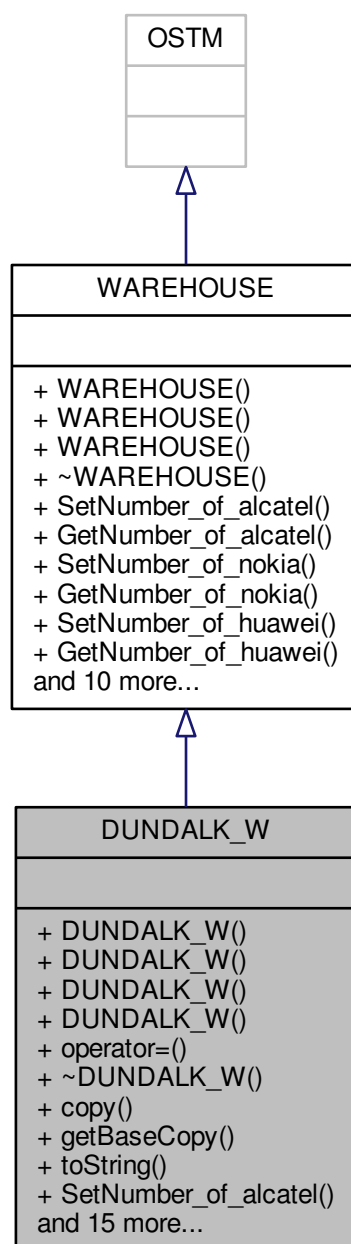
6.7 DUNDALK_W Class Reference

```
#include <DUNDALK_W.h>
```

Inheritance diagram for DUNDALK_W:



Collaboration diagram for DUNDALK_W:



Public Member Functions

- [DUNDALK_W](#) ()
- [DUNDALK_W](#) (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel)
- [DUNDALK_W](#) (std::shared_ptr< [WAREHOUSE](#) > obj, int _version, int _unique_id)
- [DUNDALK_W](#) (const [DUNDALK_W](#) &orig)

- [DUNDALK_W operator=](#) (const [DUNDALK_W](#) &orig)
- virtual [~DUNDALK_W](#) ()
- virtual void [copy](#) (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)
copy function, make deep copy of the object/pointer
- virtual std::shared_ptr< OSTM > [getBaseCopy](#) (std::shared_ptr< OSTM > object)
getBaseCopy function, make deep copy of the object/pointer and Return a new BANK type object*
- virtual void [toString](#) ()
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void [SetNumber_of_alcatel](#) (int _number_of_alcatel)
- virtual int [GetNumber_of_alcatel](#) ()
- virtual void [SetNumber_of_nokia](#) (int _number_of_nokia)
- virtual int [GetNumber_of_nokia](#) ()
- virtual void [SetNumber_of_huawei](#) (int _number_of_huawei)
- virtual int [GetNumber_of_huawei](#) ()
- virtual void [SetNumber_of_sony](#) (int _number_of_sony)
- virtual int [GetNumber_of_sony](#) ()
- virtual void [SetNumber_of_samsung](#) (int _number_of_samsung)
- virtual int [GetNumber_of_samsung](#) ()
- virtual void [SetNumber_of_iphones](#) (int _number_of_iphones)
- virtual int [GetNumber_of_iphones](#) ()
- virtual void [SetShop_name](#) (std::string _shop_name)
- virtual std::string [GetShop_name](#) ()
- virtual void [SetShop_address](#) (std::string _shop_address)
- virtual std::string [GetShop_address](#) ()

6.7.1 Detailed Description

Inherit from [WAREHOUSE](#)

Definition at line 19 of file [DUNDALK_W.h](#).

6.7.2 Constructor & Destructor Documentation

6.7.2.1 [DUNDALK_W::DUNDALK_W \(\)](#) [inline]

Constructor

Definition at line 24 of file [DUNDALK_W.h](#).

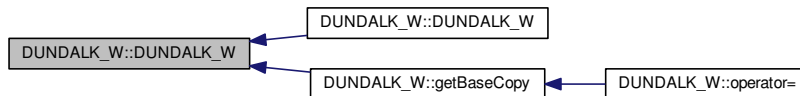
Referenced by [DUNDALK_W\(\)](#), and [getBaseCopy\(\)](#).

```

00024             : WAREHOUSE() {
00025
00026             this->_shop_address = "Dundalk Busy Street";
00027             this->_shop_name = "DUNDALK D_WAREHOUSE";
00028             this->_number_of_iphones = 200;
00029             this->_number_of_samsung = 200;
00030             this->_number_of_sony = 200;
00031             this->_number_of_huawei = 200;
00032             this->_number_of_nokia = 200;
00033             this->_number_of_alcatel = 200;
00034         };

```

Here is the caller graph for this function:



6.7.2.2 DUNDALK_W::DUNDALK_W (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel) [inline]

Custom constructor

Definition at line 38 of file [DUNDALK_W.h](#).

```

00038
                                : WAREHOUSE() {
00039     /*
00040      * copy over values
00041      */
00042     this->_shop_address = address;
00043     this->_shop_name = shop_name;
00044     this->_number_of_iphones = iphone;
00045     this->_number_of_samsung = samsung;
00046     this->_number_of_sony = sony;
00047     this->_number_of_huawei = huawei;
00048     this->_number_of_nokia = nokia;
00049     this->_number_of_alcatel = alcatel;
00050
00051     };
  
```

6.7.2.3 DUNDALK_W::DUNDALK_W (std::shared_ptr< WAREHOUSE > obj, int_version, int_unique_id) [inline]

Custom constructor, used by the library for deep copying

Definition at line 55 of file [DUNDALK_W.h](#).

References [DUNDALK_W\(\)](#).

```

00055     WAREHOUSE(_version, _unique_id) {
00056     /*
00057      * copy over values
00058      */
00059     this->_shop_address = obj->GetShop_address();
00060     this->_shop_name = obj->GetShop_name();
00061     this->_number_of_iphones = obj->GetNumber_of_iphones();
00062     this->_number_of_samsung = obj->GetNumber_of_samsung();
00063     this->_number_of_sony = obj->GetNumber_of_sony();
00064     this->_number_of_huawei = obj->GetNumber_of_huawei();
00065     this->_number_of_nokia = obj->GetNumber_of_nokia();
00066     this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067     }
  
```

Here is the call graph for this function:



6.7.2.4 DUNDALK_W::DUNDALK_W (const DUNDALK_W & orig)

Copy constructor

Definition at line 15 of file [DUNDALK_W.cpp](#).

```
00015                                     {
00016 }
```

6.7.2.5 DUNDALK_W::~~DUNDALK_W () [virtual]

de-constructor

Definition at line 12 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00012                                     {
00013 }
```

Here is the caller graph for this function:



6.7.3 Member Function Documentation

6.7.3.1 void DUNDALK_W::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>

Definition at line 35 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00035                                     {
00036
00037     std::shared_ptr<DUNDALK_W> objTO = std::dynamic_pointer_cast<DUNDALK_W>(to);
00038     std::shared_ptr<DUNDALK_W> objFROM = std::dynamic_pointer_cast<DUNDALK_W>(from);
00039     objTO->_shop_address = objFROM->GetShop_address();
00040 }
```

```

00040     objTO->_shop_name = objFROM->GetShop_name();
00041     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00042     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00043     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00044     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00045     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00046     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00047     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00048     objTO->Set_Version(objFROM->Get_Version());
00049
00050
00051 }

```

Here is the caller graph for this function:



6.7.3.2 `std::shared_ptr< OSTM > DUNDALK_W::getBaseCopy (std::shared_ptr< OSTM > object)` [virtual]

getBaseCopy function, make deep copy of the object/pointer and Return a new BANK* type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a BANK* return type

Definition at line 22 of file [DUNDALK_W.cpp](#).

References [DUNDALK_W\(\)](#).

Referenced by [operator=\(\)](#).

```

00023 {
00024
00025     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00026     std::shared_ptr<WAREHOUSE> obj(new DUNDALK_W(objTO, object->Get_Version(), object->
    Get_Unique_ID()));
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00028     return ostm_obj;
00029 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.7.3.3 `int DUNDALK_W::GetNumber_of_alcatel () [virtual]`

Implements [WAREHOUSE](#).

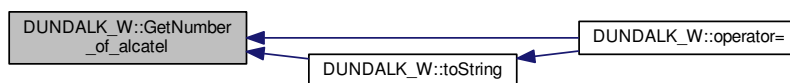
Definition at line 73 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00073 {
00074     return _number_of_alcatel;
00075 }
```

Here is the caller graph for this function:



6.7.3.4 `int DUNDALK_W::GetNumber_of_huawei () [virtual]`

Implements [WAREHOUSE](#).

Definition at line 89 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00089 {
00090     return _number_of_huawei;
00091 }
```

Here is the caller graph for this function:



6.7.3.5 int DUNDALK_W::GetNumber_of_iphones () [virtual]

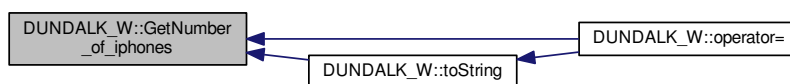
Implements [WAREHOUSE](#).

Definition at line 113 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00113 {  
00114     return _number_of_iphones;  
00115 }
```

Here is the caller graph for this function:



6.7.3.6 int DUNDALK_W::GetNumber_of_nokia () [virtual]

Implements [WAREHOUSE](#).

Definition at line 81 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00081 {  
00082     return _number_of_nokia;  
00083 }
```

Here is the caller graph for this function:



6.7.3.7 int DUNDALK_W::GetNumber_of_samsung() [virtual]

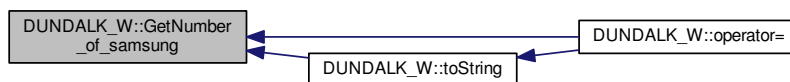
Implements [WAREHOUSE](#).

Definition at line 105 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00105 {  
00106     return _number_of_samsung;  
00107 }
```

Here is the caller graph for this function:



6.7.3.8 int DUNDALK_W::GetNumber_of_sony() [virtual]

Implements [WAREHOUSE](#).

Definition at line 97 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00097 {  
00098     return _number_of_sony;  
00099 }
```

Here is the caller graph for this function:



6.7.3.9 std::string DUNDALK_W::GetShop_address () [virtual]

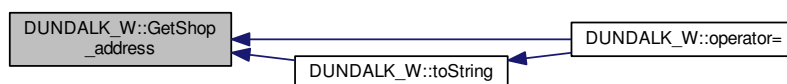
Implements [WAREHOUSE](#).

Definition at line 129 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00129                                     {  
00130     return _shop_address;  
00131 }
```

Here is the caller graph for this function:



6.7.3.10 std::string DUNDALK_W::GetShop_name () [virtual]

Implements [WAREHOUSE](#).

Definition at line 121 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00121                                     {  
00122     return _shop_name;  
00123 }
```

Here is the caller graph for this function:



6.7.3.11 `DUNDALK_W DUNDALK_W::operator= (const DUNDALK_W & orig) [inline]`

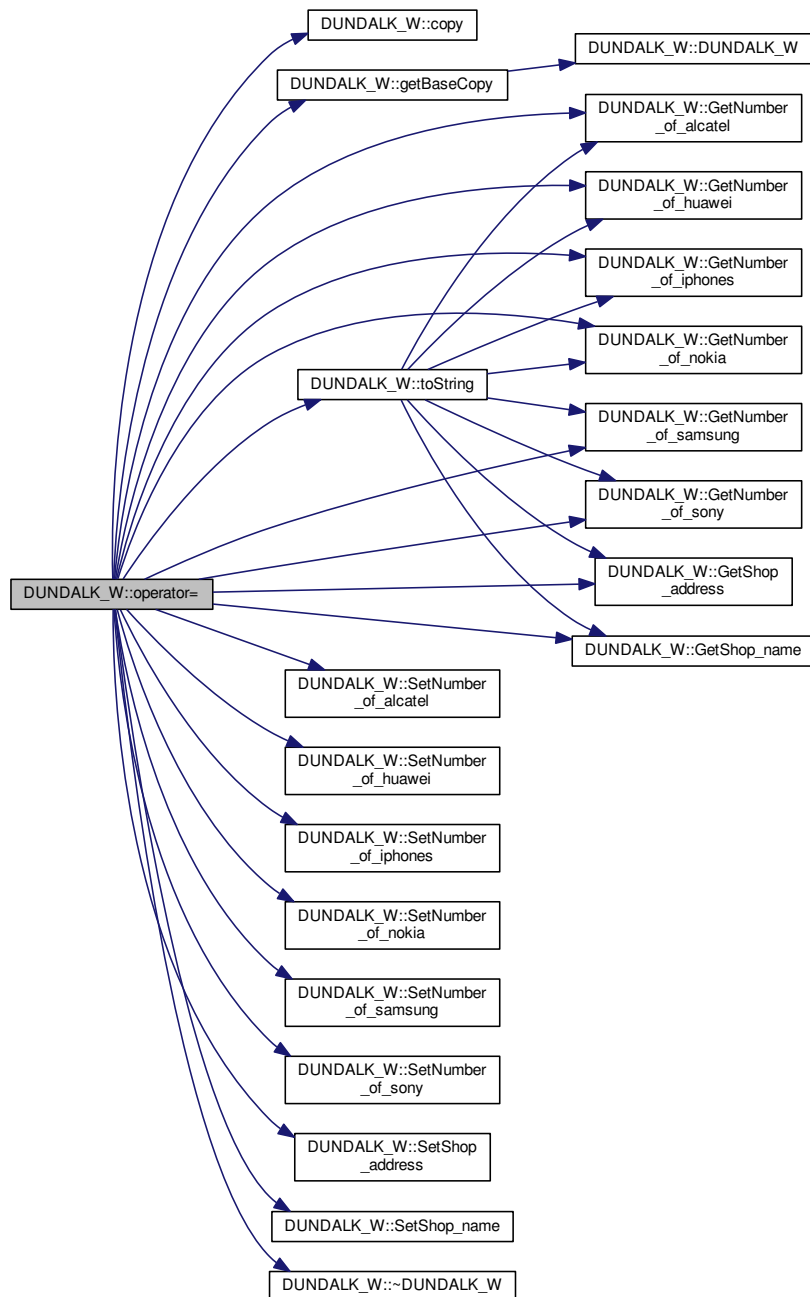
Operator

Definition at line 75 of file [DUNDALK_W.h](#).

References [copy\(\)](#), [getBaseCopy\(\)](#), [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), [GetShop_name\(\)](#), [SetNumber_of_alcatel\(\)](#), [SetNumber_of_huawei\(\)](#), [SetNumber_of_iphones\(\)](#), [SetNumber_of_nokia\(\)](#), [SetNumber_of_samsung\(\)](#), [SetNumber_of_sony\(\)](#), [SetShop_address\(\)](#), [SetShop_name\(\)](#), [toString\(\)](#), and [~DUNDALK_W\(\)](#).

00075 {};

Here is the call graph for this function:



6.7.3.12 `void DUNDALK_W::SetNumber_of_alcatel (int _number_of_alcatel)` [virtual]

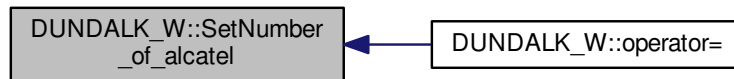
Implements [WAREHOUSE](#).

Definition at line 69 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00069                                     {
00070     this->_number_of_alcatel = _number_of_alcatel;
00071 }
```

Here is the caller graph for this function:



6.7.3.13 void DUNDALK_W::SetNumber_of_huawei(int _number_of_huawei) [virtual]

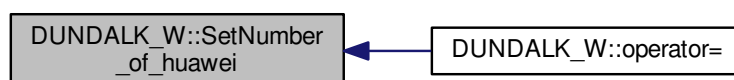
Implements [WAREHOUSE](#).

Definition at line 85 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00085                                     {
00086     this->_number_of_huawei = _number_of_huawei;
00087 }
```

Here is the caller graph for this function:



6.7.3.14 void DUNDALK_W::SetNumber_of_iphones(int _number_of_iphones) [virtual]

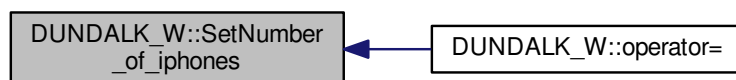
Implements [WAREHOUSE](#).

Definition at line 109 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00109                                     {
00110     this->_number_of_iphones = _number_of_iphones;
00111 }
```

Here is the caller graph for this function:



6.7.3.15 `void DUNDALK_W::SetNumber_of_nokia (int _number_of_nokia) [virtual]`

Implements [WAREHOUSE](#).

Definition at line 77 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00077                                     {  
00078     this->_number_of_nokia = _number_of_nokia;  
00079 }
```

Here is the caller graph for this function:



6.7.3.16 `void DUNDALK_W::SetNumber_of_samsung (int _number_of_samsung) [virtual]`

Implements [WAREHOUSE](#).

Definition at line 101 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00101                                     {  
00102     this->_number_of_samsung = _number_of_samsung;  
00103 }
```

Here is the caller graph for this function:



6.7.3.17 void DUNDALK_W::SetNumber_of_sony (int *_number_of_sony*) [virtual]

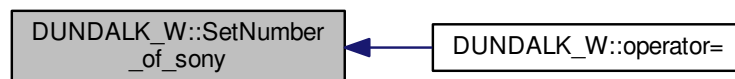
Implements [WAREHOUSE](#).

Definition at line 93 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00093      {  
00094          this->_number_of_sony = _number_of_sony;  
00095      }
```

Here is the caller graph for this function:



6.7.3.18 void DUNDALK_W::SetShop_address (std::string *_shop_address*) [virtual]

Implements [WAREHOUSE](#).

Definition at line 125 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00125      {  
00126          this->_shop_address = _shop_address;  
00127      }
```

Here is the caller graph for this function:



6.7.3.19 void DUNDALK_W::SetShop_name (std::string _shop_name) [virtual]

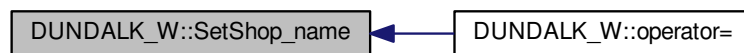
Implements [WAREHOUSE](#).

Definition at line 117 of file [DUNDALK_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00117 {
00118     this->_shop_name = _shop_name;
00119 }
```

Here is the caller graph for this function:



6.7.3.20 void DUNDALK_W::toString () [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

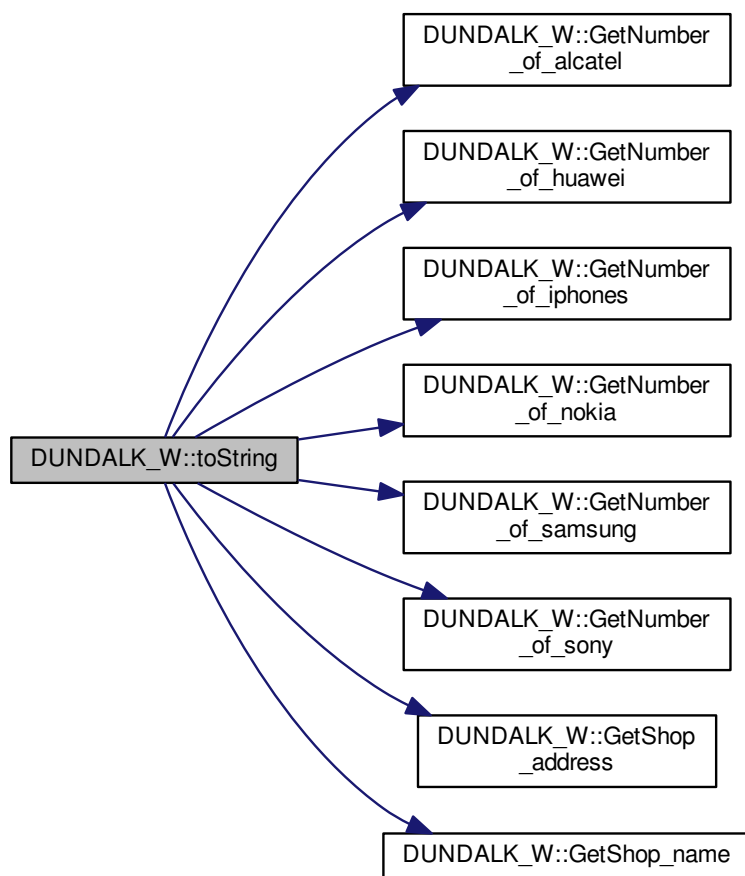
Definition at line 62 of file [DUNDALK_W.cpp](#).

References [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), and [GetShop_name\(\)](#).

Referenced by [operator=\(\)](#).

```
00063 {
00064     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
    \nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
    GetShop_address() << "\nNo. Iphones : " << this->
    GetNumber_of_iphones() << "\nNo. Samsung : " << this->
    GetNumber_of_samsung() << "\nNo. Sony : " << this->
    GetNumber_of_sony() << "\nNo. Huawei : " << this->
    GetNumber_of_huawei() << "\nNo. Nokia : " << this->
    GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
    GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00065 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



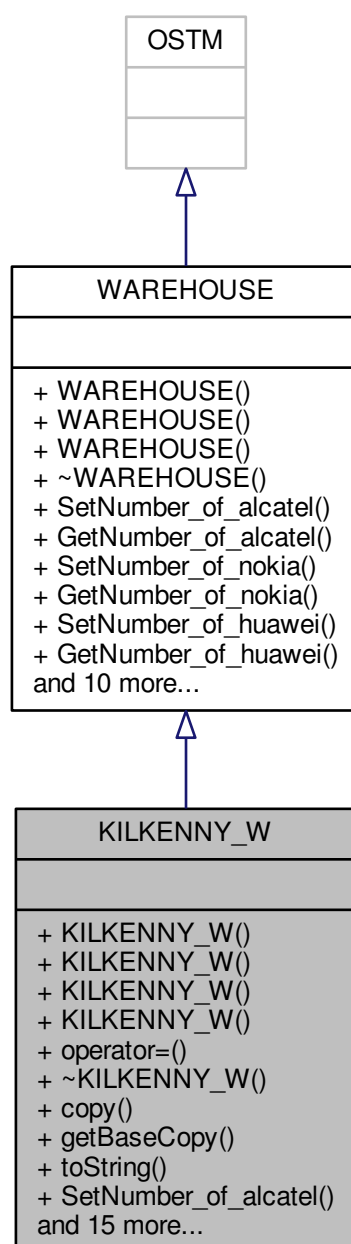
The documentation for this class was generated from the following files:

- [DUNDALK_W.h](#)
- [DUNDALK_W.cpp](#)

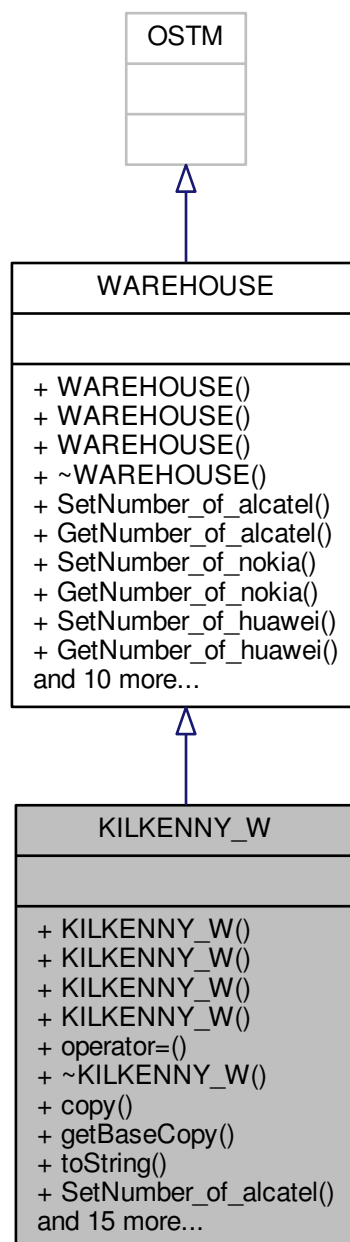
6.8 KILKENNY_W Class Reference

```
#include <KILKENNY_W.h>
```

Inheritance diagram for KILKENNY_W:



Collaboration diagram for KILKENNY_W:



Public Member Functions

- [KILKENNY_W \(\)](#)
- [KILKENNY_W \(std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel\)](#)
- [KILKENNY_W \(std::shared_ptr< \[WAREHOUSE\]\(#\) > obj, int _version, int _unique_id\)](#)
- [KILKENNY_W \(const \[KILKENNY_W\]\(#\) &orig\)](#)

- [KILKENNY_W operator=](#) (const [KILKENNY_W](#) &orig)
- virtual [~KILKENNY_W](#) ()
- virtual void [copy](#) (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)
copy function, make deep copy of the object/pointer
- virtual std::shared_ptr< OSTM > [getBaseCopy](#) (std::shared_ptr< OSTM > object)
getBaseCopy function, make deep copy of the object/pointer and Return a new BANK type object*
- virtual void [toString](#) ()
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void [SetNumber_of_alcatel](#) (int _number_of_alcatel)
- virtual int [GetNumber_of_alcatel](#) ()
- virtual void [SetNumber_of_nokia](#) (int _number_of_nokia)
- virtual int [GetNumber_of_nokia](#) ()
- virtual void [SetNumber_of_huawei](#) (int _number_of_huawei)
- virtual int [GetNumber_of_huawei](#) ()
- virtual void [SetNumber_of_sony](#) (int _number_of_sony)
- virtual int [GetNumber_of_sony](#) ()
- virtual void [SetNumber_of_samsung](#) (int _number_of_samsung)
- virtual int [GetNumber_of_samsung](#) ()
- virtual void [SetNumber_of_iphones](#) (int _number_of_iphones)
- virtual int [GetNumber_of_iphones](#) ()
- virtual void [SetShop_name](#) (std::string _shop_name)
- virtual std::string [GetShop_name](#) ()
- virtual void [SetShop_address](#) (std::string _shop_address)
- virtual std::string [GetShop_address](#) ()

6.8.1 Detailed Description

Inherit from [WAREHOUSE](#)

Definition at line 19 of file [KILKENNY_W.h](#).

6.8.2 Constructor & Destructor Documentation

6.8.2.1 KILKENNY_W::KILKENNY_W() [inline]

Constructor

Definition at line 24 of file [KILKENNY_W.h](#).

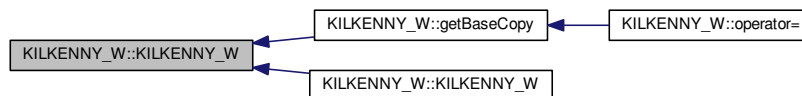
Referenced by [getBaseCopy\(\)](#), and [KILKENNY_W\(\)](#).

```

00024             : WAREHOUSE() {
00025
00026             this->_shop_address = "Kilkenny High Street";
00027             this->_shop_name = "KILKENNY_K_WAREHOUSE";
00028             this->_number_of_iphones = 200;
00029             this->_number_of_samsung = 200;
00030             this->_number_of_sony = 200;
00031             this->_number_of_huawei = 200;
00032             this->_number_of_nokia = 200;
00033             this->_number_of_alcatel = 200;
00034         };

```

Here is the caller graph for this function:



6.8.2.2 KILKENNY_W::KILKENNY_W (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel) [inline]

Custom constructor

Definition at line 38 of file [KILKENNY_W.h](#).

```

00038
00039         : WAREHOUSE() {
00040             /*
00041              * copy over values
00042              */
00043             this->_shop_address = address;
00044             this->_shop_name = shop_name;
00045             this->_number_of_iphones = iphone;
00046             this->_number_of_samsung = samsung;
00047             this->_number_of_sony = sony;
00048             this->_number_of_huawei = huawei;
00049             this->_number_of_nokia = nokia;
00050             this->_number_of_alcatel = alcatel;
00051         };
  
```

6.8.2.3 KILKENNY_W::KILKENNY_W (std::shared_ptr< WAREHOUSE > obj, int _version, int _unique_id) [inline]

Custom constructor, used by the library for deep copying

Definition at line 55 of file [KILKENNY_W.h](#).

References [KILKENNY_W\(\)](#).

```

00055         :
00056         WAREHOUSE(_version, _unique_id) {
00057             /*
00058              * copy over values
00059              */
00060             this->_shop_address = obj->GetShop_address();
00061             this->_shop_name = obj->GetShop_name();
00062             this->_number_of_iphones = obj->GetNumber_of_iphones();
00063             this->_number_of_samsung = obj->GetNumber_of_samsung();
00064             this->_number_of_sony = obj->GetNumber_of_sony();
00065             this->_number_of_huawei = obj->GetNumber_of_huawei();
00066             this->_number_of_nokia = obj->GetNumber_of_nokia();
00067             this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00068         };
  
```

Here is the call graph for this function:



6.8.2.4 KILKENNY_W::KILKENNY_W (const KILKENNY_W & orig)

Copy constructor

Definition at line 15 of file [KILKENNY_W.cpp](#).

```
00015                                     {
00016 }
```

6.8.2.5 KILKENNY_W::~~KILKENNY_W () [virtual]

de-constructor

Definition at line 12 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00012                                     {
00013 }
```

Here is the caller graph for this function:



6.8.3 Member Function Documentation

6.8.3.1 void KILKENNY_W::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>

Definition at line 35 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00035                                     {
00036
00037     std::shared_ptr<KILKENNY_W> objTO = std::dynamic_pointer_cast<KILKENNY_W>(to);
00038     std::shared_ptr<KILKENNY_W> objFROM = std::dynamic_pointer_cast<KILKENNY_W>(from);
00039     objTO->_shop_address = objFROM->GetShop_address();
00040 }
```

```

00040     objTO->_shop_name = objFROM->GetShop_name();
00041     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00042     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00043     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00044     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00045     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00046     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00047     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00048     objTO->Set_Version(objFROM->Get_Version());
00049
00050
00051 }

```

Here is the caller graph for this function:



6.8.3.2 `std::shared_ptr< OSTM > KILKENNY_W::getBaseCopy (std::shared_ptr< OSTM > object)` [virtual]

getBaseCopy function, make deep copy of the object/pointer and Return a new BANK* type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a BANK* return type

Definition at line 22 of file [KILKENNY_W.cpp](#).

References [KILKENNY_W\(\)](#).

Referenced by [operator=\(\)](#).

```

00023 {
00024
00025     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00026     std::shared_ptr<WAREHOUSE> obj(new KILKENNY_W(objTO, object->Get_Version(),object->
Get_Unique_ID()));
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00028     return ostm_obj;
00029 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.3.3 `int KILKENNY_W::GetNumber_of_alcatel () [virtual]`

Implements [WAREHOUSE](#).

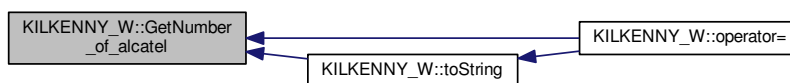
Definition at line 73 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00073 {
00074     return _number_of_alcatel;
00075 }
```

Here is the caller graph for this function:



6.8.3.4 `int KILKENNY_W::GetNumber_of_huawei () [virtual]`

Implements [WAREHOUSE](#).

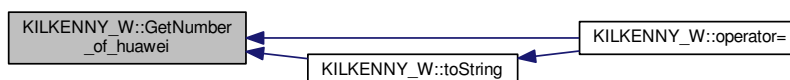
Definition at line 89 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00089 {
00090     return _number_of_huawei;
00091 }
```

Here is the caller graph for this function:



6.8.3.5 int KILKENNY_W::GetNumber_of_iphones () [virtual]

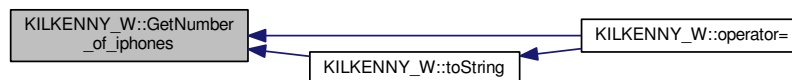
Implements [WAREHOUSE](#).

Definition at line 113 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00113 {  
00114     return _number_of_iphones;  
00115 }
```

Here is the caller graph for this function:



6.8.3.6 int KILKENNY_W::GetNumber_of_nokia () [virtual]

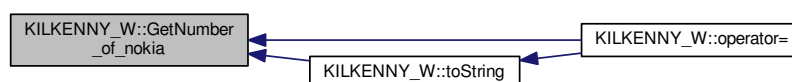
Implements [WAREHOUSE](#).

Definition at line 81 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00081 {  
00082     return _number_of_nokia;  
00083 }
```

Here is the caller graph for this function:



6.8.3.7 int KILKENNY_W::GetNumber_of_samsung() [virtual]

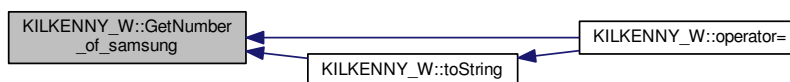
Implements [WAREHOUSE](#).

Definition at line 105 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00105 {  
00106     return _number_of_samsung;  
00107 }
```

Here is the caller graph for this function:



6.8.3.8 int KILKENNY_W::GetNumber_of_sony() [virtual]

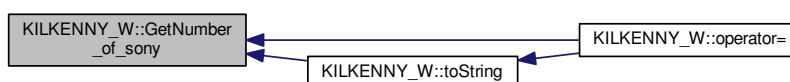
Implements [WAREHOUSE](#).

Definition at line 97 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00097 {  
00098     return _number_of_sony;  
00099 }
```

Here is the caller graph for this function:



6.8.3.9 `std::string KILKENNY_W::GetShop_address () [virtual]`

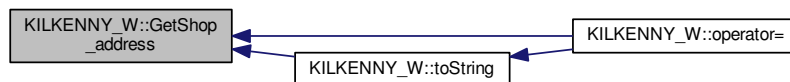
Implements [WAREHOUSE](#).

Definition at line 129 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00129                                     {  
00130     return _shop_address;  
00131 }
```

Here is the caller graph for this function:



6.8.3.10 `std::string KILKENNY_W::GetShop_name () [virtual]`

Implements [WAREHOUSE](#).

Definition at line 121 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00121                                     {  
00122     return _shop_name;  
00123 }
```

Here is the caller graph for this function:



6.8.3.11 KILKENNY_W KILKENNY_W::operator= (const KILKENNY_W & orig) [inline]

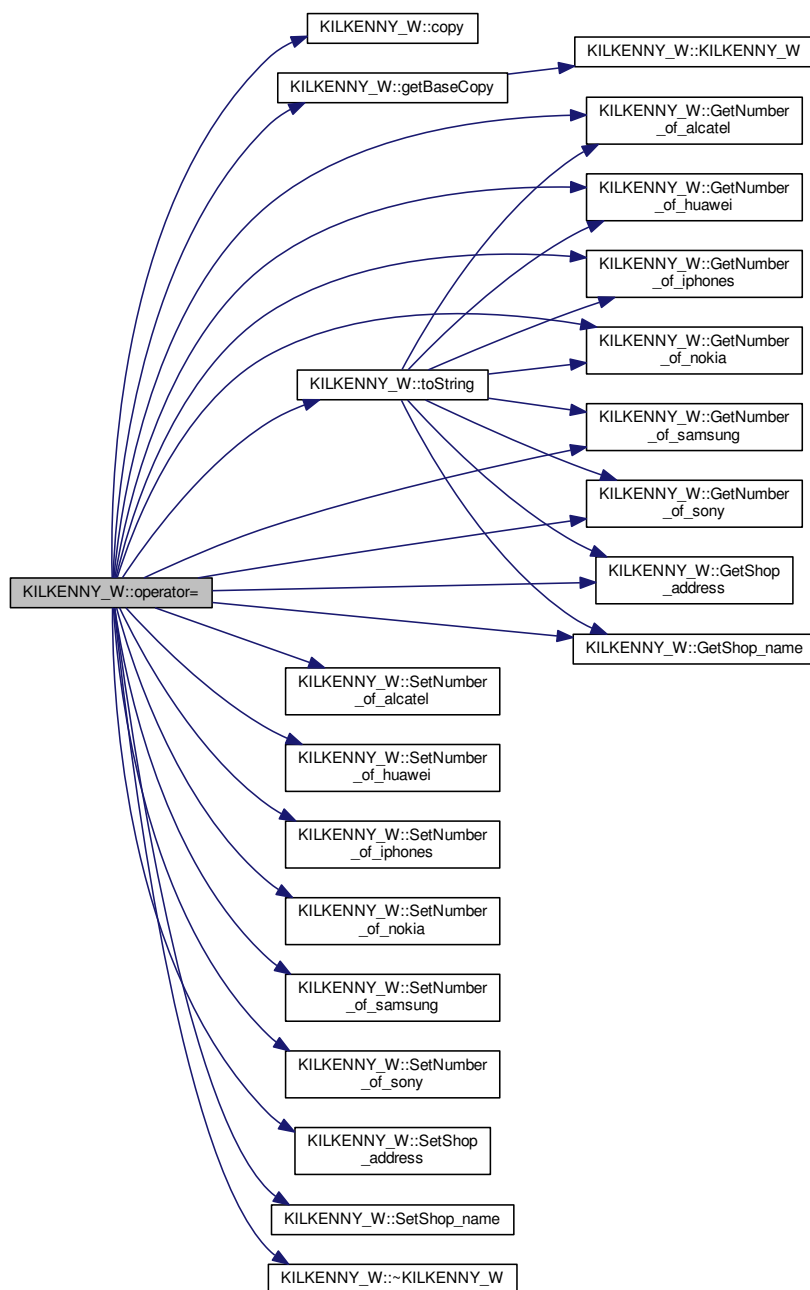
Operator

Definition at line 75 of file KILKENNY_W.h.

References [copy\(\)](#), [getBaseCopy\(\)](#), [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), [GetShop_name\(\)](#), [SetNumber_of_alcatel\(\)](#), [SetNumber_of_huawei\(\)](#), [SetNumber_of_iphones\(\)](#), [SetNumber_of_nokia\(\)](#), [SetNumber_of_samsung\(\)](#), [SetNumber_of_sony\(\)](#), [SetShop_address\(\)](#), [SetShop_name\(\)](#), [toString\(\)](#), and [~KILKENNY_W\(\)](#).

```
00075 {};
```

Here is the call graph for this function:



6.8.3.12 `void KILKENNY_W::SetNumber_of_alcatel (int _number_of_alcatel) [virtual]`

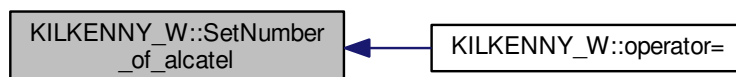
Implements [WAREHOUSE](#).

Definition at line 69 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00069                                     {  
00070     this->_number_of_alcatel = _number_of_alcatel;  
00071 }
```

Here is the caller graph for this function:



6.8.3.13 `void KILKENNY_W::SetNumber_of_huawei (int _number_of_huawei) [virtual]`

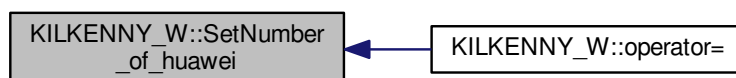
Implements [WAREHOUSE](#).

Definition at line 85 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00085                                     {  
00086     this->_number_of_huawei = _number_of_huawei;  
00087 }
```

Here is the caller graph for this function:



6.8.3.14 void KILKENNY_W::SetNumber_of_iphones (int *_number_of_iphones*) [virtual]

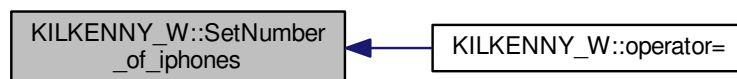
Implements [WAREHOUSE](#).

Definition at line 109 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00109                                     {  
00110     this->_number_of_iphones = _number_of_iphones;  
00111 }
```

Here is the caller graph for this function:



6.8.3.15 void KILKENNY_W::SetNumber_of_nokia (int *_number_of_nokia*) [virtual]

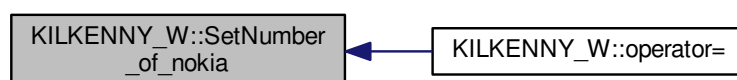
Implements [WAREHOUSE](#).

Definition at line 77 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00077                                     {  
00078     this->_number_of_nokia = _number_of_nokia;  
00079 }
```

Here is the caller graph for this function:



6.8.3.16 void KILKENNY_W::SetNumber_of_samsung (int *_number_of_samsung*) [virtual]

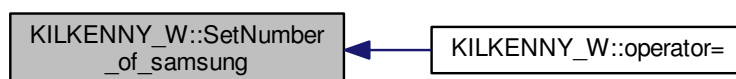
Implements [WAREHOUSE](#).

Definition at line 101 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00101                                     {  
00102     this->_number_of_samsung = _number_of_samsung;  
00103 }
```

Here is the caller graph for this function:



6.8.3.17 void KILKENNY_W::SetNumber_of_sony (int *_number_of_sony*) [virtual]

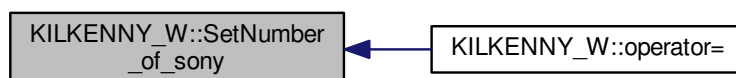
Implements [WAREHOUSE](#).

Definition at line 93 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00093                                     {  
00094     this->_number_of_sony = _number_of_sony;  
00095 }
```

Here is the caller graph for this function:



6.8.3.18 void KILKENNY_W::SetShop_address (std::string _shop_address) [virtual]

Implements [WAREHOUSE](#).

Definition at line 125 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00125                                     {  
00126     this->_shop_address = _shop_address;  
00127 }
```

Here is the caller graph for this function:



6.8.3.19 void KILKENNY_W::SetShop_name (std::string _shop_name) [virtual]

Implements [WAREHOUSE](#).

Definition at line 117 of file [KILKENNY_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00117                                     {  
00118     this->_shop_name = _shop_name;  
00119 }
```

Here is the caller graph for this function:



6.8.3.20 void KILKENNY_W::toString () [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

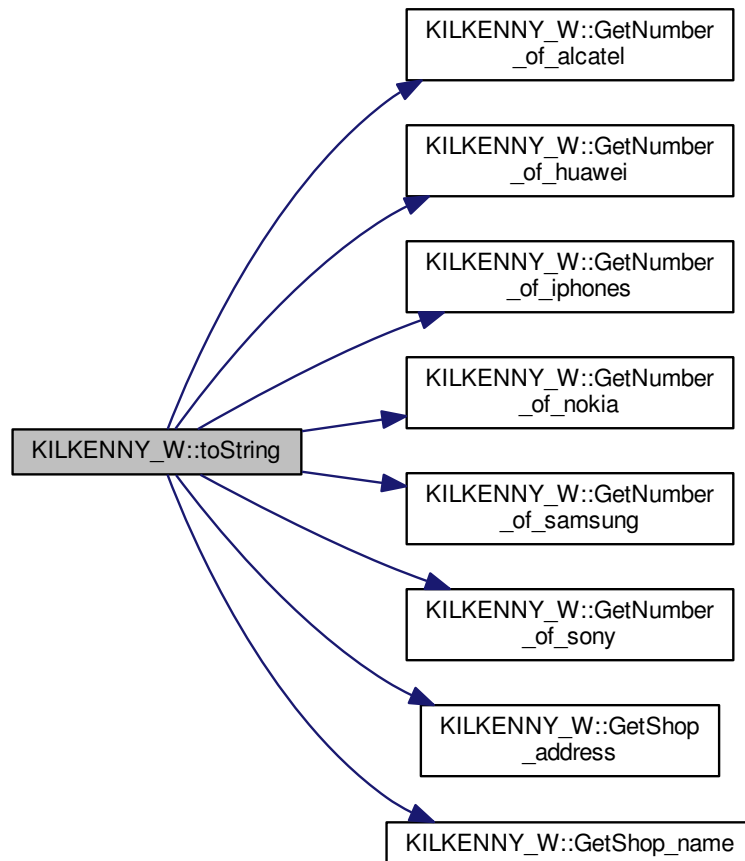
Definition at line 62 of file KILKENNY_W.cpp.

References [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), and [GetShop_name\(\)](#).

Referenced by [operator=\(\)](#).

```
00063 {
00064     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
    \nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
    GetShop_address() << "\nNo. Iphones : " << this->
    GetNumber_of_iphones() << "\nNo. Samsung : " << this->
    GetNumber_of_samsung() << "\nNo. Sony : " << this->
    GetNumber_of_sony() << "\nNo. Huawei : " << this->
    GetNumber_of_huawei() << "\nNo. Nokia : " << this->
    GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
    GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00065 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



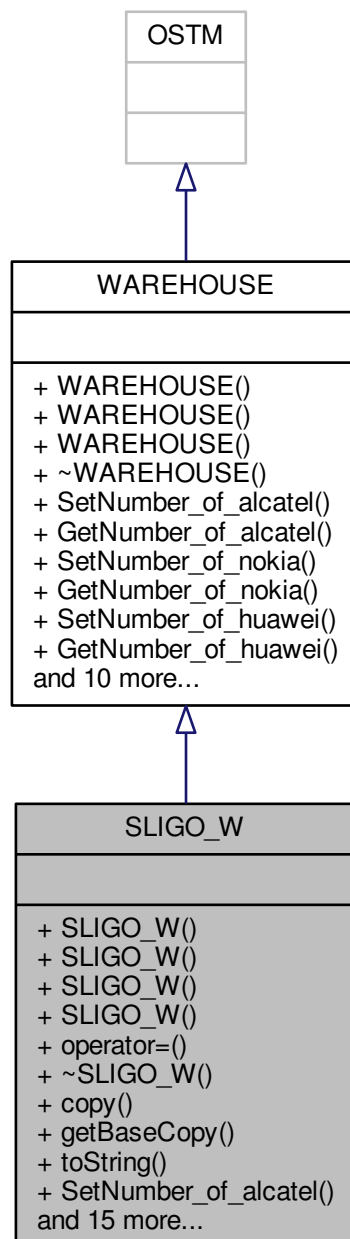
The documentation for this class was generated from the following files:

- [KILKENNY_W.h](#)
- [KILKENNY_W.cpp](#)

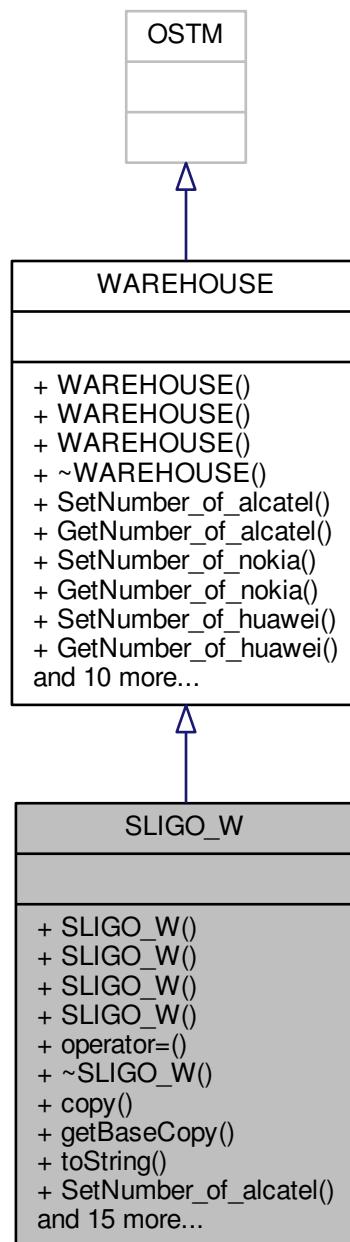
6.9 SLIGO_W Class Reference

```
#include <SLIGO_W.h>
```

Inheritance diagram for SLIGO_W:



Collaboration diagram for SLIGO_W:



Public Member Functions

- [SLIGO_W](#) ()
- [SLIGO_W](#) (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel)
- [SLIGO_W](#) (std::shared_ptr< [WAREHOUSE](#) > obj, int _version, int _unique_id)
- [SLIGO_W](#) (const [SLIGO_W](#) &orig)

- [SLIGO_W operator=](#) (const [SLIGO_W](#) &orig)
- virtual [~SLIGO_W](#) ()
- virtual void [copy](#) (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)
copy function, make deep copy of the object/pointer
- virtual std::shared_ptr< OSTM > [getBaseCopy](#) (std::shared_ptr< OSTM > object)
getBaseCopy function, make deep copy of the object/pointer and Return a new BANK type object*
- virtual void [toString](#) ()
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void [SetNumber_of_alcatel](#) (int _number_of_alcatel)
- virtual int [GetNumber_of_alcatel](#) ()
- virtual void [SetNumber_of_nokia](#) (int _number_of_nokia)
- virtual int [GetNumber_of_nokia](#) ()
- virtual void [SetNumber_of_huawei](#) (int _number_of_huawei)
- virtual int [GetNumber_of_huawei](#) ()
- virtual void [SetNumber_of_sony](#) (int _number_of_sony)
- virtual int [GetNumber_of_sony](#) ()
- virtual void [SetNumber_of_samsung](#) (int _number_of_samsung)
- virtual int [GetNumber_of_samsung](#) ()
- virtual void [SetNumber_of_iphones](#) (int _number_of_iphones)
- virtual int [GetNumber_of_iphones](#) ()
- virtual void [SetShop_name](#) (std::string _shop_name)
- virtual std::string [GetShop_name](#) ()
- virtual void [SetShop_address](#) (std::string _shop_address)
- virtual std::string [GetShop_address](#) ()

6.9.1 Detailed Description

Inherit from [WAREHOUSE](#)

Definition at line 19 of file [SLIGO_W.h](#).

6.9.2 Constructor & Destructor Documentation

6.9.2.1 [SLIGO_W::SLIGO_W](#) () [inline]

Constructor

Definition at line 24 of file [SLIGO_W.h](#).

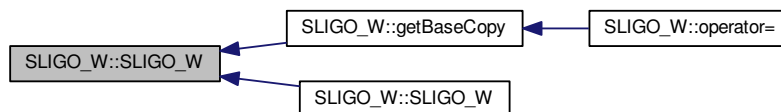
Referenced by [getBaseCopy\(\)](#), and [SLIGO_W\(\)](#).

```

00024         : WAREHOUSE() {
00025
00026         this->_shop_address = "Sligo River Street";
00027         this->_shop_name = "SLIGO S_WAREHOUSE";
00028         this->_number_of_iphones = 200;
00029         this->_number_of_samsung = 200;
00030         this->_number_of_sony = 200;
00031         this->_number_of_huawei = 200;
00032         this->_number_of_nokia = 200;
00033         this->_number_of_alcatel = 200;
00034     };

```

Here is the caller graph for this function:



6.9.2.2 SLIGO_W::SLIGO_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel) [inline]

Custom constructor

Definition at line 38 of file [SLIGO_W.h](#).

```

00038
                                : WAREHOUSE() {
00039     /*
00040      * copy over values
00041      */
00042     this->_shop_address = address;
00043     this->_shop_name = shop_name;
00044     this->_number_of_iphones = iphone;
00045     this->_number_of_samsung = samsung;
00046     this->_number_of_sony = sony;
00047     this->_number_of_huawei = huawei;
00048     this->_number_of_nokia = nokia;
00049     this->_number_of_alcatel = alcatel;
00050
00051 };
  
```

6.9.2.3 SLIGO_W::SLIGO_W(std::shared_ptr< WAREHOUSE > obj, int _version, int _unique_id) [inline]

Custom constructor, used by the library for deep copying

Definition at line 55 of file [SLIGO_W.h](#).

References [SLIGO_W\(\)](#).

```

00055
                                :
00056     WAREHOUSE(_version, _unique_id) {
00057     /*
00058      * copy over values
00059      */
00059     this->_shop_address = obj->GetShop_address();
00060     this->_shop_name = obj->GetShop_name();
00061     this->_number_of_iphones = obj->GetNumber_of_iphones();
00062     this->_number_of_samsung = obj->GetNumber_of_samsung();
00063     this->_number_of_sony = obj->GetNumber_of_sony();
00064     this->_number_of_huawei = obj->GetNumber_of_huawei();
00065     this->_number_of_nokia = obj->GetNumber_of_nokia();
00066     this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067 }
  
```

Here is the call graph for this function:



6.9.2.4 SLIGO_W::SLIGO_W (const SLIGO_W & orig)

Copy constructor

Definition at line 15 of file [SLIGO_W.cpp](#).

```
00015                                     {
00016 }
```

6.9.2.5 SLIGO_W::~~SLIGO_W () [virtual]

de-constructor

Definition at line 12 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00012                                     {
00013 }
```

Here is the caller graph for this function:



6.9.3 Member Function Documentation

6.9.3.1 void SLIGO_W::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>

Definition at line 35 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00035                                     {
00036
00037     std::shared_ptr<SLIGO_W> objTO = std::dynamic_pointer_cast<SLIGO_W>(to);
00038     std::shared_ptr<SLIGO_W> objFROM = std::dynamic_pointer_cast<SLIGO_W>(from);
```

```

00039     objTO->_shop_address = objFROM->GetShop_address();
00040     objTO->_shop_name = objFROM->GetShop_name();
00041     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00042     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00043     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00044     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00045     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00046     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00047     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00048     objTO->Set_Version(objFROM->Get_Version());
00049
00050
00051 }

```

Here is the caller graph for this function:



6.9.3.2 `std::shared_ptr< OSTM > SLIGO_W::getBaseCopy (std::shared_ptr< OSTM > object)` [virtual]

`getBaseCopy` function, make deep copy of the object/pointer and Return a new `BANK*` type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a <code>BANK*</code> return type

Definition at line 22 of file [SLIGO_W.cpp](#).

References [SLIGO_W\(\)](#).

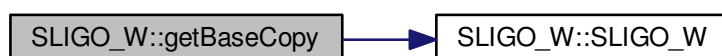
Referenced by [operator=\(\)](#).

```

00023 {
00024
00025     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00026     std::shared_ptr<WAREHOUSE> obj(new SLIGO_W(objTO, object->Get_Version(), object->Get_Unique_ID()));
00027 };
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00028
00028     return ostm_obj;
00029 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.3.3 `int SLIGO_W::GetNumber_of_alcatel () [virtual]`

Implements [WAREHOUSE](#).

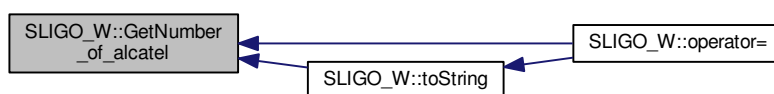
Definition at line 73 of file `SLIGO_W.cpp`.

Referenced by `operator=()`, and `toString()`.

```

00073 {
00074     return _number_of_alcatel;
00075 }
  
```

Here is the caller graph for this function:



6.9.3.4 `int SLIGO_W::GetNumber_of_huawei () [virtual]`

Implements [WAREHOUSE](#).

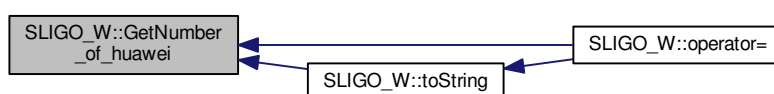
Definition at line 89 of file `SLIGO_W.cpp`.

Referenced by `operator=()`, and `toString()`.

```

00089 {
00090     return _number_of_huawei;
00091 }
  
```

Here is the caller graph for this function:



6.9.3.5 int SLIGO_W::GetNumber_of_iphones () [virtual]

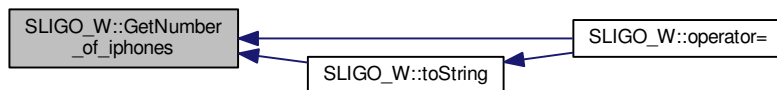
Implements [WAREHOUSE](#).

Definition at line 113 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00113 {  
00114     return _number_of_iphones;  
00115 }
```

Here is the caller graph for this function:



6.9.3.6 int SLIGO_W::GetNumber_of_nokia () [virtual]

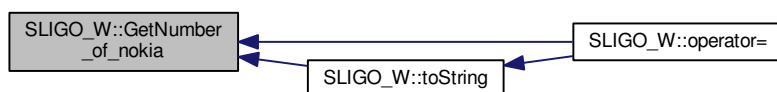
Implements [WAREHOUSE](#).

Definition at line 81 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00081 {  
00082     return _number_of_nokia;  
00083 }
```

Here is the caller graph for this function:



6.9.3.7 int SLIGO_W::GetNumber_of_samsung () [virtual]

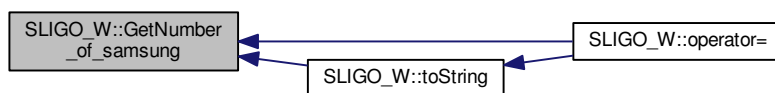
Implements [WAREHOUSE](#).

Definition at line 105 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00105 {  
00106     return _number_of_samsung;  
00107 }
```

Here is the caller graph for this function:



6.9.3.8 int SLIGO_W::GetNumber_of_sony () [virtual]

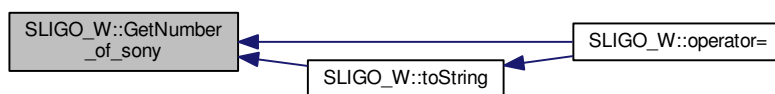
Implements [WAREHOUSE](#).

Definition at line 97 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00097 {  
00098     return _number_of_sony;  
00099 }
```

Here is the caller graph for this function:



6.9.3.9 std::string SLIGO_W::GetShop_address () [virtual]

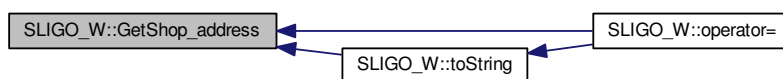
Implements [WAREHOUSE](#).

Definition at line 129 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00129                                     {  
00130     return _shop_address;  
00131 }
```

Here is the caller graph for this function:



6.9.3.10 std::string SLIGO_W::GetShop_name () [virtual]

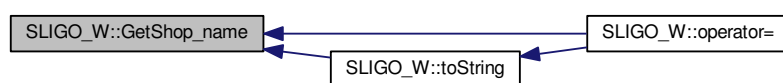
Implements [WAREHOUSE](#).

Definition at line 121 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00121                                     {  
00122     return _shop_name;  
00123 }
```

Here is the caller graph for this function:



6.9.3.11 `SLIGO_W SLIGO_W::operator= (const SLIGO_W & orig) [inline]`

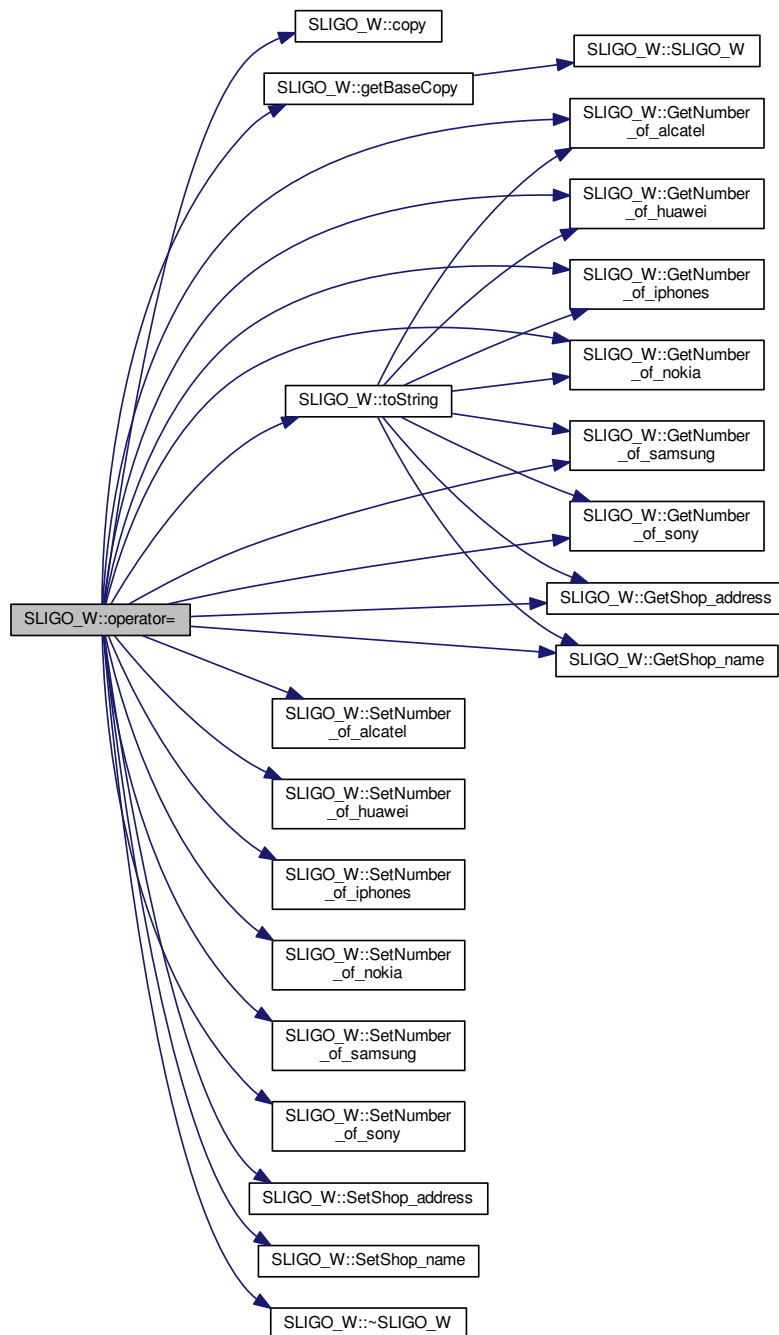
Operator

Definition at line 75 of file [SLIGO_W.h](#).

References [copy\(\)](#), [getBaseCopy\(\)](#), [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), [GetShop_name\(\)](#), [SetNumber_of_alcatel\(\)](#), [SetNumber_of_huawei\(\)](#), [SetNumber_of_iphones\(\)](#), [SetNumber_of_nokia\(\)](#), [SetNumber_of_samsung\(\)](#), [SetNumber_of_sony\(\)](#), [SetShop_address\(\)](#), [SetShop_name\(\)](#), [toString\(\)](#), and [~SLIGO_W\(\)](#).

00075 {};

Here is the call graph for this function:



6.9.3.12 `void SLIGO_W::SetNumber_of_alcatel (int number_of_alcatel)` [virtual]

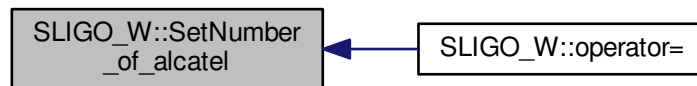
Implements [WAREHOUSE](#).

Definition at line 69 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00069                                     {  
00070     this->_number_of_alcatel = _number_of_alcatel;  
00071 }
```

Here is the caller graph for this function:



6.9.3.13 void SLIGO_W::SetNumber_of_huawei (int _number_of_huawei) [virtual]

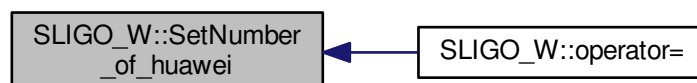
Implements [WAREHOUSE](#).

Definition at line 85 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00085                                     {  
00086     this->_number_of_huawei = _number_of_huawei;  
00087 }
```

Here is the caller graph for this function:



6.9.3.14 void SLIGO_W::SetNumber_of_iphones (int _number_of_iphones) [virtual]

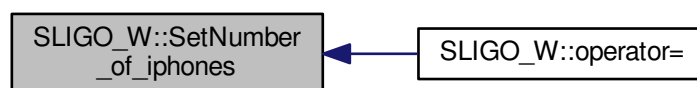
Implements [WAREHOUSE](#).

Definition at line 109 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00109                                     {  
00110     this->_number_of_iphones = _number_of_iphones;  
00111 }
```

Here is the caller graph for this function:



6.9.3.15 void SLIGO_W::SetNumber_of_nokia (int _number_of_nokia) [virtual]

Implements [WAREHOUSE](#).

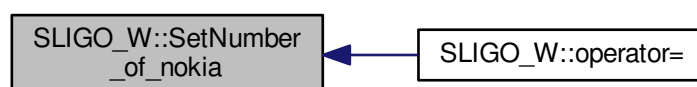
Definition at line 77 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```

00077                                     {
00078     this->_number_of_nokia = _number_of_nokia;
00079 }
  
```

Here is the caller graph for this function:



6.9.3.16 void SLIGO_W::SetNumber_of_samsung (int _number_of_samsung) [virtual]

Implements [WAREHOUSE](#).

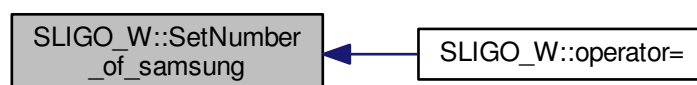
Definition at line 101 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```

00101                                     {
00102     this->_number_of_samsung = _number_of_samsung;
00103 }
  
```

Here is the caller graph for this function:



6.9.3.17 void SLIGO_W::SetNumber_of_sony (int *_number_of_sony*) [virtual]

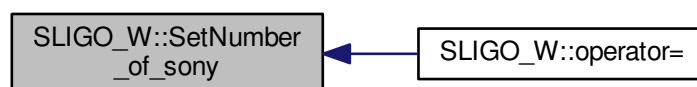
Implements [WAREHOUSE](#).

Definition at line 93 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00093                                     {  
00094     this->_number_of_sony = _number_of_sony;  
00095 }
```

Here is the caller graph for this function:



6.9.3.18 void SLIGO_W::SetShop_address (std::string *_shop_address*) [virtual]

Implements [WAREHOUSE](#).

Definition at line 125 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00125                                     {  
00126     this->_shop_address = _shop_address;  
00127 }
```

Here is the caller graph for this function:



6.9.3.19 void SLIGO_W::SetShop_name (std::string _shop_name) [virtual]

Implements [WAREHOUSE](#).

Definition at line 117 of file [SLIGO_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00117 {
00118     this->_shop_name = _shop_name;
00119 }
```

Here is the caller graph for this function:



6.9.3.20 void SLIGO_W::toString () [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

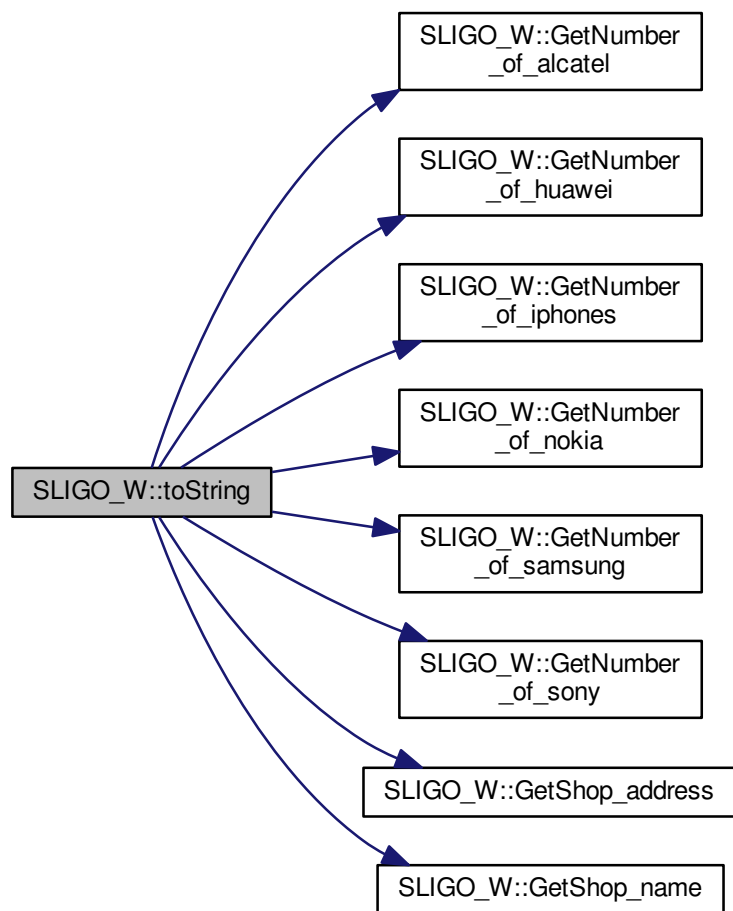
Definition at line 62 of file [SLIGO_W.cpp](#).

References [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), and [GetShop_name\(\)](#).

Referenced by [operator=\(\)](#).

```
00063 {
00064     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() <<
"\nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
GetShop_address() << "\nNo. Iphones : " << this->
GetNumber_of_iphones() << "\nNo. Samsung : " << this->
GetNumber_of_samsung() << "\nNo. Sony : " << this->
GetNumber_of_sony() << "\nNo. Huawei : " << this->
GetNumber_of_huawei() << "\nNo. Nokia : " << this->
GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00065 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



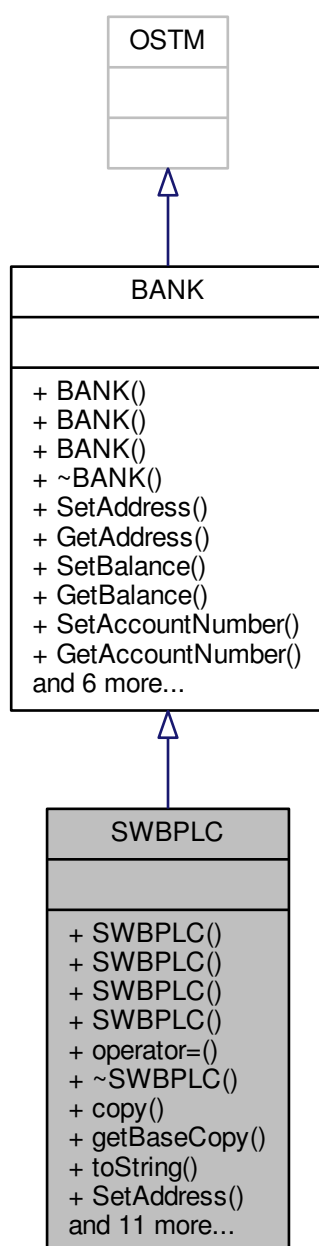
The documentation for this class was generated from the following files:

- [SLIGO_W.h](#)
- [SLIGO_W.cpp](#)

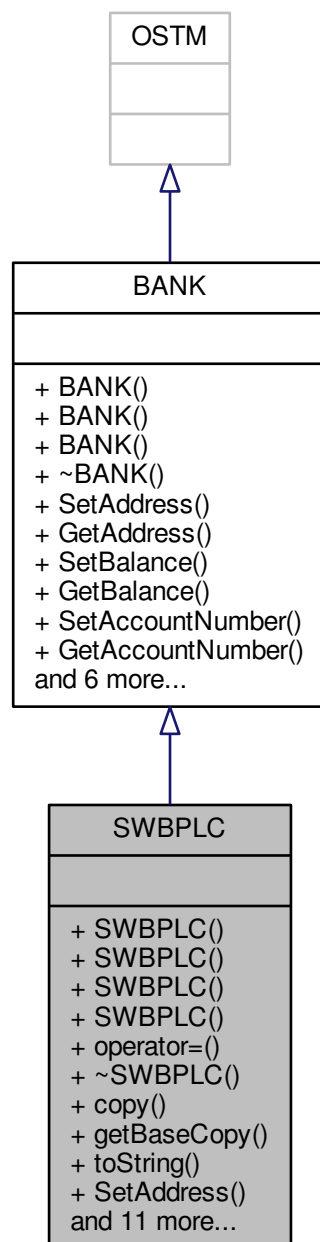
6.10 SWBPLC Class Reference

```
#include <SWBPLC.h>
```

Inheritance diagram for SWBPLC:



Collaboration diagram for SWBPLC:



Public Member Functions

- `SWBPLC ()`
- `SWBPLC (int accountNumber, double balance, std::string firstName, std::string lastName, std::string address)`
- `SWBPLC (std::shared_ptr< BANK > obj, int _version, int _unique_id)`
- `SWBPLC (const SWBPLC &orig)`
- `SWBPLC operator= (const SWBPLC &orig)`

- virtual `~SWBPLC()`
- virtual void `copy` (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)
copy function, make deep copy of the object/pointer
- virtual std::shared_ptr< OSTM > `getBaseCopy` (std::shared_ptr< OSTM > object)
getBaseCopy function, make deep copy of the object/pointer and Return a new std::shared_ptr< BANK > type object
- virtual void `toString` ()
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void `SetAddress` (std::string address)
- virtual std::string `GetAddress` () const
- virtual void `SetBalance` (double balance)
- virtual double `GetBalance` () const
- virtual void `SetAccountNumber` (int accountNumber)
- virtual int `GetAccountNumber` () const
- virtual void `SetLastName` (std::string lastName)
- virtual std::string `GetLastName` () const
- virtual void `SetFirstName` (std::string firstName)
- virtual std::string `GetFirstName` () const
- virtual void `SetFullname` (std::string fullname)
- virtual std::string `GetFullname` () const

6.10.1 Detailed Description

Inherit from [BANK](#)

Definition at line 19 of file [SWBPLC.h](#).

6.10.2 Constructor & Destructor Documentation

6.10.2.1 SWBPLC::SWBPLC() [inline]

Constructor

Definition at line 24 of file [SWBPLC.h](#).

Referenced by [getBaseCopy\(\)](#), and [SWBPLC\(\)](#).

```

00024         : BANK() {
00025             this->accountNumber = 0;
00026             this->balance = 50;
00027             this->firstName = "Joe";
00028             this->lastName = "Blog";
00029             this->address = "High street, Carlow";
00030             this->fullname = firstName + " " + lastName;
00031     };

```

Here is the caller graph for this function:



6.10.2.2 SWBPLC::SWBPLC (int *accountNumber*, double *balance*, std::string *firstName*, std::string *lastName*, std::string *address*) [inline]

Custom constructor

Definition at line 35 of file [SWBPLC.h](#).

```

00035                                     :
00036     BANK() {
00037         this->accountNumber = accountNumber;
00038         this->balance = balance;
00039         this->firstName = firstName;
00040         this->lastName = lastName;
00041         this->address = address;
00042         this->fullname = firstName + " " + lastName;
00043     };

```

6.10.2.3 SWBPLC::SWBPLC (std::shared_ptr< BANK > *obj*, int *_version*, int *_unique_id*) [inline]

Custom constructor, used by the library for deep copying

Definition at line 46 of file [SWBPLC.h](#).

References [SWBPLC\(\)](#).

```

00046                                     : BANK(_version, _unique_id) {
00047
00048         this->accountNumber = obj->GetAccountNumber();
00049         this->balance = obj->GetBalance();
00050         this->firstName = obj->GetFirstName();
00051         this->lastName = obj->GetLastName();
00052         this->address = obj->GetAddress();
00053         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00054
00055     };

```

Here is the call graph for this function:



6.10.2.4 SWBPLC::SWBPLC (const SWBPLC & *orig*)

Copy constructor

Definition at line 12 of file [SWBPLC.cpp](#).

```

00012     {
00013 }

```

6.10.2.5 SWBPLC::~~SWBPLC () [virtual]

de-constructor

Definition at line 15 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#).

```
00015         {
00016     }
```

Here is the caller graph for this function:



6.10.3 Member Function Documentation

6.10.3.1 void SWBPLC::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>

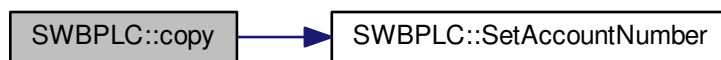
Definition at line 34 of file [SWBPLC.cpp](#).

References [SetAccountNumber\(\)](#).

Referenced by [operator=\(\)](#).

```
00034         {
00035
00036         std::shared_ptr<SWBPLC> objTO = std::dynamic_pointer_cast<SWBPLC>(to);
00037         std::shared_ptr<SWBPLC> objFROM = std::dynamic_pointer_cast<SWBPLC>(from);
00038         objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00039         objTO->Set_Version(objFROM->Get_Version());
00040         objTO->SetAccountNumber(objFROM->GetAccountNumber());
00041         objTO->SetBalance(objFROM->GetBalance());
00042
00043
00044     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.10.3.2 `int SWBPLC::GetAccountNumber () const [virtual]`

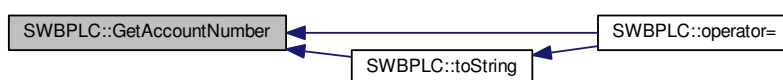
Implements [BANK](#).

Definition at line 80 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00080                                     {  
00081     return accountNumber;  
00082 }
```

Here is the caller graph for this function:



6.10.3.3 `std::string SWBPLC::GetAddress () const [virtual]`

Implements [BANK](#).

Definition at line 64 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#).

```
00064                                     {
00065     return address;
00066 }
```

Here is the caller graph for this function:



6.10.3.4 `double SWBPLC::GetBalance () const [virtual]`

Implements [BANK](#).

Definition at line 72 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00072                                     {
00073     return balance;
00074 }
```

Here is the caller graph for this function:



6.10.3.5 `std::shared_ptr<OSTM> SWBPLC::getBaseCopy (std::shared_ptr<OSTM> object) [virtual]`

`getBaseCopy` function, make deep copy of the object/pointer and Return a new `std::shared_ptr<BANK>` type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a <code>std::shared_ptr<BANK></code> return type

Definition at line 22 of file [SWBPLC.cpp](#).

References [SWBPLC\(\)](#).

Referenced by [operator=\(\)](#).

```

00023 {
00024     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00025     std::shared_ptr<BANK> obj(new SWBPLC(objTO,object->Get_Version(),object->Get_Unique_ID()));
00026     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

00027     return ostm_obj;
00028 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.10.3.6 `std::string SWBPLC::GetFirstName () const` [virtual]

Implements [BANK](#).

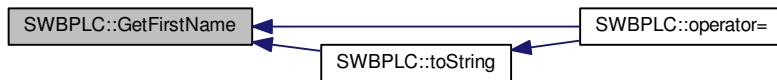
Definition at line 96 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00096                                     {
00097     return firstName;
00098 }
```

Here is the caller graph for this function:



6.10.3.7 `std::string SWBPLC::GetFullName () const` [virtual]

Implements [BANK](#).

Definition at line 104 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#).

```

00104                                     {
00105     return fullname;
00106 }
  
```

Here is the caller graph for this function:



6.10.3.8 `std::string SWBPLC::GetLastName () const` [virtual]

Implements [BANK](#).

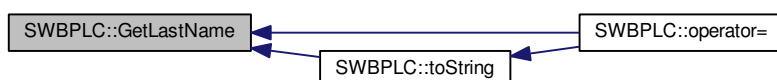
Definition at line 88 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00088                                     {
00089     return lastName;
00090 }
  
```

Here is the caller graph for this function:



6.10.3.9 SWBPLC SWBPLC::operator= (const SWBPLC & orig) [inline]

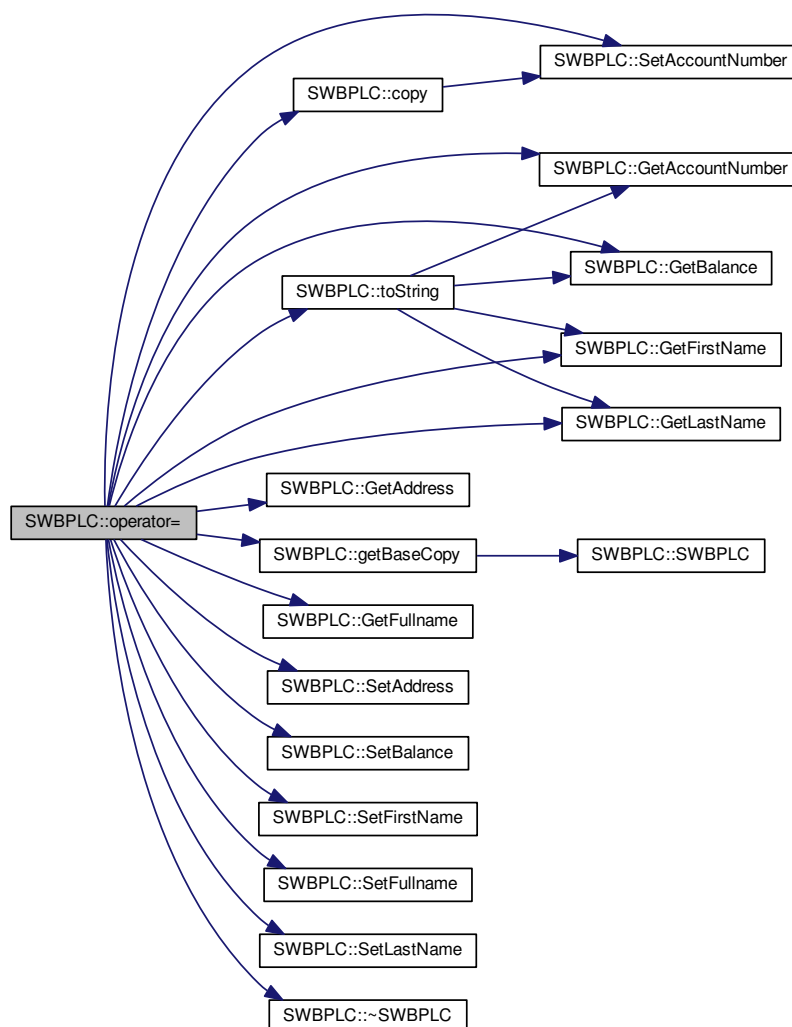
Operator

Definition at line 63 of file [SWBPLC.h](#).

References [copy\(\)](#), [GetAccountNumber\(\)](#), [GetAddress\(\)](#), [GetBalance\(\)](#), [getBaseCopy\(\)](#), [GetFirstName\(\)](#), [GetFullname\(\)](#), [GetLastName\(\)](#), [SetAccountNumber\(\)](#), [SetAddress\(\)](#), [SetBalance\(\)](#), [SetFirstName\(\)](#), [SetFullname\(\)](#), [SetLastName\(\)](#), [toString\(\)](#), and [~SWBPLC\(\)](#).

```
00063 {};
```

Here is the call graph for this function:



6.10.3.10 void SWBPLC::SetAccountNumber (int *accountNumber*) [virtual]

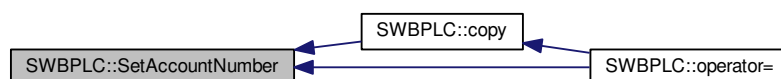
Implements [BANK](#).

Definition at line 76 of file [SWBPLC.cpp](#).

Referenced by [copy\(\)](#), and [operator=\(\)](#).

```
00076                                     {  
00077     this->accountNumber = accountNumber;  
00078 }
```

Here is the caller graph for this function:



6.10.3.11 void SWBPLC::SetAddress (std::string *address*) [virtual]

Implements [BANK](#).

Definition at line 60 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#).

```
00060                                     {  
00061     this->address = address;  
00062 }
```

Here is the caller graph for this function:



6.10.3.12 void SWBPLC::SetBalance (double *balance*) [virtual]

Implements [BANK](#).

Definition at line 68 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#).

```
00068                                     {  
00069     this->balance = balance;  
00070 }
```

Here is the caller graph for this function:



6.10.3.13 void SWBPLC::SetFirstName (std::string *firstName*) [virtual]

Implements [BANK](#).

Definition at line 92 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#).

```
00092                                     {  
00093     this->firstName = firstName;  
00094 }
```

Here is the caller graph for this function:



6.10.3.14 void SWBPLC::SetFullName (std::string *fullname*) [virtual]

Implements [BANK](#).

Definition at line 100 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#).

```
00100                                     {  
00101     this->fullname = fullname;  
00102 }
```

Here is the caller graph for this function:

**6.10.3.15** void SWBPLC::SetLastName (std::string *lastName*) [virtual]

Implements [BANK](#).

Definition at line 84 of file [SWBPLC.cpp](#).

Referenced by [operator=\(\)](#).

```
00084                                     {  
00085     this->lastName = lastName;  
00086 }
```

Here is the caller graph for this function:



6.10.3.16 void SWBPLC::toString() [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

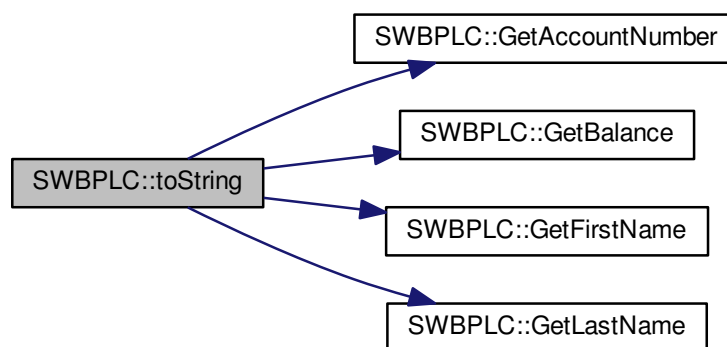
Definition at line 55 of file SWBPLC.cpp.

References [GetAccountNumber\(\)](#), [GetBalance\(\)](#), [GetFirstName\(\)](#), and [GetLastName\(\)](#).

Referenced by [operator=\(\)](#).

```
00056 {
00057     std::cout << "\nSWBPLC BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " <<
    this->GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst
    name: " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\n
    Version number : " << this->Get_Version() << std::endl;
00058 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



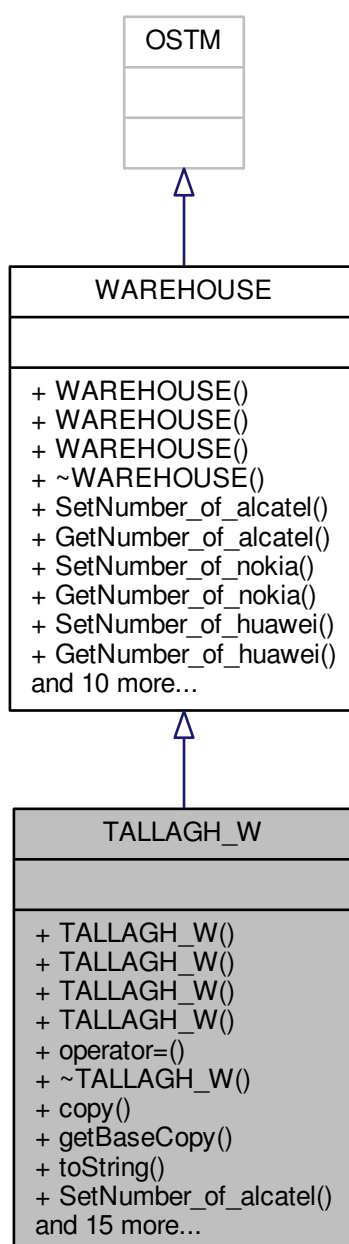
The documentation for this class was generated from the following files:

- [SWBPLC.h](#)
- [SWBPLC.cpp](#)

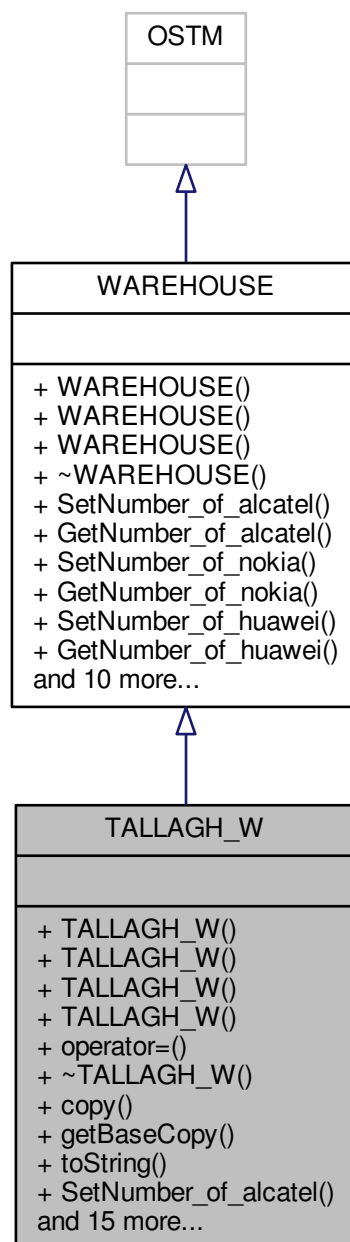
6.11 TALLAGH_W Class Reference

```
#include <TALLAGH_W.h>
```

Inheritance diagram for TALLAGH_W:



Collaboration diagram for TALLAGH_W:



Public Member Functions

- [TALLAGH_W](#) ()
- [TALLAGH_W](#) (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel)
- [TALLAGH_W](#) (std::shared_ptr< [WAREHOUSE](#) > obj, int _version, int _unique_id)
- [TALLAGH_W](#) (const [TALLAGH_W](#) &orig)

- [TALLAGH_W operator=](#) (const [TALLAGH_W](#) &orig)
- virtual [~TALLAGH_W](#) ()
- virtual void [copy](#) (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)
copy function, make deep copy of the object/pointer
- virtual std::shared_ptr< OSTM > [getBaseCopy](#) (std::shared_ptr< OSTM > object)
getBaseCopy function, make deep copy of the object/pointer and Return a new BANK type object*
- virtual void [toString](#) ()
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void [SetNumber_of_alcatel](#) (int _number_of_alcatel)
- virtual int [GetNumber_of_alcatel](#) ()
- virtual void [SetNumber_of_nokia](#) (int _number_of_nokia)
- virtual int [GetNumber_of_nokia](#) ()
- virtual void [SetNumber_of_huawei](#) (int _number_of_huawei)
- virtual int [GetNumber_of_huawei](#) ()
- virtual void [SetNumber_of_sony](#) (int _number_of_sony)
- virtual int [GetNumber_of_sony](#) ()
- virtual void [SetNumber_of_samsung](#) (int _number_of_samsung)
- virtual int [GetNumber_of_samsung](#) ()
- virtual void [SetNumber_of_iphones](#) (int _number_of_iphones)
- virtual int [GetNumber_of_iphones](#) ()
- virtual void [SetShop_name](#) (std::string _shop_name)
- virtual std::string [GetShop_name](#) ()
- virtual void [SetShop_address](#) (std::string _shop_address)
- virtual std::string [GetShop_address](#) ()

6.11.1 Detailed Description

Inherit from [WAREHOUSE](#)

Definition at line 19 of file [TALLAGH_W.h](#).

6.11.2 Constructor & Destructor Documentation

6.11.2.1 TALLAGH_W::TALLAGH_W() [inline]

Constructor

Definition at line 24 of file [TALLAGH_W.h](#).

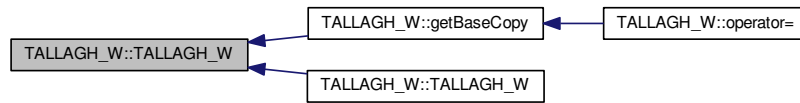
Referenced by [getBaseCopy\(\)](#), and [TALLAGH_W\(\)](#).

```

00024             : WAREHOUSE() {
00025
00026             this->_shop_address = "Tallagh Low street";
00027             this->_shop_name = "TALLAGH T_WAREHOUSE";
00028             this->_number_of_iphones = 200;
00029             this->_number_of_samsung = 200;
00030             this->_number_of_sony = 200;
00031             this->_number_of_huawei = 200;
00032             this->_number_of_nokia = 200;
00033             this->_number_of_alcatel = 200;
00034         };

```

Here is the caller graph for this function:



6.11.2.2 `TALLAGH_W::TALLAGH_W (std::string address, std::string shop_name, int iphone, int samsung, int sony, int huawei, int nokia, int alcatel) [inline]`

Custom constructor

Definition at line 38 of file [TALLAGH_W.h](#).

```

00038                                     : WAREHOUSE() {
00039     /*
00040     * copy over values
00041     */
00042     this->_shop_address = address;
00043     this->_shop_name = shop_name;
00044     this->_number_of_iphones = iphone;
00045     this->_number_of_samsung = samsung;
00046     this->_number_of_sony = sony;
00047     this->_number_of_huawei = huawei;
00048     this->_number_of_nokia = nokia;
00049     this->_number_of_alcatel = alcatel;
00050
00051 };
  
```

6.11.2.3 `TALLAGH_W::TALLAGH_W (std::shared_ptr< WAREHOUSE > obj, int _version, int _unique_id) [inline]`

Custom constructor, used by the library for deep copying

Definition at line 55 of file [TALLAGH_W.h](#).

References [TALLAGH_W\(\)](#).

```

00055                                     :
00055     WAREHOUSE(_version, _unique_id) {
00056     /*
00057     * copy over values
00058     */
00059     this->_shop_address = obj->GetShop_address();
00060     this->_shop_name = obj->GetShop_name();
00061     this->_number_of_iphones = obj->GetNumber_of_iphones();
00062     this->_number_of_samsung = obj->GetNumber_of_samsung();
00063     this->_number_of_sony = obj->GetNumber_of_sony();
00064     this->_number_of_huawei = obj->GetNumber_of_huawei();
00065     this->_number_of_nokia = obj->GetNumber_of_nokia();
00066     this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067 }
  
```

Here is the call graph for this function:



6.11.2.4 TALLAGH_W::TALLAGH_W (const TALLAGH_W & *orig*)

Copy constructor

Definition at line 15 of file [TALLAGH_W.cpp](#).

```
00015                                     {
00016 }
```

6.11.2.5 TALLAGH_W::~~TALLAGH_W () [virtual]

de-constructor

Definition at line 12 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00012                                     {
00013 }
```

Here is the caller graph for this function:



6.11.3 Member Function Documentation

6.11.3.1 void TALLAGH_W::copy (std::shared_ptr< OSTM > *to*, std::shared_ptr< OSTM > *from*) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a BANK* type object casted back from std::shared_ptr<OSTM>

Definition at line 35 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00035                                     {
00036
00037     std::shared_ptr<TALLAGH_W> objTO = std::dynamic_pointer_cast<TALLAGH_W>(to);
00038     std::shared_ptr<TALLAGH_W> objFROM = std::dynamic_pointer_cast<TALLAGH_W>(from);
00039     objTO->_shop_address = objFROM->GetShop_address();
00040 }
```

```

00040     objTO->_shop_name = objFROM->GetShop_name();
00041     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00042     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00043     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00044     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00045     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00046     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00047     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00048     objTO->Set_Version(objFROM->Get_Version());
00049
00050
00051 }

```

Here is the caller graph for this function:



6.11.3.2 `std::shared_ptr< OSTM > TALLAGH_W::getBaseCopy (std::shared_ptr< OSTM > object)` [virtual]

`getBaseCopy` function, make deep copy of the object/pointer and Return a new `BANK*` type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a <code>BANK*</code> return type

Definition at line 22 of file [TALLAGH_W.cpp](#).

References [TALLAGH_W\(\)](#).

Referenced by [operator=\(\)](#).

```

00023 {
00024
00025     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00026     std::shared_ptr<WAREHOUSE> obj(new TALLAGH_W(objTO, object->Get_Version(), object->
        Get_Unique_ID()));
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00028
00028     return ostm_obj;
00029 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.3.3 `int TALLAGH_W::GetNumber_of_alcatel()` [virtual]

Implements [WAREHOUSE](#).

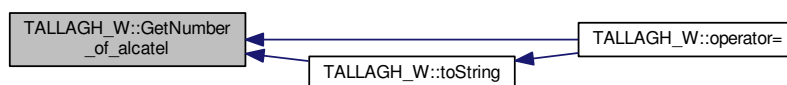
Definition at line 71 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00071 {
00072     return _number_of_alcatel;
00073 }
```

Here is the caller graph for this function:



6.11.3.4 `int TALLAGH_W::GetNumber_of_huawei()` [virtual]

Implements [WAREHOUSE](#).

Definition at line 87 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00087 {
00088     return _number_of_huawei;
00089 }
```

Here is the caller graph for this function:



6.11.3.5 int TALLAGH_W::GetNumber_of_iphones () [virtual]

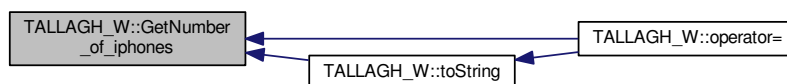
Implements [WAREHOUSE](#).

Definition at line 111 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00111 {  
00112     return _number_of_iphones;  
00113 }
```

Here is the caller graph for this function:



6.11.3.6 int TALLAGH_W::GetNumber_of_nokia () [virtual]

Implements [WAREHOUSE](#).

Definition at line 79 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00079 {  
00080     return _number_of_nokia;  
00081 }
```

Here is the caller graph for this function:



6.11.3.7 int TALLAGH_W::GetNumber_of_samsung () [virtual]

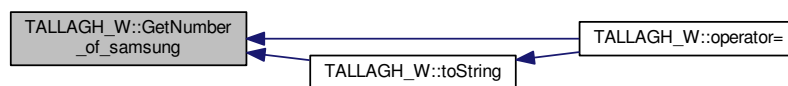
Implements [WAREHOUSE](#).

Definition at line 103 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00103 {  
00104     return _number_of_samsung;  
00105 }
```

Here is the caller graph for this function:



6.11.3.8 int TALLAGH_W::GetNumber_of_sony () [virtual]

Implements [WAREHOUSE](#).

Definition at line 95 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00095 {  
00096     return _number_of_sony;  
00097 }
```

Here is the caller graph for this function:



6.11.3.9 `std::string TALLAGH_W::GetShop_address () [virtual]`

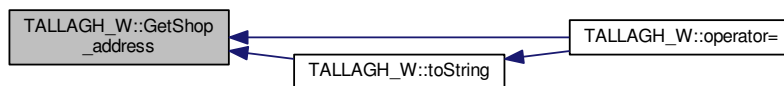
Implements [WAREHOUSE](#).

Definition at line 127 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00127                                     {  
00128     return _shop_address;  
00129 }
```

Here is the caller graph for this function:



6.11.3.10 `std::string TALLAGH_W::GetShop_name () [virtual]`

Implements [WAREHOUSE](#).

Definition at line 119 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00119                                     {  
00120     return _shop_name;  
00121 }
```

Here is the caller graph for this function:



6.11.3.11 TALLAGH_W TALLAGH_W::operator= (const TALLAGH_W & *orig*) [inline]

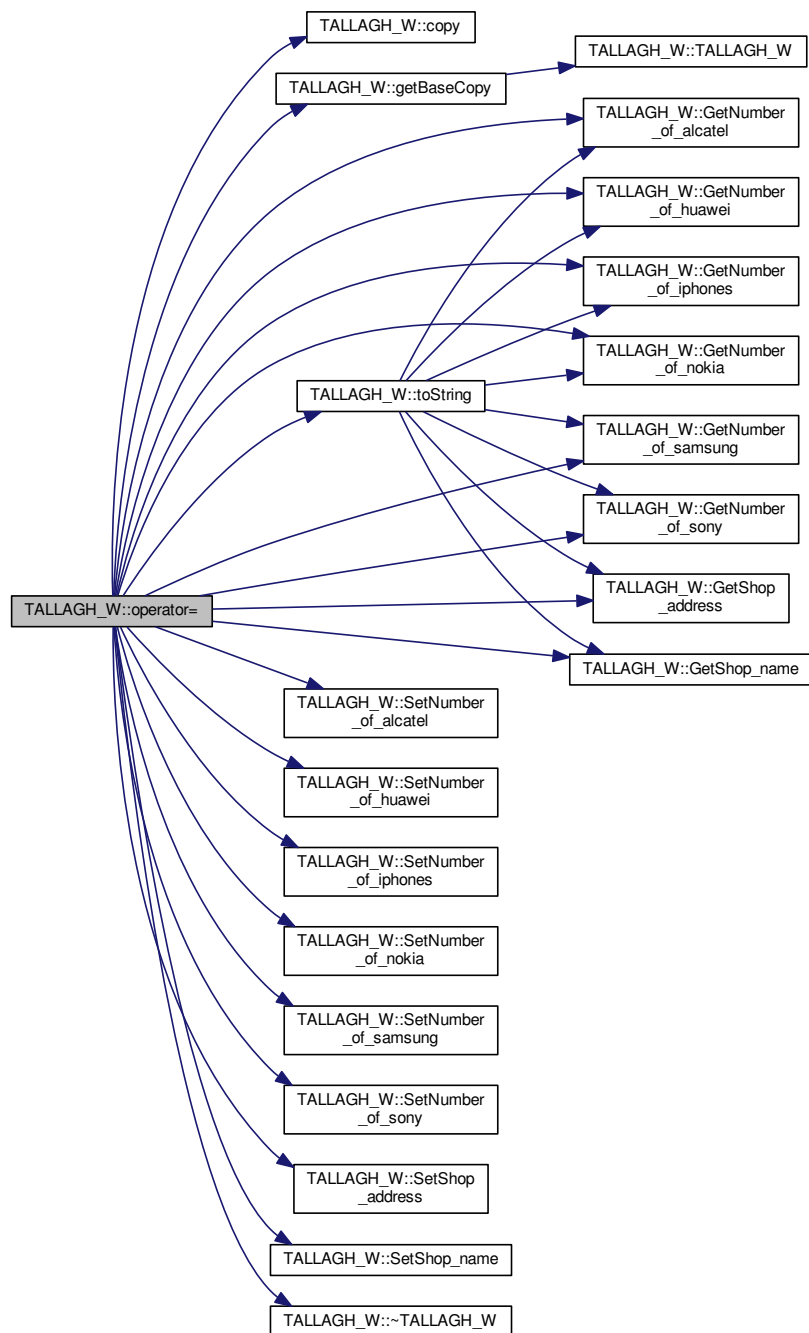
Operator

Definition at line 75 of file [TALLAGH_W.h](#).

References [copy\(\)](#), [getBaseCopy\(\)](#), [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), [GetShop_name\(\)](#), [SetNumber_of_alcatel\(\)](#), [SetNumber_of_huawei\(\)](#), [SetNumber_of_iphones\(\)](#), [SetNumber_of_nokia\(\)](#), [SetNumber_of_samsung\(\)](#), [SetNumber_of_sony\(\)](#), [SetShop_address\(\)](#), [SetShop_name\(\)](#), [toString\(\)](#), and [~TALLAGH_W\(\)](#).

00075 {};

Here is the call graph for this function:



6.11.3.12 `void TALLAGH_W::SetNumber_of_alcatel (int _number_of_alcatel) [virtual]`

Implements [WAREHOUSE](#).

Definition at line 67 of file [TALLAGH_W.cpp](#).

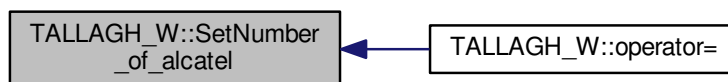
Referenced by [operator=\(\)](#).

```

00067                                     {
00068     this->_number_of_alcatel = _number_of_alcatel;
00069 }

```

Here is the caller graph for this function:



6.11.3.13 void TALLAGH_W::SetNumber_of_huawei (int *_number_of_huawei*) [virtual]

Implements [WAREHOUSE](#).

Definition at line 83 of file [TALLAGH_W.cpp](#).

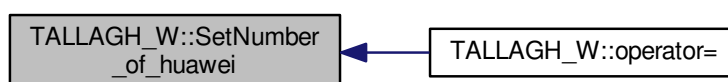
Referenced by [operator=\(\)](#).

```

00083                                     {
00084     this->_number_of_huawei = _number_of_huawei;
00085 }

```

Here is the caller graph for this function:



6.11.3.14 void TALLAGH_W::SetNumber_of_iphones (int *_number_of_iphones*) [virtual]

Implements [WAREHOUSE](#).

Definition at line 107 of file [TALLAGH_W.cpp](#).

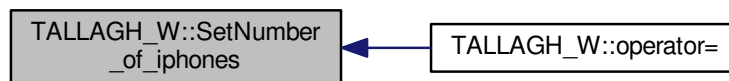
Referenced by [operator=\(\)](#).

```

00107                                     {
00108     this->_number_of_iphones = _number_of_iphones;
00109 }

```

Here is the caller graph for this function:



6.11.3.15 `void TALLAGH_W::SetNumber_of_nokia (int _number_of_nokia) [virtual]`

Implements [WAREHOUSE](#).

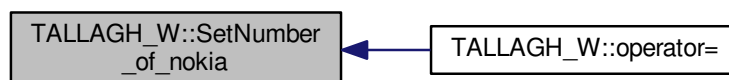
Definition at line 75 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#).

```

00075                                     {
00076     this->_number_of_nokia = _number_of_nokia;
00077 }
  
```

Here is the caller graph for this function:



6.11.3.16 `void TALLAGH_W::SetNumber_of_samsung (int _number_of_samsung) [virtual]`

Implements [WAREHOUSE](#).

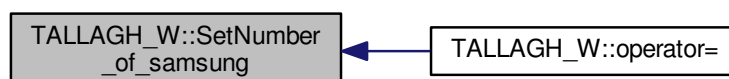
Definition at line 99 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#).

```

00099                                     {
00100     this->_number_of_samsung = _number_of_samsung;
00101 }
  
```

Here is the caller graph for this function:



6.11.3.17 void TALLAGH_W::SetNumber_of_sony (int *_number_of_sony*) [virtual]

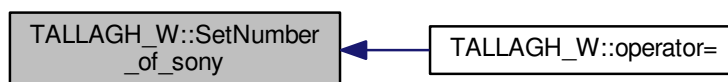
Implements [WAREHOUSE](#).

Definition at line 91 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00091      {  
00092          this->_number_of_sony = _number_of_sony;  
00093      }
```

Here is the caller graph for this function:



6.11.3.18 void TALLAGH_W::SetShop_address (std::string *_shop_address*) [virtual]

Implements [WAREHOUSE](#).

Definition at line 123 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00123      {  
00124          this->_shop_address = _shop_address;  
00125      }
```

Here is the caller graph for this function:



6.11.3.19 void TALLAGH_W::SetShop_name (std::string_shop_name) [virtual]

Implements [WAREHOUSE](#).

Definition at line 115 of file [TALLAGH_W.cpp](#).

Referenced by [operator=\(\)](#).

```
00115                                     {
00116     this->_shop_name = _shop_name;
00117 }
```

Here is the caller graph for this function:



6.11.3.20 void TALLAGH_W::toString () [virtual]

`_cast`, is use to cast bak the `std::shared_ptr<OSTM>` to the required type

`toString` function, displays the object values in formatted way

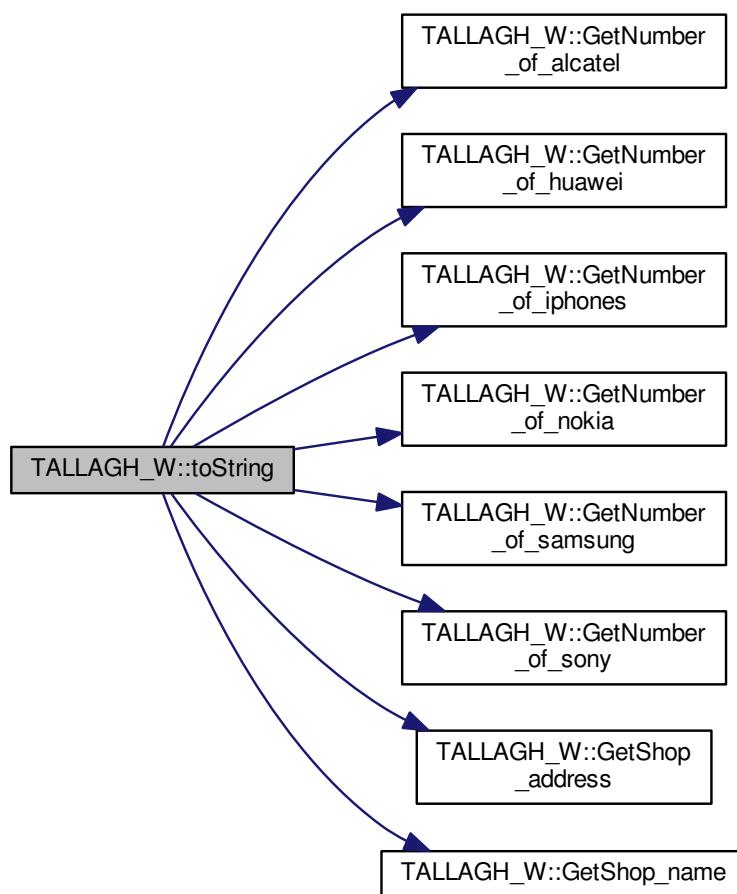
Definition at line 62 of file [TALLAGH_W.cpp](#).

References [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), and [GetShop_name\(\)](#).

Referenced by [operator=\(\)](#).

```
00063 {
00064     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
    \nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
    GetShop_address() << "\nNo. Iphones : " << this->
    GetNumber_of_iphones() << "\nNo. Samsung : " << this->
    GetNumber_of_samsung() << "\nNo. Sony : " << this->
    GetNumber_of_sony() << "\nNo. Huawei : " << this->
    GetNumber_of_huawei() << "\nNo. Nokia : " << this->
    GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
    GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00065 }
```


Here is the call graph for this function:



Here is the caller graph for this function:



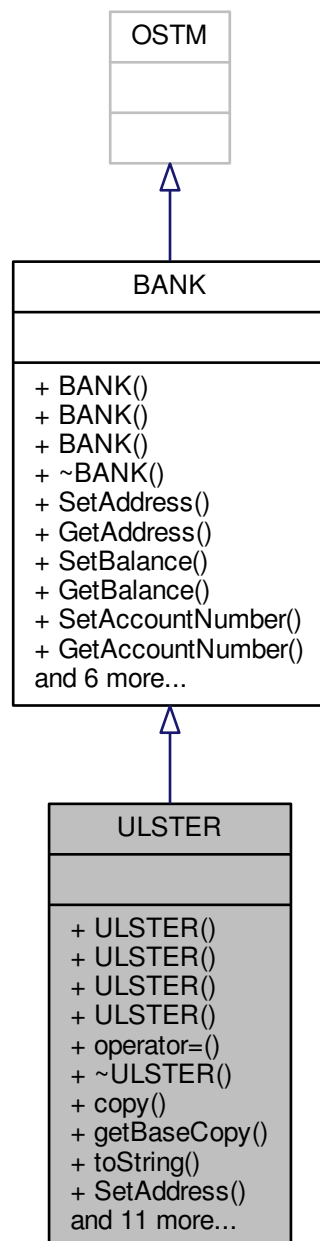
The documentation for this class was generated from the following files:

- [TALLAGH_W.h](#)
- [TALLAGH_W.cpp](#)

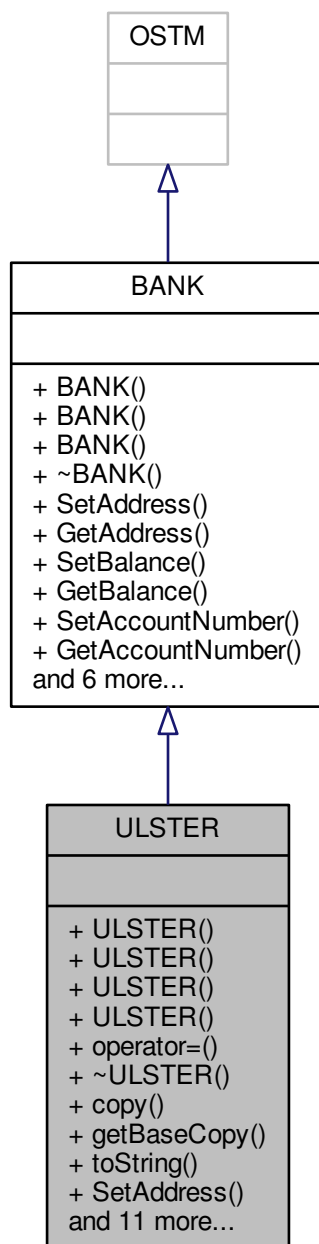
6.12 ULSTER Class Reference

```
#include <ULSTER.h>
```

Inheritance diagram for ULSTER:



Collaboration diagram for ULSTER:



Public Member Functions

- [ULSTER](#) ()
- [ULSTER](#) (int accountNumber, double balance, std::string firstName, std::string lastName, std::string address)
- [ULSTER](#) (std::shared_ptr< [BANK](#) > obj, int _version, int _unique_id)
- [ULSTER](#) (const [ULSTER](#) &orig)
- [ULSTER operator=](#) (const [ULSTER](#) &orig)

- virtual `~ULSTER ()`
- virtual void `copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)`
copy function, make deep copy of the object/pointer
- virtual `std::shared_ptr< OSTM > getBaseCopy (std::shared_ptr< OSTM > object)`
getBaseCopy function, make deep copy of the object/pointer and Return a new std::shared_ptr<BANK> type object
- virtual void `toString ()`
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void `SetAddress (std::string address)`
- virtual `std::string GetAddress () const`
- virtual void `SetBalance (double balance)`
- virtual `double GetBalance () const`
- virtual void `SetAccountNumber (int accountNumber)`
- virtual `int GetAccountNumber () const`
- virtual void `SetLastName (std::string lastName)`
- virtual `std::string GetLastName () const`
- virtual void `SetFirstName (std::string firstName)`
- virtual `std::string GetFirstName () const`
- virtual void `SetFullname (std::string fullname)`
- virtual `std::string GetFullname () const`

6.12.1 Detailed Description

Inherit from [BANK](#)

Definition at line 19 of file [ULSTER.h](#).

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `ULSTER::ULSTER ()` [`inline`]

Constructor

Definition at line 24 of file [ULSTER.h](#).

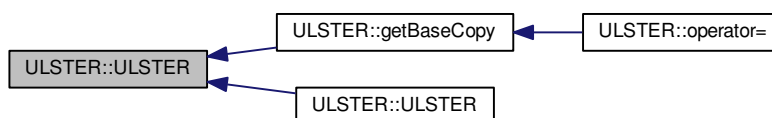
Referenced by [getBaseCopy\(\)](#), and [ULSTER\(\)](#).

```

00024         : BANK() {
00025             this->accountNumber = 0;
00026             this->balance = 50;
00027             this->firstName = "Joe";
00028             this->lastName = "Blog";
00029             this->address = "High street, Carlow";
00030             this->fullname = firstName + " " + lastName;
00031     };

```

Here is the caller graph for this function:



6.12.2.2 ULSTER::ULSTER (int *accountNumber*, double *balance*, std::string *firstName*, std::string *lastName*, std::string *address*) [inline]

Custom constructor

Definition at line 35 of file [ULSTER.h](#).

```

00035                                     :
00036     BANK() {
00037         this->accountNumber = accountNumber;
00037         this->balance = balance;
00038         this->firstName = firstName;
00039         this->lastName = lastName;
00040         this->address = address;
00041         this->fullname = firstName + " " + lastName;
00042     };

```

6.12.2.3 ULSTER::ULSTER (std::shared_ptr< **BANK**> *obj*, int *_version*, int *_unique_id*) [inline]

Custom constructor, used by the library for deep copying

Definition at line 46 of file [ULSTER.h](#).

References [ULSTER\(\)](#).

```

00046                                     : BANK(_version, _unique_id) {
00047
00048         this->accountNumber = obj->GetAccountNumber();
00049         this->balance = obj->GetBalance();
00050         this->firstName = obj->GetFirstName();
00051         this->lastName = obj->GetLastName();
00052         this->address = obj->GetAddress();
00053         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00054     };

```

Here is the call graph for this function:



6.12.2.4 ULSTER::ULSTER (const ULSTER & *orig*)

Copy constructor

Definition at line 15 of file [ULSTER.cpp](#).

```

00015     {
00016 }

```

6.12.2.5 ULSTER::~~ULSTER() [virtual]

de-constructor

Definition at line 18 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#).

```
00018         {
00019     }
```

Here is the caller graph for this function:



6.12.3 Member Function Documentation

6.12.3.1 void ULSTER::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>

Definition at line 37 of file [ULSTER.cpp](#).

References [SetAccountNumber\(\)](#).

Referenced by [operator=\(\)](#).

```
00037         {
00038
00039     std::shared_ptr<ULSTER> objTO = std::dynamic_pointer_cast<ULSTER>(to);
00040     std::shared_ptr<ULSTER> objFROM = std::dynamic_pointer_cast<ULSTER>(from);
00041     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00042     objTO->Set_Version(objFROM->Get_Version());
00043     objTO->SetAccountNumber(objFROM->GetAccountNumber());
00044     objTO->SetBalance(objFROM->GetBalance());
00045
00046
00047 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.2 int ULSTER::GetAccountNumber () const [virtual]

Implements [BANK](#).

Definition at line [83](#) of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00083                                     {  
00084     return accountNumber;  
00085 }
```

Here is the caller graph for this function:



6.12.3.3 `std::string ULSTER::GetAddress () const` [virtual]

Implements [BANK](#).

Definition at line 67 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#).

```
00067                                     {
00068     return address;
00069 }
```

Here is the caller graph for this function:



6.12.3.4 `double ULSTER::GetBalance () const` [virtual]

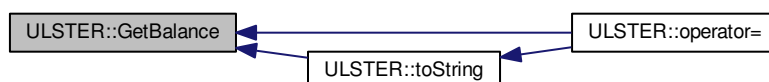
Implements [BANK](#).

Definition at line 75 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00075                                     {
00076     return balance;
00077 }
```

Here is the caller graph for this function:



6.12.3.5 `std::shared_ptr< OSTM > ULSTER::getBaseCopy (std::shared_ptr< OSTM > object)` [virtual]

`getBaseCopy` function, make deep copy of the object/pointer and Return a new `std::shared_ptr<BANK>` type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a <code>std::shared_ptr<BANK></code> return type

Definition at line 25 of file [ULSTER.cpp](#).

References [ULSTER\(\)](#).

Referenced by [operator=\(\)](#).

```

00026 {
00027     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00028     std::shared_ptr<BANK> obj(new ULSTER(objTO,object->Get_Version(),object->Get_Unique_ID()));
00029     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

00030     return ostm_obj;
00031 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.6 `std::string ULSTER::GetFirstName () const` [virtual]

Implements [BANK](#).

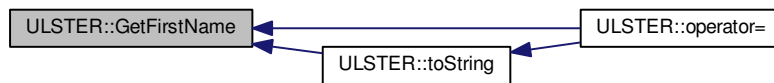
Definition at line 99 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00099                                     {
00100     return firstName;
00101 }
```

Here is the caller graph for this function:



6.12.3.7 `std::string ULSTER::GetFullName () const [virtual]`

Implements [BANK](#).

Definition at line 107 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#).

```

00107                                     {
00108     return fullname;
00109 }
  
```

Here is the caller graph for this function:



6.12.3.8 `std::string ULSTER::GetLastName () const [virtual]`

Implements [BANK](#).

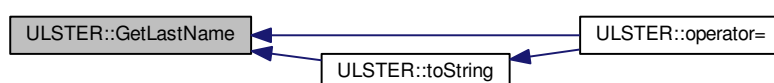
Definition at line 91 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00091                                     {
00092     return lastName;
00093 }
  
```

Here is the caller graph for this function:



6.12.3.9 ULSTER ULSTER::operator= (const ULSTER & orig) [inline]

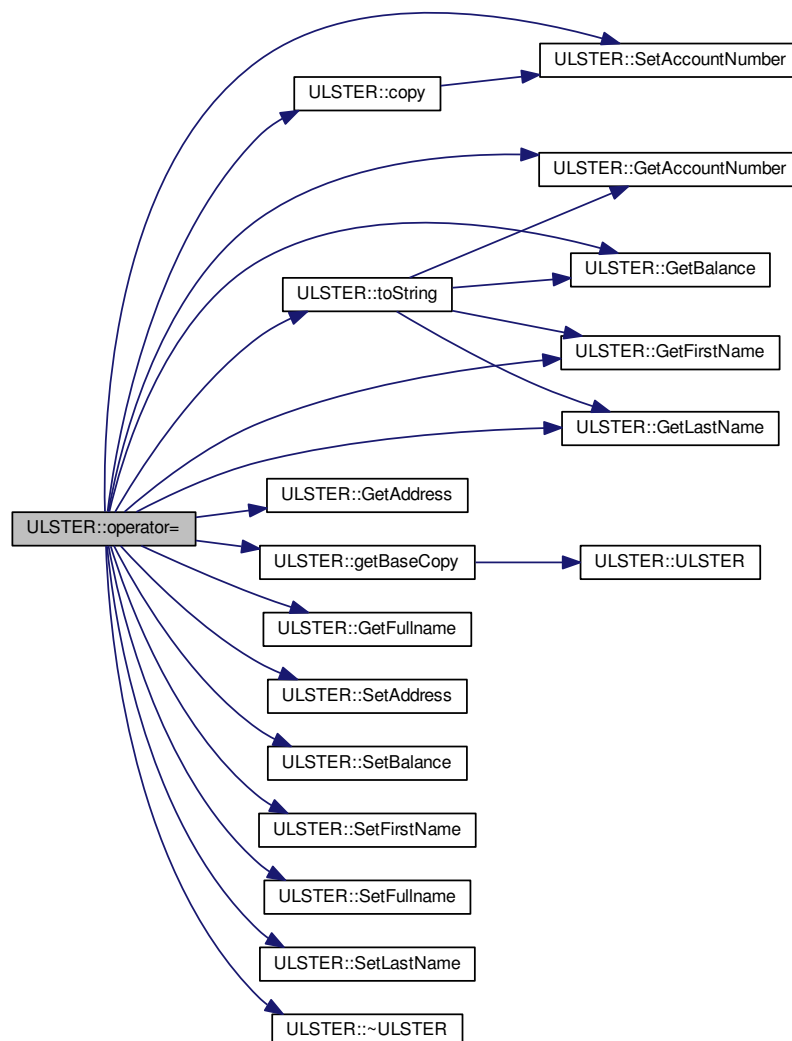
Operator

Definition at line 62 of file [ULSTER.h](#).

References [copy\(\)](#), [GetAccountNumber\(\)](#), [GetAddress\(\)](#), [GetBalance\(\)](#), [getBaseCopy\(\)](#), [GetFirstName\(\)](#), [GetFullname\(\)](#), [GetLastName\(\)](#), [SetAccountNumber\(\)](#), [SetAddress\(\)](#), [SetBalance\(\)](#), [SetFirstName\(\)](#), [SetFullname\(\)](#), [SetLastName\(\)](#), [toString\(\)](#), and [~ULSTER\(\)](#).

```
00062 {};
```

Here is the call graph for this function:



6.12.3.10 void ULSTER::SetAccountNumber (int *accountNumber*) [virtual]

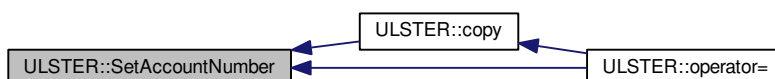
Implements [BANK](#).

Definition at line 79 of file [ULSTER.cpp](#).

Referenced by [copy\(\)](#), and [operator=\(\)](#).

```
00079                                     {  
00080     this->accountNumber = accountNumber;  
00081 }
```

Here is the caller graph for this function:



6.12.3.11 void ULSTER::SetAddress (std::string *address*) [virtual]

Implements [BANK](#).

Definition at line 63 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#).

```
00063                                     {  
00064     this->address = address;  
00065 }
```

Here is the caller graph for this function:



6.12.3.12 void ULSTER::SetBalance (double *balance*) [virtual]

Implements [BANK](#).

Definition at line 71 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#).

```
00071                                     {  
00072     this->balance = balance;  
00073 }
```

Here is the caller graph for this function:

**6.12.3.13** void ULSTER::SetFirstName (std::string *firstName*) [virtual]

Implements [BANK](#).

Definition at line 95 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#).

```
00095                                     {  
00096     this->firstName = firstName;  
00097 }
```

Here is the caller graph for this function:



6.12.3.14 void ULSTER::SetFullName (std::string *fullname*) [virtual]

Implements [BANK](#).

Definition at line 103 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#).

```
00103                                     {  
00104     this->fullname = fullname;  
00105 }
```

Here is the caller graph for this function:



6.12.3.15 void ULSTER::SetLastName (std::string *lastName*) [virtual]

Implements [BANK](#).

Definition at line 87 of file [ULSTER.cpp](#).

Referenced by [operator=\(\)](#).

```
00087                                     {  
00088     this->lastName = lastName;  
00089 }
```

Here is the caller graph for this function:



6.12.3.16 void ULSTER::toString() [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

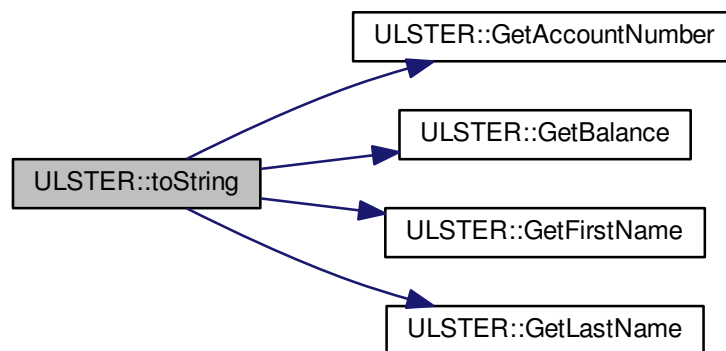
Definition at line 58 of file [ULSTER.cpp](#).

References [GetAccountNumber\(\)](#), [GetBalance\(\)](#), [GetFirstName\(\)](#), and [GetLastName\(\)](#).

Referenced by [operator=\(\)](#).

```
00059 {
00060     std::cout << "\nULSTER BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " <<
    this->GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
    " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
    number : " << this->Get_Version() << std::endl;
00061 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



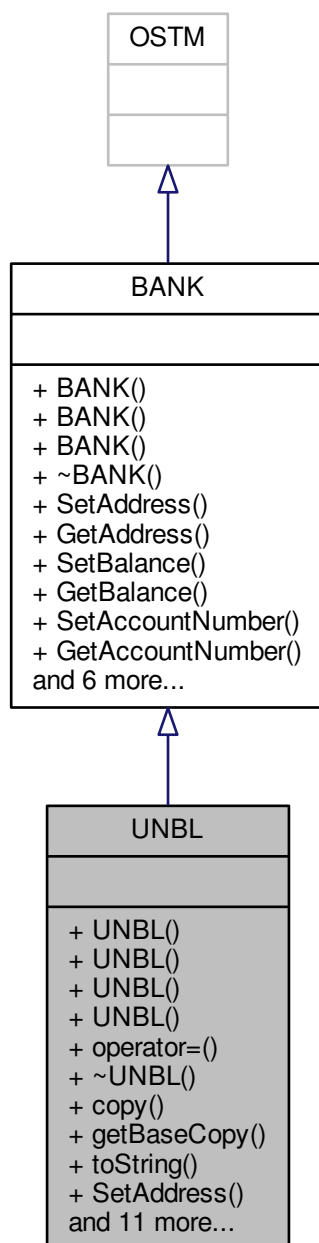
The documentation for this class was generated from the following files:

- [ULSTER.h](#)
- [ULSTER.cpp](#)

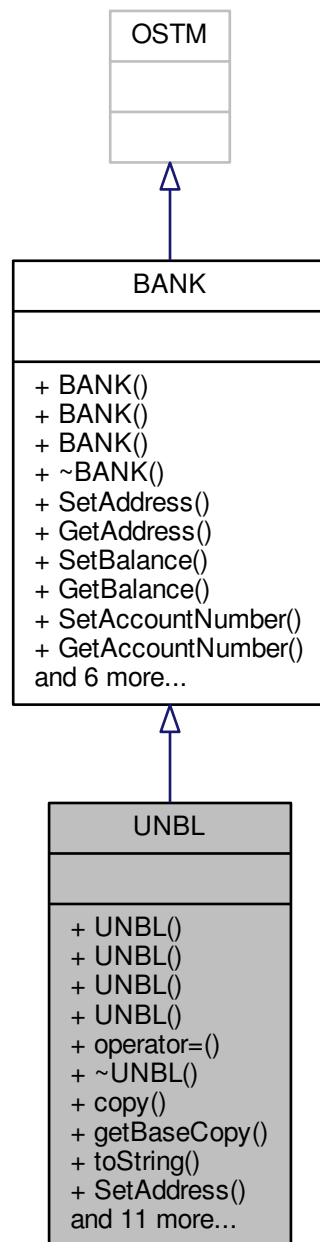
6.13 UNBL Class Reference

```
#include <UNBL.h>
```

Inheritance diagram for UNBL:



Collaboration diagram for UNBL:



Public Member Functions

- [UNBL](#) ()
- [UNBL](#) (int accountNumber, double balance, std::string firstName, std::string lastName, std::string address)
- [UNBL](#) (std::shared_ptr< [BANK](#) > obj, int _version, int _unique_id)
- [UNBL](#) (const [UNBL](#) &orig)
- [UNBL operator=](#) (const [UNBL](#) &orig)

- virtual `~UNBL ()`
- virtual void `copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from)`
copy function, make deep copy of the object/pointer
- virtual `std::shared_ptr< OSTM > getBaseCopy (std::shared_ptr< OSTM > object)`
getBaseCopy function, make deep copy of the object/pointer and Return a new std::shared_ptr< BANK> type object
- virtual void `toString ()`
_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type
- virtual void `SetAddress (std::string address)`
- virtual `std::string GetAddress () const`
- virtual void `SetBalance (double balance)`
- virtual `double GetBalance () const`
- virtual void `SetAccountNumber (int accountNumber)`
- virtual `int GetAccountNumber () const`
- virtual void `SetLastName (std::string lastName)`
- virtual `std::string GetLastName () const`
- virtual void `SetFirstName (std::string firstName)`
- virtual `std::string GetFirstName () const`
- virtual void `SetFullname (std::string fullname)`
- virtual `std::string GetFullname () const`

6.13.1 Detailed Description

Inherit from [BANK](#)

Definition at line 19 of file [UNBL.h](#).

6.13.2 Constructor & Destructor Documentation

6.13.2.1 `UNBL::UNBL () [inline]`

Constructor

Definition at line 24 of file [UNBL.h](#).

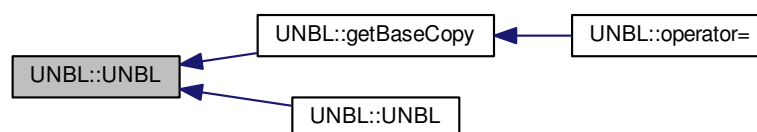
Referenced by [getBaseCopy\(\)](#), and [UNBL\(\)](#).

```

00024         : BANK() {
00025             this->accountNumber = 0;
00026             this->balance = 50;
00027             this->firstName = "Joe";
00028             this->lastName = "Blog";
00029             this->address = "High street, Carlow";
00030             this->fullname = firstName + " " + lastName;
00031     };

```

Here is the caller graph for this function:



6.13.2.2 UNBL::UNBL (int *accountNumber*, double *balance*, std::string *firstName*, std::string *lastName*, std::string *address*) [inline]

Custom constructor

Definition at line 35 of file [UNBL.h](#).

```
00035                                     :
00036     BANK() {
00037         this->accountNumber = accountNumber;
00037         this->balance = balance;
00038         this->firstName = firstName;
00039         this->lastName = lastName;
00040         this->address = address;
00041         this->fullname = firstName + " " + lastName;
00042     };
```

6.13.2.3 UNBL::UNBL (std::shared_ptr< BANK > *obj*, int *_version*, int *_unique_id*) [inline]

Custom constructor, used by the library for deep copying

Definition at line 46 of file [UNBL.h](#).

References [UNBL\(\)](#).

```
00046                                     : BANK(_version, _unique_id) {
00047
00048         this->accountNumber = obj->GetAccountNumber();
00049         this->balance = obj->GetBalance();
00050         this->firstName = obj->GetFirstName();
00051         this->lastName = obj->GetLastName();
00052         this->address = obj->GetAddress();
00053         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00054     };
```

Here is the call graph for this function:



6.13.2.4 UNBL::UNBL (const UNBL & *orig*)

Copy constructor

Definition at line 11 of file [UNBL.cpp](#).

```
00011     {
00012 }
```

6.13.2.5 UNBL::~UNBL () [virtual]

de-constructor

Definition at line 14 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#).

```
00014         {
00015     }
```

Here is the caller graph for this function:



6.13.3 Member Function Documentation

6.13.3.1 void UNBL::copy (std::shared_ptr< OSTM > to, std::shared_ptr< OSTM > from) [virtual]

copy function, make deep copy of the object/pointer

Parameters

<i>objTO</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>
<i>objFROM</i>	is a std::shared_ptr<BANK> type object casted back from std::shared_ptr<OSTM>

Definition at line 33 of file [UNBL.cpp](#).

References [SetAccountNumber\(\)](#).

Referenced by [operator=\(\)](#).

```
00033                                     {
00034
00035     std::shared_ptr<UNBL> objTO = std::dynamic_pointer_cast<UNBL>(to);
00036     std::shared_ptr<UNBL> objFROM = std::dynamic_pointer_cast<UNBL>(from);
00037     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00038     objTO->Set_Version(objFROM->Get_Version());
00039     objTO->SetAccountNumber(objFROM->GetAccountNumber());
00040     objTO->SetBalance(objFROM->GetBalance());
00041
00042 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.3.2 `int UNBL::GetAccountNumber () const [virtual]`

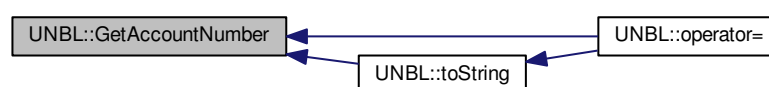
Implements [BANK](#).

Definition at line 78 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00078                                     {  
00079     return accountNumber;  
00080 }
```

Here is the caller graph for this function:



6.13.3.3 `std::string UNBL::GetAddress () const [virtual]`

Implements [BANK](#).

Definition at line 62 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#).

```
00062                                     {
00063     return address;
00064 }
```

Here is the caller graph for this function:



6.13.3.4 `double UNBL::GetBalance () const [virtual]`

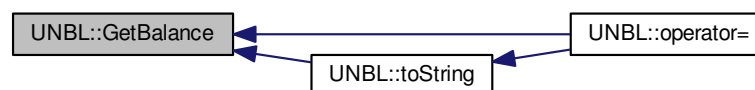
Implements [BANK](#).

Definition at line 70 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```
00070                                     {
00071     return balance;
00072 }
```

Here is the caller graph for this function:



6.13.3.5 `std::shared_ptr< OSTM > UNBL::getBaseCopy (std::shared_ptr< OSTM > object) [virtual]`

`getBaseCopy` function, make deep copy of the object/pointer and Return a new `std::shared_ptr<BANK>` type object

Parameters

<i>objTO</i>	is a BANK type pointer for casting
<i>obj</i>	is a <code>std::shared_ptr<BANK></code> return type

Definition at line 21 of file [UNBL.cpp](#).

References [UNBL\(\)](#).

Referenced by [operator=\(\)](#).

```

00022 {
00023     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00024     std::shared_ptr<BANK> obj(new UNBL(objTO,object->Get_Version(),object->Get_Unique_ID()));
00025     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

00026     return ostm_obj;
00027 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.3.6 `std::string UNBL::GetFirstName () const` `[virtual]`

Implements [BANK](#).

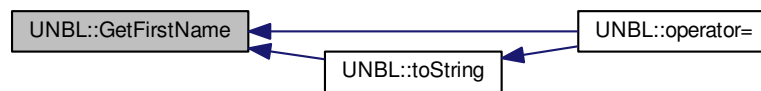
Definition at line 94 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00094     {
00095         return firstName;
00096     }
```

Here is the caller graph for this function:



6.13.3.7 `std::string UNBL::GetFullName () const [virtual]`

Implements [BANK](#).

Definition at line 102 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#).

```

00102                                     {
00103     return fullname;
00104 }
  
```

Here is the caller graph for this function:



6.13.3.8 `std::string UNBL::GetLastName () const [virtual]`

Implements [BANK](#).

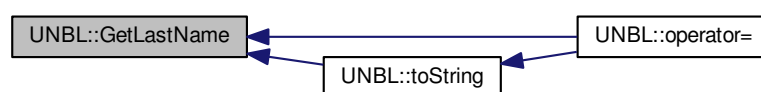
Definition at line 86 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#), and [toString\(\)](#).

```

00086                                     {
00087     return lastName;
00088 }
  
```

Here is the caller graph for this function:



6.13.3.9 UNBL UNBL::operator= (const UNBL & orig) [inline]

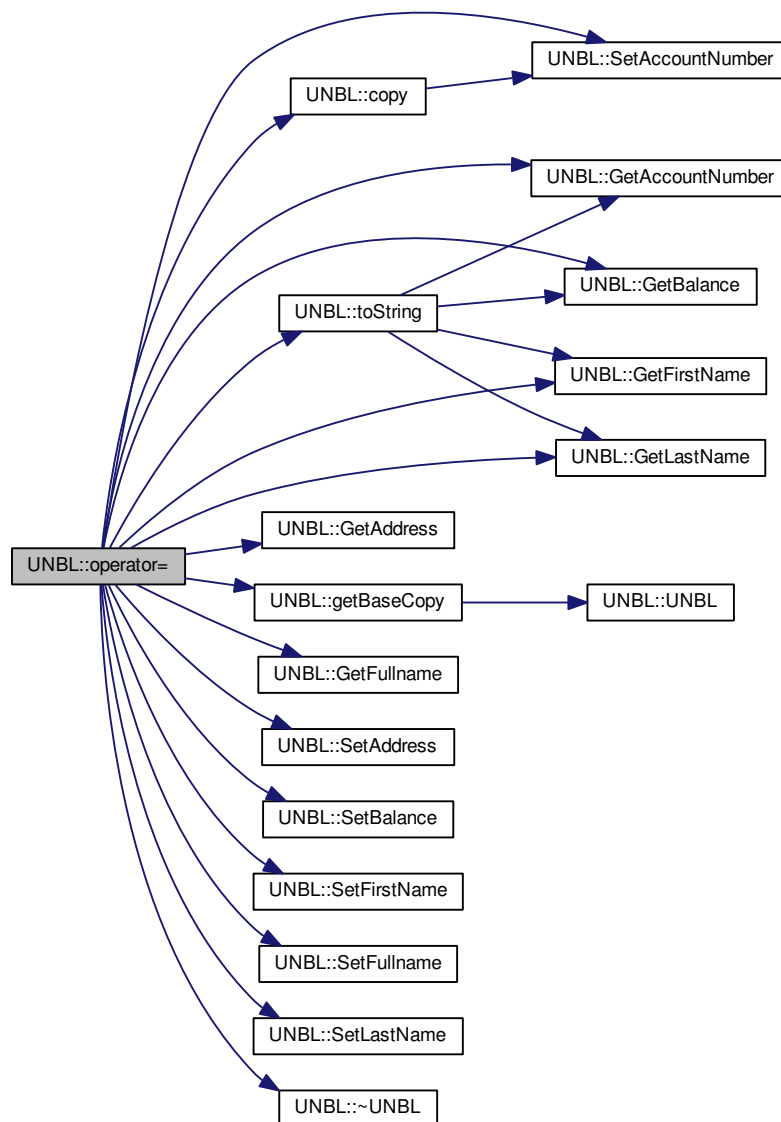
Operator

Definition at line 62 of file UNBL.h.

References [copy\(\)](#), [GetAccountNumber\(\)](#), [GetAddress\(\)](#), [GetBalance\(\)](#), [getBaseCopy\(\)](#), [GetFirstName\(\)](#), [GetFullname\(\)](#), [GetLastName\(\)](#), [SetAccountNumber\(\)](#), [SetAddress\(\)](#), [SetBalance\(\)](#), [SetFirstName\(\)](#), [SetFullname\(\)](#), [SetLastName\(\)](#), [toString\(\)](#), and [~UNBL\(\)](#).

```
00062 {};
```

Here is the call graph for this function:



6.13.3.10 void UNBL::SetAccountNumber (int *accountNumber*) [virtual]

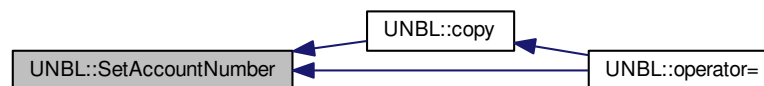
Implements [BANK](#).

Definition at line 74 of file [UNBL.cpp](#).

Referenced by [copy\(\)](#), and [operator=\(\)](#).

```
00074 {  
00075     this->accountNumber = accountNumber;  
00076 }
```

Here is the caller graph for this function:



6.13.3.11 void UNBL::SetAddress (std::string *address*) [virtual]

Implements [BANK](#).

Definition at line 58 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#).

```
00058 {  
00059     this->address = address;  
00060 }
```

Here is the caller graph for this function:



6.13.3.12 void UNBL::SetBalance (double *balance*) [virtual]

Implements [BANK](#).

Definition at line 66 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#).

```
00066                                     {  
00067     this->balance = balance;  
00068 }
```

Here is the caller graph for this function:



6.13.3.13 void UNBL::SetFirstName (std::string *firstName*) [virtual]

Implements [BANK](#).

Definition at line 90 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#).

```
00090                                     {  
00091     this->firstName = firstName;  
00092 }
```

Here is the caller graph for this function:



6.13.3.14 void UNBL::SetFullName (std::string *fullname*) [virtual]

Implements [BANK](#).

Definition at line 98 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#).

```
00098                                     {  
00099     this->fullname = fullname;  
00100 }
```

Here is the caller graph for this function:



6.13.3.15 void UNBL::SetLastName (std::string *lastName*) [virtual]

Implements [BANK](#).

Definition at line 82 of file [UNBL.cpp](#).

Referenced by [operator=\(\)](#).

```
00082                                     {  
00083     this->lastName = lastName;  
00084 }
```

Here is the caller graph for this function:



6.13.3.16 void UNBL::toString() [virtual]

_cast, is use to cast bak the std::shared_ptr<OSTM> to the required type

toString function, displays the object values in formatted way

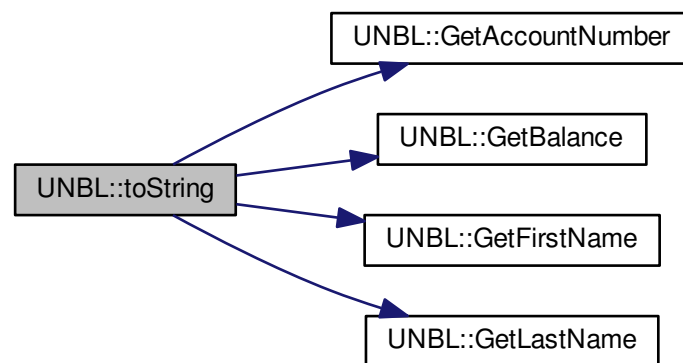
Definition at line 53 of file UNBL.cpp.

References [GetAccountNumber\(\)](#), [GetBalance\(\)](#), [GetFirstName\(\)](#), and [GetLastName\(\)](#).

Referenced by [operator=\(\)](#).

```
00054 {
00055     std::cout << "\nUNBL BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " << this->
    GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
    " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
    number : " << this->Get_Version() << std::endl;
00056 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



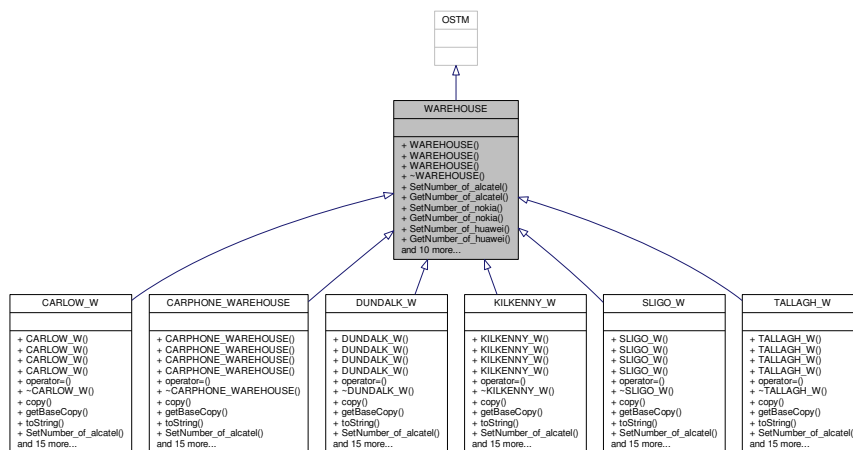
The documentation for this class was generated from the following files:

- [UNBL.h](#)
- [UNBL.cpp](#)

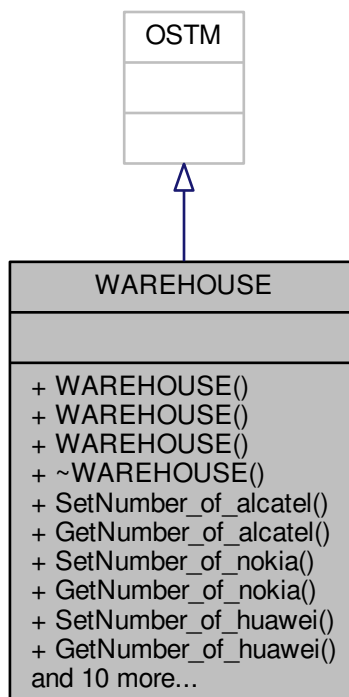
6.14 WAREHOUSE Class Reference

```
#include <WAREHOUSE.h>
```

Inheritance diagram for WAREHOUSE:



Collaboration diagram for WAREHOUSE:



Public Member Functions

- [WAREHOUSE](#) ()
- [WAREHOUSE](#) (int _version, int _unique_id)
- [WAREHOUSE](#) (const [WAREHOUSE](#) &orig)
- virtual [~WAREHOUSE](#) ()
- virtual void [SetNumber_of_alcatel](#) (int _number_of_alcatel)=0
- virtual int [GetNumber_of_alcatel](#) ()=0
- virtual void [SetNumber_of_nokia](#) (int _number_of_nokia)=0
- virtual int [GetNumber_of_nokia](#) ()=0
- virtual void [SetNumber_of_huawei](#) (int _number_of_huawei)=0
- virtual int [GetNumber_of_huawei](#) ()=0
- virtual void [SetNumber_of_sony](#) (int _number_of_sony)=0
- virtual int [GetNumber_of_sony](#) ()=0
- virtual void [SetNumber_of_samsung](#) (int _number_of_samsung)=0
- virtual int [GetNumber_of_samsung](#) ()=0
- virtual void [SetNumber_of_iphones](#) (int _number_of_iphones)=0
- virtual int [GetNumber_of_iphones](#) ()=0
- virtual void [SetShop_name](#) (std::string _shop_name)=0
- virtual std::string [GetShop_name](#) ()=0
- virtual void [SetShop_address](#) (std::string _shop_address)=0
- virtual std::string [GetShop_address](#) ()=0

6.14.1 Detailed Description

[WAREHOUSE](#) inherit from OSTM library

Definition at line 16 of file [WAREHOUSE.h](#).

6.14.2 Constructor & Destructor Documentation

6.14.2.1 [WAREHOUSE::WAREHOUSE](#) () `[inline]`

Constructor

Definition at line 21 of file [WAREHOUSE.h](#).

Referenced by [WAREHOUSE\(\)](#).

```
00021         :OSTM() {
00022
00023     };
```

Here is the caller graph for this function:



6.14.2.2 WAREHOUSE::WAREHOUSE (int _version, int _unique_id) [inline]

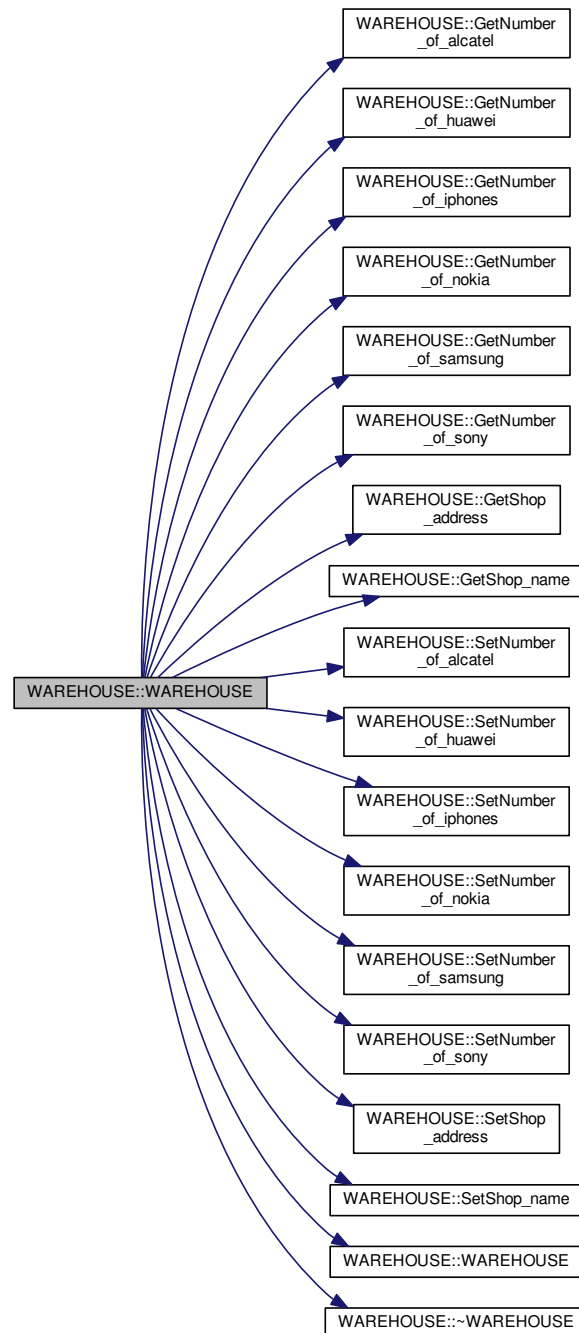
Custom Constructor

Definition at line 27 of file [WAREHOUSE.h](#).

References [GetNumber_of_alcatel\(\)](#), [GetNumber_of_huawei\(\)](#), [GetNumber_of_iphones\(\)](#), [GetNumber_of_nokia\(\)](#), [GetNumber_of_samsung\(\)](#), [GetNumber_of_sony\(\)](#), [GetShop_address\(\)](#), [GetShop_name\(\)](#), [SetNumber_of_alcatel\(\)](#), [SetNumber_of_huawei\(\)](#), [SetNumber_of_iphones\(\)](#), [SetNumber_of_nokia\(\)](#), [SetNumber_of_samsung\(\)](#), [SetNumber_of_sony\(\)](#), [SetShop_address\(\)](#), [SetShop_name\(\)](#), [WAREHOUSE\(\)](#), and [~WAREHOUSE\(\)](#).

```
00027                                     : OSTM(_version, _unique_id) {
00028
00029     };
```


Here is the call graph for this function:



6.14.2.3 WAREHOUSE::WAREHOUSE (const WAREHOUSE & orig)

Copy constructor

Definition at line 12 of file [WAREHOUSE.cpp](#).

```

00012         {
00013     }

```

6.14.2.4 WAREHOUSE::~~WAREHOUSE() [virtual]

de-constructor

Definition at line 15 of file [WAREHOUSE.cpp](#).

Referenced by [WAREHOUSE\(\)](#).

```
00015             {
00016 }
```

Here is the caller graph for this function:



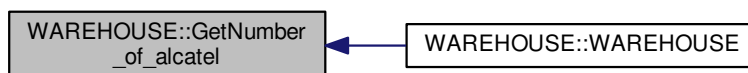
6.14.3 Member Function Documentation

6.14.3.1 virtual int WAREHOUSE::GetNumber_of_alcatel() [pure virtual]

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

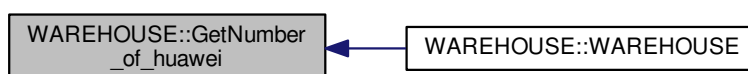


6.14.3.2 virtual int WAREHOUSE::GetNumber_of_huawei() [pure virtual]

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

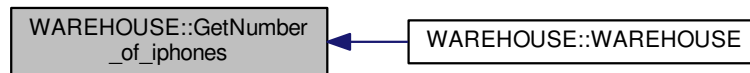


6.14.3.3 `virtual int WAREHOUSE::GetNumber_of_iphones () [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

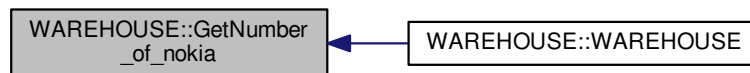


6.14.3.4 `virtual int WAREHOUSE::GetNumber_of_nokia () [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

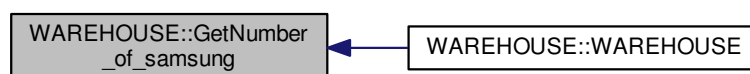


6.14.3.5 `virtual int WAREHOUSE::GetNumber_of_samsung () [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

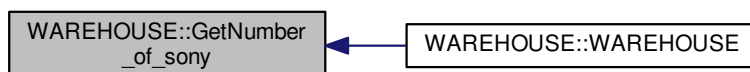


6.14.3.6 `virtual int WAREHOUSE::GetNumber_of_sony () [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

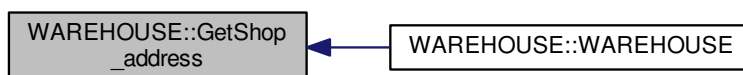


6.14.3.7 `virtual std::string WAREHOUSE::GetShop_address () [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:



6.14.3.8 `virtual std::string WAREHOUSE::GetShop_name () [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

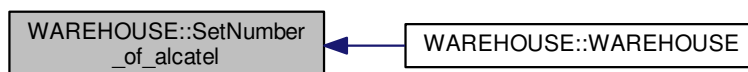


6.14.3.9 `virtual void WAREHOUSE::SetNumber_of_alcatel (int _number_of_alcatel) [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

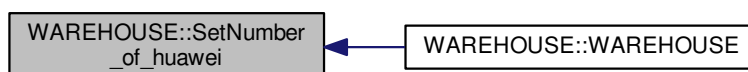


6.14.3.10 `virtual void WAREHOUSE::SetNumber_of_huawei (int _number_of_huawei) [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

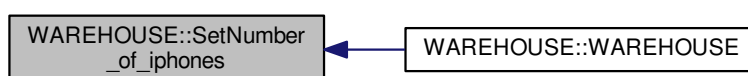


6.14.3.11 `virtual void WAREHOUSE::SetNumber_of_iphones (int _number_of_iphones) [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

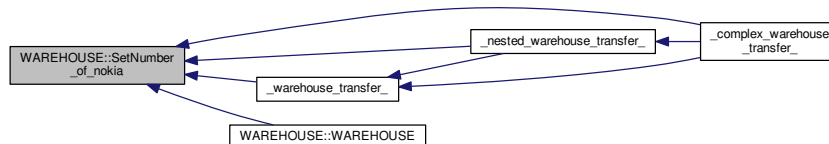


6.14.3.12 `virtual void WAREHOUSE::SetNumber_of_nokia (int _number_of_nokia) [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [_complex_warehouse_transfer_\(\)](#), [_nested_warehouse_transfer_\(\)](#), [_warehouse_transfer_\(\)](#), and [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

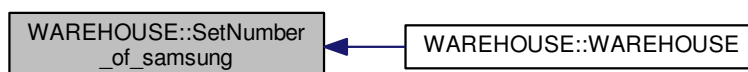


6.14.3.13 `virtual void WAREHOUSE::SetNumber_of_samsung (int _number_of_samsung) [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

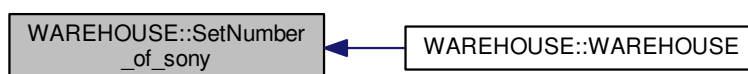


6.14.3.14 `virtual void WAREHOUSE::SetNumber_of_sony (int _number_of_sony) [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:

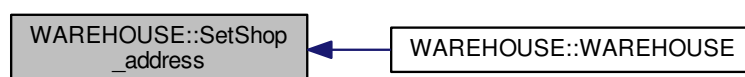


6.14.3.15 `virtual void WAREHOUSE::SetShop_address (std::string_shop_address) [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:



6.14.3.16 `virtual void WAREHOUSE::SetShop_name (std::string_shop_name) [pure virtual]`

Implemented in [CARPHONE_WAREHOUSE](#), [SLIGO_W](#), [TALLAGH_W](#), [CARLOW_W](#), [DUNDALK_W](#), and [KILKENNY_W](#).

Referenced by [WAREHOUSE\(\)](#).

Here is the caller graph for this function:



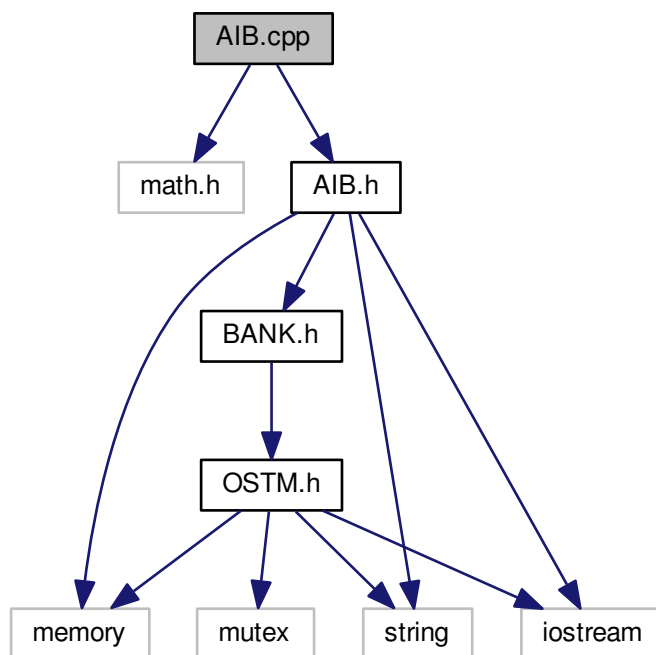
The documentation for this class was generated from the following files:

- [WAREHOUSE.h](#)
- [WAREHOUSE.cpp](#)

7 File Documentation

7.1 AIB.cpp File Reference

```
#include <math.h>
#include "AIB.h"
Include dependency graph for AIB.cpp:
```



7.2 AIB.cpp

```
00001 /*
00002  * File:   AIB.cpp
00003  * Author: Zoltan Fuzesi
00004  * IT Carlow : C00197361
00005  *
00006  * Created on January 17, 2018, 8:02 PM
00007  */
00008
00009 #include <math.h>
00010
00011 #include "AIB.h"
00012
00013
00014 AIB::AIB(const AIB& orig) {
00015 }
00016
00017 AIB::~AIB() {
00018 }
00019
00024 std::shared_ptr<OSTM> AIB::getBaseCopy(std::shared_ptr<OSTM> object)
00025 {
00026
00027     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00028     std::shared_ptr<BANK> obj(new AIB(objTO, object->Get_Version(),object->Get_Unique_ID()));
00029     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00030     return ostm_obj;
00031 }
00037 void AIB::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00038
```



```

00039     std::shared_ptr<AIB> objTO = std::dynamic_pointer_cast<AIB>(to);
00040     std::shared_ptr<AIB> objFROM = std::dynamic_pointer_cast<AIB>(from);
00041     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00042     objTO->Set_Version(objFROM->Get_Version());
00043     objTO->SetAccountNumber(objFROM->GetAccountNumber());
00044     objTO->SetBalance(objFROM->GetBalance());
00045 }
00049 //std::shared_ptr<AIB> AIB::_cast(std::shared_ptr<OSTM> _object){
00050 //
00051 //     return std::static_pointer_cast<AIB>(_object);
00052 //}
00056 void AIB::toString()
00057 {
00058     std::cout << "\nAIB BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " << this->
GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
    " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
    number : " << this->Get_Version() << std::endl;
00059 }
00060
00061 void AIB::SetAddress(std::string address) {
00062     this->address = address;
00063 }
00064
00065 std::string AIB::GetAddress() const {
00066     return address;
00067 }
00068
00069 void AIB::SetBalance(double balance) {
00070     this->balance = balance;
00071 }
00072
00073 double AIB::GetBalance() const {
00074     return balance;
00075 }
00076
00077 void AIB::SetAccountNumber(int accountNumber) {
00078     this->accountNumber = accountNumber;
00079 }
00080
00081 int AIB::GetAccountNumber() const {
00082     return accountNumber;
00083 }
00084
00085 void AIB::SetLastName(std::string lastName) {
00086     this->lastName = lastName;
00087 }
00088
00089 std::string AIB::GetLastName() const {
00090     return lastName;
00091 }
00092
00093 void AIB::SetFirstName(std::string firstName) {
00094     this->firstName = firstName;
00095 }
00096
00097 std::string AIB::GetFirstName() const {
00098     return firstName;
00099 }
00100
00101 void AIB::SetFullname(std::string fullname) {
00102     this->fullname = fullname;
00103 }
00104
00105 std::string AIB::GetFullname() const {
00106     return fullname;
00107 }

```

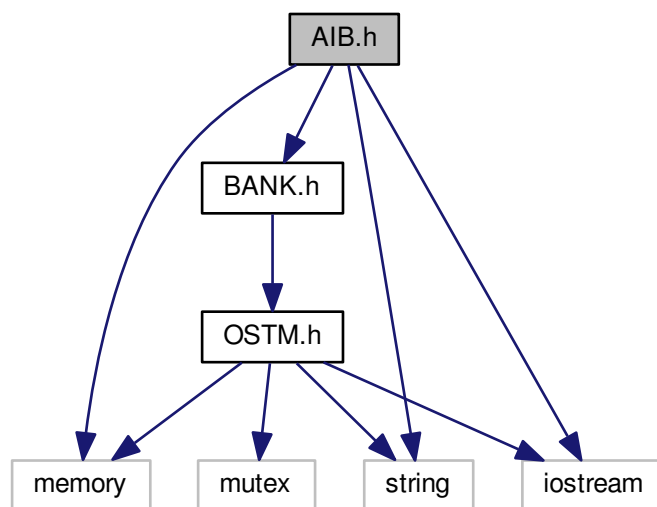
7.3 AIB.h File Reference

```

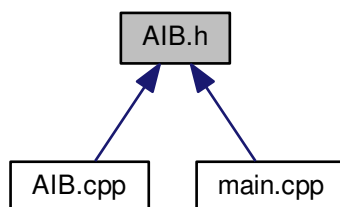
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for AIB.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [AIB](#)

7.4 AIB.h

```

00001 /*
00002  * File:   AIB.h
00003  * Author: Zoltan Fuzesi
00004  * IT Carlow : C00197361
00005  *
00006  * Created on January 17, 2018, 8:02 PM
00007  */
  
```

```

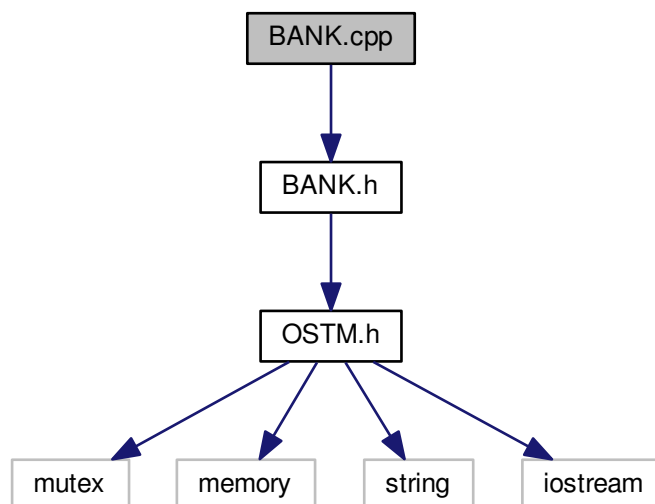
00008
00009 #ifndef AIB_H
00010 #define AIB_H
00011 #include "BANK.h"
00012 #include <string>
00013 #include <memory>
00014 #include <iostream>
00018 class AIB : public BANK {
00019 public:
00023     AIB(): BANK()
00024     {
00025         this->accountNumber = 0;
00026         this->balance = 50;
00027         this->firstName = "Joe";
00028         this->lastName = "Blog";
00029         this->address = "High street, Carlow";
00030         this->fullname = firstName + " " + lastName;
00031
00032     };
00036     AIB(int accountNumber, double balance, std::string firstName, std::string lastName, std::string
address): BANK(
00037     {
00038         this->accountNumber = accountNumber;
00039         this->balance = balance;
00040         this->firstName = firstName;
00041         this->lastName = lastName;
00042         this->address = address;
00043         this->fullname = firstName + " " + lastName;
00044     };
00048     AIB(std::shared_ptr<BANK> obj, int _version, int _unique_id): BANK(_version, _unique_id)
00049     {
00050
00051         this->accountNumber = obj->GetAccountNumber();
00052         this->balance = obj->GetBalance();
00053         this->firstName = obj->GetFirstName();
00054         this->lastName = obj->GetLastName();
00055         this->address = obj->GetAddress();
00056         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00057
00058     };
00062     AIB(const AIB& orig);
00066     AIB operator=(const AIB& orig){};
00070     virtual ~AIB();
00071
00072     /*
00073     * Implement OSTM virtual methods
00074     */
00075     // virtual std::shared_ptr<AIB> _cast(std::shared_ptr<OSTM> _object);
00076     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00077     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00078     virtual void toString();
00079
00080     /*
00081     * Implement BANK virtual methods
00082     */
00083     virtual void SetAddress(std::string address);
00084     virtual std::string GetAddress() const;
00085     virtual void SetBalance(double balance);
00086     virtual double GetBalance() const;
00087     virtual void SetAccountNumber(int accountNumber);
00088     virtual int GetAccountNumber() const;
00089     virtual void SetLastName(std::string lastName);
00090     virtual std::string GetLastName() const;
00091     virtual void SetFirstName(std::string firstName);
00092     virtual std::string GetFirstName() const;
00093     virtual void SetFullname(std::string fullname);
00094     virtual std::string GetFullname() const;
00095
00096 private:
00097     std::string fullname;
00098     std::string firstName;
00099     std::string lastName;
00100     int accountNumber;
00101     double balance;
00102     std::string address;
00103
00104
00105 };
00106
00107 #endif /* AIB_H */

```

7.5 BANK.cpp File Reference

```
#include "BANK.h"
```

Include dependency graph for BANK.cpp:



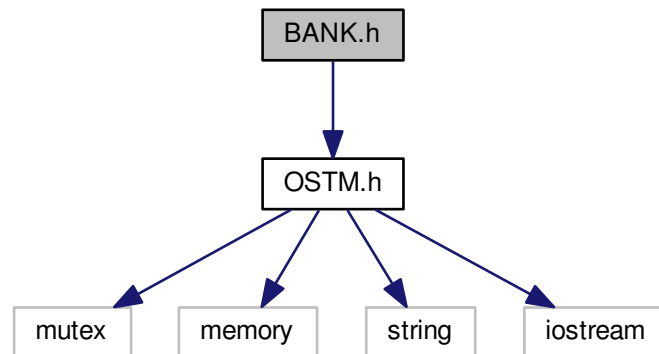
7.6 BANK.cpp

```
00001 /*
00002  * File:   BANK.cpp
00003  * Author: Zoltan Fuzesi
00004  * IT Carlow : C00197361
00005  *
00006  * Created on January 17, 2018, 8:02 PM
00007  */
00008
00009 #include "BANK.h"
00010
00011 BANK::BANK(const BANK& orig) {
00012 }
00013
00014 BANK::~BANK() {
00015 }
00016
```

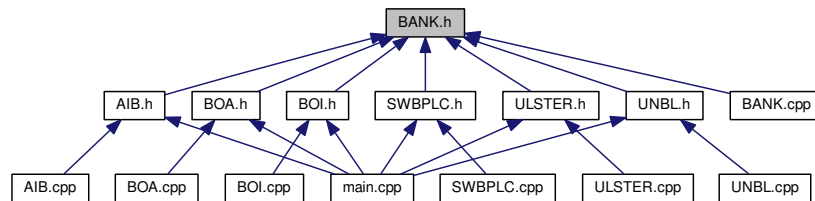
7.7 BANK.h File Reference

```
#include "OSTM.h"
```

Include dependency graph for BANK.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [BANK](#)

7.8 BANK.h

```

00001 /*
00002  * File:   BANK.h
00003  * Author: Zoltan Fuzesi
00004  * IT Carlow : C00197361
00005  *
00006  * Created on January 17, 2018, 8:02 PM
00007  */
00008
00009 #ifndef BANK_H
00010 #define BANK_H
00011 #include "OSTM.h"
00016 class BANK : public OSTM {
00017
00018
00019 public:
00023     BANK(): OSTM() {
00024
00025     };
00029     BANK(int _version, int _unique_id) : OSTM(_version, _unique_id){
  
```

```

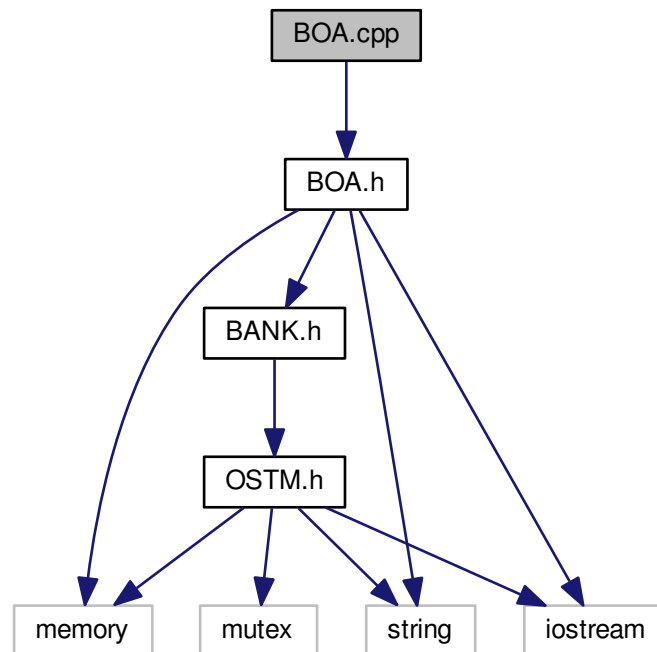
00030
00031     };
00032     BANK(const BANK& orig);
00033     virtual ~BANK();
00034
00035     /*
00036      * Bank specific virtual functions
00037      */
00038     virtual void SetAddress(std::string address) = 0;
00039     virtual std::string GetAddress() const = 0;
00040     virtual void SetBalance(double balance) = 0;
00041     virtual double GetBalance() const = 0;
00042     virtual void SetAccountNumber(int accountNumber) = 0;
00043     virtual int GetAccountNumber() const = 0;
00044     virtual void SetLastName(std::string lastName) = 0;
00045     virtual std::string GetLastName() const = 0;
00046     virtual void SetFirstName(std::string firstName) = 0;
00047     virtual std::string GetFirstName() const = 0;
00048     virtual void SetFullname(std::string fullname) = 0;
00049     virtual std::string GetFullname() const = 0;
00050 private:
00051 };
00052 #endif /* BANK_H */
00053

```

7.9 BOA.cpp File Reference

```
#include "BOA.h"
```

Include dependency graph for BOA.cpp:



7.10 BOA.cpp

```
00001 /*
```

```

00002  * File:    BOA.cpp
00003  * Author:  Zoltan Fuzesi
00004  * IT Carlow : C00197361
00005  *
00006  * Created on January 17, 2018, 8:02 PM
00007  */
00008
00009 #include "BOA.h"
00010
00011
00012 BOA::BOA(const BOA& orig) {
00013 }
00014
00015 BOA::~BOA() {
00016 }
00022 std::shared_ptr<OSTM> BOA::getBaseCopy(std::shared_ptr<OSTM> object)
00023 {
00024     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00025     std::shared_ptr<BANK> obj(new BOA(objTO, object->Get_Version(), object->Get_Unique_ID()));
00026     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00027     return ostm_obj;
00028 }
00034 void BOA::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00035
00036     std::shared_ptr<BOA> objTO = std::dynamic_pointer_cast<BOA>(to);
00037     std::shared_ptr<BOA> objFROM = std::dynamic_pointer_cast<BOA>(from);
00038     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00039     objTO->Set_Version(objFROM->Get_Version());
00040     objTO->SetAccountNumber(objFROM->GetAccountNumber());
00041     objTO->SetBalance(objFROM->GetBalance());
00042 }
00043
00047 //std::shared_ptr<BOA> BOA::_cast(std::shared_ptr<OSTM> _object){
00048 //
00049 //     return std::static_pointer_cast<BOA>(_object);
00050 //}
00054 void BOA::toString()
00055 {
00056     // std::cout << "\nUnique ID : " << this->GetUniqueID() << "\nInt value : " << this->GetV_int() <<
00057     "\nDouble value : " << this->GetV_double() << "\nFloat value : " << this->GetV_float() << "\nString value : " <<
00058     this->GetV_string() << "\nVersion number : " << this->Get_Version() << "\nLoad Counter : " <<
00059     this->GetLoadCounter() << "\nWrite Counter : " << this->GetWriteCounter() << std::endl;
00060     std::cout << "\nBOA BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " << this->
00061     GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
00062     " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
00063     number : " << this->Get_Version() << std::endl;
00064 }
00065
00066 void BOA::SetAddress(std::string address) {
00067     this->address = address;
00068 }
00069
00070 std::string BOA::GetAddress() const {
00071     return address;
00072 }
00073
00074 void BOA::SetBalance(double balance) {
00075     this->balance = balance;
00076 }
00077
00078 double BOA::GetBalance() const {
00079     return balance;
00080 }
00081
00082 void BOA::SetAccountNumber(int accountNumber) {
00083     this->accountNumber = accountNumber;
00084 }
00085
00086 int BOA::GetAccountNumber() const {
00087     return accountNumber;
00088 }
00089
00090 void BOA::SetLastName(std::string lastName) {
00091     this->lastName = lastName;
00092 }
00093
00094 std::string BOA::GetLastName() const {
00095     return lastName;
00096 }
00097
00098 void BOA::SetFirstName(std::string firstName) {
00099     this->firstName = firstName;
00100 }
00101
00102 std::string BOA::GetFirstName() const {
00103     return firstName;
00104 }

```

```

00099
00100 void BOA::SetFullname(std::string fullname) {
00101     this->fullname = fullname;
00102 }
00103
00104 std::string BOA::GetFullname() const {
00105     return fullname;
00106 }
00107

```

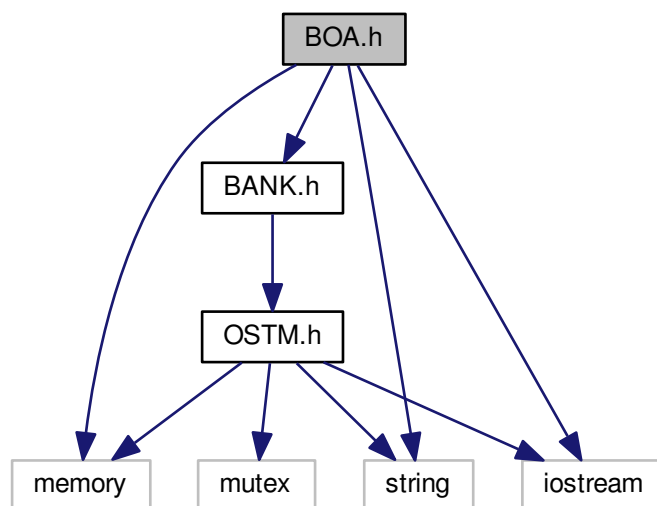
7.11 BOA.h File Reference

```

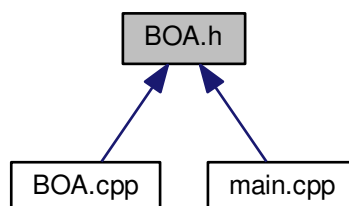
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for BOA.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BOA

7.12 BOA.h

```

00001 /*
00002  * File:   BOA.h
00003  * Author: Zoltan Fuzesi
00004  * IT Carlow : C00197361
00005  *
00006  * Created on January 17, 2018, 8:02 PM
00007  */
00008
00009 #ifndef BOA_H
00010 #define BOA_H
00011 #include "BANK.h"
00012 #include <string>
00013 #include <memory>
00014 #include <iostream>
00018 class BOA : public BANK {
00019 public:
00020
00024     BOA() : BANK() {
00025         this->accountNumber = 0;
00026         this->balance = 50;
00027         this->firstName = "Joe";
00028         this->lastName = "Blog";
00029         this->address = "High street, Carlow";
00030         this->fullname = firstName + " " + lastName;
00031     };
00035     BOA(int accountNumber, double balance, std::string firstName, std::string lastName, std::string
address) : BANK() {
00036         this->accountNumber = accountNumber;
00037         this->balance = balance;
00038         this->firstName = firstName;
00039         this->lastName = lastName;
00040         this->address = address;
00041         this->fullname = firstName + " " + lastName;
00042     };
00046     BOA(std::shared_ptr<BANK> obj, int _version, int _unique_id) : BANK(_version, _unique_id) {
00047
00048         this->accountNumber = obj->GetAccountNumber();
00049         this->balance = obj->GetBalance();
00050         this->firstName = obj->GetFirstName();
00051         this->lastName = obj->GetLastName();
00052         this->address = obj->GetAddress();
00053         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00054     };
00055
00056
00060     BOA(const BOA& orig);
00064     BOA operator=(const BOA& orig) {
00065     };
00069     virtual ~BOA();
00070
00071     /*
00072     * Implement OSTM virtual methods
00073     */
00074     //virtual std::shared_ptr<BOA> _cast(std::shared_ptr<OSTM> _object);
00075     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00076     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00077     virtual void toString();
00078
00079     /*
00080     * Implement BANK virtual methods
00081     */
00082     virtual void SetAddress(std::string address);
00083     virtual std::string GetAddress() const;
00084     virtual void SetBalance(double balance);
00085     virtual double GetBalance() const;
00086     virtual void SetAccountNumber(int accountNumber);
00087     virtual int GetAccountNumber() const;
00088     virtual void SetLastName(std::string lastName);
00089     virtual std::string GetLastName() const;
00090     virtual void SetFirstName(std::string firstName);
00091     virtual std::string GetFirstName() const;
00092     virtual void SetFullname(std::string fullname);
00093     virtual std::string GetFullname() const;
00094 private:
00095     std::string fullname;
00096     std::string firstName;

```

```

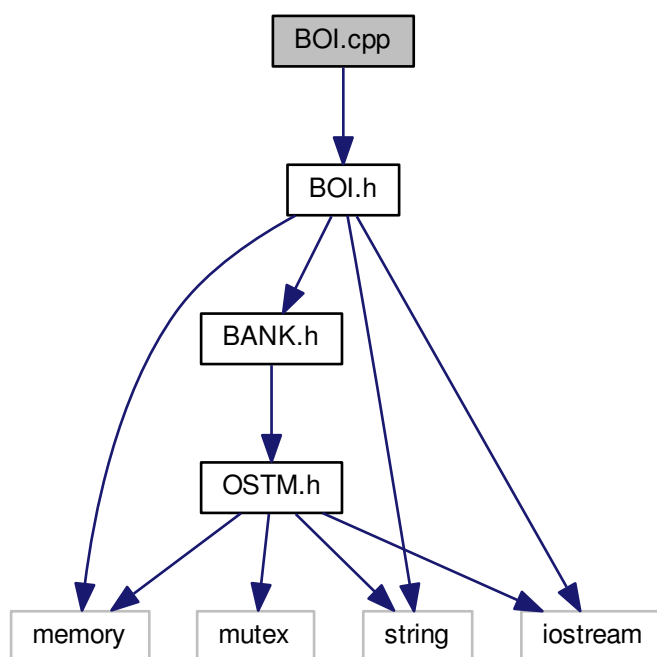
00097     std::string lastName;
00098     int accountNumber;
00099     double balance;
00100     std::string address;
00101
00102 };
00103
00104 #endif /* BOA_H */
00105

```

7.13 BOI.cpp File Reference

```
#include "BOI.h"
```

Include dependency graph for BOI.cpp:



7.14 BOI.cpp

```

00001
00002 /*
00003  * File:   BOI.cpp
00004  * Author: Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #include "BOI.h"
00011
00012 BOI::~BOI() {
00013 }
00014
00015 BOI::BOI(const BOI& orig) {
00016 }
00022 std::shared_ptr<OSTM> BOI::getBaseCopy(std::shared_ptr<OSTM> object)

```

```

00023 {
00024
00025     std::shared_ptr<BOI> objTO = std::dynamic_pointer_cast<BOI>(object);
00026     std::shared_ptr<BOI> obj(new BOI(objTO,object->Get_Version(),object->Get_Unique_ID()));
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00028     return ostm_obj;
00029 }
00035 void BOI::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00036
00037     std::shared_ptr<BOI> objTO = std::dynamic_pointer_cast<BOI>(to);
00038     std::shared_ptr<BOI> objFROM = std::dynamic_pointer_cast<BOI>(from);
00039     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00040     objTO->Set_Version(objFROM->Get_Version());
00041     objTO->SetAccountNumber(objFROM->GetAccountNumber());
00042     objTO->SetBalance(objFROM->GetBalance());
00043 }
00047 //std::shared_ptr<BOI> BOI::_cast(std::shared_ptr<OSTM> _object){
00048 //
00049 //     return std::static_pointer_cast<BOI>(_object);
00050 //}
00054 void BOI::toString()
00055 {
00056     std::cout << "\nBOI BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " << this->
GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
" << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
number : " << this->Get_Version() << std::endl;
00057 }
00058 void BOI::SetAddress(std::string address) {
00059     this->address = address;
00060 }
00061
00062 std::string BOI::GetAddress() const {
00063     return address;
00064 }
00065
00066 void BOI::SetBalance(double balance) {
00067     this->balance = balance;
00068 }
00069
00070 double BOI::GetBalance() const {
00071     return balance;
00072 }
00073
00074 void BOI::SetAccountNumber(int accountNumber) {
00075     this->accountNumber = accountNumber;
00076 }
00077
00078 int BOI::GetAccountNumber() const {
00079     return accountNumber;
00080 }
00081
00082 void BOI::SetLastName(std::string lastName) {
00083     this->lastName = lastName;
00084 }
00085
00086 std::string BOI::GetLastName() const {
00087     return lastName;
00088 }
00089
00090 void BOI::SetFirstName(std::string firstName) {
00091     this->firstName = firstName;
00092 }
00093
00094 std::string BOI::GetFirstName() const {
00095     return firstName;
00096 }
00097
00098 void BOI::SetFullname(std::string fullname) {
00099     this->fullname = fullname;
00100 }
00101
00102 std::string BOI::GetFullname() const {
00103     return fullname;
00104 }
00105

```

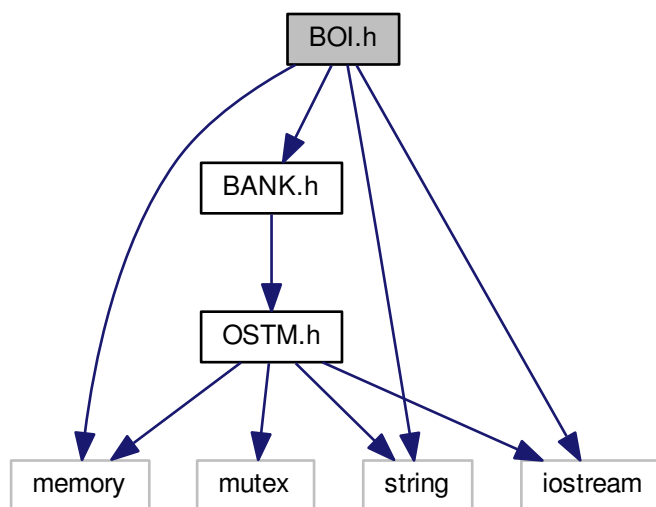
7.15 BOI.h File Reference

```

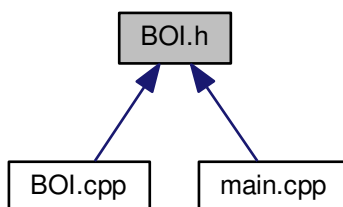
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for BOI.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [BOI](#)

7.16 BOI.h

```

00001
00002 /*
00003  * File:   BOI.h
00004  * Author: Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
  
```

```

00008  */
00009
00010 #ifndef BOI_H
00011 #define BOI_H
00012 #include "BANK.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class BOI: public BANK {
00020 public:
00024     BOI(): BANK()
00025     {
00026         this->accountNumber = 0;
00027         this->balance = 50;
00028         this->firstName = "Joe";
00029         this->lastName = "Blog";
00030         this->address = "High street, Carlow";
00031         this->fullname = firstName + " " + lastName;
00032     }
00033
00037     BOI(int accountNumber, double balance, std::string firstName, std::string lastName, std::string
address): BANK()
00038     {
00039         this->accountNumber = accountNumber;
00040         this->balance = balance;
00041         this->firstName = firstName;
00042         this->lastName = lastName;
00043         this->address = address;
00044         this->fullname = firstName + " " + lastName;
00045     };
00049     BOI(std::shared_ptr<BOI> obj, int _version, int _unique_id): BANK(_version, _unique_id)
00050     {
00051         this->accountNumber = obj->GetAccountNumber();
00052         this->balance = obj->GetBalance();
00053         this->firstName = obj->GetFirstName();
00054         this->lastName = obj->GetLastName();
00055         this->address = obj->GetAddress();
00056         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00057     };
00061     BOI(const BOI& orig);
00065     BOI operator=(const BOI& orig){};
00069     virtual ~BOI();
00070
00071     /*
00072     * Implement OSTM virtual methods
00073     */
00074     // virtual std::shared_ptr<BOI> _cast(std::shared_ptr<OSTM> _object);
00075     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00076     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00077     virtual void toString();
00078
00079     /*
00080     * Implement BANK virtual methods
00081     */
00082     virtual void SetAddress(std::string address);
00083     virtual std::string GetAddress() const;
00084     virtual void SetBalance(double balance);
00085     virtual double GetBalance() const;
00086     virtual void SetAccountNumber(int accountNumber);
00087     virtual int GetAccountNumber() const;
00088     virtual void SetLastName(std::string lastName);
00089     virtual std::string GetLastName() const;
00090     virtual void SetFirstName(std::string firstName);
00091     virtual std::string GetFirstName() const;
00092     virtual void SetFullname(std::string fullname);
00093     virtual std::string GetFullname() const;
00094
00095 private:
00096     std::string fullname;
00097     std::string firstName;
00098     std::string lastName;
00099     int accountNumber;
00100     double balance;
00101     std::string address;
00102 };
00103
00104
00105 #endif /* BOI_H */
00106
00107
00108     //virtual std::string get_class();
00109
00110     //
00111     //virtual bool get(std::shared_ptr<OSTM> object);
00112
00114 // * To change this license header, choose License Headers in Project Properties.
00115 // * To change this template file, choose Tools | Templates

```

```

00116 // * and open the template in the editor.
00117 // */
00118 //
00120 // * File:   BOI.h
00121 // * Author: zoltan
00122 // *
00123 // * Created on January 19, 2018, 6:37 PM
00124 // */
00125 //
00126 // #ifndef BOI_H
00127 // #define BOI_H
00128 // #include "OSTM.h"
00129 // #include <string>
00130 // #include <memory>
00131 // #include <iostream>
00132 //
00133 // class BOI: public OSTM {
00134 // public:
00135 //     BOI(): OSTM()
00136 //     {
00137 //         this->accountNumber = 0;
00138 //         this->balance = 50;
00139 //         this->firstName = "Joe";
00140 //         this->lastName = "Blog";
00141 //         this->address = "High street, Carlow";
00142 //         this->fullname = firstName + " " + lastName;
00143 //     }
00144 //     BOI(int accountNumber, double balance, std::string firstName, std::string lastName, std::string
00145 //         address): OSTM()
00146 //     {
00147 //         this->accountNumber = accountNumber;
00148 //         this->balance = balance;
00149 //         this->firstName = firstName;
00150 //         this->lastName = lastName;
00151 //         this->address = address;
00152 //         this->fullname = firstName + " " + lastName;
00153 //     };
00154 //
00155 //     BOI(OSTM& obj, int _version, int _unique_id): OSTM(_version, _unique_id)
00156 //     {
00157 //         this->accountNumber = obj.GetAccountNumber();
00158 //         this->balance = obj.GetBalance();
00159 //         this->firstName = obj.GetFirstName();
00160 //         this->lastName = obj.GetLastName();
00161 //         this->address = obj.GetAddress();
00162 //         this->fullname = obj.GetFirstName() + " " + obj.GetLastName();
00163 //     };
00164 //
00165 //     BOI(std::shared_ptr<OSTM> obj, int _version, int _unique_id): OSTM(_version, _unique_id)
00166 //     {
00167 //         this->accountNumber = obj->GetAccountNumber();
00168 //         this->balance = obj->GetBalance();
00169 //         this->firstName = obj->GetFirstName();
00170 //         this->lastName = obj->GetLastName();
00171 //         this->address = obj->GetAddress();
00172 //         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00173 //     };
00174 //
00175 //     BOI(const BOI& orig);
00176 //     BOI operator=(const BOI& orig){};
00177 //     virtual ~BOI();
00178 //
00179 //     virtual std::shared_ptr<BOI> _cast(std::shared_ptr<OSTM> _object);
00180 //     virtual std::shared_ptr<BOI> getBaseCopy(OSTM& object);
00181 //     virtual std::shared_ptr<BOI> getBaseCopy(std::shared_ptr<OSTM> object);
00182 //     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00183 //     virtual void toString();
00184 //     virtual void SetAddress(std::string address);
00185 //     virtual std::string GetAddress() const;
00186 //     virtual void SetBalance(double balance);
00187 //     virtual double GetBalance() const;
00188 //     virtual void SetAccountNumber(int accountNumber);
00189 //     virtual int GetAccountNumber() const;
00190 //     virtual void SetLastName(std::string lastName);
00191 //     virtual std::string GetLastName() const;
00192 //     virtual void SetFirstName(std::string firstName);
00193 //     virtual std::string GetFirstName() const;
00194 //     virtual void SetFullname(std::string fullname);
00195 //     virtual std::string GetFullname() const;
00196 //
00197 // private:
00198 //     std::string fullname;
00199 //     std::string firstName;
00200 //     std::string lastName;
00201 //     int accountNumber;

```

```

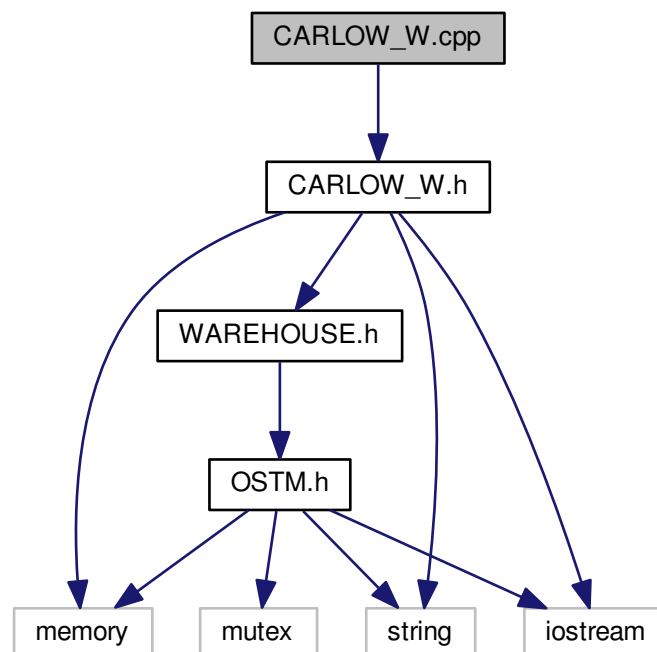
00215 //    double balance;
00216 //    std::string address;
00217 //
00218 //};
00219 //
00220 //endif /* BOI_H */
00221 //
00222 //
00223 //    //virtual std::string get_class();
00224 //
00225 //    //
00226 //    //virtual bool get(std::shared_ptr<OSTM> object);
00227 //

```

7.17 CARLOW_W.cpp File Reference

```
#include "CARLOW_W.h"
```

Include dependency graph for CARLOW_W.cpp:



7.18 CARLOW_W.cpp

```

00001
00002 /*
00003  * File:    CARLOW_W.cpp
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009 #include "CARLOW_W.h"
00010
00011
00012 //CARLOW_W::CARLOW_W() {
00013 //}

```

```

00014 CARLOW_W::~CARLOW_W() {
00015 }
00016
00017 CARLOW_W::CARLOW_W(const CARLOW_W& orig) {
00018 }
00024 std::shared_ptr<OSTM> CARLOW_W::getBaseCopy(std::shared_ptr<OSTM> object)
00025 {
00026
00027     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00028     std::shared_ptr<WAREHOUSE> obj(new CARLOW_W(objTO, object->Get_Version(), object->Get_Unique_ID(
00029 )));
00029     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00030     return ostm_obj;
00031 }
00037 void CARLOW_W::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00038
00039     std::shared_ptr<CARLOW_W> objTO = std::dynamic_pointer_cast<CARLOW_W>(to);
00040     std::shared_ptr<CARLOW_W> objFROM = std::dynamic_pointer_cast<CARLOW_W>(from);
00041     objTO->_shop_address = objFROM->GetShop_address();
00042     objTO->_shop_name = objFROM->GetShop_name();
00043     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00044     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00045     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00046     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00047     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00048     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00049     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00050     objTO->Set_Version(objFROM->Get_Version());
00051
00052
00053 }
00057 //std::shared_ptr<CARLOW_W> CARLOW_W::_cast(std::shared_ptr<OSTM> _object){
00058 //
00059 //     return std::static_pointer_cast<CARLOW_W>(_object);
00060 //}
00064 void CARLOW_W::toString()
00065 {
00066     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
        \nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
        GetShop_address() << "\nNo. Iphones : " << this->
        GetNumber_of_iphones() << "\nNo. Samsung : " << this->
        GetNumber_of_samsung() << "\nNo. Sony : " << this->
        GetNumber_of_sony() << "\nNo. Huawei : " << this->
        GetNumber_of_huawei() << "\nNo. Nokia : " << this->
        GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
        GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00067 }
00068
00069
00070
00071 void CARLOW_W::SetNumber_of_alcatel(int _number_of_alcatel) {
00072     this->_number_of_alcatel = _number_of_alcatel;
00073 }
00074
00075 int CARLOW_W::GetNumber_of_alcatel(){
00076     return _number_of_alcatel;
00077 }
00078
00079 void CARLOW_W::SetNumber_of_nokia(int _number_of_nokia) {
00080     this->_number_of_nokia = _number_of_nokia;
00081 }
00082
00083 int CARLOW_W::GetNumber_of_nokia(){
00084     return _number_of_nokia;
00085 }
00086
00087 void CARLOW_W::SetNumber_of_huawei(int _number_of_huawei) {
00088     this->_number_of_huawei = _number_of_huawei;
00089 }
00090
00091 int CARLOW_W::GetNumber_of_huawei(){
00092     return _number_of_huawei;
00093 }
00094
00095 void CARLOW_W::SetNumber_of_sony(int _number_of_sony) {
00096     this->_number_of_sony = _number_of_sony;
00097 }
00098
00099 int CARLOW_W::GetNumber_of_sony(){
00100     return _number_of_sony;
00101 }
00102
00103 void CARLOW_W::SetNumber_of_samsung(int _number_of_samsung) {
00104     this->_number_of_samsung = _number_of_samsung;
00105 }
00106
00107 int CARLOW_W::GetNumber_of_samsung(){

```



```

00108     return _number_of_samsung;
00109 }
00110
00111 void CARLOW_W::SetNumber_of_iphones(int _number_of_iphones) {
00112     this->_number_of_iphones = _number_of_iphones;
00113 }
00114
00115 int CARLOW_W::GetNumber_of_iphones() {
00116     return _number_of_iphones;
00117 }
00118
00119 void CARLOW_W::SetShop_name(std::string _shop_name) {
00120     this->_shop_name = _shop_name;
00121 }
00122
00123 std::string CARLOW_W::GetShop_name() {
00124     return _shop_name;
00125 }
00126
00127 void CARLOW_W::SetShop_address(std::string _shop_address) {
00128     this->_shop_address = _shop_address;
00129 }
00130
00131 std::string CARLOW_W::GetShop_address() {
00132     return _shop_address;
00133 }
00134
00135
00136

```

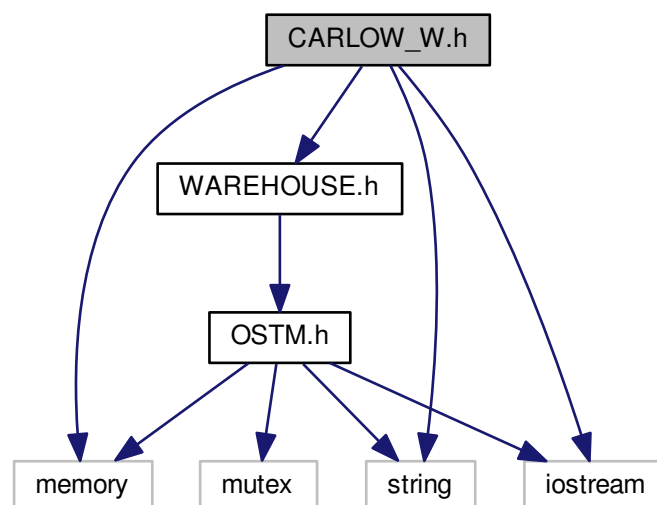
7.19 CARLOW_W.h File Reference

```

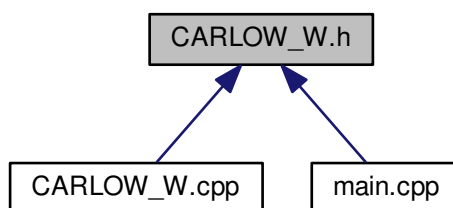
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for CARLOW_W.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CARLOW_W](#)

7.20 CARLOW_W.h

```

00001
00002 /*
00003  * File:    CARLOW_W.h
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #ifndef CARLOW_W_H
00011 #define CARLOW_W_H
00012 #include "WAREHOUSE.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class CARLOW_W :public WAREHOUSE {
00020 public:
00024     CARLOW_W() : WAREHOUSE() {
00025
00026         this->_shop_address = "Carlow potato street";
00027         this->_shop_name = "CARLOW C_WAREHOUSE";
00028         this->_number_of_iphones = 200;
00029         this->_number_of_samsung = 200;
00030         this->_number_of_sony = 200;
00031         this->_number_of_huawei = 200;
00032         this->_number_of_nokia = 200;
00033         this->_number_of_alcatel = 200;
00034     };
00038     CARLOW_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
00039 huawei, int nokia, int alcatel): WAREHOUSE() {
00039     /*
00040         * copy over values
00041     */
00042     this->_shop_address = address;
00043     this->_shop_name = shop_name;
00044     this->_number_of_iphones = iphone;
00045     this->_number_of_samsung = samsung;
00046     this->_number_of_sony = sony;
00047     this->_number_of_huawei = huawei;
00048     this->_number_of_nokia = nokia;
00049     this->_number_of_alcatel = alcatel;
00050
00051     };
00055     CARLOW_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
00056 WAREHOUSE(_version, _unique_id) {
00056     /*
00057         * copy over values
00058     */

```

```

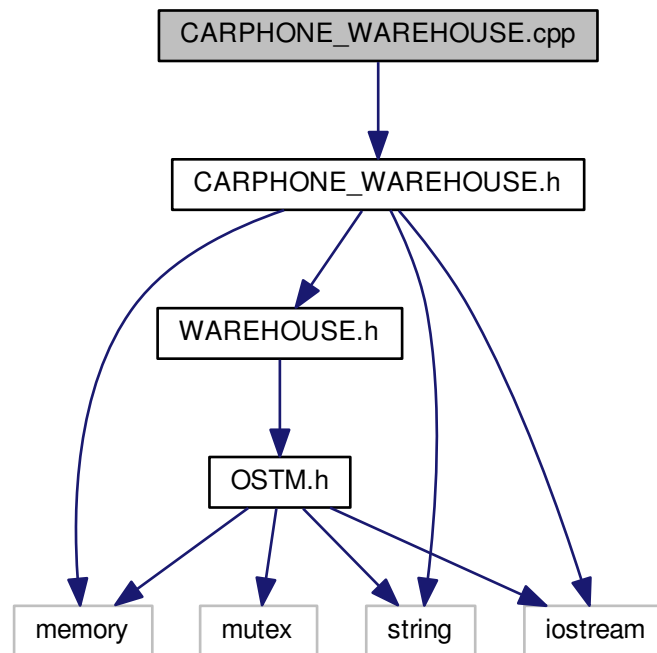
00059         this->_shop_address = obj->GetShop_address();
00060         this->_shop_name = obj->GetShop_name();
00061         this->_number_of_iphones = obj->GetNumber_of_iphones();
00062         this->_number_of_samsung = obj->GetNumber_of_samsung();
00063         this->_number_of_sony = obj->GetNumber_of_sony();
00064         this->_number_of_huawei = obj->GetNumber_of_huawei();
00065         this->_number_of_nokia = obj->GetNumber_of_nokia();
00066         this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067     }
00071     CARLOW_W(const CARLOW_W& orig);
00075     CARLOW_W operator=(const CARLOW_W& orig){};
00079     virtual ~CARLOW_W();
00080     /*
00081      * Implement OSTM virtual methods
00082      */
00083     // virtual std::shared_ptr<CARLOW_W> _cast(std::shared_ptr<OSTM> _object);
00084     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00085     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00086     virtual void toString();
00087     /*
00088      * Implement Warehouse methods
00089      */
00090     virtual void SetNumber_of_alcatel(int _number_of_alcatel);
00091     virtual int GetNumber_of_alcatel();
00092     virtual void SetNumber_of_nokia(int _number_of_nokia);
00093     virtual int GetNumber_of_nokia();
00094     virtual void SetNumber_of_huawei(int _number_of_huawei);
00095     virtual int GetNumber_of_huawei();
00096     virtual void SetNumber_of_sony(int _number_of_sony);
00097     virtual int GetNumber_of_sony();
00098     virtual void SetNumber_of_samsung(int _number_of_samsung);
00099     virtual int GetNumber_of_samsung();
00100     virtual void SetNumber_of_iphones(int _number_of_iphones);
00101     virtual int GetNumber_of_iphones();
00102     virtual void SetShop_name(std::string _shop_name);
00103     virtual std::string GetShop_name();
00104     virtual void SetShop_address(std::string _shop_address);
00105     virtual std::string GetShop_address();
00106
00107 private:
00109     std::string _shop_address;
00110     std::string _shop_name;
00111     int _number_of_iphones;
00112     int _number_of_samsung;
00113     int _number_of_sony;
00114     int _number_of_huawei;
00115     int _number_of_nokia;
00116     int _number_of_alcatel;
00117
00118 };
00119
00120 #endif /* CARLOW_W_H */
00121

```

7.21 CARPHONE_WAREHOUSE.cpp File Reference

```
#include "CARPHONE_WAREHOUSE.h"
```

Include dependency graph for CARPHONE_WAREHOUSE.cpp:



7.22 CARPHONE_WAREHOUSE.cpp

```

00001
00002 /*
00003  * File:    CARPHONE_WAREHOUSE.cpp
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009 #include "CARPHONE_WAREHOUSE.h"
00010
00011 CARPHONE_WAREHOUSE::CARPHONE_WAREHOUSE(const
00012     CARPHONE_WAREHOUSE& orig) {
00013 }
00014 CARPHONE_WAREHOUSE::~CARPHONE_WAREHOUSE() {
00015 }
00021 std::shared_ptr<OSTM> CARPHONE_WAREHOUSE::getBaseCopy(std::shared_ptr<OSTM>
00022     object)
00023 {
00024     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00025     std::shared_ptr<WAREHOUSE> obj(new CARPHONE_WAREHOUSE(objTO, object->Get_Version(),
00026         object->Get_Unique_ID()));
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00028     return ostm_obj;
00029 }
00034 void CARPHONE_WAREHOUSE::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from)
00035 {
00036     std::shared_ptr<CARPHONE_WAREHOUSE> objTO = std::dynamic_pointer_cast<
00037     CARPHONE_WAREHOUSE>(to);
00038     std::shared_ptr<CARPHONE_WAREHOUSE> objFROM = std::dynamic_pointer_cast<
00039     CARPHONE_WAREHOUSE>(from);
00040     objTO->_shop_address = objFROM->GetShop_address();
00041     objTO->_shop_name = objFROM->GetShop_name();
  
```

```

00040     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00041     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00042     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00043     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00044     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00045     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00046     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00047     objTO->Set_Version(objFROM->Get_Version());
00048
00049 }
00053 //std::shared_ptr<CARPHONE_WAREHOUSE> CARPHONE_WAREHOUSE::_cast(std::shared_ptr<OSTM> _object){
00054 //
00055 //     return std::static_pointer_cast<CARPHONE_WAREHOUSE>(_object);
00056 //}
00060 void CARPHONE_WAREHOUSE::toString()
00061 {
00062     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
        \nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
        GetShop_address() << "\nNo. Iphones : " << this->
        GetNumber_of_iphones() << "\nNo. Samsung : " << this->
        GetNumber_of_samsung() << "\nNo. Sony : " << this->
        GetNumber_of_sony() << "\nNo. Huawei : " << this->
        GetNumber_of_huawei() << "\nNo. Nokia : " << this->
        GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
        GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00063 }
00064
00065
00066
00067 void CARPHONE_WAREHOUSE::SetNumber_of_alcatel(int
    _number_of_alcatel) {
00068     this->_number_of_alcatel = _number_of_alcatel;
00069 }
00070
00071 int CARPHONE_WAREHOUSE::GetNumber_of_alcatel() {
00072     return _number_of_alcatel;
00073 }
00074
00075 void CARPHONE_WAREHOUSE::SetNumber_of_nokia(int _number_of_nokia) {
00076     this->_number_of_nokia = _number_of_nokia;
00077 }
00078
00079 int CARPHONE_WAREHOUSE::GetNumber_of_nokia() {
00080     return _number_of_nokia;
00081 }
00082
00083 void CARPHONE_WAREHOUSE::SetNumber_of_huawei(int _number_of_huawei)
    {
00084     this->_number_of_huawei = _number_of_huawei;
00085 }
00086
00087 int CARPHONE_WAREHOUSE::GetNumber_of_huawei() {
00088     return _number_of_huawei;
00089 }
00090
00091 void CARPHONE_WAREHOUSE::SetNumber_of_sony(int _number_of_sony) {
00092     this->_number_of_sony = _number_of_sony;
00093 }
00094
00095 int CARPHONE_WAREHOUSE::GetNumber_of_sony() {
00096     return _number_of_sony;
00097 }
00098
00099 void CARPHONE_WAREHOUSE::SetNumber_of_samsung(int
    _number_of_samsung) {
00100     this->_number_of_samsung = _number_of_samsung;
00101 }
00102
00103 int CARPHONE_WAREHOUSE::GetNumber_of_samsung() {
00104     return _number_of_samsung;
00105 }
00106
00107 void CARPHONE_WAREHOUSE::SetNumber_of_iphones(int
    _number_of_iphones) {
00108     this->_number_of_iphones = _number_of_iphones;
00109 }
00110
00111 int CARPHONE_WAREHOUSE::GetNumber_of_iphones() {
00112     return _number_of_iphones;
00113 }
00114
00115 void CARPHONE_WAREHOUSE::SetShop_name(std::string _shop_name) {
00116     this->_shop_name = _shop_name;
00117 }
00118
00119 std::string CARPHONE_WAREHOUSE::GetShop_name() {
00120     return _shop_name;

```

```

00121 }
00122
00123 void CARPHONE_WAREHOUSE::SetShop_address(std::string _shop_address) {
00124     this->_shop_address = _shop_address;
00125 }
00126
00127 std::string CARPHONE_WAREHOUSE::GetShop_address(){
00128     return _shop_address;
00129 }
00130
00131
00132

```

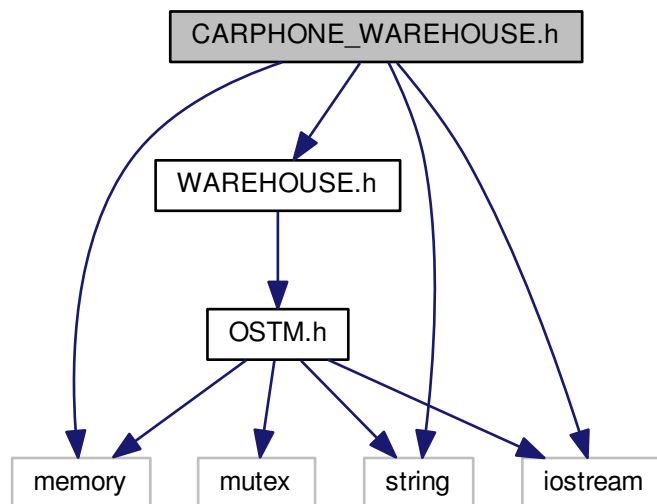
7.23 CARPHONE_WAREHOUSE.h File Reference

```

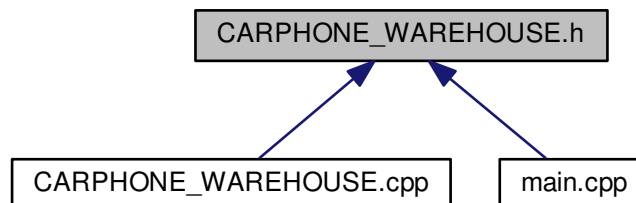
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for CARPHONE_WAREHOUSE.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CARPHONE_WAREHOUSE](#)

7.24 CARPHONE_WAREHOUSE.h

```

00001
00002 /*
00003  * File:    CARPHONE_WAREHOUSE.h
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #ifndef CARPHONE_WAREHOUSE_H
00011 #define CARPHONE_WAREHOUSE_H
00012 #include "WAREHOUSE.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class CARPHONE_WAREHOUSE :public WAREHOUSE {
00020 public:
00024     CARPHONE_WAREHOUSE(): WAREHOUSE() {
00025
00026         this->_shop_address = "DUBLIN XII";
00027         this->_shop_name = "DISTRIBUTION CENTER";
00028         this->_number_of_iphones = 10000;
00029         this->_number_of_samsung = 10000;
00030         this->_number_of_sony = 10000;
00031         this->_number_of_huawei = 10000;
00032         this->_number_of_nokia = 10000;
00033         this->_number_of_alcatel = 10000;
00034     };
00038     CARPHONE_WAREHOUSE(std::string address, std::string shop_name, int iphone, int
samsung, int sony, int huawei, int nokia, int alcatel): WAREHOUSE(){
00039         /*
00040          * copy over values
00041          */
00042         this->_shop_address = address;
00043         this->_shop_name = shop_name;
00044         this->_number_of_iphones = iphone;
00045         this->_number_of_samsung = samsung;
00046         this->_number_of_sony = sony;
00047         this->_number_of_huawei = huawei;
00048         this->_number_of_nokia = nokia;
00049         this->_number_of_alcatel = alcatel;
00050
00051     };
00055     CARPHONE_WAREHOUSE(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
WAREHOUSE(_version, _unique_id){
00056         /*
00057          * copy over values
00058          */
00059         this->_shop_address = obj->GetShop_address();
00060         this->_shop_name = obj->GetShop_name();
00061         this->_number_of_iphones = obj->GetNumber_of_iphones();
00062         this->_number_of_samsung = obj->GetNumber_of_samsung();
00063         this->_number_of_sony = obj->GetNumber_of_sony();
00064         this->_number_of_huawei = obj->GetNumber_of_huawei();
00065         this->_number_of_nokia = obj->GetNumber_of_nokia();
00066         this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067     }
00071     CARPHONE_WAREHOUSE(const CARPHONE_WAREHOUSE& orig);
00075     CARPHONE_WAREHOUSE operator=(const
CARPHONE_WAREHOUSE& orig){};
00079     virtual ~CARPHONE_WAREHOUSE();
00080     /*
00081     * Implement OSTM virtual methods
00082     */
00083     //virtual std::shared_ptr<CARPHONE_WAREHOUSE> _cast(std::shared_ptr<OSTM> _object);
00084     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00085     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00086
00087
00088
00089     virtual void toString();
00090     /*
00091     * Implement Warehouse methods
00092     */
00093     virtual void SetNumber_of_alcatel(int _number_of_alcatel);
00094     virtual int GetNumber_of_alcatel();

```

```

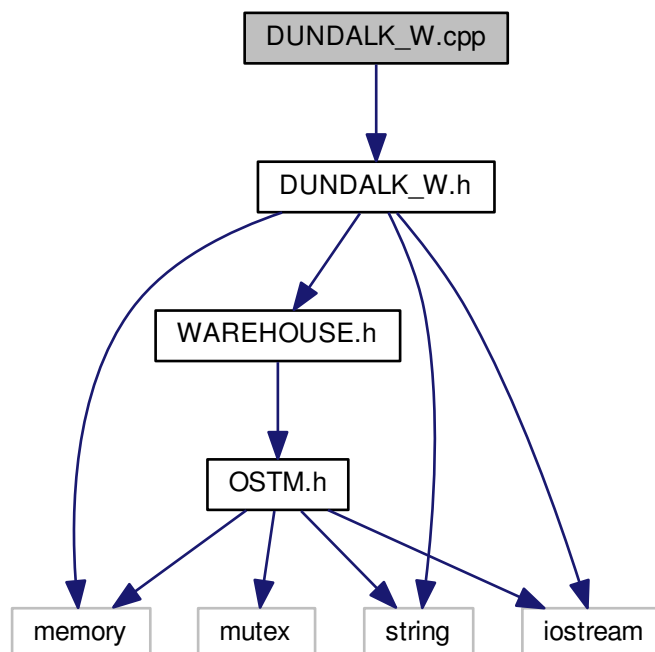
00095     virtual void SetNumber_of_nokia(int _number_of_nokia);
00096     virtual int GetNumber_of_nokia();
00097     virtual void SetNumber_of_huawei(int _number_of_huawei);
00098     virtual int GetNumber_of_huawei();
00099     virtual void SetNumber_of_sony(int _number_of_sony);
00100     virtual int GetNumber_of_sony();
00101     virtual void SetNumber_of_samsung(int _number_of_samsung);
00102     virtual int GetNumber_of_samsung();
00103     virtual void SetNumber_of_iphones(int _number_of_iphones);
00104     virtual int GetNumber_of_iphones();
00105     virtual void SetShop_name(std::string _shop_name);
00106     virtual std::string GetShop_name();
00107     virtual void SetShop_address(std::string _shop_address);
00108     virtual std::string GetShop_address();
00109
00110 private:
00111     std::string _shop_address;
00112     std::string _shop_name;
00113     int _number_of_iphones;
00114     int _number_of_samsung;
00115     int _number_of_sony;
00116     int _number_of_huawei;
00117     int _number_of_nokia;
00118     int _number_of_alcatel;
00119
00120 };
00121
00122 #endif /* CARPHONE_WAREHOUSE_H */
00123

```

7.25 DUNDALK_W.cpp File Reference

```
#include "DUNDALK_W.h"
```

Include dependency graph for DUNDALK_W.cpp:



7.26 DUNDALK_W.cpp

```

00001
00002 /*
00003  * File:    DUNDALK_W.cpp
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #include "DUNDALK_W.h"
00011
00012 DUNDALK_W::~DUNDALK_W() {
00013 }
00014
00015 DUNDALK_W::DUNDALK_W(const DUNDALK_W& orig) {
00016 }
00022 std::shared_ptr<OSTM> DUNDALK_W::getBaseCopy(std::shared_ptr<OSTM> object)
00023 {
00024
00025     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00026     std::shared_ptr<WAREHOUSE> obj(new DUNDALK_W(objTO, object->Get_Version(), object->
    Get_Unique_ID()));
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00028     return ostm_obj;
00029 }
00035 void DUNDALK_W::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00036
00037     std::shared_ptr<DUNDALK_W> objTO = std::dynamic_pointer_cast<DUNDALK_W>(to);
00038     std::shared_ptr<DUNDALK_W> objFROM = std::dynamic_pointer_cast<DUNDALK_W>(from);
00039     objTO->_shop_address = objFROM->GetShop_address();
00040     objTO->_shop_name = objFROM->GetShop_name();
00041     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00042     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00043     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00044     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00045     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00046     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00047     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00048     objTO->Set_Version(objFROM->Get_Version());
00049
00050
00051 }
00055 //std::shared_ptr<DUNDALK_W> DUNDALK_W::_cast(std::shared_ptr<OSTM> _object){
00056 //
00057 //     return std::static_pointer_cast<DUNDALK_W>(_object);
00058 //}
00062 void DUNDALK_W::toString()
00063 {
00064     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
    \nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
    GetShop_address() << "\nNo. iPhones : " << this->
    GetNumber_of_iphones() << "\nNo. Samsung : " << this->
    GetNumber_of_samsung() << "\nNo. Sony : " << this->
    GetNumber_of_sony() << "\nNo. Huawei : " << this->
    GetNumber_of_huawei() << "\nNo. Nokia : " << this->
    GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
    GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00065 }
00066
00067
00068
00069 void DUNDALK_W::SetNumber_of_alcatel(int _number_of_alcatel) {
00070     this->_number_of_alcatel = _number_of_alcatel;
00071 }
00072
00073 int DUNDALK_W::GetNumber_of_alcatel(){
00074     return _number_of_alcatel;
00075 }
00076
00077 void DUNDALK_W::SetNumber_of_nokia(int _number_of_nokia) {
00078     this->_number_of_nokia = _number_of_nokia;
00079 }
00080
00081 int DUNDALK_W::GetNumber_of_nokia(){
00082     return _number_of_nokia;
00083 }
00084
00085 void DUNDALK_W::SetNumber_of_huawei(int _number_of_huawei) {
00086     this->_number_of_huawei = _number_of_huawei;
00087 }
00088
00089 int DUNDALK_W::GetNumber_of_huawei(){
00090     return _number_of_huawei;
00091 }

```

```

00092
00093 void DUNDALK_W::SetNumber_of_sony(int _number_of_sony) {
00094     this->_number_of_sony = _number_of_sony;
00095 }
00096
00097 int DUNDALK_W::GetNumber_of_sony() {
00098     return _number_of_sony;
00099 }
00100
00101 void DUNDALK_W::SetNumber_of_samsung(int _number_of_samsung) {
00102     this->_number_of_samsung = _number_of_samsung;
00103 }
00104
00105 int DUNDALK_W::GetNumber_of_samsung() {
00106     return _number_of_samsung;
00107 }
00108
00109 void DUNDALK_W::SetNumber_of_iphones(int _number_of_iphones) {
00110     this->_number_of_iphones = _number_of_iphones;
00111 }
00112
00113 int DUNDALK_W::GetNumber_of_iphones() {
00114     return _number_of_iphones;
00115 }
00116
00117 void DUNDALK_W::SetShop_name(std::string _shop_name) {
00118     this->_shop_name = _shop_name;
00119 }
00120
00121 std::string DUNDALK_W::GetShop_name() {
00122     return _shop_name;
00123 }
00124
00125 void DUNDALK_W::SetShop_address(std::string _shop_address) {
00126     this->_shop_address = _shop_address;
00127 }
00128
00129 std::string DUNDALK_W::GetShop_address() {
00130     return _shop_address;
00131 }
00132
00133
00134

```

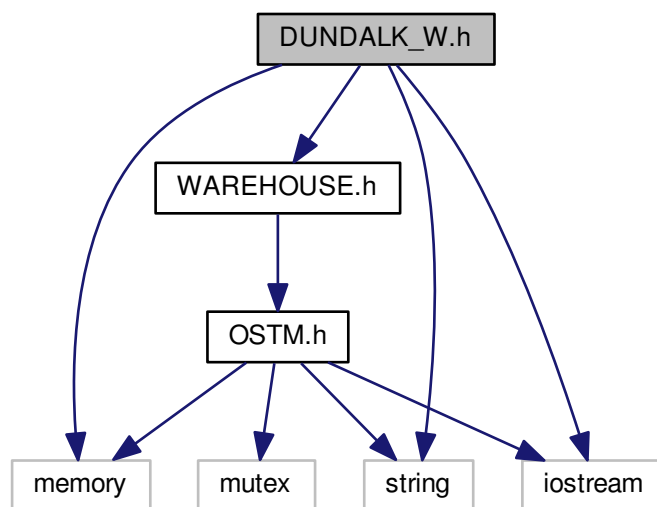
7.27 DUNDALK_W.h File Reference

```

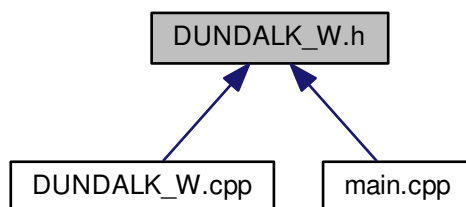
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for DUNDALK_W.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [DUNDALK_W](#)

7.28 DUNDALK_W.h

```

00001
00002 /*
00003  * File:   DUNDALK_W.h
00004  * Author: Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
  
```

```

00008  */
00009
00010 #ifndef DUNDALK_W_H
00011 #define DUNDALK_W_H
00012 #include "WAREHOUSE.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class DUNDALK_W :public WAREHOUSE {
00020 public:
00024     DUNDALK_W() : WAREHOUSE() {
00025
00026         this->_shop_address = "Dundalk Busy Street";
00027         this->_shop_name = "DUNDALK D_WAREHOUSE";
00028         this->_number_of_iphones = 200;
00029         this->_number_of_samsung = 200;
00030         this->_number_of_sony = 200;
00031         this->_number_of_huawei = 200;
00032         this->_number_of_nokia = 200;
00033         this->_number_of_alcatel = 200;
00034     };
00038     DUNDALK_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
    huawei, int nokia, int alcatel): WAREHOUSE() {
00039         /*
00040         * copy over values
00041         */
00042         this->_shop_address = address;
00043         this->_shop_name = shop_name;
00044         this->_number_of_iphones = iphone;
00045         this->_number_of_samsung = samsung;
00046         this->_number_of_sony = sony;
00047         this->_number_of_huawei = huawei;
00048         this->_number_of_nokia = nokia;
00049         this->_number_of_alcatel = alcatel;
00050
00051     };
00055     DUNDALK_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
    WAREHOUSE(_version, _unique_id) {
00056         /*
00057         * copy over values
00058         */
00059         this->_shop_address = obj->GetShop_address();
00060         this->_shop_name = obj->GetShop_name();
00061         this->_number_of_iphones = obj->GetNumber_of_iphones();
00062         this->_number_of_samsung = obj->GetNumber_of_samsung();
00063         this->_number_of_sony = obj->GetNumber_of_sony();
00064         this->_number_of_huawei = obj->GetNumber_of_huawei();
00065         this->_number_of_nokia = obj->GetNumber_of_nokia();
00066         this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067     }
00071     DUNDALK_W(const DUNDALK_W& orig);
00075     DUNDALK_W operator=(const DUNDALK_W& orig){};
00079     virtual ~DUNDALK_W();
00080     /*
00081     * Implement OSTM virtual methods
00082     */
00083     //virtual std::shared_ptr<DUNDALK_W> _cast(std::shared_ptr<OSTM> _object);
00084     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00085     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00086     virtual void toString();
00087     /*
00088     * Implement Warehouse methods
00089     */
00090     virtual void SetNumber_of_alcatel(int _number_of_alcatel);
00091     virtual int GetNumber_of_alcatel();
00092     virtual void SetNumber_of_nokia(int _number_of_nokia);
00093     virtual int GetNumber_of_nokia();
00094     virtual void SetNumber_of_huawei(int _number_of_huawei);
00095     virtual int GetNumber_of_huawei();
00096     virtual void SetNumber_of_sony(int _number_of_sony);
00097     virtual int GetNumber_of_sony();
00098     virtual void SetNumber_of_samsung(int _number_of_samsung);
00099     virtual int GetNumber_of_samsung();
00100     virtual void SetNumber_of_iphones(int _number_of_iphones);
00101     virtual int GetNumber_of_iphones();
00102     virtual void SetShop_name(std::string _shop_name);
00103     virtual std::string GetShop_name();
00104     virtual void SetShop_address(std::string _shop_address);
00105     virtual std::string GetShop_address();
00106
00107 private:
00109     std::string _shop_address;
00110     std::string _shop_name;
00111     int _number_of_iphones;
00112     int _number_of_samsung;
00113     int _number_of_sony;

```

```

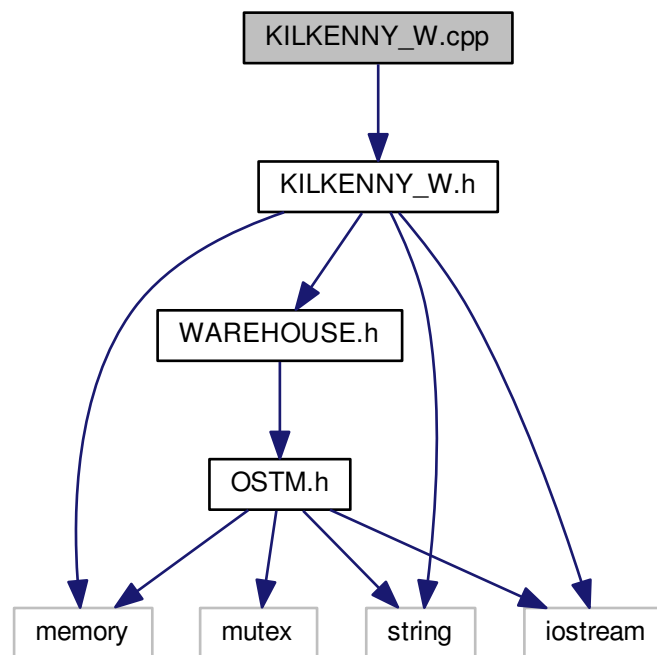
00114     int _number_of_huawei;
00115     int _number_of_nokia;
00116     int _number_of_alcatel;
00117 };
00118 };
00119
00120 #endif /* DUNDALK_W_H */
00121

```

7.29 KILKENNY_W.cpp File Reference

```
#include "KILKENNY_W.h"
```

Include dependency graph for KILKENNY_W.cpp:



7.30 KILKENNY_W.cpp

```

00001
00002 /*
00003  * File:    KILKENNY_W.cpp
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #include "KILKENNY_W.h"
00011
00012 KILKENNY_W::~KILKENNY_W() {
00013 }
00014
00015 KILKENNY_W::KILKENNY_W(const KILKENNY_W& orig) {
00016 }
00022 std::shared_ptr<OSTM> KILKENNY_W::getBaseCopy(std::shared_ptr<OSTM> object)
00023 {

```

```

00024
00025     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00026     std::shared_ptr<WAREHOUSE> obj(new KILKENNY_W(objTO, object->Get_Version(),object->
    Get_Unique_ID()));
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00028     return ostm_obj;
00029 }
00035 void KILKENNY_W::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00036
00037     std::shared_ptr<KILKENNY_W> objTO = std::dynamic_pointer_cast<KILKENNY_W>(to);
00038     std::shared_ptr<KILKENNY_W> objFROM = std::dynamic_pointer_cast<KILKENNY_W>(from);
00039     objTO->_shop_address = objFROM->GetShop_address();
00040     objTO->_shop_name = objFROM->GetShop_name();
00041     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00042     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00043     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00044     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00045     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00046     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00047     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00048     objTO->Set_Version(objFROM->Get_Version());
00049
00050
00051 }
00055 //std::shared_ptr<KILKENNY_W> KILKENNY_W::_cast(std::shared_ptr<OSTM> _object){
00056 //
00057 //     return static_cast<std::shared_ptr<KILKENNY_W>>(_object);
00058 //}
00062 void KILKENNY_W::toString()
00063 {
00064     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
    \nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
    GetShop_address() << "\nNo. iPhones : " << this->
    GetNumber_of_iphones() << "\nNo. Samsung : " << this->
    GetNumber_of_samsung() << "\nNo. Sony : " << this->
    GetNumber_of_sony() << "\nNo. Huawei : " << this->
    GetNumber_of_huawei() << "\nNo. Nokia : " << this->
    GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
    GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00065 }
00066
00067
00068
00069 void KILKENNY_W::SetNumber_of_alcatel(int _number_of_alcatel) {
00070     this->_number_of_alcatel = _number_of_alcatel;
00071 }
00072
00073 int KILKENNY_W::GetNumber_of_alcatel(){
00074     return _number_of_alcatel;
00075 }
00076
00077 void KILKENNY_W::SetNumber_of_nokia(int _number_of_nokia) {
00078     this->_number_of_nokia = _number_of_nokia;
00079 }
00080
00081 int KILKENNY_W::GetNumber_of_nokia(){
00082     return _number_of_nokia;
00083 }
00084
00085 void KILKENNY_W::SetNumber_of_huawei(int _number_of_huawei) {
00086     this->_number_of_huawei = _number_of_huawei;
00087 }
00088
00089 int KILKENNY_W::GetNumber_of_huawei(){
00090     return _number_of_huawei;
00091 }
00092
00093 void KILKENNY_W::SetNumber_of_sony(int _number_of_sony) {
00094     this->_number_of_sony = _number_of_sony;
00095 }
00096
00097 int KILKENNY_W::GetNumber_of_sony(){
00098     return _number_of_sony;
00099 }
00100
00101 void KILKENNY_W::SetNumber_of_samsung(int _number_of_samsung) {
00102     this->_number_of_samsung = _number_of_samsung;
00103 }
00104
00105 int KILKENNY_W::GetNumber_of_samsung(){
00106     return _number_of_samsung;
00107 }
00108
00109 void KILKENNY_W::SetNumber_of_iphones(int _number_of_iphones) {
00110     this->_number_of_iphones = _number_of_iphones;
00111 }
00112

```

```

00113 int KILKENNY_W::GetNumber_of_iphones() {
00114     return _number_of_iphones;
00115 }
00116
00117 void KILKENNY_W::SetShop_name(std::string _shop_name) {
00118     this->_shop_name = _shop_name;
00119 }
00120
00121 std::string KILKENNY_W::GetShop_name() {
00122     return _shop_name;
00123 }
00124
00125 void KILKENNY_W::SetShop_address(std::string _shop_address) {
00126     this->_shop_address = _shop_address;
00127 }
00128
00129 std::string KILKENNY_W::GetShop_address() {
00130     return _shop_address;
00131 }
00132

```

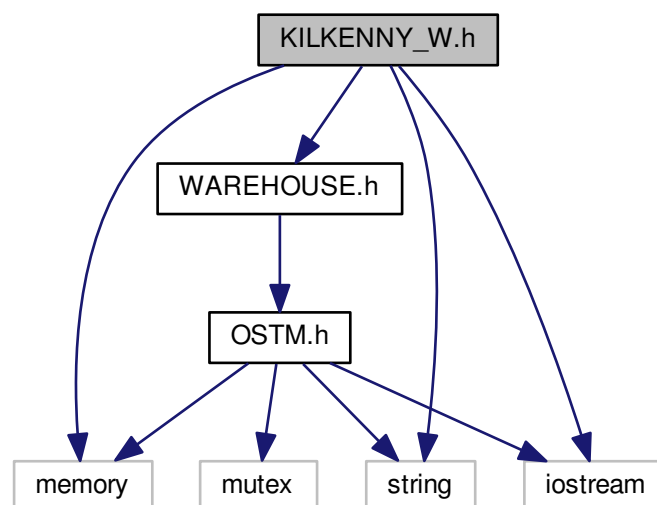
7.31 KILKENNY_W.h File Reference

```

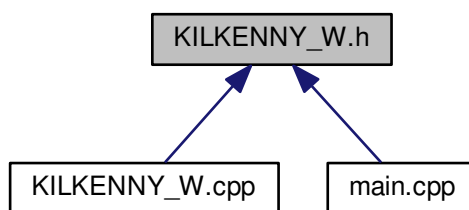
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for KILKENNY_W.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [KILKENNY_W](#)

7.32 KILKENNY_W.h

```

00001
00002 /*
00003  * File:    KILKENNY_W.h
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #ifndef KILKENNY_W_H
00011 #define KILKENNY_W_H
00012 #include "WAREHOUSE.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class KILKENNY_W : public WAREHOUSE {
00020 public:
00024     KILKENNY_W(): WAREHOUSE() {
00025
00026         this->_shop_address = "Kilkenny High Street";
00027         this->_shop_name = "KILKENNY K_WAREHOUSE";
00028         this->_number_of_iphones = 200;
00029         this->_number_of_samsung = 200;
00030         this->_number_of_sony = 200;
00031         this->_number_of_huawei = 200;
00032         this->_number_of_nokia = 200;
00033         this->_number_of_alcatel = 200;
00034     };
00038     KILKENNY_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
00039     huawei, int nokia, int alcatel): WAREHOUSE(){
00039         /*
00040          * copy over values
00041          */
00042         this->_shop_address = address;
00043         this->_shop_name = shop_name;
00044         this->_number_of_iphones = iphone;
00045         this->_number_of_samsung = samsung;
00046         this->_number_of_sony = sony;
00047         this->_number_of_huawei = huawei;
00048         this->_number_of_nokia = nokia;
00049         this->_number_of_alcatel = alcatel;
00050
00051     };
00055     KILKENNY_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
00056     WAREHOUSE(_version, _unique_id){
00056         /*
00057          * copy over values
00058          */

```



```

00059         this->_shop_address = obj->GetShop_address();
00060         this->_shop_name = obj->GetShop_name();
00061         this->_number_of_iphones = obj->GetNumber_of_iphones();
00062         this->_number_of_samsung = obj->GetNumber_of_samsung();
00063         this->_number_of_sony = obj->GetNumber_of_sony();
00064         this->_number_of_huawei = obj->GetNumber_of_huawei();
00065         this->_number_of_nokia = obj->GetNumber_of_nokia();
00066         this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067     }
00071     KILKENNY_W(const KILKENNY_W& orig);
00075     KILKENNY_W operator=(const KILKENNY_W& orig){};
00079     virtual ~KILKENNY_W();
00080     /*
00081      * Implement OSTM virtual methods
00082      */
00083     //virtual std::shared_ptr<KILKENNY_W> _cast(std::shared_ptr<OSTM> _object);
00084     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00085     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00086     virtual void toString();
00087     /*
00088      * Implement Warehouse methods
00089      */
00090     virtual void SetNumber_of_alcatel(int _number_of_alcatel);
00091     virtual int GetNumber_of_alcatel();
00092     virtual void SetNumber_of_nokia(int _number_of_nokia);
00093     virtual int GetNumber_of_nokia();
00094     virtual void SetNumber_of_huawei(int _number_of_huawei);
00095     virtual int GetNumber_of_huawei();
00096     virtual void SetNumber_of_sony(int _number_of_sony);
00097     virtual int GetNumber_of_sony();
00098     virtual void SetNumber_of_samsung(int _number_of_samsung);
00099     virtual int GetNumber_of_samsung();
00100     virtual void SetNumber_of_iphones(int _number_of_iphones);
00101     virtual int GetNumber_of_iphones();
00102     virtual void SetShop_name(std::string _shop_name);
00103     virtual std::string GetShop_name();
00104     virtual void SetShop_address(std::string _shop_address);
00105     virtual std::string GetShop_address();
00106
00107
00108 private:
00109     std::string _shop_address;
00110     std::string _shop_name;
00111     int _number_of_iphones;
00112     int _number_of_samsung;
00113     int _number_of_sony;
00114     int _number_of_huawei;
00115     int _number_of_nokia;
00116     int _number_of_alcatel;
00117
00118 };
00119
00120 #endif /* KILKENNY_W_H */
00121

```

7.33 main.cpp File Reference

```
#include <windows.h>
```

```

#include <cstdlib>
#include <iostream>
#include <thread>
#include <process.h>
#include "TM.h"
#include "AIB.h"
#include "BOI.h"
#include "BOA.h"
#include "SWBPLC.h"
#include "ULSTER.h"
#include "UNBL.h"
#include "WAREHOUSE.h"
#include "CARPHONE_WAREHOUSE.h"
#include "CARLOW_W.h"
#include "KILKENNY_W.h"
#include "TALLAGH_W.h"
#include "DUNDALK_W.h"
#include "SLIGO_W.h"
#include <mutex>
#include <memory>
#include <condition_variable>
#include <vector>

```

Include dependency graph for main.cpp:



Functions

- void [_six_account_transfer_](#) (std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _from_one_, std::shared_ptr< OSTM > _from_two_, std::shared_ptr< OSTM > _from_three_, std::shared_ptr< OSTM > _from_four_, std::shared_ptr< OSTM > _from_five_, TM & _tm, double _amount)
six_account_transfer function, takes six std::shared_ptr<OSTM> pointer, the Transaction manager, and the amount to use in the transaction and transfer the _amount value from five account to one account
- void [_two_account_transfer_](#) (std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _from_, TM & _tm, double _amount)
two_account_transfer function, takes two std::shared_ptr<OSTM> pointer, the Transaction manager, and the amount to use in the transaction and transfer the _amount value from one account to the another account
- void [_nesting_](#) (std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _from_, TM & _tm, double _amount)
nesting function, takes two std::shared_ptr<OSTM> pointer, the Transaction manager, and the amount to use in the transaction and transfer the _amount value from one account to the another account This function create nested transactions inside the transaction, and call other function to nesting the transaction as well
- void [_complex_transfer_](#) (std::shared_ptr< OSTM > _from_, std::shared_ptr< OSTM > _from_two_, std::vector< std::shared_ptr< OSTM > > _customer_vec, TM & _tm, double _amount)
complex_transfer function, takes two std::shared_ptr<OSTM> pointer, a vector of std::shared_ptr<OSTM> pointers, the Transaction manager, and the amount to use in the transaction, and transfer the _amount value from booth single objects to the objects to the vector collection
- void [_warehouse_transfer_](#) (std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _from_, TM & _tm, double _amount)
warehouse_transfer function, takes two std::shared_ptr<OSTM> pointer, the Transaction manager, and the amount to use in the transaction and transfer the _amount value from one account to the another account
- void [_nested_warehouse_transfer_](#) (std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _to_two_, std::shared_ptr< OSTM > _to_three_, std::shared_ptr< OSTM > _from_, TM & _tm, double _amount)

nested_warehouse_transfer function, takes three `std::shared_ptr<OSTM>` pointer, the Transaction manager, and the amount to use in the transaction and transfer the `_amount` value from one account to the another account

- void `_complex_warehouse_transfer_` (`std::shared_ptr< OSTM > _to_`, `std::shared_ptr< OSTM > _to_`↵
two, `std::shared_ptr< OSTM > _to_three`, `std::vector< std::shared_ptr< OSTM >> _warehouse_vec`, `std`↵
: `shared_ptr< OSTM > _from_`, `TM & _tm`, `double _amount`)
- int `main` (void)

7.33.1 Function Documentation

7.33.1.1 void `_complex_transfer_` (`std::shared_ptr< OSTM > _from_`, `std::shared_ptr< OSTM > _from_two_`, `std::vector< std::shared_ptr< OSTM >> _customer_vec`, `TM & _tm`, `double _amount`)

complex_transfer function, takes two `std::shared_ptr<OSTM>` pointer, a vector of `std::shared_ptr<OSTM>` pointers, the Transaction manager, and the amount to use in the transaction, and transfer the `_amount` value from booth single objects to the objects to the vector collection

Parameters

<code>std::shared_ptr<TX></code>	tx, Transaction Object
<code>std::shared_ptr<BANK></code>	type, <i>FROM</i> & <i>FROM_TWO</i> & <i>TO</i>
<code>std::shared_ptr<OSTM></code>	type, <i>FROM_OSTM_ONE</i> & <i>FROM_OSTM_TWO</i> & <i>TO_OSTM</i>

Register the two single account

Declare required pointers

Register customers accounts from the collection (vector)

From `std::shared_ptr<OSTM>` to `std::shared_ptr<BANK>` to access the virtual methods

Make changes with the objects

From `std::shared_ptr<BANK>` to `std::shared_ptr<OSTM>` to store the memory spaces

Store changes

Commit changes

Definition at line 296 of file `main.cpp`.

References `BANK::SetBalance()`.

```

00296
00297         std::shared_ptr<TX> tx = _tm._get_tx();
00301         tx->_register(_from_);
00302         tx->_register(_from_two_);
00306         std::shared_ptr<OSTM> _FROM_OSTM_ONE_, _FROM_OSTM_TWO_, _TO_OSTM_;
00307         std::shared_ptr<BANK> _FROM_, _FROM_TWO_, _TO_;
00308
00309         bool done = false;
00310         try {
00311             while (!done) {
00312                 // for (int i = 0; i < vector_number; ++i) {
00313                 for (auto&& obj : _customer_vec) {
00317                     // auto&& obj = _customer_vec.at(i);
00318                     tx->_register(obj);
00322                     _FROM_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
00323                     _FROM_TWO_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_two_));
00324                     _TO_ = std::dynamic_pointer_cast<BANK> (tx->load(obj));

```

```

00328         _FROM->SetBalance(_FROM->GetBalance() - _amount);
00329         _FROM_TWO->SetBalance(_FROM_TWO->GetBalance() - _amount);
00330         _TO->SetBalance(_TO->GetBalance() + (_amount * 2));
00334         _FROM_OSTM_ONE_ = std::dynamic_pointer_cast<OSTM> (_FROM_);
00335         _FROM_OSTM_TWO_ = std::dynamic_pointer_cast<OSTM> (_FROM_TWO_);
00336         _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_);
00340         tx->store(_FROM_OSTM_ONE_);
00341         tx->store(_FROM_OSTM_TWO_);
00342         tx->store(_TO_OSTM_);
00343     }
00347     done = tx->commit();
00348 }
00349 } catch (std::runtime_error& e) {
00350     std::cout << e.what() << std::endl;
00351 }
00352 }

```

Here is the call graph for this function:



7.33.1.2 void _complex_warehouse_transfer_ (std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _to_two,
std::shared_ptr< OSTM > _to_three, std::vector< std::shared_ptr< OSTM >> _warehouse_vec, std::shared_ptr<
OSTM > _from_, TM & _tm, double _amount)

Register the two single account

Declare required pointers

Register customers accounts from the collection (vector)

From std::shared_ptr<OSTM> to std::shared_ptr<BANK> to access the virtual methods

Make changes with the objects

From std::shared_ptr<WAREHOUSE> to std::shared_ptr<OSTM> to store the memory spaces

Store changes

NESTED [WAREHOUSE TEST](#) _to_two

Make changes with the objects

From std::shared_ptr<BANK> to std::shared_ptr<OSTM> to store the memory spaces

Store changes

Commit changes

Definition at line 520 of file [main.cpp](#).

References [_nested_warehouse_transfer_\(\)](#), [_warehouse_transfer_\(\)](#), and [WAREHOUSE::SetNumber_of_nokia\(\)](#).

```

00520
00521     {
00522     std::shared_ptr<TX> tx = _tm._get_tx();
00523     tx->_register(_to_);
00524     tx->_register(_to_two);
00525     tx->_register(_to_three);
00526     tx->_register(_from_);
00527     std::shared_ptr<WAREHOUSE> _TO_SHOP_, _TO_SHOP_TWO_, _TO_SHOP_VEC_, _FROM_DIST_;
00528     std::shared_ptr<OSTM> _TO_OSTM_, _TO_OSTM_TWO_, _TO_OSTM_VEC_, _FROM_OSTM_;
00529
00530     bool done = false;
00531     try {
00532         while (!done) {
00533
00534             // for (int i = 0; i < vector_number; ++i) {
00535             for (auto&& obj : _warehouse_vec) {
00536                 //auto&& obj = _warehouse_vec.at(i);
00537                 tx->_register(obj);
00538                 _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
00539                 _TO_SHOP_TWO_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_two));
00540                 _TO_SHOP_VEC_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(obj));
00541                 _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));
00542
00543                 _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
00544                 _TO_SHOP_TWO_->SetNumber_of_nokia(_TO_SHOP_TWO_->GetNumber_of_nokia() + _amount);
00545                 _TO_SHOP_VEC_->SetNumber_of_nokia(_TO_SHOP_VEC_->GetNumber_of_nokia() + _amount);
00546                 _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - (_amount * 3));
00547
00548                 _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
00549                 _TO_SHOP_TWO_->SetNumber_of_samsung(_TO_SHOP_TWO_->GetNumber_of_samsung() + _amount);
00550                 _TO_SHOP_VEC_->SetNumber_of_samsung(_TO_SHOP_VEC_->GetNumber_of_samsung() + _amount);
00551                 _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - (_amount * 3));
00552
00553                 _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
00554                 _TO_SHOP_TWO_->SetNumber_of_iphones(_TO_SHOP_TWO_->GetNumber_of_iphones() + _amount);
00555                 _TO_SHOP_VEC_->SetNumber_of_iphones(_TO_SHOP_VEC_->GetNumber_of_iphones() + _amount);
00556                 _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - (_amount * 3));
00557
00558                 _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
00559                 _TO_SHOP_TWO_->SetNumber_of_sony(_TO_SHOP_TWO_->GetNumber_of_sony() + _amount);
00560                 _TO_SHOP_VEC_->SetNumber_of_sony(_TO_SHOP_VEC_->GetNumber_of_sony() + _amount);
00561                 _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - (_amount * 3));
00562
00563                 _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00564                 _TO_OSTM_TWO_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_TWO_);
00565                 _TO_OSTM_VEC_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_VEC_);
00566                 _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00567                 tx->store(_TO_OSTM_);
00568                 tx->store(_TO_SHOP_TWO_);
00569                 tx->store(_TO_SHOP_VEC_);
00570                 tx->store(_FROM_OSTM_);
00571
00572             }
00573             std::shared_ptr<TX> txTwo = _tm._get_tx();
00574             bool nestedDone = false;
00575             while (!nestedDone) {
00576                 _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_to_two));
00577                 _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_from_));
00578                 _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
00579                 _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);
00580
00581                 _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
00582                 _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);
00583
00584                 _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
00585                 _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);
00586
00587                 _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
00588                 _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
00589                 _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00590                 _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00591                 txTwo->store(_TO_OSTM_);
00592                 txTwo->store(_FROM_OSTM_);
00593
00594                 /*
00595                 * NESTED TRANSACTION TEST _to_three
00596                 */
00597                 _warehouse_transfer_(_to_three, _from_, _tm, _amount);
00598                 _nested_warehouse_transfer_(_to_, _to_two, _to_three, _from_,
00599                 _tm, _amount);
00600
00601                 nestedDone = tx->commit();
00602             }
00603         }
00604     }
00605

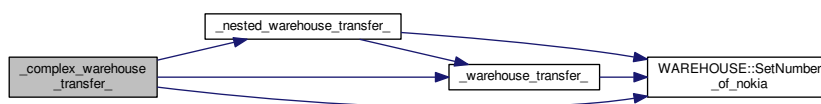
```

```

00640         done = tx->commit();
00641     }
00642 }
00643 } catch (std::runtime_error& e) {
00644     std::cout << e.what() << std::endl;
00645 }
00646 }

```

Here is the call graph for this function:



7.33.1.3 void `_nested_warehouse_transfer_` (`std::shared_ptr< OSTM > _to_`, `std::shared_ptr< OSTM > _to_two`, `std::shared_ptr< OSTM > _to_three`, `std::shared_ptr< OSTM > _from_`, `TM & _tm`, `double _amount`)

`nested_warehouse_transfer` function, takes three `std::shared_ptr<OSTM>` pointer, the Transaction manager, and the amount to use in the transaction and transfer the `_amount` value from one account to the another account

Parameters

<code>std::shared_ptr< TX ></code>	tx, Transaction Object
<code>std::shared_ptr< WAREHOUSE ></code>	type, <code>TO_SHOP</code> & <code>FROM_DIST</code>
<code>std::shared_ptr< OSTM ></code>	type, <code>TO_OSTM</code> & <code>FROM_OSTM</code>

Register the two single account

Declare required pointers

From `std::shared_ptr<OSTM>` to `std::shared_ptr<BANK>` to access the virtual methods

Make changes with the objects

From `std::shared_ptr<BANK>` to `std::shared_ptr<OSTM>` to store the memory spaces

Store changes

NESTED **WAREHOUSE** TEST `_to_two`

Make changes with the objects

From `std::shared_ptr<BANK>` to `std::shared_ptr<OSTM>` to store the memory spaces

Store changes

Commit changes

Definition at line 421 of file `main.cpp`.

References `_warehouse_transfer_()`, and `WAREHOUSE::SetNumber_of_nokia()`.

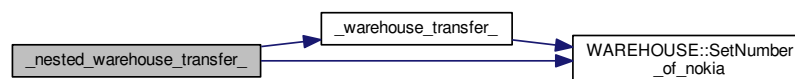
Referenced by `_complex_warehouse_transfer_()`.

```

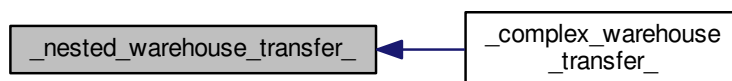
00421
00422         std::shared_ptr<TX> tx = _tm._get_tx();
00426         tx->_register(_to_);
00427         tx->_register(_to_two);
00428         tx->_register(_to_three);
00429         tx->_register(_from_);
00433         std::shared_ptr<WAREHOUSE> _TO_SHOP_, _FROM_DIST_;
00434         std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;
00435
00436         bool done = false;
00437         try {
00438             while (!done) {
00442                 _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
00443                 _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));
00447                 _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
00448                 _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);
00449
00450                 _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
00451                 _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);
00452
00453                 _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
00454                 _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);
00455
00456                 _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
00457                 _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
00461                 _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00462                 _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00466                 tx->store(_TO_OSTM_);
00467                 tx->store(_FROM_OSTM_);
00468
00472                 std::shared_ptr<TX> txTwo = _tm._get_tx();
00473                 bool nestedDone = false;
00474                 while (!nestedDone) {
00475                     _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_to_two));
00476                     _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_from_));
00480                     _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
00481                     _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);
00482
00483                     _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
00484                     _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);
00485
00486                     _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
00487                     _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);
00488
00489                     _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
00490                     _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
00494                     _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00495                     _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00499                     txTwo->store(_TO_OSTM_);
00500                     txTwo->store(_FROM_OSTM_);
00501
00502                     /*
00503                      * NESTED TRANSACTION TEST _to_three
00504                      */
00505                     _warehouse_transfer_(_to_three, _from_, _tm, _amount);
00506
00507                     nestedDone = tx->commit();
00508                 }
00509                 done = tx->commit();
00513             }
00514         } catch (std::runtime_error& e) {
00515             std::cout << e.what() << std::endl;
00516         }
00517     }
00518 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.1.4 void _nesting_ (std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _from_, TM & _tm, double _amount)

nesting function, takes two std::shared_ptr<OSTM> pointer, the Transaction manager, and the amount to use in the transaction and transfer the _amount value from one account to the another account This function create nested transactions inside the transaction, and call other function to nesting the transaction as well

Parameters

<i>std::shared_ptr< TX></i>	tx, Transaction Object
<i>std::shared_ptr< BANK></i>	type, <i>TO_BANK</i> & <i>FROM_BANK</i>
<i>std::shared_ptr< OSTM></i>	type, <i>TO_OSTM</i> & <i>FROM_OSTM</i>

Register the two single account

Declare required pointers

From std::shared_ptr<OSTM> to std::shared_ptr<BANK> to access the virtual methods

Make changes with the objects

From std::shared_ptr<BANK> to std::shared_ptr<OSTM> to store the memory spaces

Store changes

NESTED TRANSACTION

Make changes with the objects

From std::shared_ptr<BANK> to std::shared_ptr<OSTM> to store the memory spaces

Store changes

NESTED TRANSACTION IN THE NESTED TRANSACTION *two_account_transfer* function call

Commit changes

Definition at line 208 of file [main.cpp](#).

References [_two_account_transfer_\(\)](#), and [BANK::SetBalance\(\)](#).

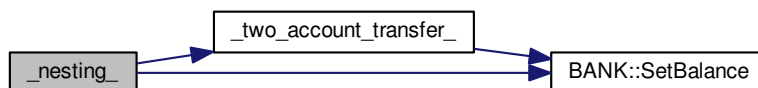
Referenced by [main\(\)](#).


```

00208                                     {
00209         std::shared_ptr<TX> tx = _tm._get_tx();
00213         tx->_register(_to_);
00214         tx->_register(_from_);
00218         std::shared_ptr<BANK> _TO_BANK_, _FROM_BANK_;
00219         std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;
00220
00221
00222         bool done = false;
00223         try {
00224             while (!done) {
00228                 _TO_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
00229                 _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
00233                 _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
00234                 _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
00238                 _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
00239                 _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
00243                 tx->store(_TO_OSTM_);
00244                 tx->store(_FROM_OSTM_);
00245
00249                 std::shared_ptr<TX> txTwo = _tm._get_tx();
00250
00251                 bool nestedDone = false;
00252                 while (!nestedDone) {
00253                     _TO_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_to_));
00254                     _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_from_));
00258                     _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
00259                     _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
00263                     _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
00264                     _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
00268                     txTwo->store(_TO_OSTM_);
00269                     txTwo->store(_FROM_OSTM_);
00274                     _two_account_transfer_(_to_, _from_, _tm, _amount);
00275
00276                     nestedDone = txTwo->commit();
00277                 }
00278
00282                 done = tx->commit();
00283             }
00284         } catch (std::runtime_error& e) {
00285             std::cout << e.what() << std::endl;
00286         }
00287     }

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.1.5 `void _six_account_transfer(std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _from_one_, std::shared_ptr< OSTM > _from_two_, std::shared_ptr< OSTM > _from_three_, std::shared_ptr< OSTM > _from_four_, std::shared_ptr< OSTM > _from_five_, TM & _tm, double _amount)`

`six_account_transfer` function, takes six `std::shared_ptr<OSTM>` pointer, the Transaction manager, and the amount to use in the transaction and transfer the `_amount` value from five account to one account

Parameters

<code>std::shared_ptr<TX></code>	tx, Transaction Object
<code>std::shared_ptr<BANK></code>	type, <code>TO & FROM_ONE & FROM_TWO & FROM_THREE & FROM_FOUR & FROM_FIVE</code>
<code>std::shared_ptr<OSTM></code>	type, <code>_TO_OSTM & _FROM_ONE_OSTM & _FROM_TWO_OSTM & _FROM_THREE_OSTM & _FROM_FOUR_OSTM & _FROM_FIVE_OSTM</code>

Register the two single account

Required pointers to use in transaction

From `std::shared_ptr<OSTM>` to `std::shared_ptr<BANK>` to access the virtual methods

Make changes with the objects

From `std::shared_ptr<BANK>` to `std::shared_ptr<OSTM>` to store the memory spaces

Store changes

Commit changes

Definition at line 53 of file [main.cpp](#).

References [BANK::SetBalance\(\)](#).

```

00053
00054         {
00055             std::shared_ptr<TX> tx = _tm._get_tx();
00056             tx->_register(_to_);
00057             tx->_register(_from_one_);
00058             tx->_register(_from_two_);
00059             tx->_register(_from_three_);
00060             tx->_register(_from_four_);
00061             tx->_register(_from_five_);
00062             tx->_register(_from_five_);
00063             tx->_register(_from_five_);
00064
00065             std::shared_ptr<OSTM> _TO_OSTM, _FROM_ONE_OSTM, _FROM_TWO_OSTM, _FROM_THREE_OSTM, _FROM_FOUR_OSTM,
00066             _FROM_FIVE_OSTM;
00067             std::shared_ptr<BANK> _TO_, _FROM_ONE_, _FROM_TWO_, _FROM_THREE_, _FROM_FOUR_, _FROM_FIVE_;
00068             try {
00069                 bool done = false;
00070                 while (!done) {
00071                     _TO_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
00072                     _FROM_ONE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_one_));
00073                     _FROM_TWO_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_two_));
00074                     _FROM_THREE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_three_));
00075                     _FROM_FOUR_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_four_));
00076                     _FROM_FIVE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_five_));
00077                     _TO_->SetBalance(_TO_->GetBalance() + (_amount * 5));
00078                     _FROM_ONE_->SetBalance(_FROM_ONE_->GetBalance() - _amount);
00079                     _FROM_TWO_->SetBalance(_FROM_TWO_->GetBalance() - _amount);
00080                     _FROM_THREE_->SetBalance(_FROM_THREE_->GetBalance() - _amount);
00081                     _FROM_FOUR_->SetBalance(_FROM_FOUR_->GetBalance() - _amount);
00082                     _FROM_FIVE_->SetBalance(_FROM_FIVE_->GetBalance() - _amount);
00083                     _TO_OSTM = std::dynamic_pointer_cast<OSTM> (_TO_);
00084                     _FROM_ONE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_ONE_);
00085                     _FROM_TWO_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_TWO_);
00086                     _FROM_THREE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_THREE_);
00087                     _FROM_FOUR_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_FOUR_);
00088                     _FROM_FIVE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_FIVE_);
00089                 }
00090             }
00091         }

```

```

00103         tx->store(_TO_OSTM);
00104         tx->store(_FROM_ONE_OSTM);
00105         tx->store(_FROM_TWO_OSTM);
00106         tx->store(_FROM_THREE_OSTM);
00107         tx->store(_FROM_FOUR_OSTM);
00108         tx->store(_FROM_FIVE_OSTM);
00112         done = tx->commit();
00113     }
00114     } catch (std::runtime_error& e) {
00115         std::cout << e.what() << std::endl;
00116     }
00117 }

```

Here is the call graph for this function:



7.33.1.6 `void two_account_transfer_ (std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _from_, TM & _tm, double _amount)`

two_account_transfer function, takes two `std::shared_ptr<OSTM>` pointer, the Transaction manager, and the amount to use in the transaction and transfer the `_amount` value from one account to the another account

Parameters

<code>std::shared_ptr<TX></code>	tx, Transaction Object
<code>std::shared_ptr<BANK></code>	type, <i>TO_BANK</i> & <i>FROM_BANK</i>
<code>std::shared_ptr<OSTM></code>	type, <i>TO_OSTM</i> & <i>FROM_OSTM</i>

Register the two single account

Declare required pointers

From `std::shared_ptr<OSTM>` to `std::shared_ptr<BANK>` to access the virtual methods

Make changes with the objects

From `std::shared_ptr<BANK>` to `std::shared_ptr<OSTM>` to store the memory spaces

Store changes

NESTED TRANSACTION

Make changes with the objects

From `std::shared_ptr<BANK>` to `std::shared_ptr<OSTM>` to store the memory spaces

Store changes

Commit changes

Commit changes

Definition at line 125 of file [main.cpp](#).

References [BANK::SetBalance\(\)](#).

Referenced by [_nesting_\(\)](#).

```

00125 {
00126     std::shared_ptr<TX> tx = _tm._get_tx();
00130     tx->_register(_to_);
00131     tx->_register(_from_);
00135     std::shared_ptr<BANK> _TO_BANK_, _FROM_BANK_;
00136     std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;
00137
00138     bool done = false;
00139     try {
00140         while (!done) {
00144             _TO_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
00145             _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
00149             _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
00150             _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
00154             _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
00155             _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
00159             tx->store(_TO_OSTM_);
00160             tx->store(_FROM_OSTM_);
00161
00165             std::shared_ptr<TX> txTwo = _tm._get_tx();
00166
00167             bool nestedDone = false;
00168             while (!nestedDone) {
00169                 _TO_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_to_));
00170                 _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_from_));
00174                 _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
00175                 _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
00179                 _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
00180                 _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
00184                 txTwo->store(_TO_OSTM_);
00185                 txTwo->store(_FROM_OSTM_);
00189                 nestedDone = txTwo->commit();
00190             }
00194             done = tx->commit();
00195         }
00196     } catch (std::runtime_error& e) {
00197         std::cout << e.what() << std::endl;
00198     }
00199 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.1.7 `void _warehouse_transfer_ (std::shared_ptr< OSTM > _to_, std::shared_ptr< OSTM > _from_, TM & _tm, double _amount)`

warehouse_transfer function, takes two `std::shared_ptr<OSTM>` pointer, the Transaction manager, and the amount to use in the transaction and transfer the `_amount` value from one account to the another account

Parameters

<code>std::shared_ptr<TX></code>	tx, Transaction Object
<code>std::shared_ptr<WAREHOUSE></code>	type, <i>TO_SHOP</i> & <i>FROM_DIST</i>
<code>std::shared_ptr<OSTM></code>	type, <i>TO_OSTM</i> & <i>FROM_OSTM</i>

Register the two single account

Declare required pointers

From `std::shared_ptr<OSTM>` to `std::shared_ptr<BANK>` to access the virtual methods

Make changes with the objects

From `std::shared_ptr<BANK>` to `std::shared_ptr<OSTM>` to store the memory spaces

Store changes

Commit changes

Definition at line 360 of file [main.cpp](#).

References [WAREHOUSE::SetNumber_of_nokia\(\)](#).

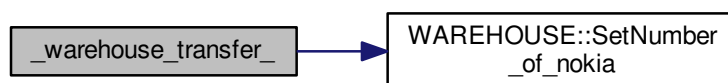
Referenced by [_complex_warehouse_transfer_\(\)](#), and [_nested_warehouse_transfer_\(\)](#).

```

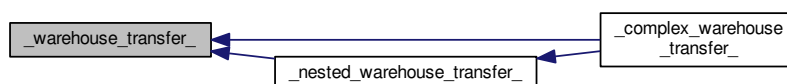
00360
00361     std::shared_ptr<TX> tx = _tm._get_tx();
00362     tx->_register(_to_);
00363     tx->_register(_from_);
00364     std::shared_ptr<WAREHOUSE> _TO_SHOP_, _FROM_DIST_;
00365     std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;
00366
00367     bool done = false;
00368     try {
00369         while (!done) {
00370             _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
00371             _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));
00372             _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
00373             _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);
00374
00375             _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
00376             _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);
00377
00378             _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
00379             _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);
00380
00381             _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
00382             _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
00383             _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00384             _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00385             tx->store(_TO_OSTM_);
00386             tx->store(_FROM_OSTM_);
00387             done = tx->commit();
00388         }
00389     } catch (std::runtime_error& e) {
00390         std::cout << e.what() << std::endl;
00391     }
00392 }
00393

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.1.8 int main (void)

main method to run test Get the Transaction Manager

```
TM& tm = TM::Instance();
```

Create vector to store std::shared_ptr<OSTM> pointers. All object will have unique ID by default

```
std::vector<std::shared_ptr<OSTM>> _customer_vec(vector_number);
std::vector<std::shared_ptr<OSTM>> _warehouse_vec(vector_number);
```

Create objects type of **BANK**. All object will have unique ID by default

```
std::shared_ptr<OSTM> aib_ptr = new AIB(100, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny");
std::shared_ptr<OSTM> boi_ptr = new BOI(200, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny");
std::shared_ptr<OSTM> boa_ptr = new BOA(300, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny");
std::shared_ptr<OSTM> swplc_ptr = new SWBPLC(400, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny");
std::shared_ptr<OSTM> ulster_ptr = new ULSTER(500, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny");
std::shared_ptr<OSTM> unbl_ptr = new UNBL(600, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny");
```

Create objects type of **WAREHOUSE**. All object will have unique ID by default

```
std::shared_ptr<OSTM> w_dist = new CARPHONE_WAREHOUSE();
std::shared_ptr<OSTM> c_shop = new CARLOW_W();
std::shared_ptr<OSTM> k_shop = new KILKENNY_W();
std::shared_ptr<OSTM> t_shop = new TALLAGH_W();
std::shared_ptr<OSTM> d_shop = new DUNDALK_W();
std::shared_ptr<OSTM> s_shop = new SLIGO_W();
```

Create vector of std::shared_ptr<OSTM> **BANK** pointers

vector_number is 100 at the moment

```
for(int i=0;i<vector_number;++i)
```

Create vector of `std::shared_ptr<OSTM>` **WAREHOUSE** pointers
vector_number is 100 at the moment
for(int i=0;i<vector_number;++i)

Display **WAREHOUSE** objects before transaction

```
w_dist->toString();
c_shop->toString();
k_shop->toString();
t_shop->toString();
d_shop->toString();
s_shop->toString();
```

Display **BANK** objects before transaction

```
aib_ptr->toString();
boi_ptr->toString();
boa_ptr->toString();
swplc_ptr->toString();
ulster_ptr->toString();
unbl_ptr->toString();
```

Parameters

<i>transferAmount</i>	in the transaction, control the value in the transaction between objetscs
<i>threadArraySize</i>	control number of threads The logic in the IF ELSE statement distribute the threads between three different thread creating option. If the threadArraySize is divisible with three, the threads will be distributed between function. However, you can creates any number of threads, but to follow the correct output should increase the IF ELSE statement to distribute the threads in equal number.

Creating threadsⁿ -> threadArraySize
for (int i = 0; i < threadArraySize; ++i)

TEST 1 : Nested transaction Test

```
thArray[i] = std::thread(nesting, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
```

TEST 2 :Three different type of function call where the objects are participating in multiple type of transactions

```
thArray[i] = std::thread(two_account_transfer, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
thArray[i] = std::thread(six_account_transfer, boi_ptr, boa_ptr, swplc_ptr, ulster_ptr, aib_ptr, unbl_ptr, std::ref(tm),
transferAmount)
thArray[i] = std::thread(complex_transfer, aib_ptr, boi_ptr, std::ref(_customer_vec), std::ref(tm), transferAmount);
```

TEST 3 : Testing **WAREHOUSE** type pointers within transactions

```
thArray[i] = std::thread(phone_transfer, c_shop, w_dist, std::ref(tm), transferAmount);
```

TEST 4 : Testing **WAREHOUSE** type pointers within nested transactions

```
thArray[i] = std::thread(nested_warehouse_transfer, c_shop, d_shop, k_shop, w_dist, std::ref(tm), transferAmount);
```

TEST 5 : Testing **WAREHOUSE** type pointers within mixed and nested transactions

```
thArray[i] = std::thread(warehouse_transfer, c_shop, w_dist, std::ref(tm), transferAmount);
thArray[i] = std::thread(nested_warehouse_transfer, c_shop, d_shop, k_shop, w_dist, std::ref(tm), transferAmount);
thArray[i] = std::thread(complex_warehouse_transfer, d_shop, c_shop, std::ref(_warehouse_vec), w_dist, std::ref(tm), transferAmount);
```

Display objects after all transactions are finished
 Uncomment the required corresponding TEST to display results

Extra tx to call and display ROLLBACK value
`std::shared_ptr<TX> tx = tm._get_tx();`

Display the number of ROLLBACK by all the threads
`std::cout << "Rollback counter is : " << tx->getTest_counter() << std::endl;`

Display object from vector

Clean up Transaction Manager from all main process associated transactions
`tm._TX_EXIT();`

Display all Transactions associated with the main process. It should be empty after `_TX_EXIT()` function call!!!
`tm.print_all();`

Definition at line 651 of file `main.cpp`.

References `_nesting_()`.

```

00651     {
00652         TM& tm = TM::Instance();
00653
00654         std::vector<std::shared_ptr< OSTM>>_customer_vec; //(vector_number);
00655         std::vector<std::shared_ptr< OSTM>>_warehouse_vec; //(vector_number);
00656
00657         std::shared_ptr<OSTM> aib_ptr(new AIB(100, 500, "Joe", "Blog", "High street, Kilkenney, Co.Kilkenney")
00658 );
00659         std::shared_ptr<OSTM> boi_ptr(new BOI(200, 500, "Joe", "Blog", "High street, Kilkenney, Co.Kilkenney")
00660 );
00661         std::shared_ptr<OSTM> boa_ptr(new BOA(300, 500, "Joe", "Blog", "High street, Kilkenney, Co.Kilkenney")
00662 );
00663         std::shared_ptr<OSTM> swplc_ptr(new SWBPLC(400, 500, "Joe", "Blog", "High street, Kilkenney,
00664 Co.Kilkenney"));
00665         std::shared_ptr<OSTM> ulster_ptr(new ULSTER(500, 500, "Joe", "Blog", "High street, Kilkenney,
00666 Co.Kilkenney"));
00667         std::shared_ptr<OSTM> unbl_ptr(new UNBL(600, 500, "Joe", "Blog", "High street, Kilkenney,
00668 Co.Kilkenney"));
00669
00670         std::shared_ptr<OSTM> w_dist(new CARPHONE_WAREHOUSE());
00671         std::shared_ptr<OSTM> c_shop(new CARLOW_W());
00672         std::shared_ptr<OSTM> k_shop(new KILKENNY_W());
00673         std::shared_ptr<OSTM> t_shop(new TALLAGH_W());
00674         std::shared_ptr<OSTM> d_shop(new DUNDALK_W());
00675         std::shared_ptr<OSTM> s_shop(new SLIGO_W());
00676
00677         for (int i = 0; i < vector_number; ++i) {
00678             if (i % 5 == 0) {
00679                 std::shared_ptr<OSTM> sharedptr(new CARLOW_W());
00680                 _warehouse_vec.push_back(std::move(sharedptr));
00681             } else if (i % 4 == 0) {
00682                 std::shared_ptr<OSTM> sharedptr(new KILKENNY_W());
00683                 _warehouse_vec.push_back(std::move(sharedptr));
00684             } else if (i % 3 == 0) {
00685                 std::shared_ptr<OSTM> sharedptr(new TALLAGH_W());
00686                 _warehouse_vec.push_back(std::move(sharedptr));
00687             } else if (i % 2 == 0) {
00688                 std::shared_ptr<OSTM> sharedptr(new DUNDALK_W());
00689                 _warehouse_vec.push_back(std::move(sharedptr));
00690             } else if (i % 1 == 0) {
00691                 std::shared_ptr<OSTM> sharedptr(new SLIGO_W());
00692                 _warehouse_vec.push_back(std::move(sharedptr));
00693             }
00694         }
00695
00696         for (int i = 0; i < vector_number; ++i) {
00697             if (i % 6 == 0) {
00698                 std::shared_ptr<OSTM> sharedptr(new AIB(i, 50, "Joe", "Blog", "High street, Kilkenney,
00699 Co.Kilkenney"));
00700                 _customer_vec.push_back(std::move(sharedptr));
00701             } else if (i % 5 == 0) {
00702                 std::shared_ptr<OSTM> sharedptr(new BOI(i, 50, "Joe", "Blog", "High street, Kilkenney,
00703 Co.Kilkenney"));
00704                 _customer_vec.push_back(std::move(sharedptr));
00705             }
00706         }
00707     }

```



```

00736         } else if (i % 4 == 0) {
00737             std::shared_ptr<OSTM> sharedptr(new BOA(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00738             _customer_vec.push_back(std::move(sharedptr));
00739         } else if (i % 3 == 0) {
00740             std::shared_ptr<OSTM> sharedptr(new SWBPLC(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00741             _customer_vec.push_back(std::move(sharedptr));
00742         } else if (i % 2 == 0) {
00743             std::shared_ptr<OSTM> sharedptr(new ULSTER(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00744             _customer_vec.push_back(std::move(sharedptr));
00745         } else if (i % 1 == 0) {
00746             std::shared_ptr<OSTM> sharedptr(new UNBL(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00747             _customer_vec.push_back(std::move(sharedptr));
00748         }
00749     }
00750
00760     // w_dist->toString();
00761     // c_shop->toString();
00762     // k_shop->toString();
00763     // t_shop->toString();
00764     // d_shop->toString();
00765     // s_shop->toString();
00766
00777     /*
00778     * TEST 1 : object requirements
00779     */
00780     aib_ptr->toString();
00781     boi_ptr->toString();
00782     boa_ptr->toString();
00783     swplc_ptr->toString();
00784     ulster_ptr->toString();
00785     unbl_ptr->toString();
00786
00787     /*
00788     * TEST 2 : object requirements
00789     */
00790     // aib_ptr->toString();
00791     // boi_ptr->toString();
00792     // boa_ptr->toString();
00793     // swplc_ptr->toString();
00794     // ulster_ptr->toString();
00795     // unbl_ptr->toString();
00796     // for(int i=0; i<vector_number; ++i){
00797     //     _customer_vec[i]->toString();
00798     // }
00799
00800     /*
00801     * TEST 3 : object requirements
00802     */
00803     // w_dist->toString();
00804     // c_shop->toString();
00805     // k_shop->toString();
00806     // t_shop->toString();
00807
00808     /*
00809     * TEST 4 : objects requirements
00810     */
00811     // w_dist->toString();
00812     // c_shop->toString();
00813     // k_shop->toString();
00814     // t_shop->toString();
00815     // d_shop->toString();
00816     // s_shop->toString();
00817
00818     /*
00819     * TEST 5 : objects requirements
00820     */
00821     // w_dist->toString();
00822     // c_shop->toString();
00823     // k_shop->toString();
00824     // t_shop->toString();
00825     // d_shop->toString();
00826     // s_shop->toString();
00827
00828     // for(auto&& elem: _warehouse_vec){
00829     //     elem->toString(); // virtual dispatch
00830     // }
00831
00832
00833
00834
00835
00839     int transferAmount = 1;
00846     int threadArraySize = 300;

```

```

00847
00848     std::thread thArray[300];
00849
00854     for (int i = 0; i < threadArraySize; ++i) {
00855
00860         //thArray[i] = std::thread(_nesting_, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
00861         if (i % 3 == 0)
00862             thArray[i] = std::thread(_nesting_, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
00863         else if (i % 2 == 0)
00864             thArray[i] = std::thread(_nesting_, boa_ptr, swplc_ptr, std::ref(tm), transferAmount);
00865         else if (i % 1 == 0)
00866             thArray[i] = std::thread(_nesting_, ulster_ptr, unbl_ptr, std::ref(tm), transferAmount);
00867
00874         //     if (i % 3 == 0)
00875         //         thArray[i] = std::thread(_two_account_transfer_, aib_ptr, boi_ptr, std::ref(tm),
transferAmount);
00876         //     else if (i % 2 == 0)
00877         //         thArray[i] = std::thread(_six_account_transfer_, boi_ptr, boa_ptr, swplc_ptr, ulster_ptr,
aib_ptr, unbl_ptr, std::ref(tm), transferAmount);
00878         //     else if (i % 1 == 0)
00879         //         thArray[i] = std::thread(_complex_transfer_, aib_ptr, boi_ptr, std::ref(_customer_vec),
std::ref(tm), transferAmount);
00880
00881
00886         //         if (i % 3 == 0)
00887         //             thArray[i] = std::thread(_warehouse_transfer_, c_shop, w_dist, std::ref(tm),
transferAmount);
00888         //         else if (i % 2 == 0)
00889         //             thArray[i] = std::thread(_warehouse_transfer_, k_shop, w_dist, std::ref(tm),
transferAmount);
00890         //         else if (i % 1 == 0)
00891         //             thArray[i] = std::thread(_warehouse_transfer_, t_shop, w_dist, std::ref(tm),
transferAmount);
00892
00897         //         if (i % 3 == 0)
00898         //             thArray[i] = std::thread(_nested_warehouse_transfer_, c_shop, d_shop, k_shop, w_dist,
std::ref(tm), transferAmount);
00899         //         else if (i % 2 == 0)
00900         //             thArray[i] = std::thread(_nested_warehouse_transfer_, k_shop, s_shop, t_shop, w_dist,
std::ref(tm), transferAmount);
00901         //         else if (i % 1 == 0)
00902         //             thArray[i] = std::thread(_nested_warehouse_transfer_, t_shop, c_shop, s_shop, w_dist,
std::ref(tm), transferAmount);
00903
00912         //         if (i % 3 == 0)
00913         //             thArray[i] = std::thread(_warehouse_transfer_, c_shop, w_dist, std::ref(tm),
transferAmount);
00914         //         else if (i % 2 == 0)
00915         //             thArray[i] = std::thread(_nested_warehouse_transfer_, k_shop, s_shop, t_shop, w_dist,
std::ref(tm), transferAmount);
00916         //         else if (i % 1 == 0)
00917         //             thArray[i] = std::thread(_complex_warehouse_transfer_, d_shop, s_shop, c_shop,
std::ref(_warehouse_vec), w_dist, std::ref(tm), transferAmount);
00918
00919
00920     }
00921     /*
00922     * Join threads^n -> threadArraySize<br>
00923     * thArray[i].join();
00924     */
00925     for (int i = 0; i < threadArraySize; ++i) {
00926         thArray[i].join();
00927     }
00928
00929
00930     std::cout << "\nMain process print " << std::endl;
00931
00936     /*
00937     * TEST 1 : object requirements
00938     */
00939     aib_ptr->toString();
00940     boi_ptr->toString();
00941     boa_ptr->toString();
00942     swplc_ptr->toString();
00943     ulster_ptr->toString();
00944     unbl_ptr->toString();
00945
00946     /*
00947     * TEST 2 : object requirements
00948     */
00949     //     aib_ptr->toString();
00950     //     boi_ptr->toString();
00951     //     boa_ptr->toString();
00952     //     swplc_ptr->toString();
00953     //     ulster_ptr->toString();
00954     //     unbl_ptr->toString();
00955     //     for(int i=0; i<vector_number; ++i){
00956     //         _customer_vec[i]->toString();

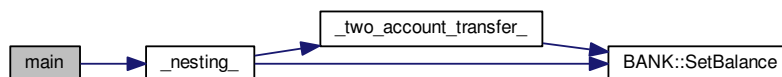
```

```

00957     //    }
00958
00959     /*
00960     * TEST 3 : object requirements
00961     */
00962     //          w_dist->toString();
00963     //          c_shop->toString();
00964     //          k_shop->toString();
00965     //          t_shop->toString();
00966
00967     /*
00968     * TEST 4 : objects requirements
00969     */
00970     //          w_dist->toString();
00971     //          c_shop->toString();
00972     //          k_shop->toString();
00973     //          t_shop->toString();
00974     //          d_shop->toString();
00975     //          s_shop->toString();
00976
00977     /*
00978     * TEST 5 : objects requirements
00979     */
00980     //          w_dist->toString();
00981     //          c_shop->toString();
00982     //          k_shop->toString();
00983     //          t_shop->toString();
00984     //          d_shop->toString();
00985     //          s_shop->toString();
00986
00987     //          for(auto&& elem: _warehouse_vec){
00988     //              elem->toString(); // virtual dispatch
00989     //          }
00990
00991     /* TEST 5 FINISH */
00992
00993     std::cout << "\nMAIN PROCESS EXIT !!!! " << std::endl;
01000     std::shared_ptr<TX> tx = tm._get_tx();
01001
01006     std::cout << "Rollback counter is : " << tx->getTest_counter() << std::endl;
01010     //     std::cout << "[vector_number]" << std::endl;
01011     //     for (int i = 0; i < vector_number; ++i) {
01012     //         //_customer_vec[i]->toString();
01013     //         auto&& os = _customer_vec.at(i);
01014     //         os->toString();
01015     //     }
01016     //     std::cout << "[_warehouse_vec]" << std::endl;
01017     //     for(auto&& elem: _warehouse_vec){
01018     //         elem->toString(); // virtual dispatch
01019     //     }
01020     //     //_customer_vec[10]->toString();
01021
01022
01027     tm._TX_EXIT();
01028     std::cout << "\nPRINT ALL FROM TM !!!! SHOULD BE EMPTY AFTER _TX_EXIT() !" << std::endl;
01033     tm.print_all();
01034     int t = 0;
01035     std::cin >> t;
01036     return 0;
01037 }

```

Here is the call graph for this function:



7.34 main.cpp

```
00001 /*
```

```

00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005  */
00006
00007  /*
00008  * File:   main.cpp
00009  * Author: zoltan
00010  *
00011  * Created on November 27, 2017, 9:26 PM
00012  */
00013  #include <windows.h>
00014  #include <cstdlib>
00015  #include <iostream>
00016  #include <thread>
00017  #include <process.h>
00018
00019  // #include <unistd.h> // used for pid_t
00020
00021  // STM library requirement
00022  #include "TM.h"
00023  #include "AIB.h" // Bank Account
00024  #include "BOI.h" // Bank Account
00025  #include "BOA.h" // Bank Account
00026  #include "SWBPLC.h" // Bank Account
00027  #include "ULSTER.h" // Bank Account
00028  #include "UNBL.h" // Bank Account
00029  #include "WAREHOUSE.h" // WAREHOUSE
00030  #include "CARPHONE_WAREHOUSE.h" // WAREHOUSE
00031  #include "CARLOW_W.h" // WAREHOUSE
00032  #include "KILKENNY_W.h" // WAREHOUSE
00033  #include "TALLAGH_W.h" // WAREHOUSE
00034  #include "DUNDALK_W.h" // WAREHOUSE
00035  #include "SLIGO_W.h" // WAREHOUSE
00036  #include <mutex>
00037  #include <memory>
00038  #include <condition_variable>
00039  #include <vector>
00040
00041
00042  static int vector_number = 600;
00043
00044  void _six_account_transfer(std::shared_ptr<OSTM> _to_, std::shared_ptr<OSTM>
    _from_one_, std::shared_ptr<OSTM> _from_two_, std::shared_ptr<OSTM> _from_three_, std::shared_ptr<OSTM> _from_four_
    , std::shared_ptr<OSTM> _from_five_, TM& _tm, double _amount) {
00045
00046      std::shared_ptr<TX> tx = _tm.get_tx();
00047      tx->register(_to_);
00048      tx->register(_from_one_);
00049      tx->register(_from_two_);
00050      tx->register(_from_three_);
00051      tx->register(_from_four_);
00052      tx->register(_from_five_);
00053
00054      std::shared_ptr<OSTM> _TO_OSTM, _FROM_ONE_OSTM, _FROM_TWO_OSTM, _FROM_THREE_OSTM, _FROM_FOUR_OSTM,
    _FROM_FIVE_OSTM;
00055      std::shared_ptr<BANK> _TO_, _FROM_ONE_, _FROM_TWO_, _FROM_THREE_, _FROM_FOUR_, _FROM_FIVE_;
00056      try {
00057          bool done = false;
00058          while (!done) {
00059              _TO_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
00060              _FROM_ONE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_one_));
00061              _FROM_TWO_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_two_));
00062              _FROM_THREE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_three_));
00063              _FROM_FOUR_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_four_));
00064              _FROM_FIVE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_five_));
00065              _TO_->SetBalance(_TO_->GetBalance() + (_amount * 5));
00066              _FROM_ONE_->SetBalance(_FROM_ONE_->GetBalance() - _amount);
00067              _FROM_TWO_->SetBalance(_FROM_TWO_->GetBalance() - _amount);
00068              _FROM_THREE_->SetBalance(_FROM_THREE_->GetBalance() - _amount);
00069              _FROM_FOUR_->SetBalance(_FROM_FOUR_->GetBalance() - _amount);
00070              _FROM_FIVE_->SetBalance(_FROM_FIVE_->GetBalance() - _amount);
00071              _TO_OSTM = std::dynamic_pointer_cast<OSTM> (_TO_);
00072              _FROM_ONE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_ONE_);
00073              _FROM_TWO_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_TWO_);
00074              _FROM_THREE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_THREE_);
00075              _FROM_FOUR_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_FOUR_);
00076              _FROM_FIVE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_FIVE_);
00077              tx->store(_TO_OSTM);
00078              tx->store(_FROM_ONE_OSTM);
00079              tx->store(_FROM_TWO_OSTM);
00080              tx->store(_FROM_THREE_OSTM);
00081              tx->store(_FROM_FOUR_OSTM);
00082              tx->store(_FROM_FIVE_OSTM);
00083              done = tx->commit();
00084          }
00085      } catch (std::runtime_error& e) {
00086          std::cout << e.what() << std::endl;
00087      }
00088  }

```

```

00116     }
00117 }
00118
00125 void _two_account_transfer(std::shared_ptr<OSTM> _to_, std::shared_ptr<OSTM> _from_,
    TM& _tm, double _amount) {
00126     std::shared_ptr<TX> tx = _tm._get_tx();
00130     tx->_register(_to_);
00131     tx->_register(_from_);
00135     std::shared_ptr<BANK> _TO_BANK_, _FROM_BANK_;
00136     std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;
00137
00138     bool done = false;
00139     try {
00140         while (!done) {
00144             _TO_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
00145             _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
00149             _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
00150             _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
00154             _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
00155             _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
00159             tx->store(_TO_OSTM_);
00160             tx->store(_FROM_OSTM_);
00161
00165             std::shared_ptr<TX> txTwo = _tm._get_tx();
00166
00167             bool nestedDone = false;
00168             while (!nestedDone) {
00169                 _TO_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_to_));
00170                 _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_from_));
00174                 _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
00175                 _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
00179                 _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
00180                 _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
00184                 txTwo->store(_TO_OSTM_);
00185                 txTwo->store(_FROM_OSTM_);
00189                 nestedDone = txTwo->commit();
00190             }
00194             done = tx->commit();
00195         }
00196     } catch (std::runtime_error& e) {
00197         std::cout << e.what() << std::endl;
00198     }
00199 }
00200
00208 void _nesting_(std::shared_ptr<OSTM> _to_, std::shared_ptr<OSTM> _from_, TM& _tm, double _amount)
    {
00209     std::shared_ptr<TX> tx = _tm._get_tx();
00213     tx->_register(_to_);
00214     tx->_register(_from_);
00218     std::shared_ptr<BANK> _TO_BANK_, _FROM_BANK_;
00219     std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;
00220
00221
00222     bool done = false;
00223     try {
00224         while (!done) {
00228             _TO_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
00229             _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
00233             _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
00234             _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
00238             _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
00239             _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
00243             tx->store(_TO_OSTM_);
00244             tx->store(_FROM_OSTM_);
00245
00249             std::shared_ptr<TX> txTwo = _tm._get_tx();
00250
00251             bool nestedDone = false;
00252             while (!nestedDone) {
00253                 _TO_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_to_));
00254                 _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_from_));
00258                 _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
00259                 _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
00263                 _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
00264                 _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
00268                 txTwo->store(_TO_OSTM_);
00269                 txTwo->store(_FROM_OSTM_);
00274                 _two_account_transfer(_to_, _from_, _tm, _amount);
00275
00276                 nestedDone = txTwo->commit();
00277             }
00278
00282             done = tx->commit();
00283         }
00284     } catch (std::runtime_error& e) {
00285         std::cout << e.what() << std::endl;
00286     }

```

```

00287 }
00288
00296 void _complex_transfer_(std::shared_ptr<OSTM> _from_, std::shared_ptr<OSTM> _from_two_,
std::vector<std::shared_ptr<OSTM>> _customer_vec, TM& _tm, double _amount) {
00297     std::shared_ptr<TX> tx = _tm._get_tx();
00301     tx->_register(_from_);
00302     tx->_register(_from_two_);
00306     std::shared_ptr<OSTM> _FROM_OSTM_ONE_, _FROM_OSTM_TWO_, _TO_OSTM_;
00307     std::shared_ptr<BANK> _FROM_, _FROM_TWO_, _TO_;
00308
00309     bool done = false;
00310     try {
00311         while (!done) {
00312             // for (int i = 0; i < vector_number; ++i) {
00313                 for (auto&& obj : _customer_vec) {
00317                     // auto&& obj = _customer_vec.at(i);
00318                     tx->_register(obj);
00322                     _FROM_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
00323                     _FROM_TWO_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_two_));
00324                     _TO_ = std::dynamic_pointer_cast<BANK> (tx->load(obj));
00328                     _FROM_->SetBalance(_FROM_->GetBalance() - _amount);
00329                     _FROM_TWO_->SetBalance(_FROM_TWO_->GetBalance() - _amount);
00330                     _TO_->SetBalance(_TO_->GetBalance() + (_amount * 2));
00334                     _FROM_OSTM_ONE_ = std::dynamic_pointer_cast<OSTM> (_FROM_);
00335                     _FROM_OSTM_TWO_ = std::dynamic_pointer_cast<OSTM> (_FROM_TWO_);
00336                     _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_);
00340                     tx->store(_FROM_OSTM_ONE_);
00341                     tx->store(_FROM_OSTM_TWO_);
00342                     tx->store(_TO_OSTM_);
00343                 }
00347                 done = tx->commit();
00348             }
00349         } catch (std::runtime_error& e) {
00350             std::cout << e.what() << std::endl;
00351         }
00352     }
00353
00360 void _warehouse_transfer_(std::shared_ptr<OSTM> _to_, std::shared_ptr<OSTM> _from_, TM&
_tm, double _amount) {
00361     std::shared_ptr<TX> tx = _tm._get_tx();
00365     tx->_register(_to_);
00366     tx->_register(_from_);
00370     std::shared_ptr<WAREHOUSE> _TO_SHOP_, _FROM_DIST_;
00371     std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;
00372
00373     bool done = false;
00374     try {
00375         while (!done) {
00379             _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
00380             _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));
00384             _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
00385             _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);
00386
00387             _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
00388             _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);
00389
00390             _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
00391             _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);
00392
00393             _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
00394             _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
00398             _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00399             _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00403             tx->store(_TO_OSTM_);
00404             tx->store(_FROM_OSTM_);
00408             done = tx->commit();
00409         }
00410     } catch (std::runtime_error& e) {
00411         std::cout << e.what() << std::endl;
00412     }
00413 }
00414
00421 void _nested_warehouse_transfer_(std::shared_ptr<OSTM> _to_,
std::shared_ptr<OSTM> _to_two, std::shared_ptr<OSTM> _to_three, std::shared_ptr<OSTM> _from_, TM& _tm, double _amount)
00422 {
00426     std::shared_ptr<TX> tx = _tm._get_tx();
00427     tx->_register(_to_);
00428     tx->_register(_to_two);
00429     tx->_register(_to_three);
00433     std::shared_ptr<WAREHOUSE> _TO_SHOP_, _FROM_DIST_;
00434     std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;
00435
00436     bool done = false;
00437     try {
00438         while (!done) {
00442             _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
00443             _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));

```

```

00447     _TO_SHOP_>SetNumber_of_nokia(_TO_SHOP_>GetNumber_of_nokia() + _amount);
00448     _FROM_DIST_>SetNumber_of_nokia(_FROM_DIST_>GetNumber_of_nokia() - _amount);
00449
00450     _TO_SHOP_>SetNumber_of_samsung(_TO_SHOP_>GetNumber_of_samsung() + _amount);
00451     _FROM_DIST_>SetNumber_of_samsung(_FROM_DIST_>GetNumber_of_samsung() - _amount);
00452
00453     _TO_SHOP_>SetNumber_of_iphones(_TO_SHOP_>GetNumber_of_iphones() + _amount);
00454     _FROM_DIST_>SetNumber_of_iphones(_FROM_DIST_>GetNumber_of_iphones() - _amount);
00455
00456     _TO_SHOP_>SetNumber_of_sony(_TO_SHOP_>GetNumber_of_sony() + _amount);
00457     _FROM_DIST_>SetNumber_of_sony(_FROM_DIST_>GetNumber_of_sony() - _amount);
00461     _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00462     _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00466     tx->store(_TO_OSTM_);
00467     tx->store(_FROM_OSTM_);
00468
00472     std::shared_ptr<TX> txTwo = _tm._get_tx();
00473     bool nestedDone = false;
00474     while (!nestedDone) {
00475         _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_to_two));
00476         _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_from));
00480         _TO_SHOP_>SetNumber_of_nokia(_TO_SHOP_>GetNumber_of_nokia() + _amount);
00481         _FROM_DIST_>SetNumber_of_nokia(_FROM_DIST_>GetNumber_of_nokia() - _amount);
00482
00483         _TO_SHOP_>SetNumber_of_samsung(_TO_SHOP_>GetNumber_of_samsung() + _amount);
00484         _FROM_DIST_>SetNumber_of_samsung(_FROM_DIST_>GetNumber_of_samsung() - _amount);
00485
00486         _TO_SHOP_>SetNumber_of_iphones(_TO_SHOP_>GetNumber_of_iphones() + _amount);
00487         _FROM_DIST_>SetNumber_of_iphones(_FROM_DIST_>GetNumber_of_iphones() - _amount);
00488
00489         _TO_SHOP_>SetNumber_of_sony(_TO_SHOP_>GetNumber_of_sony() + _amount);
00490         _FROM_DIST_>SetNumber_of_sony(_FROM_DIST_>GetNumber_of_sony() - _amount);
00494         _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00495         _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00499         txTwo->store(_TO_OSTM_);
00500         txTwo->store(_FROM_OSTM_);
00501
00502         /*
00503          * NESTED TRANSACTION TEST _to_three
00504          */
00505         _warehouse_transfer_(_to_three, _from_, _tm, _amount);
00506
00507         nestedDone = tx->commit();
00508     }
00509     done = tx->commit();
00513 }
00514 }
00515 } catch (std::runtime_error& e) {
00516     std::cout << e.what() << std::endl;
00517 }
00518 }
00519
00520 void _complex_warehouse_transfer_(std::shared_ptr<OSTM> _to_,
std::shared_ptr<OSTM> _to_two, std::shared_ptr<OSTM> _to_three, std::vector<std::shared_ptr<OSTM>> _warehouse_vec,
std::shared_ptr<OSTM> _from_, TM& _tm, double _amount) {
00521     std::shared_ptr<TX> tx = _tm._get_tx();
00525     tx->_register(_to_);
00526     tx->_register(_to_two);
00527     tx->_register(_to_three);
00528     tx->_register(_from_);
00532     std::shared_ptr<WAREHOUSE> _TO_SHOP_, _TO_SHOP_TWO_, _TO_SHOP_VEC_, _FROM_DIST_;
00533     std::shared_ptr<OSTM> _TO_OSTM_, _TO_OSTM_TWO_, _TO_OSTM_VEC_, _FROM_OSTM_;
00534
00535     bool done = false;
00536     try {
00537         while (!done) {
00538
00539             // for (int i = 0; i < vector_number; ++i) {
00540             for (auto&& obj : _warehouse_vec) {
00544                 //auto&& obj = _warehouse_vec.at(i);
00545                 tx->_register(obj);
00549                 _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
00550                 _TO_SHOP_TWO_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_two));
00551                 _TO_SHOP_VEC_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(obj));
00552                 _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from));
00553
00557                 _TO_SHOP_>SetNumber_of_nokia(_TO_SHOP_>GetNumber_of_nokia() + _amount);
00558                 _TO_SHOP_TWO_>SetNumber_of_nokia(_TO_SHOP_TWO_>GetNumber_of_nokia() + _amount);
00559                 _TO_SHOP_VEC_>SetNumber_of_nokia(_TO_SHOP_VEC_>GetNumber_of_nokia() + _amount);
00560                 _FROM_DIST_>SetNumber_of_nokia(_FROM_DIST_>GetNumber_of_nokia() - (_amount * 3));
00561
00562                 _TO_SHOP_>SetNumber_of_samsung(_TO_SHOP_>GetNumber_of_samsung() + _amount);
00563                 _TO_SHOP_TWO_>SetNumber_of_samsung(_TO_SHOP_TWO_>GetNumber_of_samsung() + _amount);
00564                 _TO_SHOP_VEC_>SetNumber_of_samsung(_TO_SHOP_VEC_>GetNumber_of_samsung() + _amount);
00565                 _FROM_DIST_>SetNumber_of_samsung(_FROM_DIST_>GetNumber_of_samsung() - (_amount * 3));
00566
00567                 _TO_SHOP_>SetNumber_of_iphones(_TO_SHOP_>GetNumber_of_iphones() + _amount);

```

```

00568         _TO_SHOP_TWO->SetNumber_of_iphones(_TO_SHOP_TWO->GetNumber_of_iphones() + _amount);
00569         _TO_SHOP_VEC->SetNumber_of_iphones(_TO_SHOP_VEC->GetNumber_of_iphones() + _amount);
00570         _FROM_DIST->SetNumber_of_iphones(_FROM_DIST->GetNumber_of_iphones() - (_amount * 3));
00571
00572         _TO_SHOP->SetNumber_of_sony(_TO_SHOP->GetNumber_of_sony() + _amount);
00573         _TO_SHOP_TWO->SetNumber_of_sony(_TO_SHOP_TWO->GetNumber_of_sony() + _amount);
00574         _TO_SHOP_VEC->SetNumber_of_sony(_TO_SHOP_VEC->GetNumber_of_sony() + _amount);
00575         _FROM_DIST->SetNumber_of_sony(_FROM_DIST->GetNumber_of_sony() - (_amount * 3));
00576
00580         _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00581         _TO_OSTM_TWO = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_TWO);
00582         _TO_OSTM_VEC = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_VEC);
00583         _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00587         tx->store(_TO_OSTM_);
00588         tx->store(_TO_SHOP_TWO);
00589         tx->store(_TO_SHOP_VEC);
00590         tx->store(_FROM_OSTM_);
00591
00592
00593
00594     }
00598     std::shared_ptr<TX> txTwo = _tm._get_tx();
00599     bool nestedDone = false;
00600     while (!nestedDone) {
00601         _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_to_two));
00602         _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_from));
00606         _TO_SHOP->SetNumber_of_nokia(_TO_SHOP->GetNumber_of_nokia() + _amount);
00607         _FROM_DIST->SetNumber_of_nokia(_FROM_DIST->GetNumber_of_nokia() - _amount);
00608
00609         _TO_SHOP->SetNumber_of_samsung(_TO_SHOP->GetNumber_of_samsung() + _amount);
00610         _FROM_DIST->SetNumber_of_samsung(_FROM_DIST->GetNumber_of_samsung() - _amount);
00611
00612         _TO_SHOP->SetNumber_of_iphones(_TO_SHOP->GetNumber_of_iphones() + _amount);
00613         _FROM_DIST->SetNumber_of_iphones(_FROM_DIST->GetNumber_of_iphones() - _amount);
00614
00615         _TO_SHOP->SetNumber_of_sony(_TO_SHOP->GetNumber_of_sony() + _amount);
00616         _FROM_DIST->SetNumber_of_sony(_FROM_DIST->GetNumber_of_sony() - _amount);
00620         _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
00621         _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
00625         txTwo->store(_TO_OSTM_);
00626         txTwo->store(_FROM_OSTM_);
00627
00628         /*
00629          * NESTED TRANSACTION TEST _to_three
00630          */
00631         _warehouse_transfer_(_to_three, _from_, _tm, _amount);
00632         _nested_warehouse_transfer_(_to_, _to_two, _to_three, _from_,
00633         _tm, _amount);
00634
00634         nestedDone = tx->commit();
00635     }
00636
00640     done = tx->commit();
00641
00642     }
00643     } catch (std::runtime_error& e) {
00644         std::cout << e.what() << std::endl;
00645     }
00646 }
00647
00651 int main(void) {
00652     TM& tm = TM::Instance();
00653
00654     std::vector<std::shared_ptr<OSTM>>_customer_vec; //(vector_number);
00655     std::vector<std::shared_ptr<OSTM>>_warehouse_vec; //(vector_number);
00656
00676     std::shared_ptr<OSTM> aib_ptr(new AIB(100, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny")
00677 );
00677     std::shared_ptr<OSTM> boi_ptr(new BOI(200, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny")
00678 );
00678     std::shared_ptr<OSTM> boa_ptr(new BOA(300, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny")
00679 );
00679     std::shared_ptr<OSTM> swplc_ptr(new SWBPLC(400, 500, "Joe", "Blog", "High street, Kilkenny,
00680 Co.Kilkenny"));
00680     std::shared_ptr<OSTM> ulster_ptr(new ULSTER(500, 500, "Joe", "Blog", "High street, Kilkenny,
00681 Co.Kilkenny"));
00681     std::shared_ptr<OSTM> unbl_ptr(new UNBL(600, 500, "Joe", "Blog", "High street, Kilkenny,
00682 Co.Kilkenny"));
00682
00693     std::shared_ptr<OSTM> w_dist(new CARPHONE_WAREHOUSE());
00694     std::shared_ptr<OSTM> c_shop(new CARLOW_W());
00695     std::shared_ptr<OSTM> k_shop(new KILKENNY_W());
00696     std::shared_ptr<OSTM> t_shop(new TALLAGH_W());
00697     std::shared_ptr<OSTM> d_shop(new DUNDALK_W());
00698     std::shared_ptr<OSTM> s_shop(new SLIGO_W());
00699
00705     for (int i = 0; i < vector_number; ++i) {

```



```

00706         if (i % 5 == 0) {
00707             std::shared_ptr<OSTM> sharedptr(new CARLOW_W());
00708             _warehouse_vec.push_back(std::move(sharedptr));
00709         } else if (i % 4 == 0) {
00710             std::shared_ptr<OSTM> sharedptr(new KILKENNY_W());
00711             _warehouse_vec.push_back(std::move(sharedptr));
00712         } else if (i % 3 == 0) {
00713             std::shared_ptr<OSTM> sharedptr(new TALLAGH_W());
00714             _warehouse_vec.push_back(std::move(sharedptr));
00715         } else if (i % 2 == 0) {
00716             std::shared_ptr<OSTM> sharedptr(new DUNDALK_W());
00717             _warehouse_vec.push_back(std::move(sharedptr));
00718         } else if (i % 1 == 0) {
00719             std::shared_ptr<OSTM> sharedptr(new SLIGO_W());
00720             _warehouse_vec.push_back(std::move(sharedptr));
00721         }
00722     }
00723
00729     for (int i = 0; i < vector_number; ++i) {
00730         if (i % 6 == 0) {
00731             std::shared_ptr<OSTM> sharedptr(new AIB(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00732             _customer_vec.push_back(std::move(sharedptr));
00733         } else if (i % 5 == 0) {
00734             std::shared_ptr<OSTM> sharedptr(new BOI(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00735             _customer_vec.push_back(std::move(sharedptr));
00736         } else if (i % 4 == 0) {
00737             std::shared_ptr<OSTM> sharedptr(new BOA(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00738             _customer_vec.push_back(std::move(sharedptr));
00739         } else if (i % 3 == 0) {
00740             std::shared_ptr<OSTM> sharedptr(new SWBPLC(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00741             _customer_vec.push_back(std::move(sharedptr));
00742         } else if (i % 2 == 0) {
00743             std::shared_ptr<OSTM> sharedptr(new ULSTER(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00744             _customer_vec.push_back(std::move(sharedptr));
00745         } else if (i % 1 == 0) {
00746             std::shared_ptr<OSTM> sharedptr(new UNBL(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
00747             _customer_vec.push_back(std::move(sharedptr));
00748         }
00749     }
00750
00760     // w_dist->toString();
00761     // c_shop->toString();
00762     // k_shop->toString();
00763     // t_shop->toString();
00764     // d_shop->toString();
00765     // s_shop->toString();
00766
00777     /*
00778     * TEST 1 : object requirements
00779     */
00780     aib_ptr->toString();
00781     boi_ptr->toString();
00782     boa_ptr->toString();
00783     swplc_ptr->toString();
00784     ulster_ptr->toString();
00785     unbl_ptr->toString();
00786
00787     /*
00788     * TEST 2 : object requirements
00789     */
00790     // aib_ptr->toString();
00791     // boi_ptr->toString();
00792     // boa_ptr->toString();
00793     // swplc_ptr->toString();
00794     // ulster_ptr->toString();
00795     // unbl_ptr->toString();
00796     // for(int i=0; i<vector_number; ++i){
00797     //     _customer_vec[i]->toString();
00798     // }
00799
00800     /*
00801     * TEST 3 : object requirements
00802     */
00803     // w_dist->toString();
00804     // c_shop->toString();
00805     // k_shop->toString();
00806     // t_shop->toString();
00807
00808     /*
00809     * TEST 4 : objects requirements
00810     */

```

```

00811 //      w_dist->toString();
00812 //      c_shop->toString();
00813 //      k_shop->toString();
00814 //      t_shop->toString();
00815 //      d_shop->toString();
00816 //      s_shop->toString();
00817
00818
00819 /*
00820  * TEST 5 : objects requirements
00821  */
00822 //      w_dist->toString();
00823 //      c_shop->toString();
00824 //      k_shop->toString();
00825 //      t_shop->toString();
00826 //      d_shop->toString();
00827 //      s_shop->toString();
00828
00829 //      for(auto&& elem: _warehouse_vec){
00830 //          elem->toString(); // virtual dispatch
00831 //      }
00832
00833
00834
00835
00839 int transferAmount = 1;
00846 int threadArraySize = 300;
00847
00848 std::thread thArray[300];
00849
00854 for (int i = 0; i < threadArraySize; ++i) {
00855
00860 //thArray[i] = std::thread(_nesting_, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
00861 if (i % 3 == 0)
00862     thArray[i] = std::thread(_nesting_, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
00863 else if (i % 2 == 0)
00864     thArray[i] = std::thread(_nesting_, boa_ptr, swplc_ptr, std::ref(tm), transferAmount);
00865 else if (i % 1 == 0)
00866     thArray[i] = std::thread(_nesting_, ulster_ptr, unbl_ptr, std::ref(tm), transferAmount);
00867
00874 //      if (i % 3 == 0)
00875 //          thArray[i] = std::thread(_two_account_transfer_, aib_ptr, boi_ptr, std::ref(tm),
transferAmount);
00876 //      else if (i % 2 == 0)
00877 //          thArray[i] = std::thread(_six_account_transfer_, boi_ptr, boa_ptr, swplc_ptr, ulster_ptr,
aib_ptr, unbl_ptr, std::ref(tm), transferAmount);
00878 //      else if (i % 1 == 0)
00879 //          thArray[i] = std::thread(_complex_transfer_, aib_ptr, boi_ptr, std::ref(_customer_vec),
std::ref(tm), transferAmount);
00880
00881
00886 //      if (i % 3 == 0)
00887 //          thArray[i] = std::thread(_warehouse_transfer_, c_shop, w_dist, std::ref(tm),
transferAmount);
00888 //      else if (i % 2 == 0)
00889 //          thArray[i] = std::thread(_warehouse_transfer_, k_shop, w_dist, std::ref(tm),
transferAmount);
00890 //      else if (i % 1 == 0)
00891 //          thArray[i] = std::thread(_warehouse_transfer_, t_shop, w_dist, std::ref(tm),
transferAmount);
00892
00897 //      if (i % 3 == 0)
00898 //          thArray[i] = std::thread(_nested_warehouse_transfer_, c_shop, d_shop, k_shop, w_dist,
std::ref(tm), transferAmount);
00899 //      else if (i % 2 == 0)
00900 //          thArray[i] = std::thread(_nested_warehouse_transfer_, k_shop, s_shop, t_shop, w_dist,
std::ref(tm), transferAmount);
00901 //      else if (i % 1 == 0)
00902 //          thArray[i] = std::thread(_nested_warehouse_transfer_, t_shop, c_shop, s_shop, w_dist,
std::ref(tm), transferAmount);
00903
00912 //      if (i % 3 == 0)
00913 //          thArray[i] = std::thread(_warehouse_transfer_, c_shop, w_dist, std::ref(tm),
transferAmount);
00914 //      else if (i % 2 == 0)
00915 //          thArray[i] = std::thread(_nested_warehouse_transfer_, k_shop, s_shop, t_shop, w_dist,
std::ref(tm), transferAmount);
00916 //      else if (i % 1 == 0)
00917 //          thArray[i] = std::thread(_complex_warehouse_transfer_, d_shop, s_shop, c_shop,
std::ref(_warehouse_vec), w_dist, std::ref(tm), transferAmount);
00918
00919
00920 }
00921 /*
00922  * Join threads^n -> threadArraySize<br>
00923  * thArray[i].join();
00924  */

```

```

00925     for (int i = 0; i < threadArraySize; ++i) {
00926         thArray[i].join();
00927     }
00928
00929
00930     std::cout << "\nMain process print " << std::endl;
00931     /*
00932      * TEST 1 : object requirements
00933      */
00934     aib_ptr->toString();
00935     boi_ptr->toString();
00936     boa_ptr->toString();
00937     swplc_ptr->toString();
00938     ulster_ptr->toString();
00939     unbl_ptr->toString();
00940
00941     /*
00942      * TEST 2 : object requirements
00943      */
00944     // aib_ptr->toString();
00945     // boi_ptr->toString();
00946     // boa_ptr->toString();
00947     // swplc_ptr->toString();
00948     // ulster_ptr->toString();
00949     // unbl_ptr->toString();
00950     // for(int i=0; i<vector_number; ++i){
00951     //     _customer_vec[i]->toString();
00952     // }
00953
00954     /*
00955      * TEST 3 : object requirements
00956      */
00957     // w_dist->toString();
00958     // c_shop->toString();
00959     // k_shop->toString();
00960     // t_shop->toString();
00961
00962     /*
00963      * TEST 4 : objects requirements
00964      */
00965     // w_dist->toString();
00966     // c_shop->toString();
00967     // k_shop->toString();
00968     // t_shop->toString();
00969     // d_shop->toString();
00970     // s_shop->toString();
00971
00972     /*
00973      * TEST 5 : objects requirements
00974      */
00975     // w_dist->toString();
00976     // c_shop->toString();
00977     // k_shop->toString();
00978     // t_shop->toString();
00979     // d_shop->toString();
00980     // s_shop->toString();
00981
00982     // for(auto&& elem: _warehouse_vec){
00983     //     elem->toString(); // virtual dispatch
00984     // }
00985
00986     /* TEST 5 FINISH */
00987
00988     std::cout << "\nMAIN PROCESS EXIT !!!! " << std::endl;
00989     std::shared_ptr<TX> tx = tm._get_tx();
00990
00991
00992     std::cout << "Rollback counter is : " << tx->getTest_counter() << std::endl;
00993     // std::cout << "[vector_number]" << std::endl;
00994     // for (int i = 0; i < vector_number; ++i) {
00995     //     _customer_vec[i]->toString();
00996     //     auto&& os = _customer_vec.at(i);
00997     //     os->toString();
00998     // }
00999     // std::cout << "[_warehouse_vec]" << std::endl;
01000     // for(auto&& elem: _warehouse_vec){
01001     //     elem->toString(); // virtual dispatch
01002     // }
01003     // _customer_vec[10]->toString();
01004
01005     tm._TX_EXIT();
01006     std::cout << "\nPRINT ALL FROM TM !!!! SHOULD BE EMPTY AFTER _TX_EXIT() !" << std::endl;
01007     tm.print_all();
01008     int t = 0;
01009     std::cin >> t;

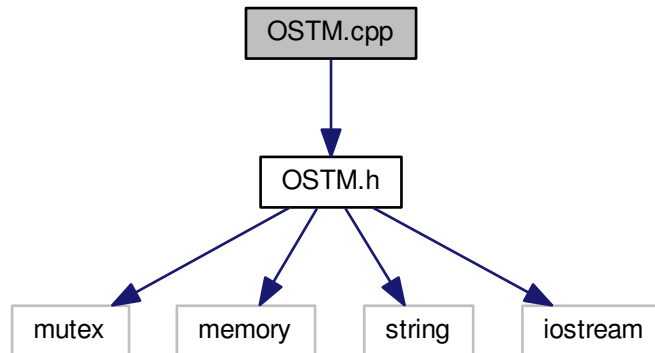
```

```
01036     return 0;
01037 }
```

7.35 OSTM.cpp File Reference

```
#include "OSTM.h"
```

Include dependency graph for OSTM.cpp:



7.36 OSTM.cpp

```

00001 /*
00002  * File:   OSTM.cpp
00003  * Author: Zoltan Fuzesi
00004  *
00005  * Created on December 18, 2017, 2:09 PM
00006  * OSTM cpp file methods implementations
00007  */
00008
00009 #include "OSTM.h"
00010
00011 int OSTM::global_Unique_ID_Number = 0;
00012
00020 OSTM::OSTM()
00021 {
00022     this->version = ZERO;
00023     this->uniqueID = Get_global_Unique_ID_Number(); //++global_Unique_ID_Number;
00024     this->canCommit = true;
00025     this->abort_Transaction = false;
00026 }
00027
00028
00036 OSTM::OSTM(int _version_number_, int _unique_id_)
00037 {
00038     // std::cout << "OSTM COPY CONSTRUCTOR" << global_Unique_ID_Number << std::endl;
00039     this->uniqueID = _unique_id_;
00040     this->version = _version_number_;
00041     this->canCommit = true;
00042     this->abort_Transaction = false;
00043 }
00044
00048 OSTM::~OSTM() {
00049     //std::cout << "[OSTM DELETE]" << std::endl;
00050 }
00056 int OSTM::Get_global_Unique_ID_Number() {
00057     if(global_Unique_ID_Number > 10000000)
00058         global_Unique_ID_Number = 0;
00059     return ++global_Unique_ID_Number;
00060 }
```

```

00061
00066 void OSTM::Set_Unique_ID(int uniqueID) {
00067     this->uniqueID = uniqueID;
00068 }
00073 int OSTM::Get_Unique_ID() const
00074 {
00075     return uniqueID;
00076 }
00081 void OSTM::Set_Version(int version)
00082 {
00083     this->version = version;
00084 }
00089 int OSTM::Get_Version() const
00090 {
00091     return version;
00092 }
00097 void OSTM::increase_VersionNumber()
00098 {
00099     this->version += 1;
00100 }
00105 void OSTM::Set_Can_Commit(bool canCommit) {
00106     this->canCommit = canCommit;
00107 }
00112 bool OSTM::Is_Can_Commit() const {
00113     return canCommit;
00114 }
00119 void OSTM::Set_Abort_Transaction(bool abortTransaction) {
00120     this->abort_Transaction = abortTransaction;
00121 }
00126 bool OSTM::Is_Abort_Transaction() const {
00127     return abort_Transaction;
00128 }
00133 void OSTM::lock_Mutex() {
00134     this->mutex.lock();
00135 }
00140 void OSTM::unlock_Mutex() {
00141     this->mutex.unlock();
00142 }
00147 bool OSTM::is_Locked(){
00148     return this->mutex.try_lock();
00149 }

```

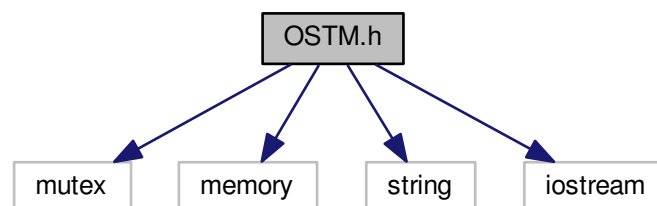
7.37 OSTM.h File Reference

```

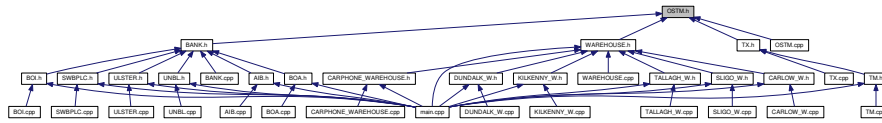
#include <mutex>
#include <memory>
#include <string>
#include <iostream>

```

Include dependency graph for OSTM.h:



This graph shows which files directly or indirectly include this file:



Functions

- class `__declspec (dllexport)` OSTM

7.37.1 Function Documentation

7.37.1.1 class `__declspec (dllexport)`

OSTM Constructor

OSTM Custom Constructor

De-constructor

OSTM required virtual method for deep copy

OSTM required virtual method for returning a pointer that is copy of the original pointer

OSTM required virtual method for display object

setter for unique id

getter for unique id

setter for version number

getter for version number

commit time increase version number to child object

NOT USED YET

NOT USED YET

NOT USED YET

NOT USED YET

object unique lock, locks mutex

object unique lock, unlocks mutex

object unique lock, try locks mutex return boolean value depends on the lock state

Unique object number increase at object creation

Meaningful display for value 0

Object built in lock

Returning global_Unique_ID_Number to the constructor

Definition at line 17 of file [OSTM.h](#).

```

00017     {
00018 public:
00022     OSTM();
00026     OSTM(int _version_number_, int _unique_id_);
00030     virtual ~OSTM();
00034     virtual void copy(std::shared_ptr<OSTM> from, std::shared_ptr<OSTM> to){};
00038     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object) = 0; //std::cout << "[OSTM
GETBASECOPY]" << std::endl;};
00042     virtual void toString(){};
00046     void Set_Unique_ID(int uniqueID);
00050     int Get_Unique_ID() const;
00054     void Set_Version(int version);
00058     int Get_Version() const;
00062     void increase_VersionNumber();
00066     bool Is_Can_Commit() const;
00070     void Set_Can_Commit(bool canCommit);
00074     void Set_Abort_Transaction(bool abortTransaction);
00078     bool Is_Abort_Transaction() const;
00082     void lock_Mutex();
00086     void unlock_Mutex();
00090     bool is_Locked();
00091
00092 private:
00093     /*
00094     * \brief Object version number
00095     */
00096     int version;
00097     /*
00098     * \brief Object unique identifier
00099     */
00100     int uniqueID;
00101     /*
00102     * \brief Boolean value to check any other thread failed to commit
00103     */
00104     bool canCommit;
00105     /*
00106     * \brief Abort the transaction
00107     */
00108     bool abort_Transaction;
00112     static int global_Unique_ID_Number;
00116     const int ZERO = 0;
00120     std::mutex mutex;
00124     int Get_global_Unique_ID_Number();
00125
00126 };

```

7.38 OSTM.h

```

00001 /*
00002  * File:   OSTM.h
00003  * Author: Zoltan FÜzesi
00004  *
00005  * Created on December 18, 2017, 2:09 PM
00006  * OSTM header file fields and methods declarations
00007  */
00008
00009 #ifndef OSTM_H
00010 #define OSTM_H
00011 #include <mutex>
00012 #include <memory>
00013 #include <string>
00014 #include <iostream>
00015 #include <string>
00016
00017 class __declspec(dllexport) OSTM {
00018 public:
00022     OSTM();
00026     OSTM(int _version_number_, int _unique_id_);
00030     virtual ~OSTM();
00034     virtual void copy(std::shared_ptr<OSTM> from, std::shared_ptr<OSTM> to){};
00038     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object) = 0; //std::cout << "[OSTM
GETBASECOPY]" << std::endl;};
00042     virtual void toString(){};
00046     void Set_Unique_ID(int uniqueID);
00050     int Get_Unique_ID() const;
00054     void Set_Version(int version);
00058     int Get_Version() const;
00062     void increase_VersionNumber();
00066     bool Is_Can_Commit() const;
00070     void Set_Can_Commit(bool canCommit);
00074     void Set_Abort_Transaction(bool abortTransaction);
00078     bool Is_Abort_Transaction() const;
00082     void lock_Mutex();

```

```

00086     void unlock_Mutex();
00090     bool is_Locked();
00091
00092 private:
00093     /*
00094      * \brief Object version number
00095      */
00096     int version;
00097     /*
00098      * \brief Object unique identifier
00099      */
00100     int uniqueID;
00101     /*
00102      * \brief Boolean value to check any other thread failed to commit
00103      */
00104     bool canCommit;
00105     /*
00106      * \brief Abort the transaction
00107      */
00108     bool abort_Transaction;
00112     static int global_Unique_ID_Number;
00116     const int ZERO = 0;
00120     std::mutex mutex;
00124     int Get_global_Unique_ID_Number();
00125
00126 };
00127
00128 #endif /* OSTM_H */

```

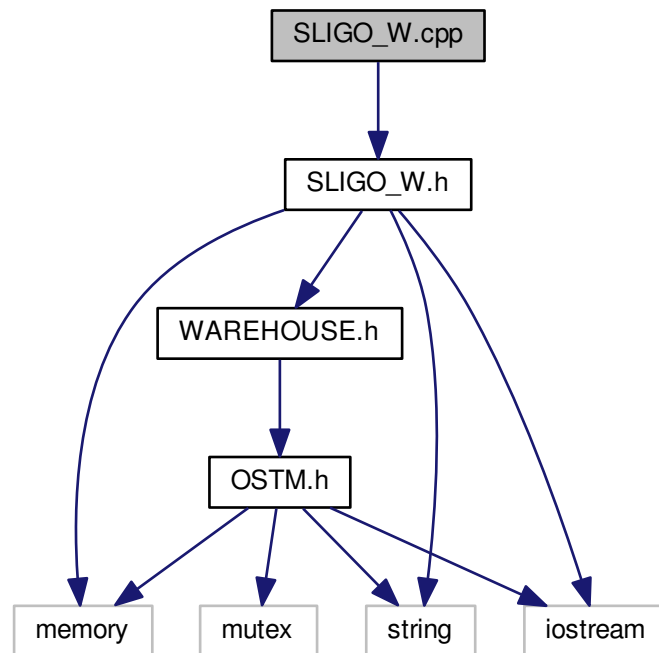
7.39 README.md File Reference

7.40 README.md

```

00001 C++ Software Transactional Memory (STM)
00002
00003 This documentation includes all the project specific files that required to build the STM library and
00004 the
00005 client code to use the library. The client code is demonstrate the usage of the STM API (Application
00006 Programming Interface).
00007 The STM library is a object based implementation, where the client need to inherit from the library
00008 on order to
00009 achieve the polymorphic Object Oriented Programming (OOP) behaviour.
00010
00011 The client application use a middle class to declare the child (Classes inherit from BANK) specific
00012 behaviour as a virtual methods.
00013 Whit this implementation the client application need to casting back the OSTM object to BANK object to
00014 use the child class
00015 implemented specific behaviours.
00016
00017
00018
00019
00020
00021
00022
00023
00024
00025
00026
00027
00028
00029
00030
00031
00032
00033
00034
00035
00036
00037
00038
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054
00055
00056
00057
00058
00059
00060
00061
00062
00063
00064
00065
00066
00067
00068
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
```


Include dependency graph for SLIGO_W.cpp:



7.42 SLIGO_W.cpp

```

00001
00002 /*
00003  * File:    SLIGO_W.cpp
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #include "SLIGO_W.h"
00011
00012 SLIGO_W::~SLIGO_W() {
00013 }
00014
00015 SLIGO_W::SLIGO_W(const SLIGO_W& orig) {
00016 }
00022 std::shared_ptr<OSTM> SLIGO_W::getBaseCopy(std::shared_ptr<OSTM> object)
00023 {
00024
00025     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00026     std::shared_ptr<WAREHOUSE> obj(new SLIGO_W(objTO, object->Get_Version(), object->Get_Unique_ID()));
00027
00028     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00029
00030     return ostm_obj;
00031 }
00035 void SLIGO_W::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00036
00037     std::shared_ptr<SLIGO_W> objTO = std::dynamic_pointer_cast<SLIGO_W>(to);
00038     std::shared_ptr<SLIGO_W> objFROM = std::dynamic_pointer_cast<SLIGO_W>(from);
00039     objTO->_shop_address = objFROM->GetShop_address();
00040     objTO->_shop_name = objFROM->GetShop_name();
00041     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00042     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00043     objTO->_number_of_sony = objFROM->GetNumber_of_sony();

```

```

00044     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00045     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00046     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00047     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00048     objTO->Set_Version(objFROM->Get_Version());
00049
00050
00051 }
00052 //std::shared_ptr<SLIGO_W> SLIGO_W::_cast(std::shared_ptr<OSTM> _object) {
00053 //
00054 //     return static_cast<std::shared_ptr<SLIGO_W>>(_object);
00055 //}
00062 void SLIGO_W::toString()
00063 {
00064     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() <<
"\nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
GetShop_address() << "\nNo. iPhones : " << this->
GetNumber_of_iphones() << "\nNo. Samsung : " << this->
GetNumber_of_samsung() << "\nNo. Sony : " << this->
GetNumber_of_sony() << "\nNo. Huawei : " << this->
GetNumber_of_huawei() << "\nNo. Nokia : " << this->
GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00065 }
00066
00067
00068
00069 void SLIGO_W::SetNumber_of_alcatel(int _number_of_alcatel) {
00070     this->_number_of_alcatel = _number_of_alcatel;
00071 }
00072
00073 int SLIGO_W::GetNumber_of_alcatel() {
00074     return _number_of_alcatel;
00075 }
00076
00077 void SLIGO_W::SetNumber_of_nokia(int _number_of_nokia) {
00078     this->_number_of_nokia = _number_of_nokia;
00079 }
00080
00081 int SLIGO_W::GetNumber_of_nokia() {
00082     return _number_of_nokia;
00083 }
00084
00085 void SLIGO_W::SetNumber_of_huawei(int _number_of_huawei) {
00086     this->_number_of_huawei = _number_of_huawei;
00087 }
00088
00089 int SLIGO_W::GetNumber_of_huawei() {
00090     return _number_of_huawei;
00091 }
00092
00093 void SLIGO_W::SetNumber_of_sony(int _number_of_sony) {
00094     this->_number_of_sony = _number_of_sony;
00095 }
00096
00097 int SLIGO_W::GetNumber_of_sony() {
00098     return _number_of_sony;
00099 }
00100
00101 void SLIGO_W::SetNumber_of_samsung(int _number_of_samsung) {
00102     this->_number_of_samsung = _number_of_samsung;
00103 }
00104
00105 int SLIGO_W::GetNumber_of_samsung() {
00106     return _number_of_samsung;
00107 }
00108
00109 void SLIGO_W::SetNumber_of_iphones(int _number_of_iphones) {
00110     this->_number_of_iphones = _number_of_iphones;
00111 }
00112
00113 int SLIGO_W::GetNumber_of_iphones() {
00114     return _number_of_iphones;
00115 }
00116
00117 void SLIGO_W::SetShop_name(std::string _shop_name) {
00118     this->_shop_name = _shop_name;
00119 }
00120
00121 std::string SLIGO_W::GetShop_name() {
00122     return _shop_name;
00123 }
00124
00125 void SLIGO_W::SetShop_address(std::string _shop_address) {
00126     this->_shop_address = _shop_address;
00127 }
00128

```

```

00129 std::string SLIGO_W::GetShop_address() {
00130     return _shop_address;
00131 }
00132
00133
00134

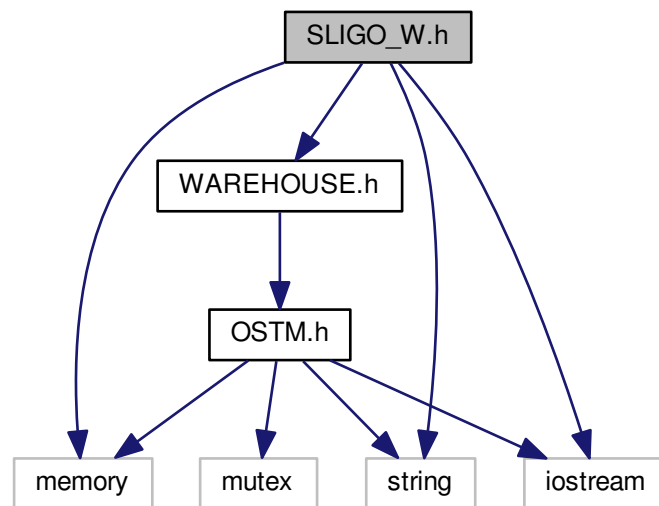
```

7.43 SLIGO_W.h File Reference

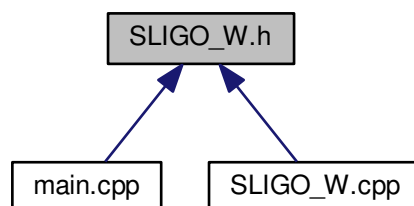
```

#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>
Include dependency graph for SLIGO_W.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class [SLIGO_W](#)

7.44 SLIGO_W.h

```

00001
00002 /*
00003  * File:    SLIGO_W.h
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #ifndef SLIGO_W_H
00011 #define SLIGO_W_H
00012 #include "WAREHOUSE.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class SLIGO_W :public WAREHOUSE {
00020 public:
00024     SLIGO_W() : WAREHOUSE() {
00025
00026         this->_shop_address = "Sligo River Street";
00027         this->_shop_name = "SLIGO S_WAREHOUSE";
00028         this->_number_of_iphones = 200;
00029         this->_number_of_samsung = 200;
00030         this->_number_of_sony = 200;
00031         this->_number_of_huawei = 200;
00032         this->_number_of_nokia = 200;
00033         this->_number_of_alcatel = 200;
00034     };
00038     SLIGO_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
00039     huawei, int nokia, int alcatel): WAREHOUSE() {
00040         /*
00041          * copy over values
00042          */
00042         this->_shop_address = address;
00043         this->_shop_name = shop_name;
00044         this->_number_of_iphones = iphone;
00045         this->_number_of_samsung = samsung;
00046         this->_number_of_sony = sony;
00047         this->_number_of_huawei = huawei;
00048         this->_number_of_nokia = nokia;
00049         this->_number_of_alcatel = alcatel;
00050
00051     };
00055     SLIGO_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
00056     WAREHOUSE(_version, _unique_id) {
00057         /*
00058          * copy over values
00059          */
00059         this->_shop_address = obj->GetShop_address();
00060         this->_shop_name = obj->GetShop_name();
00061         this->_number_of_iphones = obj->GetNumber_of_iphones();
00062         this->_number_of_samsung = obj->GetNumber_of_samsung();
00063         this->_number_of_sony = obj->GetNumber_of_sony();
00064         this->_number_of_huawei = obj->GetNumber_of_huawei();
00065         this->_number_of_nokia = obj->GetNumber_of_nokia();
00066         this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067     }
00071     SLIGO_W(const SLIGO_W& orig);
00075     SLIGO_W operator=(const SLIGO_W& orig) {};
00079     virtual ~SLIGO_W();
00080
00081     /*
00082      * Implement OSTM virtual methods
00083      */
00084     // virtual std::shared_ptr<SLIGO_W> _cast(std::shared_ptr<OSTM> _object);
00085     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00086     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00087     virtual void toString();
00088     /*
00089      * Implement Warehouse methods
00090      */
00091     virtual void SetNumber_of_alcatel(int _number_of_alcatel);
00092     virtual int GetNumber_of_alcatel();
00093     virtual void SetNumber_of_nokia(int _number_of_nokia);
00094     virtual int GetNumber_of_nokia();
00095     virtual void SetNumber_of_huawei(int _number_of_huawei);

```

```

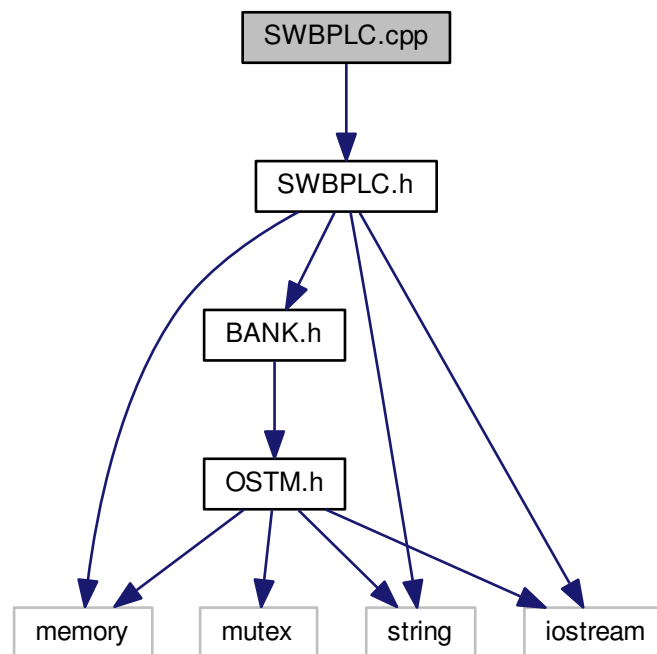
00096     virtual int GetNumber_of_huawei();
00097     virtual void SetNumber_of_sony(int _number_of_sony);
00098     virtual int GetNumber_of_sony();
00099     virtual void SetNumber_of_samsung(int _number_of_samsung);
00100     virtual int GetNumber_of_samsung();
00101     virtual void SetNumber_of_iphones(int _number_of_iphones);
00102     virtual int GetNumber_of_iphones();
00103     virtual void SetShop_name(std::string _shop_name);
00104     virtual std::string GetShop_name();
00105     virtual void SetShop_address(std::string _shop_address);
00106     virtual std::string GetShop_address();
00107
00108
00109 private:
00110     std::string _shop_address;
00111     std::string _shop_name;
00112     int _number_of_iphones;
00113     int _number_of_samsung;
00114     int _number_of_sony;
00115     int _number_of_huawei;
00116     int _number_of_nokia;
00117     int _number_of_alcatel;
00118
00119 };
00120
00121 #endif /* SLIGO_W_H */
00122

```

7.45 SWBPLC.cpp File Reference

```
#include "SWBPLC.h"
```

Include dependency graph for SWBPLC.cpp:



7.46 SWBPLC.cpp

```
00001
```

```

00002  /*
00003  * File:   SWBPLC.cpp
00004  * Author: Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #include "SWBPLC.h"
00011
00012 SWBPLC::SWBPLC(const SWBPLC& orig) {
00013 }
00014
00015 SWBPLC::~SWBPLC() {
00016 }
00022 std::shared_ptr<OSTM> SWBPLC::getBaseCopy(std::shared_ptr<OSTM> object)
00023 {
00024     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00025     std::shared_ptr<BANK> obj(new SWBPLC(objTO, object->Get_Version(), object->Get_Unique_ID()));
00026     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00027
00028     return ostm_obj;
00029 }
00034 void SWBPLC::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00035
00036     std::shared_ptr<SWBPLC> objTO = std::dynamic_pointer_cast<SWBPLC>(to);
00037     std::shared_ptr<SWBPLC> objFROM = std::dynamic_pointer_cast<SWBPLC>(from);
00038     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00039     objTO->Set_Version(objFROM->Get_Version());
00040     objTO->SetAccountNumber(objFROM->GetAccountNumber());
00041     objTO->SetBalance(objFROM->GetBalance());
00042
00043 }
00044
00048 //std::shared_ptr<SWBPLC> SWBPLC::_cast(std::shared_ptr<OSTM> _object){
00049 //
00050 //     return static_cast<std::shared_ptr<SWBPLC>>(_object);
00051 //}
00055 void SWBPLC::toString()
00056 {
00057     std::cout << "\nSWBPLC BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " <<
    this->GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst
    name: " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\n
    Version number : " << this->Get_Version() << std::endl;
00058 }
00059
00060 void SWBPLC::SetAddress(std::string address) {
00061     this->address = address;
00062 }
00063
00064 std::string SWBPLC::GetAddress() const {
00065     return address;
00066 }
00067
00068 void SWBPLC::SetBalance(double balance) {
00069     this->balance = balance;
00070 }
00071
00072 double SWBPLC::GetBalance() const {
00073     return balance;
00074 }
00075
00076 void SWBPLC::SetAccountNumber(int accountNumber) {
00077     this->accountNumber = accountNumber;
00078 }
00079
00080 int SWBPLC::GetAccountNumber() const {
00081     return accountNumber;
00082 }
00083
00084 void SWBPLC::SetLastName(std::string lastName) {
00085     this->lastName = lastName;
00086 }
00087
00088 std::string SWBPLC::GetLastName() const {
00089     return lastName;
00090 }
00091
00092 void SWBPLC::SetFirstName(std::string firstName) {
00093     this->firstName = firstName;
00094 }
00095
00096 std::string SWBPLC::GetFirstName() const {
00097     return firstName;
00098 }
00099
00100 void SWBPLC::SetFullname(std::string fullname) {

```

```

00101     this->fullname = fullname;
00102 }
00103
00104 std::string SWBPLC::GetFullname() const {
00105     return fullname;
00106 }
00107

```

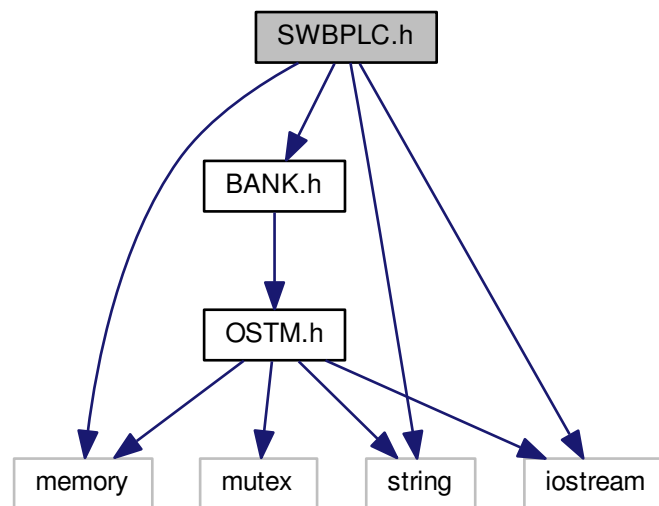
7.47 SWBPLC.h File Reference

```

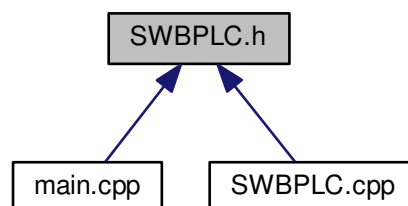
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for SWBPLC.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SWBPLC](#)

7.48 SWBPLC.h

```

00001
00002 /*
00003  * File:    SWBPLC.h
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #ifndef SWBPLC_H
00011 #define SWBPLC_H
00012 #include "BANK.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class SWBPLC : public BANK {
00020 public:
00024     SWBPLC() : BANK() {
00025         this->accountNumber = 0;
00026         this->balance = 50;
00027         this->firstName = "Joe";
00028         this->lastName = "Blog";
00029         this->address = "High street, Carlow";
00030         this->fullname = firstName + " " + lastName;
00031     };
00035     SWBPLC(int accountNumber, double balance, std::string firstName, std::string lastName,
std::string address) : BANK() {
00036         this->accountNumber = accountNumber;
00037         this->balance = balance;
00038         this->firstName = firstName;
00039         this->lastName = lastName;
00040         this->address = address;
00041         this->fullname = firstName + " " + lastName;
00042     };
00046     SWBPLC(std::shared_ptr<BANK> obj, int _version, int _unique_id) : BANK(_version, _unique_id)
{
00047
00048         this->accountNumber = obj->GetAccountNumber();
00049         this->balance = obj->GetBalance();
00050         this->firstName = obj->GetFirstName();
00051         this->lastName = obj->GetLastName();
00052         this->address = obj->GetAddress();
00053         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00054
00055     };
00059     SWBPLC(const SWBPLC& orig);
00063     SWBPLC operator=(const SWBPLC& orig) {};
00067     virtual ~SWBPLC();
00068
00069     /*
00070     * Implement OSTM virtual methods
00071     */
00072     //virtual std::shared_ptr<SWBPLC> _cast(std::shared_ptr<OSTM> _object);
00073     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00074     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00075     virtual void toString();
00076
00077     /*
00078     * Implement BANK virtual methods
00079     */
00080     virtual void SetAddress(std::string address);
00081     virtual std::string GetAddress() const;
00082     virtual void SetBalance(double balance);
00083     virtual double GetBalance() const;
00084     virtual void SetAccountNumber(int accountNumber);
00085     virtual int GetAccountNumber() const;
00086     virtual void SetLastName(std::string lastName);
00087     virtual std::string GetLastName() const;
00088     virtual void SetFirstName(std::string firstName);
00089     virtual std::string GetFirstName() const;
00090     virtual void SetFullname(std::string fullname);
00091     virtual std::string GetFullname() const;
00092 private:
00093     std::string fullname;
00094     std::string firstName;
00095     std::string lastName;

```



```

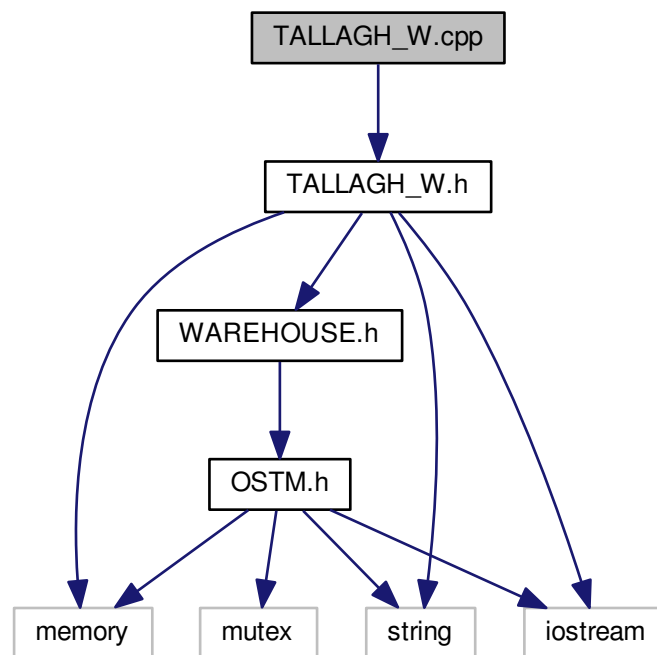
00096     int accountNumber;
00097     double balance;
00098     std::string address;
00099
00100 };
00101
00102 #endif /* SWBPLC_H */
00103

```

7.49 TALLAGH_W.cpp File Reference

```
#include "TALLAGH_W.h"
```

Include dependency graph for TALLAGH_W.cpp:



7.50 TALLAGH_W.cpp

```

00001
00002 /*
00003  * File:    TALLAGH_W.cpp
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #include "TALLAGH_W.h"
00011
00012 TALLAGH_W::~TALLAGH_W() {
00013 }
00014
00015 TALLAGH_W::TALLAGH_W(const TALLAGH_W& orig) {
00016 }
00022 std::shared_ptr<OSTM> TALLAGH_W::getBaseCopy(std::shared_ptr<OSTM> object)
00023 {

```

```

00024
00025     std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
00026     std::shared_ptr<WAREHOUSE> obj(new TALLAGH_W(objTO, object->Get_Version(), object->
Get_Unique_ID()));
00027     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

00028     return ostm_obj;
00029 }
00030 void TALLAGH_W::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00031
00032     std::shared_ptr<TALLAGH_W> objTO = std::dynamic_pointer_cast<TALLAGH_W>(to);
00033     std::shared_ptr<TALLAGH_W> objFROM = std::dynamic_pointer_cast<TALLAGH_W>(from);
00034     objTO->_shop_address = objFROM->GetShop_address();
00035     objTO->_shop_name = objFROM->GetShop_name();
00036     objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
00037     objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
00038     objTO->_number_of_sony = objFROM->GetNumber_of_sony();
00039     objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
00040     objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
00041     objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
00042     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00043     objTO->Set_Version(objFROM->Get_Version());
00044
00045 }
00046 //std::shared_ptr<TALLAGH_W> TALLAGH_W::_cast(std::shared_ptr<OSTM> _object){
00047 //
00048 //     return static_cast<std::shared_ptr<TALLAGH_W>>(_object);
00049 //}
00050 void TALLAGH_W::toString()
00051 {
00052     std::cout << "\n" << this->GetShop_name() << "\nUnique ID : " << this->Get_Unique_ID() << "
\nShop Name : " << this->GetShop_name() << "\nShop Address : " << this->
GetShop_address() << "\nNo. Iphones : " << this->
GetNumber_of_iphones() << "\nNo. Samsung : " << this->
GetNumber_of_samsung() << "\nNo. Sony : " << this->
GetNumber_of_sony() << "\nNo. Huawei : " << this->
GetNumber_of_huawei() << "\nNo. Nokia : " << this->
GetNumber_of_nokia() << "\nNo. Alcatel : " << this->
GetNumber_of_alcatel() << "\nVersion number : " << this->Get_Version() << std::endl;
00053 }
00054 void TALLAGH_W::SetNumber_of_alcatel(int _number_of_alcatel) {
00055     this->_number_of_alcatel = _number_of_alcatel;
00056 }
00057 int TALLAGH_W::GetNumber_of_alcatel() {
00058     return _number_of_alcatel;
00059 }
00060 void TALLAGH_W::SetNumber_of_nokia(int _number_of_nokia) {
00061     this->_number_of_nokia = _number_of_nokia;
00062 }
00063 int TALLAGH_W::GetNumber_of_nokia() {
00064     return _number_of_nokia;
00065 }
00066 void TALLAGH_W::SetNumber_of_huawei(int _number_of_huawei) {
00067     this->_number_of_huawei = _number_of_huawei;
00068 }
00069 int TALLAGH_W::GetNumber_of_huawei() {
00070     return _number_of_huawei;
00071 }
00072 void TALLAGH_W::SetNumber_of_sony(int _number_of_sony) {
00073     this->_number_of_sony = _number_of_sony;
00074 }
00075 int TALLAGH_W::GetNumber_of_sony() {
00076     return _number_of_sony;
00077 }
00078 void TALLAGH_W::SetNumber_of_samsung(int _number_of_samsung) {
00079     this->_number_of_samsung = _number_of_samsung;
00080 }
00081 int TALLAGH_W::GetNumber_of_samsung() {
00082     return _number_of_samsung;
00083 }
00084 void TALLAGH_W::SetNumber_of_iphones(int _number_of_iphones) {
00085     this->_number_of_iphones = _number_of_iphones;
00086 }
00087 int TALLAGH_W::GetNumber_of_iphones() {

```

```

00112     return _number_of_iphones;
00113 }
00114
00115 void TALLAGH_W::SetShop_name(std::string _shop_name) {
00116     this->_shop_name = _shop_name;
00117 }
00118
00119 std::string TALLAGH_W::GetShop_name() {
00120     return _shop_name;
00121 }
00122
00123 void TALLAGH_W::SetShop_address(std::string _shop_address) {
00124     this->_shop_address = _shop_address;
00125 }
00126
00127 std::string TALLAGH_W::GetShop_address() {
00128     return _shop_address;
00129 }

```

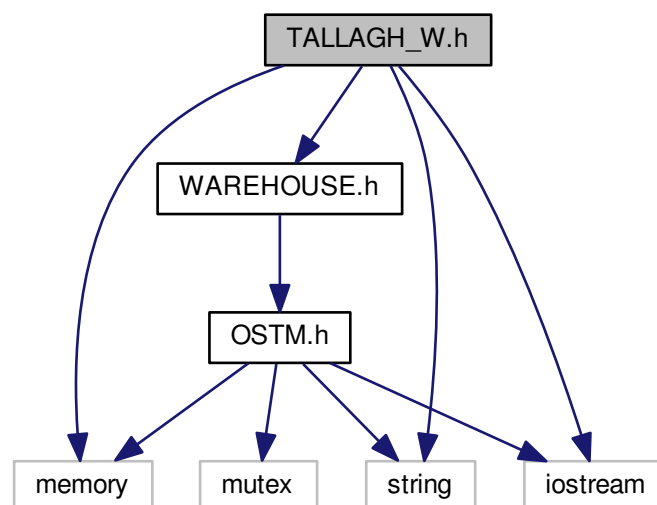
7.51 TALLAGH_W.h File Reference

```

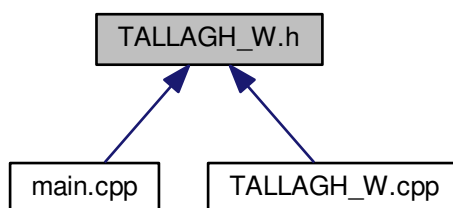
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for TALLAGH_W.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TALLAGH_W](#)

7.52 TALLAGH_W.h

```

00001
00002 /*
00003  * File:    TALLAGH_W.h
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #ifndef TALLAGH_W_H
00011 #define TALLAGH_W_H
00012 #include "WAREHOUSE.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class TALLAGH_W :public WAREHOUSE {
00020 public:
00024     TALLAGH_W() : WAREHOUSE(){
00025
00026         this->_shop_address = "Tallagh Low street";
00027         this->_shop_name = "TALLAGH T_WAREHOUSE";
00028         this->_number_of_iphones = 200;
00029         this->_number_of_samsung = 200;
00030         this->_number_of_sony = 200;
00031         this->_number_of_huawei = 200;
00032         this->_number_of_nokia = 200;
00033         this->_number_of_alcatel = 200;
00034     };
00038     TALLAGH_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
00039     huawei, int nokia, int alcatel): WAREHOUSE(){
00039         /*
00040         * copy over values
00041         */
00042         this->_shop_address = address;
00043         this->_shop_name = shop_name;
00044         this->_number_of_iphones = iphone;
00045         this->_number_of_samsung = samsung;
00046         this->_number_of_sony = sony;
00047         this->_number_of_huawei = huawei;
00048         this->_number_of_nokia = nokia;
00049         this->_number_of_alcatel = alcatel;
00050
00051     };
00055     TALLAGH_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
00056     WAREHOUSE(_version, _unique_id){
00056         /*
00057         * copy over values
00058         */

```

```

00059         this->_shop_address = obj->GetShop_address();
00060         this->_shop_name = obj->GetShop_name();
00061         this->_number_of_iphones = obj->GetNumber_of_iphones();
00062         this->_number_of_samsung = obj->GetNumber_of_samsung();
00063         this->_number_of_sony = obj->GetNumber_of_sony();
00064         this->_number_of_huawei = obj->GetNumber_of_huawei();
00065         this->_number_of_nokia = obj->GetNumber_of_nokia();
00066         this->_number_of_alcatel = obj->GetNumber_of_alcatel();
00067     }
00071     TALLAGH_W(const TALLAGH_W& orig);
00075     TALLAGH_W operator=(const TALLAGH_W& orig){};
00079     virtual ~TALLAGH_W();
00080
00081     /*
00082     * Implement OSTM virtual methods
00083     */
00084     //virtual std::shared_ptr<TALLAGH_W> _cast(std::shared_ptr<OSTM> _object);
00085     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00086     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00087     virtual void toString();
00088     /*
00089     * Implement Warehouse methods
00090     */
00091     virtual void SetNumber_of_alcatel(int _number_of_alcatel);
00092     virtual int GetNumber_of_alcatel();
00093     virtual void SetNumber_of_nokia(int _number_of_nokia);
00094     virtual int GetNumber_of_nokia();
00095     virtual void SetNumber_of_huawei(int _number_of_huawei);
00096     virtual int GetNumber_of_huawei();
00097     virtual void SetNumber_of_sony(int _number_of_sony);
00098     virtual int GetNumber_of_sony();
00099     virtual void SetNumber_of_samsung(int _number_of_samsung);
00100     virtual int GetNumber_of_samsung();
00101     virtual void SetNumber_of_iphones(int _number_of_iphones);
00102     virtual int GetNumber_of_iphones();
00103     virtual void SetShop_name(std::string _shop_name);
00104     virtual std::string GetShop_name();
00105     virtual void SetShop_address(std::string _shop_address);
00106     virtual std::string GetShop_address();
00107
00108 private:
00109     std::string _shop_address;
00110     std::string _shop_name;
00111     int _number_of_iphones;
00112     int _number_of_samsung;
00113     int _number_of_sony;
00114     int _number_of_huawei;
00115     int _number_of_nokia;
00116     int _number_of_alcatel;
00117
00118 };
00119
00120
00121 #endif /* TALLAGH_W_H */
00122

```

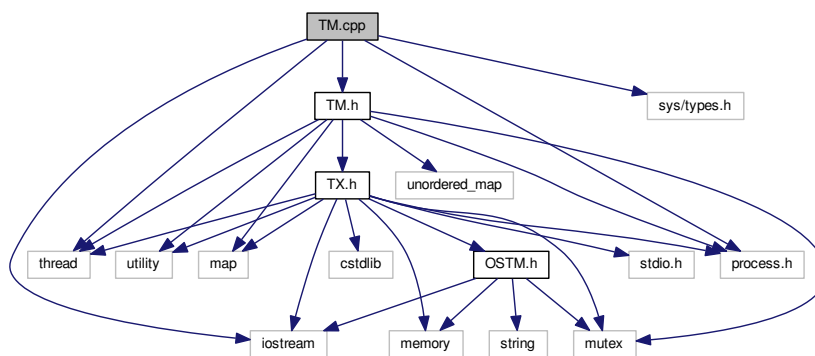
7.53 TM.cpp File Reference

```

#include "TM.h"
#include <thread>
#include <process.h>
#include <sys/types.h>
#include <iostream>

```

Include dependency graph for TM.cpp:



7.54 TM.cpp

```

00001 /*
00002  * File: TM.cpp
00003  * Author: Zoltan Fuzesi
00004  *
00005  * Created on December 18, 2017, 2:09 PM
00006  * Transaction Manager class methods implementation
00007  */
00008 #include "TM.h"
00009 #include <thread>
00010 // #include <unistd.h>
00011 #include <process.h>
00012 #include <sys/types.h>
00013 #include <iostream>
00014
00018 int TM::_tm_id;
00022 std::map<int, std::map< std::thread::id, int >> TM::process_map_collection;
00028 TM& TM::Instance() {
00029     static TM _instance;
00030     _instance._tm_id = _getpid();
00031
00032     return _instance;
00033 }
00034
00035 //TM Transaction manager checking the Process ID existence in the map
00036 //If not in the map then register
00043 void TM::registerTX()
00044 {
00045     std::lock_guard<std::mutex> guard(register_Lock);
00046     int ppid = _getpid();
00047     std::map<int, std::map< std::thread::id, int >>::iterator process_map_collection_Iterator =
00048     TM::process_map_collection.find(ppid);
00049     if (process_map_collection_Iterator == TM::process_map_collection.end()) {
00050         /*
00051          * Register main process/application to the global map
00052          */
00053         std::map< std::thread::id, int >map = get_thread_Map();
00054         TM::process_map_collection.insert({ppid, map});
00055     }
00056     std::map<std::thread::id, std::shared_ptr < TX>>::iterator it = txMap.find(std::this_thread::get_id());
00057     if (it == txMap.end()) {
00058         std::shared_ptr<TX> _transaction_object(new TX(std::this_thread::get_id()));
00059         txMap.insert({std::this_thread::get_id(), _transaction_object});
00060         /*
00061          * Get the map if registered first time
00062          */
00063         process_map_collection_Iterator = TM::process_map_collection.find(ppid);
00064         /*
00065          * Insert to the GLOBAL MAP as a helper to clean up at end of main process
00066          */
00067         process_map_collection_Iterator->second.insert({std::this_thread::get_id(), 1});
00068     }
00069 }
00070

```

```

00071 }
00072
00078 std::shared_ptr<TX>const TM::_get_tx()
00079 {
00080     std::lock_guard<std::mutex> guard(get_Lock);
00081
00082     std::map<std::thread::id, std::shared_ptr<TX>>::iterator it = txMap.find(std::this_thread::get_id());
00083     if(it == txMap.end())
00084     {
00085         registerTX();
00086         it = txMap.find(std::this_thread::get_id());
00087     } else {
00088         it->second->_increase_tx_nesting();
00089     }
00090     //it = txMap.find(std::this_thread::get_id());
00091
00092     return it->second;
00093 }
00094
00095 void TM::_TX_EXIT() {
00096     TX tx(std::this_thread::get_id());
00097     int ppid = _getpid();
00098     std::map<int, std::map< std::thread::id, int >>::iterator process_map_collection_Iterator =
00099     TM::process_map_collection.find(ppid);
00100     if (process_map_collection_Iterator != TM::process_map_collection.end()) {
00101         for (auto current = process_map_collection_Iterator->second.begin(); current !=
00102 process_map_collection_Iterator->second.end(); ++current) {
00103             /*
00104              * Delete all transaction associated with the actual main process
00105              */
00106             txMap.erase(current->first);
00107         }
00108         TM::process_map_collection.erase(ppid);
00109     }
00110     tx.ostm_exit();
00111 }
00112
00121 void TM::print_all() {
00122     get_Lock.lock();
00123     for (auto current = txMap.begin(); current != txMap.end(); ++current) {
00124         std::cout << "KEY : " << current->first << std::endl;
00125     }
00126     get_Lock.unlock();
00127 }
00128
00133 std::map< std::thread::id, int > TM::get_thread_Map() {
00134     std::map< std::thread::id, int > thread_Map;
00135     return thread_Map;
00136 }

```

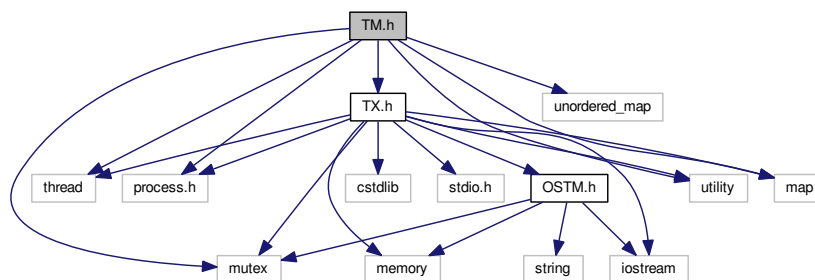
7.55 TM.h File Reference

```

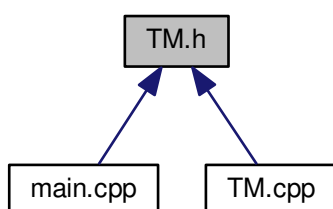
#include <thread>
#include <process.h>
#include <mutex>
#include <unordered_map>
#include <utility>
#include <map>
#include "TX.h"

```

Include dependency graph for TM.h:



This graph shows which files directly or indirectly include this file:



Functions

- class `__declspec (dllexport)` TM

7.55.1 Function Documentation

7.55.1.1 class `__declspec (dllexport)`

TM constructor, prevent from multiple instantiation

TM de-constructor, prevent from deletion

TM copy constructor, prevent from copying the Transaction Manager

TM copy operator, prevent from copying the Transaction Manager

Parameters

<i>txMap</i>	std::map, store all transactional objects created with Transaction Manager
--------------	--

STATIC GLOBAL MAP Collection to store all process associated keys to find when deleting transactions

Parameters

<i>process_map_collection</i>	std::map
-------------------------------	----------

get_thread_Map returning and map to insert to the process_map_collection as an inner value

registerTX void, register transaction into txMap

Parameters

<i>register_Lock</i>	std::mutex, used in the registerTX function
<i>register_Lock</i>	std::mutex, used in the _get_tx function
<i>_tm_id</i>	pid_t, process id determine the actual process between process in the shared OSTM library

Scott Meyer's Singleton creation, what is thread safe

_get_tx std::shared_ptr<TX>, returning a shared pointer with the transaction

_TX_EXIT void, the thread calls the ostm_exit function in the transaction, and clear all elements from the shared global collection associated with the main process

ONLY FOR TESTING print_all void, prints all object in the txMap

Definition at line 48 of file [TM.h](#).

```

00048                                     {
00049 private:
00053     TM() = default;
00057     ~TM() = default;
00061     TM(const TM&) = delete;
00065     TM& operator=(const TM&) = delete;
00069     std::map<std::thread::id, std::shared_ptr<TX>>txMap;
00074     static std::map<int, std::map< std::thread::id, int >> process_map_collection;
00078     std::map< std::thread::id, int > get_thread_Map();
00082     void registerTX();
00086     std::mutex register_Lock;
00090     std::mutex get_Lock;
00094     static int _tm_id;
00095
00096 public:
00097
00101     static TM& Instance();
00105     std::shared_ptr<TX>const _get_tx();
00109     void _TX_EXIT();
00113     void print_all();
00114
00115
00116 };

```

7.56 TM.h

```

00001 /*
00002  * File:    TM.h
00003  * Author:  Zoltan Fuzesi
00004  *
00005  * Created on December 18, 2017, 2:09 PM
00006  * Transaction Manager class fields and methods declarations
00007  */
00037 #ifndef TM_H
00038 #define TM_H
00039
00040 #include <thread>

```

```

00041 #include <process.h>
00042 #include <mutex>
00043 #include <unordered_map>
00044 #include <utility>
00045 #include <map>
00046 #include "TX.h"
00047
00048 class __declspec(dllexport) TM {
00049 private:
00053     TM() = default;
00057     ~TM() = default;
00061     TM(const TM&) = delete;
00065     TM& operator=(const TM&) = delete;
00069     std::map<std::thread::id, std::shared_ptr<TX>> txMap;
00074     static std::map<int, std::map< std::thread::id, int >> process_map_collection;
00078     std::map< std::thread::id, int > get_thread_Map();
00082     void registerTX();
00086     std::mutex register_Lock;
00090     std::mutex get_Lock;
00094     static int _tm_id;
00095
00096 public:
00097
00101     static TM& Instance();
00105     std::shared_ptr<TX> const _get_tx();
00109     void _TX_EXIT();
00113     void print_all();
00114
00115 };
00116
00117
00118
00119 #endif // TM_H

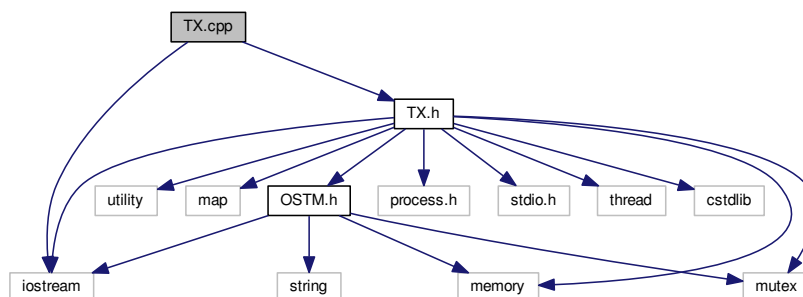
```

7.57 TX.cpp File Reference

```
#include "TX.h"
```

```
#include <iostream>
```

Include dependency graph for TX.cpp:



7.58 TX.cpp

```

00001 /*
00002  * File:   TX.cpp
00003  * Author: Zoltan Fuzesi
00004  *
00005  * Created on December 18, 2017, 2:09 PM
00006  * TX cpp file methods implementations
00007  */
00008 #include "TX.h"
00009 #include <iostream>
00013 std::map<int, std::shared_ptr<OSTM> > TX::main_Process_Map_collection;
00017 std::map<int, std::map< int, int >> TX::process_map_collection;
00021 std::mutex TX::register_Lock;

```

```

00025 int TX::test_counter = 0;
00031 TX::TX(std::thread::id id) {
00032     this->transaction_Number = id;
00033     this->_tx_nesting_level = 0;
00034 }
00038 TX::~TX() {
00039 }
00040 }
00044 TX::TX(const TX& orig) {
00045 }
00046 }
00047
00052 void TX::th_exit() {
00053
00054     if (this->_tx_nesting_level > 0) {
00055         /*
00056          * Active nested transactions running in background, do not delete anything yet
00057          */
00058     } else {
00059         /*
00060          * Remove all elements map entries from transaction and clear the map
00061          */
00062         working_Map_collection.clear();
00063     }
00064 }
00065
00072 void TX::ostm_exit() {
00073     std::map<int, std::shared_ptr<OSTM>>::iterator main_Process_Map_collection_Iterator;
00074
00075     int ppid = _getpid();
00076     std::map<int, std::map< int, int >>::iterator process_map_collection_Iterator =
TX::process_map_collection.find(ppid);
00077     if (process_map_collection_Iterator != TX::process_map_collection.end()) {
00078
00079         for (auto current = process_map_collection_Iterator->second.begin(); current !=
process_map_collection_Iterator->second.end(); ++current) {
00080             main_Process_Map_collection_Iterator = TX::main_Process_Map_collection.find(current->first);
00081
00082             if (main_Process_Map_collection_Iterator != TX::main_Process_Map_collection.end()){
00083                 /*
00084                  * Delete element from shared main_Process_Map_collection by object unique key value,
shared_ptr will destroy automatically
00085                  */
00086                 TX::main_Process_Map_collection.erase(main_Process_Map_collection_Iterator->first);
00087             }
00088         }
00089         /*
00090          * Delete from Process_map_collection, Main process exits delete association with library
00091          */
00092         TX::process_map_collection.erase(process_map_collection_Iterator->first);
00093     }
00094 }
00095
00104 void TX::_register(std::shared_ptr<OSTM> object) {
00105     /*
00106      * MUST USE SHARED LOCK TO PROTECT SHARED GLOBAL MAP/COLLECTION
00107      */
00108     std::lock_guard<std::mutex> guard(TX::register_Lock);
00109
00110     /*
00111      * Check for null pointer !
00112      * Null pointer can cause segmentation fault!!!
00113      */
00114     if(object == nullptr){
00115         throw std::runtime_error(std::string("[RUNTIME ERROR : NULL POINTER IN REGISTER FUNCTION]") );
00116     }
00117
00118     int ppid = _getpid();
00119     std::map<int, std::map< int, int >>::iterator process_map_collection_Iterator =
TX::process_map_collection.find(ppid);
00120     if (process_map_collection_Iterator == TX::process_map_collection.end()) {
00121         /*
00122          * Register main process/application to the global map
00123          */
00124         std::map< int, int >map = get_thread_Map();
00125         TX::process_map_collection.insert({ppid, map});
00126         /*
00127          * Get the map if registered first time
00128          */
00129         process_map_collection_Iterator = TX::process_map_collection.find(ppid);
00130     }
00131     std::map<int, std::shared_ptr<OSTM>>::iterator main_Process_Map_collection_Iterator =
TX::main_Process_Map_collection.find(object->Get_Unique_ID());
00132     if (main_Process_Map_collection_Iterator == TX::main_Process_Map_collection.end()) {
00133         /*
00134          * Insert to the GLOBAL MAP
00135          */

```

```

00136         TX::main_Process_Map_collection.insert({object->Get_Unique_ID(), object});
00137         /*
00138          * Insert to the GLOBAL MAP as a helper to clean up at end of main process
00139          */
00140         process_map_collection_Iterator->second.insert({object->Get_Unique_ID(), 1});
00141     }
00142
00143
00144     std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator
= working_Map_collection.find(object->Get_Unique_ID());
00145     if (working_Map_collection_Object_Shared_Pointer_Iterator == working_Map_collection.end()) {
00146
00147         working_Map_collection.insert({object->Get_Unique_ID(), object->getBaseCopy(object)});
00148     }
00149
00150 }
00151 std::shared_ptr<OSTM> TX::load(std::shared_ptr<OSTM> object) {
00152
00153     std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator;
00154     /*
00155      * Check for null pointer !
00156      * Null pointer can cause segmentation fault!!!
00157      */
00158     if(object == nullptr){
00159         throw std::runtime_error(std::string("[RUNTIME ERROR : NULL POINTER IN LOAD FUNCTION]") );
00160     }
00161
00162     working_Map_collection_Object_Shared_Pointer_Iterator = working_Map_collection.find(object->
Get_Unique_ID());
00163
00164     if (working_Map_collection_Object_Shared_Pointer_Iterator != working_Map_collection.end()) {
00165
00166         return working_Map_collection_Object_Shared_Pointer_Iterator->second->getBaseCopy(
working_Map_collection_Object_Shared_Pointer_Iterator->second);
00167
00168     } else { throw std::runtime_error(std::string("[RUNTIME ERROR : NO OBJECT FOUND LOAD FUNCTION]") );}
00169 }
00170 void TX::store(std::shared_ptr<OSTM> object) {
00171     /*
00172      * Check for null pointer !
00173      * Null pointer can cause segmentation fault!!!
00174      */
00175     if(object == nullptr){
00176         throw std::runtime_error(std::string("[RUNTIME ERROR : NULL POINTER IN STORE FUNCTION]") );
00177     }
00178
00179     std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator;
00180
00181     working_Map_collection_Object_Shared_Pointer_Iterator = working_Map_collection.find(object->
Get_Unique_ID());
00182
00183     if (working_Map_collection_Object_Shared_Pointer_Iterator != working_Map_collection.end()) {
00184
00185         working_Map_collection_Object_Shared_Pointer_Iterator->second = object;
00186
00187     } else { std::cout << "[ERROR STORE]" << std::endl; }
00188 }
00189 bool TX::commit() {
00190
00191     bool can_Commit = true;
00192
00193     /*
00194      * Dealing with nested transactions first
00195      */
00196     if (this->tx_nesting_level > 0) {
00197         _decrease_tx_nesting();
00198         return true;
00199     }
00200
00201     std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator;
00202
00203     std::map<int, std::shared_ptr<OSTM>>::iterator main_Process_Map_collection_Iterator;
00204     for (working_Map_collection_Object_Shared_Pointer_Iterator = working_Map_collection.begin();
working_Map_collection_Object_Shared_Pointer_Iterator != working_Map_collection.end();
working_Map_collection_Object_Shared_Pointer_Iterator++) {
00205
00206         main_Process_Map_collection_Iterator = TX::main_Process_Map_collection.find(
working_Map_collection_Object_Shared_Pointer_Iterator->second->Get_Unique_ID());
00207
00208         /*
00209          * Throws runtime error if object can not find
00210          */
00211         if(main_Process_Map_collection_Iterator == TX::main_Process_Map_collection.end())
00212         {
00213             throw std::runtime_error(std::string("[RUNTIME ERROR : CAN'T FIND OBJECT COMMIT FUNCTION]")
);
00214         }
00215
00216     }
00217
00218     /*

```

```

00229         * Busy wait WHILE object locked by other thread
00230         */
00231         while (! (main_Process_Map_collection_Iterator->second->is_Locked()));
00232
00233         if (main_Process_Map_collection_Iterator->second->Get_Version() >
working_Map_collection_Object_Shared_Pointer_Iterator->second->Get_Version()) {
00234
00235             working_Map_collection_Object_Shared_Pointer_Iterator->second->Set_Can_Commit(false);
00236             can_Commit = false;
00237             break;
00238         } else {
00239
00240             working_Map_collection_Object_Shared_Pointer_Iterator->second->Set_Can_Commit(true);
00241         }
00242     }
00243     if (!can_Commit) {
00244         TX::test_counter += 1;
00245         for (working_Map_collection_Object_Shared_Pointer_Iterator = working_Map_collection.begin();
working_Map_collection_Object_Shared_Pointer_Iterator != working_Map_collection.end();
working_Map_collection_Object_Shared_Pointer_Iterator++) {
00246
00247             main_Process_Map_collection_Iterator = TX::main_Process_Map_collection.find(
working_Map_collection_Object_Shared_Pointer_Iterator->second->Get_Unique_ID());
00248             (working_Map_collection_Object_Shared_Pointer_Iterator->second->copy(
working_Map_collection_Object_Shared_Pointer_Iterator->second, main_Process_Map_collection_Iterator->second);
00249         }
00250     }
00251     _release_object_lock();
00252
00253     return false;
00254 } else {
00255     /*
00256     * Commit changes
00257     */
00258     for (working_Map_collection_Object_Shared_Pointer_Iterator = working_Map_collection.begin();
working_Map_collection_Object_Shared_Pointer_Iterator != working_Map_collection.end();
working_Map_collection_Object_Shared_Pointer_Iterator++) {
00260
00261         main_Process_Map_collection_Iterator = TX::main_Process_Map_collection.find((
working_Map_collection_Object_Shared_Pointer_Iterator->second->Get_Unique_ID());
00262         if (main_Process_Map_collection_Iterator != TX::main_Process_Map_collection.end()) {
00263
00264             (main_Process_Map_collection_Iterator->second->copy(
main_Process_Map_collection_Iterator->second, working_Map_collection_Object_Shared_Pointer_Iterator->second);
00265             main_Process_Map_collection_Iterator->second->increase_VersionNumber();
00266
00267         } else {
00268             throw std::runtime_error(std::string("[RUNTIME ERROR : CAN'T FIND OBJECT COMMIT
FUNCTION]"));
00270         }
00271     }
00272 }
00273
00274     _release_object_lock();
00275     this->th_exit();
00276     return true;
00277 }
00278 } //Commit finish
00279
00280 void TX::_release_object_lock() {
00281
00282     std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator;
00283     std::map<int, std::shared_ptr<OSTM>>::iterator main_Process_Map_collection_Iterator;
00284     for (working_Map_collection_Object_Shared_Pointer_Iterator = working_Map_collection.begin();
working_Map_collection_Object_Shared_Pointer_Iterator != working_Map_collection.end();
working_Map_collection_Object_Shared_Pointer_Iterator++) {
00291
00292         main_Process_Map_collection_Iterator = TX::main_Process_Map_collection.find((
working_Map_collection_Object_Shared_Pointer_Iterator->second->Get_Unique_ID());
00293         if (main_Process_Map_collection_Iterator != TX::main_Process_Map_collection.end()) {
00294             /*
00295             * Release object lock
00296             */
00297             (main_Process_Map_collection_Iterator->second->unlock_Mutex();
00298         }
00299     }
00300 }
00301 }
00302
00303 void TX::_increase_tx_nesting() {
00304
00305     this->_tx_nesting_level += 1;
00306     // std::cout << "[this->_tx_nesting_level] = " << this->_tx_nesting_level << std::endl;
00307 }
00308 }

```

```

00316 void TX::_decrease_tx_nesting() {
00317     // std::cout << "[this->_tx_nesting_level] = " << this->_tx_nesting_level << std::endl;
00318     this->_tx_nesting_level -= 1;
00319 }
00320 }
00324 int TX::getTest_counter() {
00325     return TX::test_counter;
00326 }
00331 const std::thread::id TX::_get_tx_number() const {
00332     return transaction_Number;
00333 }
00338 std::map< int, int > TX::get_thread_Map() {
00339     std::map< int, int > thread_Map;
00340     return thread_Map;
00341 }
00342
00346 void TX::_print_all_tx() {
00347
00348     std::cout << "[PRINTALLTHREAD]" << std::endl;
00349     std::map< int, std::shared_ptr<OSTM> >::iterator it;
00350     /*
00351     * All registered thread id in the TX global
00352     */
00353     int ppid = _getpid();
00354     std::map<int, std::map< int, int >::iterator process_map_collection_Iterator =
TX::process_map_collection.find(ppid);
00355     if (process_map_collection_Iterator != TX::process_map_collection.end()) {
00356
00357         for (auto current = process_map_collection_Iterator->second.begin(); current !=
process_map_collection_Iterator->second.end(); ++current) {
00358             it = working_Map_collection.find(current->first);
00359             if (it != working_Map_collection.end()) {
00360                 std::cout << "[Unique number ] : " << it->second->Get_Unique_ID() << std::endl;
00361             }
00362         }
00363     }
00364 }
00365
00366 }
00367 }

```

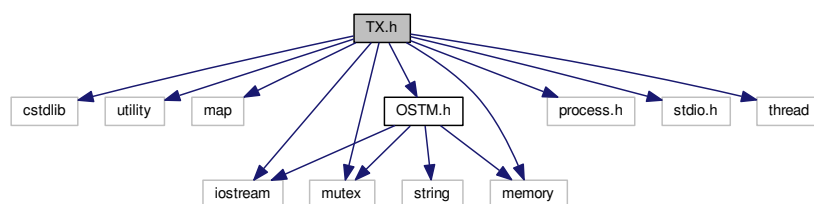
7.59 TX.h File Reference

```

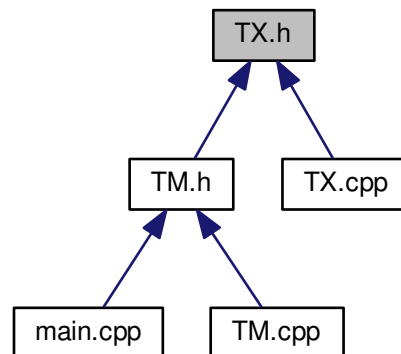
#include <cstdlib>
#include <utility>
#include <map>
#include <iostream>
#include <mutex>
#include <process.h>
#include <memory>
#include <stdio.h>
#include <thread>
#include "OSTM.h"

```

Include dependency graph for TX.h:



This graph shows which files directly or indirectly include this file:



Functions

- class `__declspec (dllexport) TX`

7.59.1 Function Documentation

7.59.1.1 class `__declspec (dllexport)`

Constructor

De-constructor

Default copy constructor

Delete all map entries associated with the main process

Register OSTM pointer into STM library

Register OSTM pointer into STM library

Store transactional changes

Commit transactional changes

Add TX nesting level by one

Remove TX nesting level by one

Only TM Transaction Manager can create instance of TX Transaction

Parameters

<code>test_counter</code>	int ONLY FOR TESTING!!!
---------------------------	-------------------------

MAP Collection to store OSTM* parent based pointers to make invisible changes during isolated transaction

Parameters

<i>working_Map_collection</i>	std::map
-------------------------------	----------

Returning the transaction number

Parameters

<i>transaction_Number</i>	std::thread::id NOT USED YET
<i>_tx_nesting_level</i>	int

STATIC GLOBAL MAP Collection to store OSTM* parent based pointers to control/lock and compare objects version number within transactions

Parameters

<i>main_Process_Map_collection</i>	std::map
------------------------------------	----------

STATIC GLOBAL MAP Collection to store all process associated keys to find when deleting transactions

Parameters

<i>process_map_collection</i>	std::map
-------------------------------	----------

get_thread_Map returning and map to insert to the process_map_collection as an inner value

Parameters

<i>register_Lock</i>	std::mutex to control shared access on MAIN MAP
----------------------	---

_get_tx_number returning the transaction unique identifier

Release the locks in objects with transaction associated collection

Clean up all associated values by the thread delete from working_Map_collection, it is an automated function

Definition at line 24 of file TX.h.

```

00024                                     {
00025 public:
00029     TX(std::thread::id id);
00033     ~TX();
00037     TX(const TX& orig);
00041     void ostm_exit();
00042
00046     void _register(std::shared_ptr<OSTM> object);
00050     std::shared_ptr<OSTM> load(std::shared_ptr<OSTM> object);
00054     void store(std::shared_ptr<OSTM> object);
00058     bool commit();
00062     void _increase_tx_nesting();
00066     void _decrease_tx_nesting();
00070     friend class TM;
00071     /*
00072     * \brief ONLY FOR TESTING!!! returning the number of rollback happened during transactions

```



```

00073     */
00074     int getTest_counter();
00078     static int test_counter;
00079     /*
00080     * TESTING ONLY
00081     */
00082     void _print_all_tx() ;
00083
00084
00085 private:
00090     std::map< int, std::shared_ptr<OSTM> > working_Map_collection;
00096     std::thread::id transaction_Number;
00100     int _tx_nesting_level;
00101
00106     static std::map<int, std::shared_ptr<OSTM> > main_Process_Map_collection;
00111     static std::map<int, std::map< int, int >> process_map_collection;
00112     //static std::map<pid_t, std::map< int, std::pair<ppid, int> >> process_map_collection;
00116     std::map< int , int > get_thread_Map();
00120     static std::mutex register_Lock;
00124     const std::thread::id _get_tx_number() const;
00125
00129     void _release_object_lock();
00133     void th_exit();
00134
00135
00136
00137 };

```

7.60 TX.h

```

00001 /*
00002 * File:    TX.h
00003 * Author:  Zoltan Fuzesi
00004 *
00005 * Created on December 18, 2017, 2:09 PM
00006 * Transaction class fields and methods declarations
00007 */
00008
00009 #ifndef TX_H
00010 #define TX_H
00011 #include <cstdlib>
00012 #include <utility>
00013 #include <map>
00014 #include <iostream>
00015 #include <mutex>
00016 #include <process.h>
00017 #include <memory>
00018 #include <stdio.h>
00019 #include <thread>
00020 #include "OSTM.h"
00021
00022 class TM;
00023
00024 class __declspec(dllexport) TX {
00025 public:
00029     TX(std::thread::id id);
00033     ~TX();
00037     TX(const TX& orig);
00041     void ostm_exit();
00042
00046     void _register(std::shared_ptr<OSTM> object);
00050     std::shared_ptr<OSTM> load(std::shared_ptr<OSTM> object);
00054     void store(std::shared_ptr<OSTM> object);
00058     bool commit();
00062     void _increase_tx_nesting();
00066     void _decrease_tx_nesting();
00070     friend class TM;
00071
00072     /* \brief ONLY FOR TESTING!!! returning the number of rollback happened during transactions
00073     */
00074     int getTest_counter();
00078     static int test_counter;
00079     /*
00080     * TESTING ONLY
00081     */
00082     void _print_all_tx() ;
00083
00084
00085 private:
00090     std::map< int, std::shared_ptr<OSTM> > working_Map_collection;
00096     std::thread::id transaction_Number;
00100     int _tx_nesting_level;
00101

```

```

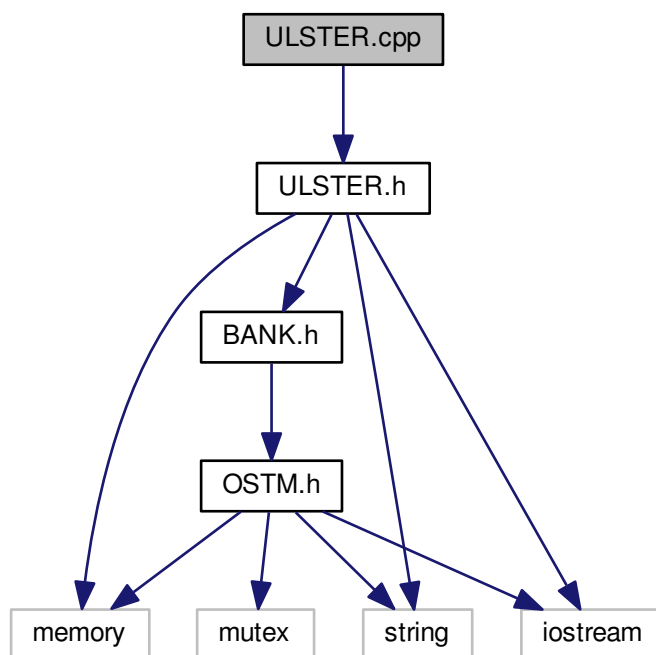
00106     static std::map<int, std::shared_ptr<OSTM> > main_Process_Map_collection;
00111     static std::map<int, std::map< int, int >> process_map_collection;
00112     //static std::map<pid_t, std::map< int, std::pair<ppid, int> >> process_map_collection;
00116     std::map< int , int > get_thread_Map();
00120     static std::mutex register_Lock;
00124     const std::thread::id _get_tx_number() const;
00125
00129     void _release_object_lock();
00133     void th_exit();
00134
00135
00136
00137 };
00138 #endif // _TX_H_

```

7.61 ULSTER.cpp File Reference

```
#include "ULSTER.h"
```

Include dependency graph for ULSTER.cpp:



7.62 ULSTER.cpp

```

00001
00002 /*
00003  * File:    ULSTER.cpp
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #include "ULSTER.h"
00011
00012 //ULSTER::ULSTER() {

```

```

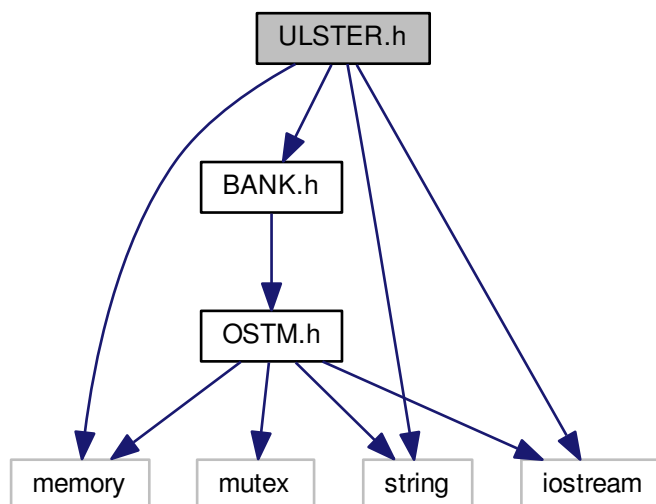
00013 //}
00014
00015 ULSTER::ULSTER(const ULSTER& orig) {
00016 }
00017
00018 ULSTER::~ULSTER() {
00019 }
00020
00021 std::shared_ptr<OSTM> ULSTER::getBaseCopy(std::shared_ptr<OSTM> object)
00022 {
00023     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00024     std::shared_ptr<BANK> obj(new ULSTER(objTO, object->Get_Version(), object->Get_Unique_ID()));
00025     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00026
00027     return ostm_obj;
00028 }
00029
00030 void ULSTER::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00031
00032     std::shared_ptr<ULSTER> objTO = std::dynamic_pointer_cast<ULSTER>(to);
00033     std::shared_ptr<ULSTER> objFROM = std::dynamic_pointer_cast<ULSTER>(from);
00034     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00035     objTO->Set_Version(objFROM->Get_Version());
00036     objTO->SetAccountNumber(objFROM->GetAccountNumber());
00037     objTO->SetBalance(objFROM->GetBalance());
00038
00039 }
00040
00041 //std::shared_ptr<ULSTER> ULSTER::_cast(std::shared_ptr<OSTM> _object){
00042 //
00043 //     return static_cast<std::shared_ptr<ULSTER>>(_object);
00044 //}
00045
00046 void ULSTER::toString()
00047 {
00048     std::cout << "\nULSTER BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " <<
this->GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
" << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
number : " << this->Get_Version() << std::endl;
00049 }
00050
00051 void ULSTER::SetAddress(std::string address) {
00052     this->address = address;
00053 }
00054
00055 std::string ULSTER::GetAddress() const {
00056     return address;
00057 }
00058
00059 void ULSTER::SetBalance(double balance) {
00060     this->balance = balance;
00061 }
00062
00063 double ULSTER::GetBalance() const {
00064     return balance;
00065 }
00066
00067 void ULSTER::SetAccountNumber(int accountNumber) {
00068     this->accountNumber = accountNumber;
00069 }
00070
00071 int ULSTER::GetAccountNumber() const {
00072     return accountNumber;
00073 }
00074
00075 void ULSTER::SetLastName(std::string lastName) {
00076     this->lastName = lastName;
00077 }
00078
00079 std::string ULSTER::GetLastName() const {
00080     return lastName;
00081 }
00082
00083 void ULSTER::SetFirstName(std::string firstName) {
00084     this->firstName = firstName;
00085 }
00086
00087 std::string ULSTER::GetFirstName() const {
00088     return firstName;
00089 }
00090
00091 void ULSTER::SetFullname(std::string fullname) {
00092     this->fullname = fullname;
00093 }
00094
00095 std::string ULSTER::GetFullname() const {
00096     return fullname;
00097 }
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110

```

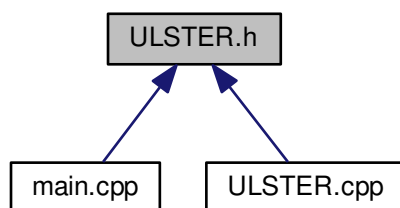
7.63 ULSTER.h File Reference

```
#include "BANK.h"  
#include <string>  
#include <memory>  
#include <iostream>
```

Include dependency graph for ULSTER.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ULSTER](#)

7.64 ULSTER.h

```

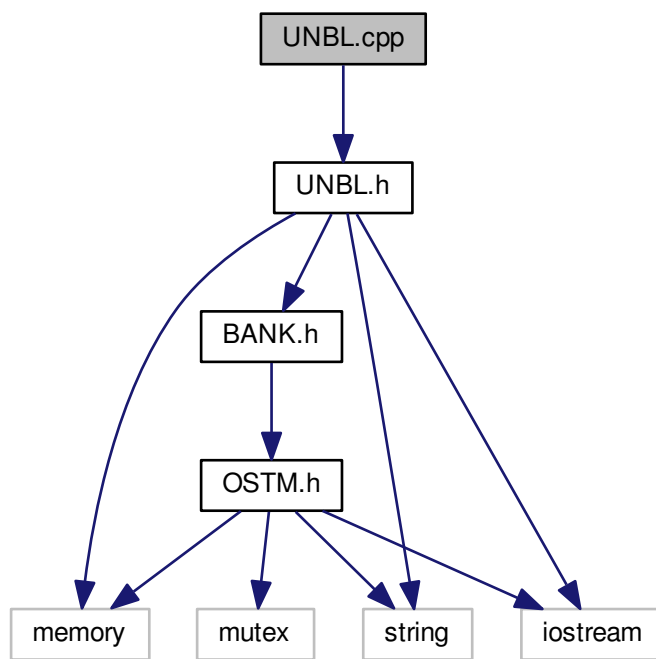
00001
00002 /*
00003  * File:    ULSTER.h
00004  * Author:  Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #ifndef ULSTER_H
00011 #define ULSTER_H
00012 #include "BANK.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class ULSTER : public BANK {
00020 public:
00024     ULSTER() : BANK() {
00025         this->accountNumber = 0;
00026         this->balance = 50;
00027         this->firstName = "Joe";
00028         this->lastName = "Blog";
00029         this->address = "High street, Carlow";
00030         this->fullname = firstName + " " + lastName;
00031     };
00035     ULSTER(int accountNumber, double balance, std::string firstName, std::string lastName,
std::string address) : BANK() {
00036         this->accountNumber = accountNumber;
00037         this->balance = balance;
00038         this->firstName = firstName;
00039         this->lastName = lastName;
00040         this->address = address;
00041         this->fullname = firstName + " " + lastName;
00042     };
00046     ULSTER(std::shared_ptr<BANK> obj, int _version, int _unique_id) : BANK(_version, _unique_id)
{
00047
00048         this->accountNumber = obj->GetAccountNumber();
00049         this->balance = obj->GetBalance();
00050         this->firstName = obj->GetFirstName();
00051         this->lastName = obj->GetLastName();
00052         this->address = obj->GetAddress();
00053         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00054     };
00058     ULSTER(const ULSTER& orig);
00062     ULSTER operator=(const ULSTER& orig) {};
00066     virtual ~ULSTER();
00067
00068     /*
00069     * Implement OSTM virtual methods
00070     */
00071     //virtual std::shared_ptr<ULSTER> _cast(std::shared_ptr<OSTM> _object);
00072     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00073     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00074     virtual void toString();
00075
00076     /*
00077     * Implement BANK virtual methods
00078     */
00079     virtual void SetAddress(std::string address);
00080     virtual std::string GetAddress() const;
00081     virtual void SetBalance(double balance);
00082     virtual double GetBalance() const;
00083     virtual void SetAccountNumber(int accountNumber);
00084     virtual int GetAccountNumber() const;
00085     virtual void SetLastName(std::string lastName);
00086     virtual std::string GetLastName() const;
00087     virtual void SetFirstName(std::string firstName);
00088     virtual std::string GetFirstName() const;
00089     virtual void SetFullname(std::string fullname);
00090     virtual std::string GetFullname() const;
00091 private:
00092     std::string fullname;
00093     std::string firstName;
00094     std::string lastName;
00095     int accountNumber;
00096     double balance;
00097     std::string address;
00098
00099 };
00100
00101 #endif /* ULSTER_H */
00102

```

7.65 UNBL.cpp File Reference

```
#include "UNBL.h"
```

Include dependency graph for UNBL.cpp:



7.66 UNBL.cpp

```

00001 /*
00002  * File:    UNBL.cpp
00003  * Author:  Zoltan Fuzesi
00004  * IT Carlow : C00197361
00005  *
00006  * Created on January 17, 2018, 8:02 PM
00007  */
00008
00009 #include "UNBL.h"
00010
00011 UNBL::UNBL(const UNBL& orig) {
00012 }
00013
00014 UNBL::~UNBL() {
00015 }
00021 std::shared_ptr<OSTM> UNBL::getBaseCopy(std::shared_ptr<OSTM> object)
00022 {
00023     std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
00024     std::shared_ptr<BANK> obj(new UNBL(objTO, object->Get_Version(), object->Get_Unique_ID()));
00025     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
00026
00027     return ostm_obj;
00033 void UNBL::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){
00034
00035     std::shared_ptr<UNBL> objTO = std::dynamic_pointer_cast<UNBL>(to);
00036     std::shared_ptr<UNBL> objFROM = std::dynamic_pointer_cast<UNBL>(from);
00037     objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
00038     objTO->Set_Version(objFROM->Get_Version());
00039     objTO->SetAccountNumber(objFROM->GetAccountNumber());
  
```

```

00040     objTO->SetBalance(objFROM->GetBalance());
00041
00042 }
00046 //std::shared_ptr<UNBL> UNBL::_cast(std::shared_ptr<OSTM> _object) {
00047 //
00048 //     return static_cast<std::shared_ptr<UNBL>>(_object);
00049 //}
00053 void UNBL::toString()
00054 {
00055     std::cout << "\nUNBL BANK" << "\nUnique ID : " << this->Get_Unique_ID() << "\nInt account : " << this->
GetAccountNumber() << "\nDouble value : " << this->GetBalance() << "\nFirst name:
    " << this->GetFirstName() << "\nLast name : " << this->GetLastName() << "\nVersion
    number : " << this->Get_Version() << std::endl;
00056 }
00057
00058 void UNBL::SetAddress(std::string address) {
00059     this->address = address;
00060 }
00061
00062 std::string UNBL::GetAddress() const {
00063     return address;
00064 }
00065
00066 void UNBL::SetBalance(double balance) {
00067     this->balance = balance;
00068 }
00069
00070 double UNBL::GetBalance() const {
00071     return balance;
00072 }
00073
00074 void UNBL::SetAccountNumber(int accountNumber) {
00075     this->accountNumber = accountNumber;
00076 }
00077
00078 int UNBL::GetAccountNumber() const {
00079     return accountNumber;
00080 }
00081
00082 void UNBL::SetLastName(std::string lastName) {
00083     this->lastName = lastName;
00084 }
00085
00086 std::string UNBL::GetLastName() const {
00087     return lastName;
00088 }
00089
00090 void UNBL::SetFirstName(std::string firstName) {
00091     this->firstName = firstName;
00092 }
00093
00094 std::string UNBL::GetFirstName() const {
00095     return firstName;
00096 }
00097
00098 void UNBL::SetFullname(std::string fullname) {
00099     this->fullname = fullname;
00100 }
00101
00102 std::string UNBL::GetFullname() const {
00103     return fullname;
00104 }
00105

```

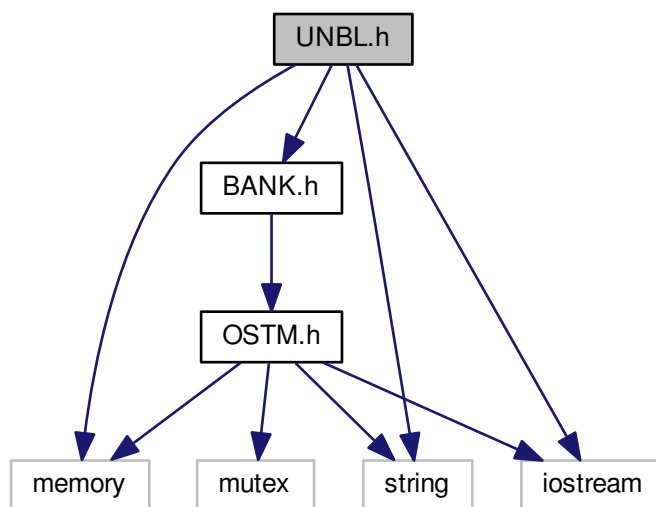
7.67 UNBL.h File Reference

```

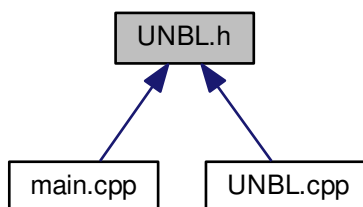
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>

```

Include dependency graph for UNBL.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [UNBL](#)

7.68 UNBL.h

```

00001
00002 /*
00003  * File:   UNBL.h
00004  * Author: Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
  
```



```

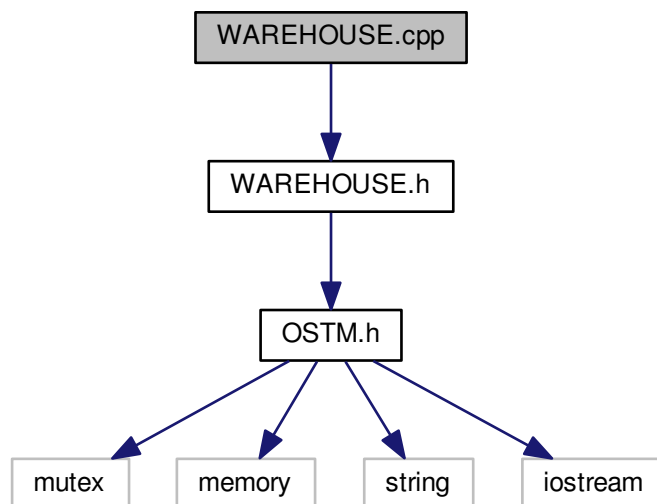
00008  */
00009
00010 #ifndef UNBL_H
00011 #define UNBL_H
00012 #include "BANK.h"
00013 #include <string>
00014 #include <memory>
00015 #include <iostream>
00019 class UNBL : public BANK {
00020 public:
00024     UNBL() : BANK() {
00025         this->accountNumber = 0;
00026         this->balance = 50;
00027         this->firstName = "Joe";
00028         this->lastName = "Blog";
00029         this->address = "High street, Carlow";
00030         this->fullname = firstName + " " + lastName;
00031     };
00035     UNBL(int accountNumber, double balance, std::string firstName, std::string lastName, std::string
address) : BANK() {
00036         this->accountNumber = accountNumber;
00037         this->balance = balance;
00038         this->firstName = firstName;
00039         this->lastName = lastName;
00040         this->address = address;
00041         this->fullname = firstName + " " + lastName;
00042     };
00046     UNBL(std::shared_ptr<BANK> obj, int _version, int _unique_id) : BANK(_version, _unique_id) {
00047
00048         this->accountNumber = obj->GetAccountNumber();
00049         this->balance = obj->GetBalance();
00050         this->firstName = obj->GetFirstName();
00051         this->lastName = obj->GetLastName();
00052         this->address = obj->GetAddress();
00053         this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
00054     };
00058     UNBL(const UNBL& orig);
00062     UNBL operator=(const UNBL& orig) {};
00066     virtual ~UNBL();
00067
00068     /*
00069     * Implement OSTM virtual methods
00070     */
00071     //virtual std::shared_ptr<UNBL> _cast(std::shared_ptr<OSTM> _object);
00072     virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
00073     virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
00074     virtual void toString();
00075
00076     /*
00077     * Implement BANK virtual methods
00078     */
00079     virtual void SetAddress(std::string address);
00080     virtual std::string GetAddress() const;
00081     virtual void SetBalance(double balance);
00082     virtual double GetBalance() const;
00083     virtual void SetAccountNumber(int accountNumber);
00084     virtual int GetAccountNumber() const;
00085     virtual void SetLastName(std::string lastName);
00086     virtual std::string GetLastName() const;
00087     virtual void SetFirstName(std::string firstName);
00088     virtual std::string GetFirstName() const;
00089     virtual void SetFullname(std::string fullname);
00090     virtual std::string GetFullname() const;
00091 private:
00092     std::string fullname;
00093     std::string firstName;
00094     std::string lastName;
00095     int accountNumber;
00096     double balance;
00097     std::string address;
00098
00099 };
00100
00101 #endif /* UNBL_H */
00102

```

7.69 WAREHOUSE.cpp File Reference

```
#include "WAREHOUSE.h"
```

Include dependency graph for WAREHOUSE.cpp:



7.70 WAREHOUSE.cpp

```

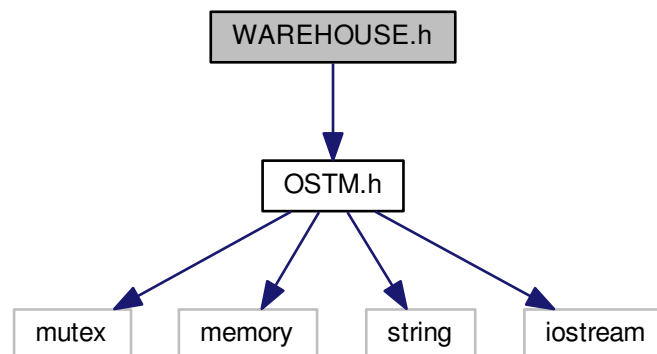
00001
00002 /*
00003  * File:   WAREHOUSE.cpp
00004  * Author: Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #include "WAREHOUSE.h"
00011
00012 WAREHOUSE::WAREHOUSE(const WAREHOUSE& orig) {
00013 }
00014
00015 WAREHOUSE::~WAREHOUSE() {
00016 }
00017

```

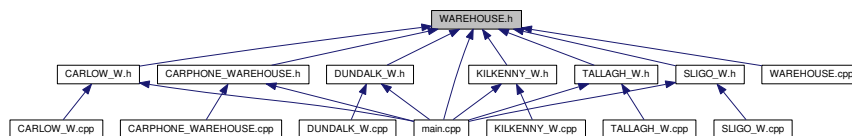
7.71 WAREHOUSE.h File Reference

```
#include "OSTM.h"
```

Include dependency graph for WAREHOUSE.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [WAREHOUSE](#)

7.72 WAREHOUSE.h

```

00001
00002 /*
00003  * File:   WAREHOUSE.h
00004  * Author: Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
00008  */
00009
00010 #ifndef WAREHOUSE_H
00011 #define WAREHOUSE_H
00012 #include "OSTM.h"
00016 class WAREHOUSE : public OSTM {
00017 public:
00021     WAREHOUSE():OSTM(){
00022     };
00023     };
00027     WAREHOUSE(int _version, int _unique_id) : OSTM(_version, _unique_id){
00028     };
00029     };
00033     WAREHOUSE(const WAREHOUSE& orig);
00037     virtual ~WAREHOUSE();
00038
00039     /*
  
```

```
00040      * WAREHOUSE BASE METHODS
00041      */
00042
00043      virtual void SetNumber_of_alcatel(int _number_of_alcatel) = 0;
00044      virtual int GetNumber_of_alcatel() = 0;
00045      virtual void SetNumber_of_nokia(int _number_of_nokia) = 0;
00046      virtual int GetNumber_of_nokia() = 0;
00047      virtual void SetNumber_of_huawei(int _number_of_huawei) = 0;
00048      virtual int GetNumber_of_huawei() = 0;
00049      virtual void SetNumber_of_sony(int _number_of_sony) = 0;
00050      virtual int GetNumber_of_sony() = 0;
00051      virtual void SetNumber_of_samsung(int _number_of_samsung) = 0;
00052      virtual int GetNumber_of_samsung() = 0;
00053      virtual void SetNumber_of_iphones(int _number_of_iphones) = 0;
00054      virtual int GetNumber_of_iphones() = 0;
00055      virtual void SetShop_name(std::string _shop_name) = 0;
00056      virtual std::string GetShop_name() = 0;
00057      virtual void SetShop_address(std::string _shop_address) = 0;
00058      virtual std::string GetShop_address() = 0;
00059
00060 private:
00061
00062 };
00063
00064 #endif /* WAREHOUSE_H */
00065
```