C++ Software Transactional memory

Zoltan Fuzesi

C00197361 IT Carlow

Supervisor : Joe Kehoe

Contents

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 demostrate 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 speciffic behaviours.

3 Hierarchical Index

3.1 Class Hierarchy

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

4 Class Index

4.1 Class List

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

5 File Index

5.1 File List

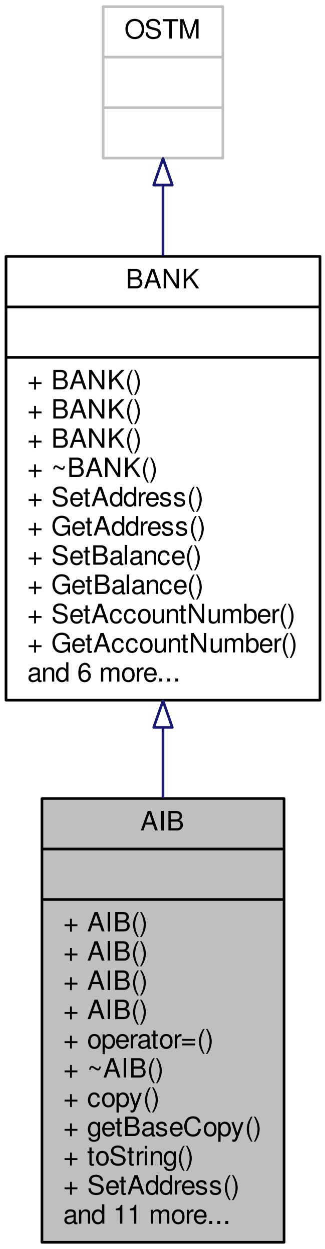
Here is a list of all files with brief descriptions:

6 Class Documentation

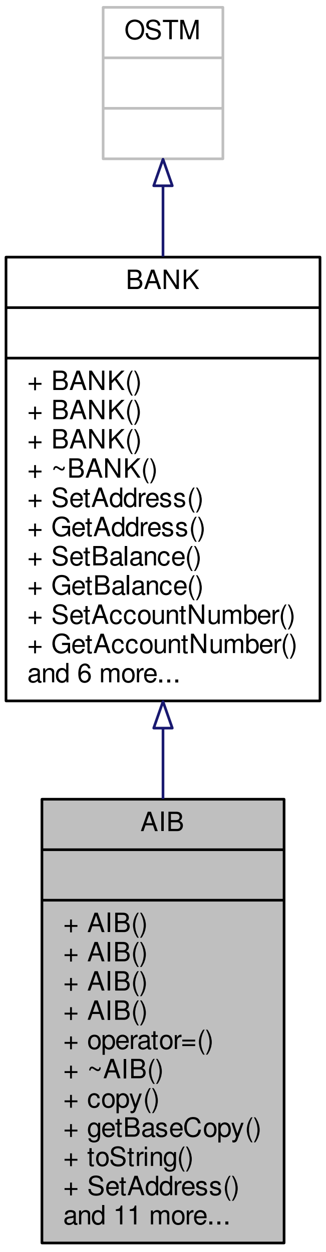
6.1 AIB Class Reference

```
#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]

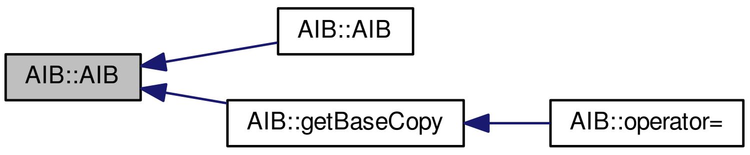
Constructor

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.

```
                                                                                        :
      BANK()
    {
        this->accountNumber = accountNumber;
        this->balance = balance;
        this->firstName = firstName;
        this->lastName = lastName;
        this->address = address;
        this->fullname = firstName + " " + lastName;
    };
```

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

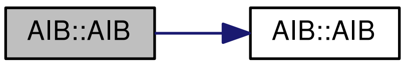
Custom constructor, used by the library for deep copying

Definition at line 48 of file AIB.h.

References AIB().

```
                                                             : BANK(_version, _unique_id)
    {

        this->accountNumber = obj->GetAccountNumber();
        this->balance = obj->GetBalance();
        this->firstName = obj->GetFirstName();
        this->lastName = obj->GetLastName();
        this->address = obj->GetAddress();
        this->fullname = obj->GetFirstName() + " " + obj->GetLastName();

    };
```

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.

```
                         {
}
```

6.1.2.5 AIB::∼AIB ( ) [virtual]

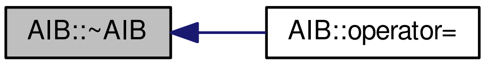
de-constructor

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

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

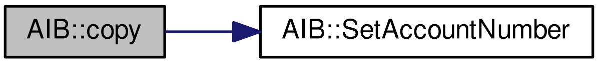
Definition at line 37 of file AIB.cpp.

References SetAccountNumber().

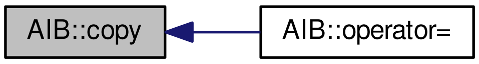
Referenced by operator=().

```
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]

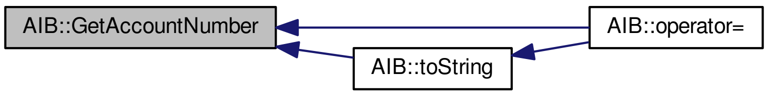
Implements BANK.

Definition at line 81 of file AIB.cpp.

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

```
                              {
    return accountNumber;
}
```

Here is the caller graph for this function:



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

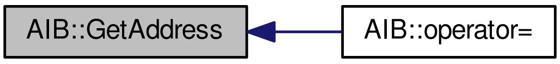
Implements BANK.

Definition at line 65 of file AIB.cpp.

Referenced by operator=().

```
                                   {
    return address;
}
```

Here is the caller graph for this function:



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

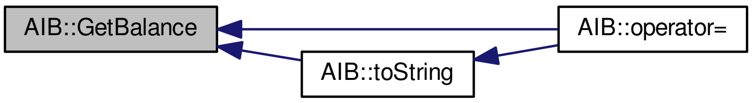
Implements BANK.

Definition at line 73 of file AIB.cpp.

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

```
                                {
    return balance;
}
```

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

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

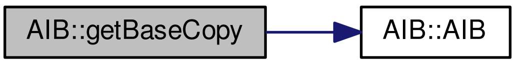
Definition at line 24 of file AIB.cpp.

References AIB().

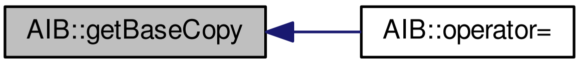
Referenced by operator=().

```
{

    std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
    std::shared_ptr<BANK> obj(new AIB(objTO, object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
```

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]

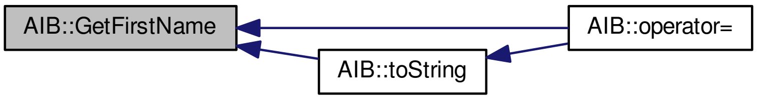
Implements BANK.

Definition at line 97 of file AIB.cpp.

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

```
                                  {
    return firstName;
}
```

Here is the caller graph for this function:



6.1.3.7 std::string AIB::GetFullname ( ) const [virtual]

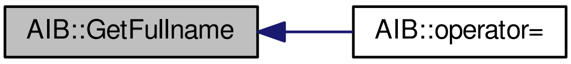
Implements BANK.

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]

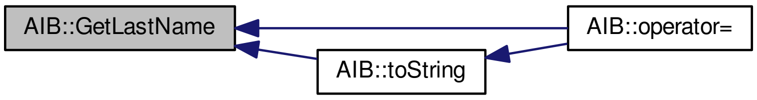
Implements BANK.

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]

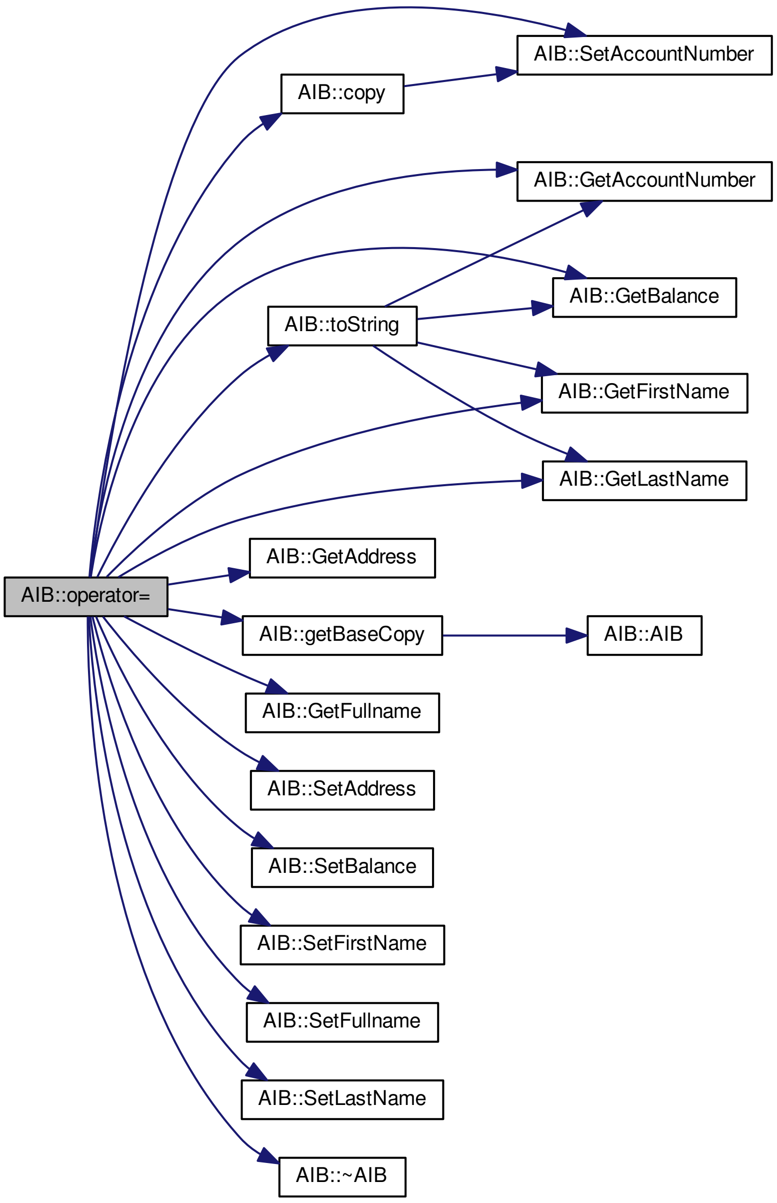
Operator

Definition at line 66 of file AIB.h.

References copy(), GetAccountNumber(), GetAddress(), GetBalance(), getBaseCopy(), GetFirstName(), Get←
Fullname(), 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]

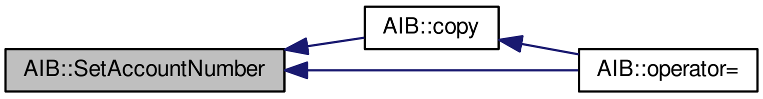
Implements BANK.

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]

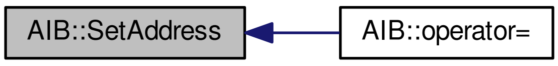
Implements BANK.

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]

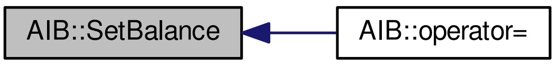
Implements BANK.

Definition at line 69 of file AIB.cpp.

Referenced by operator=().

```
                                        {
    this->balance = balance;
}
```

Here is the caller graph for this function:



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

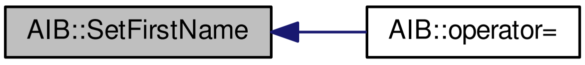
Implements BANK.

Definition at line 93 of file AIB.cpp.

Referenced by operator=().

```
                                                  {
    this->firstName = firstName;
}
```

Here is the caller graph for this function:



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

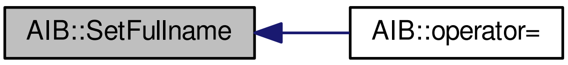
Implements BANK.

Definition at line 101 of file AIB.cpp.

Referenced by operator=().

```
                                    {
    this->fullname = fullname;
}
```

Here is the caller graph for this function:



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

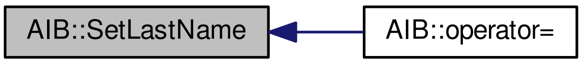
Implements BANK.

Definition at line 85 of file AIB.cpp.

Referenced by operator=().

```
                                    {
    this->lastName = lastName;
}
```

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

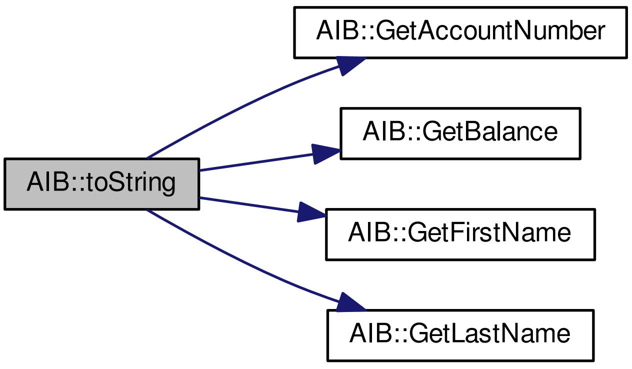
Definition at line 56 of file AIB.cpp.

References GetAccountNumber(), GetBalance(), GetFirstName(), and GetLastName().

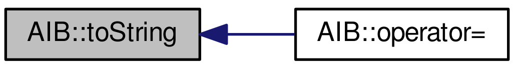
Referenced by operator=().

```
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:



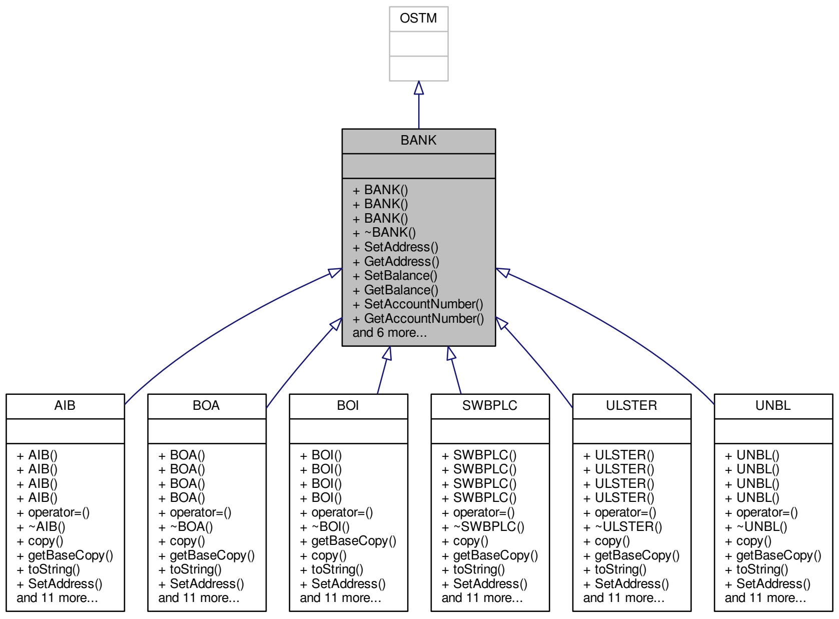
The documentation for this class was generated from the following files:

- AIB.h
- AIB.cpp

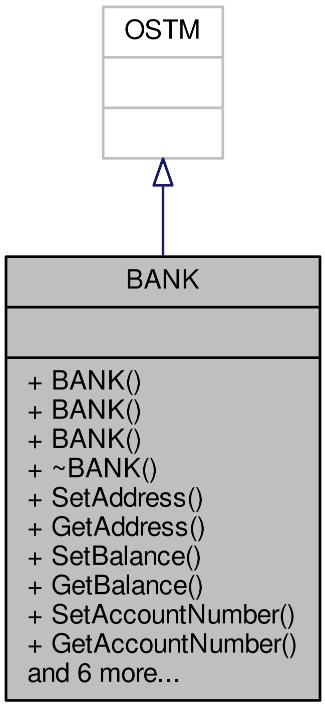
6.2 BANK Class Reference

```
#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]

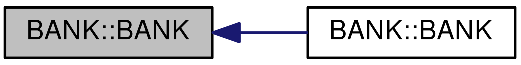
Constructor

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]

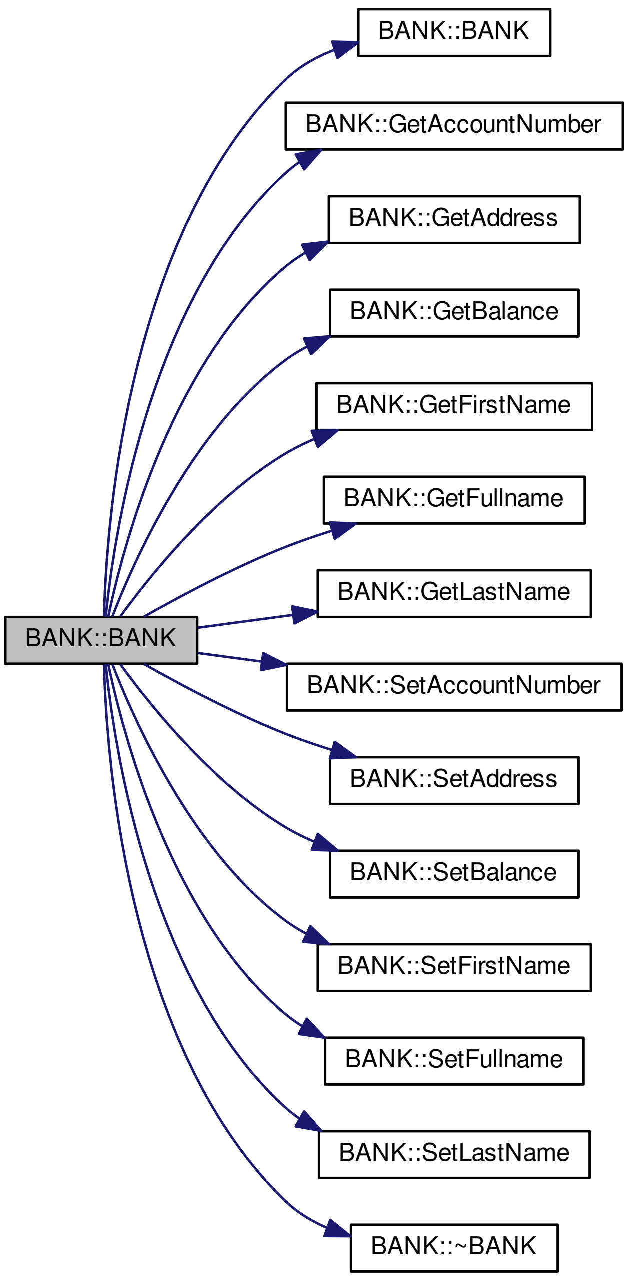
Custom Constructor

Definition at line 29 of file BANK.h.

References BANK(), GetAccountNumber(), GetAddress(), GetBalance(), GetFirstName(), GetFullname(), Get↩ LastName(), 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]

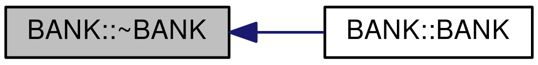
de-constructor

Definition at line 14 of file BANK.cpp.

Referenced by BANK().

```
00014                {
00015 }
```

Here is the caller graph for this function:



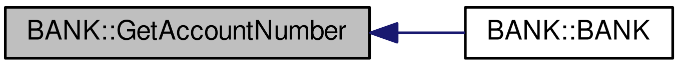
6.2.3 Member Function Documentation

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:

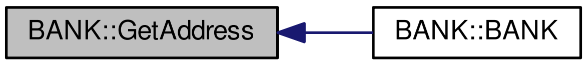


6.2.3.2 virtual std::string BANK::GetAddress ( ) const [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:

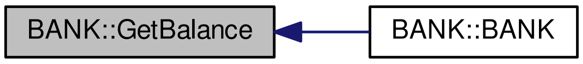


6.2.3.3 virtual double BANK::GetBalance ( ) const [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:

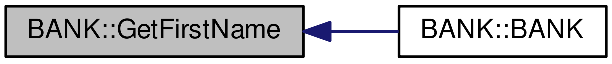


6.2.3.4 virtual std::string BANK::GetFirstName ( ) const [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:

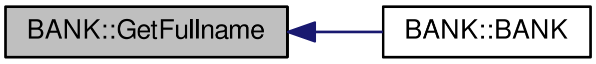


6.2.3.5 virtual std::string BANK::GetFullname ( ) const [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:

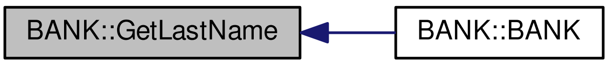


6.2.3.6 virtual std::string BANK::GetLastName ( ) const [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:

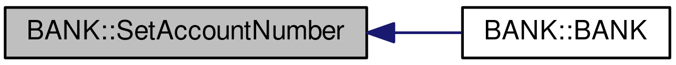


6.2.3.7 virtual void BANK::SetAccountNumber (int *accountNumber*) [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:

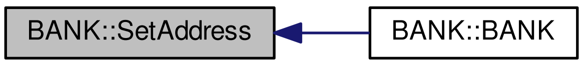


6.2.3.8 virtual void BANK::SetAddress (std::string *address*) [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:

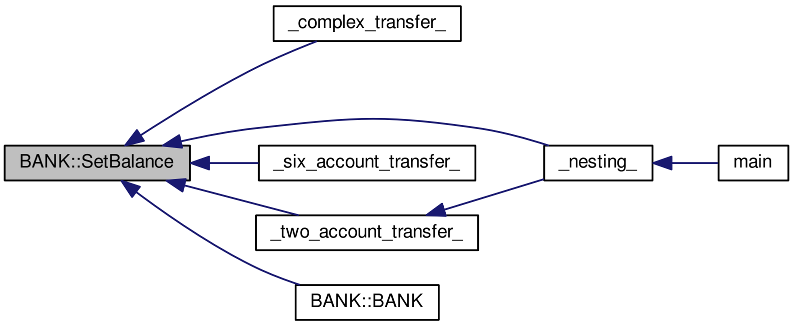


6.2.3.9 virtual void BANK::SetBalance (double *balance*) [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by _complex_transfer_(), _nesting_(), _six_account_transfer_(), _two_account_transfer_(), and BAN↩K().

Here is the caller graph for this function:

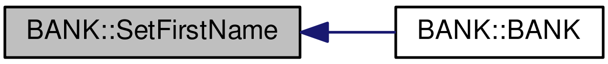


6.2.3.10 virtual void BANK::SetFirstName (std::string *firstName*) [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:

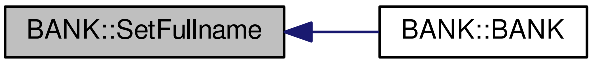


6.2.3.11 virtual void BANK::SetFullname (std::string *fullname*) [pure virtual]

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:

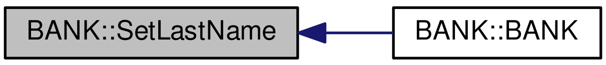


6.2.3.12 virtual void BANK::SetLastName (std::string *lastName*) `[pure virtual]`

Implemented in AIB, BOA, BOI, SWBPLC, ULSTER, and UNBL.

Referenced by BANK().

Here is the caller graph for this function:



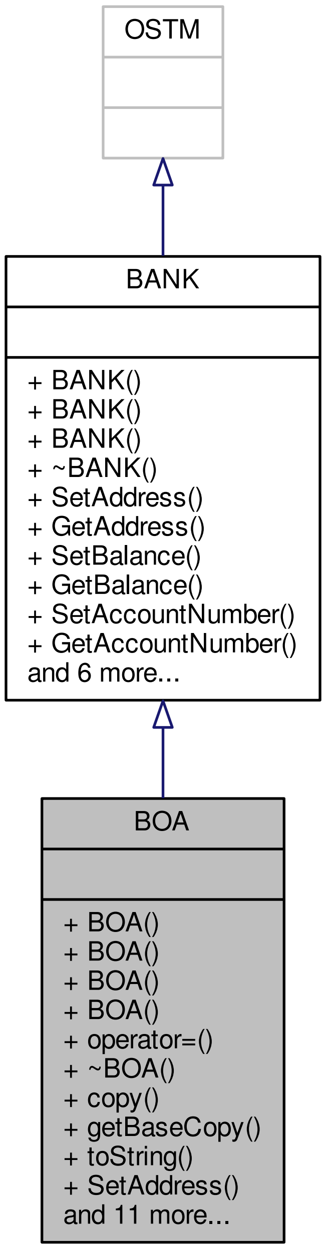
The documentation for this class was generated from the following files:

- BANK.h
- BANK.cpp

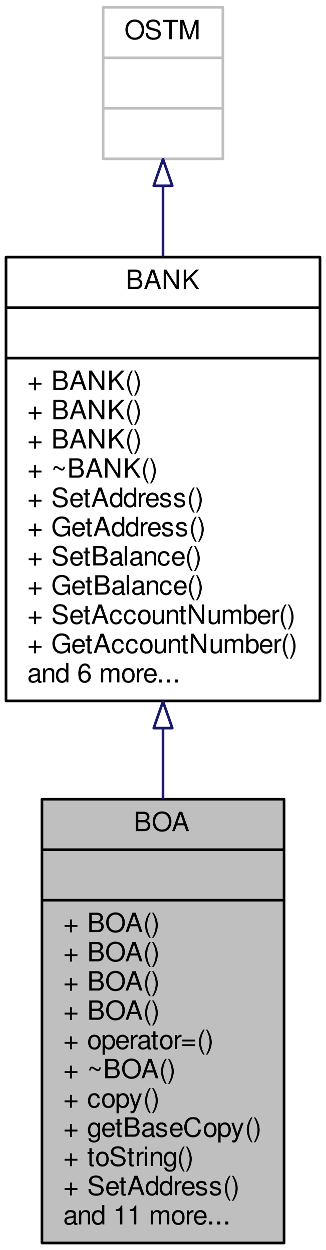
6.3 BOA Class Reference

```
#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]

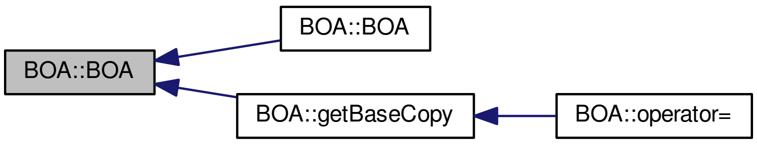
Constructor

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.

```
                                                                                :
      BANK() {
            this->accountNumber = accountNumber;
            this->balance = balance;
            this->firstName = firstName;
            this->lastName = lastName;
            this->address = address;
            this->fullname = firstName + " " + lastName;
        };
```

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

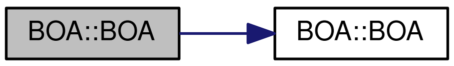
Custom constructor, used by the library for deep copying

Definition at line 46 of file BOA.h.

References BOA().

```
                                                        : BANK(_version, _unique_id) {

            this->accountNumber = obj->GetAccountNumber();
            this->balance = obj->GetBalance();
            this->firstName = obj->GetFirstName();
            this->lastName = obj->GetLastName();
            this->address = obj->GetAddress();
            this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
        };
```

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.

```
                         {
 }
```

6.3.2.5 BOA::∼BOA ( ) [virtual]

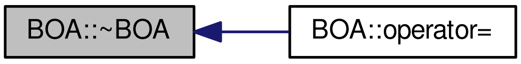
de-constructor

Definition at line 15 of file BOA.cpp.

Referenced by operator=().

```
          {
}
```

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

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

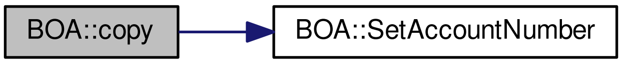
Definition at line 34 of file BOA.cpp.

References SetAccountNumber().

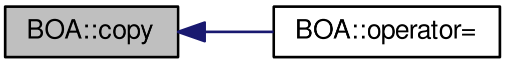
Referenced by operator=().

```
                                                              {

    std::shared_ptr<BOA> objTO = std::dynamic_pointer_cast<BOA>(to);
    std::shared_ptr<BOA> objFROM = std::dynamic_pointer_cast<BOA>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());

}
```

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]

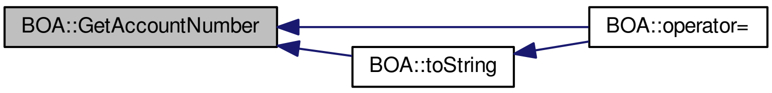
Implements BANK.

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]

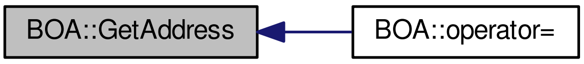
Implements BANK.

Definition at line 64 of file BOA.cpp.

Referenced by operator=().

```
                                {
    return address;
}
```

Here is the caller graph for this function:



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

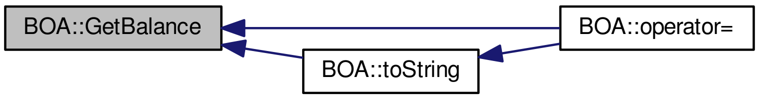
Implements BANK.

Definition at line 72 of file BOA.cpp.

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

```
                              {
    return balance;
}
```

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

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

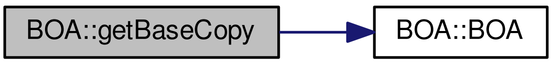
Definition at line 22 of file BOA.cpp.

References BOA().

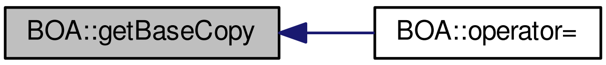
Referenced by operator=().

```
{
        std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
    std::shared_ptr<BANK> obj(new BOA(objTO,object->Get_Version(),object->Get_Unique_ID()));
        std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
```

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]

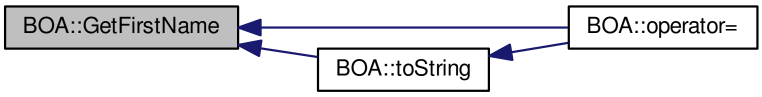
Implements BANK.

Definition at line 96 of file BOA.cpp.

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

```
                                 {
    return firstName;
}
```

Here is the caller graph for this function:



6.3.3.7 std::string BOA::GetFullname ( ) const [virtual]

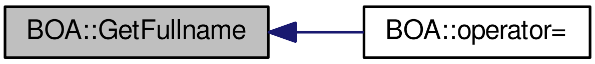
Implements BANK.

Definition at line 104 of file BOA.cpp.

Referenced by operator=().

```
                              {
    return fullname;
}
```

Here is the caller graph for this function:



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

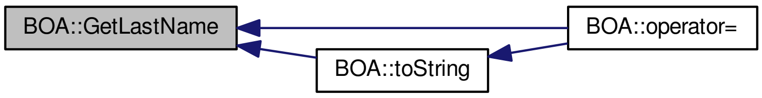
Implements BANK.

Definition at line 88 of file BOA.cpp.

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

```
                              {
    return lastName;
}
```

Here is the caller graph for this function:



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

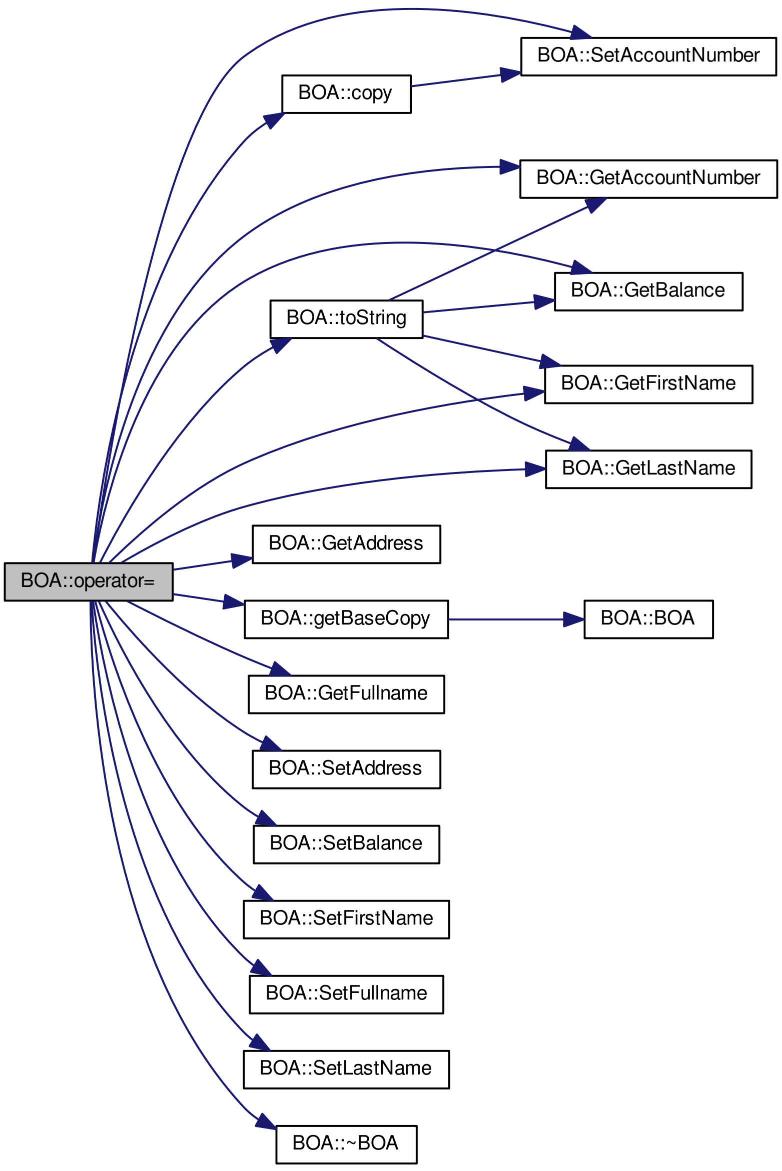
Operator

Definition at line 64 of file BOA.h.

References copy(), GetAccountNumber(), GetAddress(), GetBalance(), getBaseCopy(), GetFirstName(), Get←
Fullname(), 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]

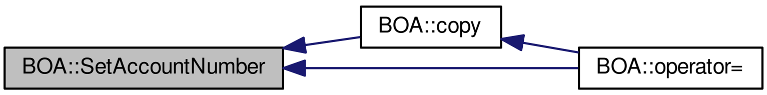
Implements BANK.

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]

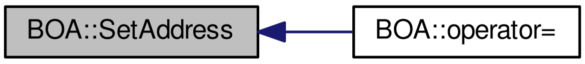
Implements BANK.

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]

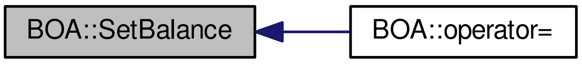
Implements BANK.

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]

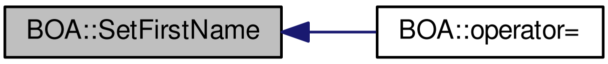
Implements BANK.

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]

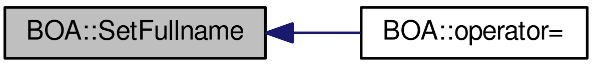
Implements BANK.

Definition at line 100 of file BOA.cpp.

Referenced by operator=().

```
{
    this->fullname = fullname;
}
```

Here is the caller graph for this function:



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

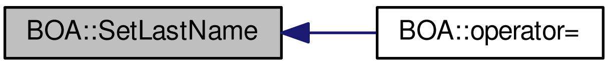
Implements BANK.

Definition at line 84 of file BOA.cpp.

Referenced by operator=().

```
{
    this->lastName = lastName;
}
```

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

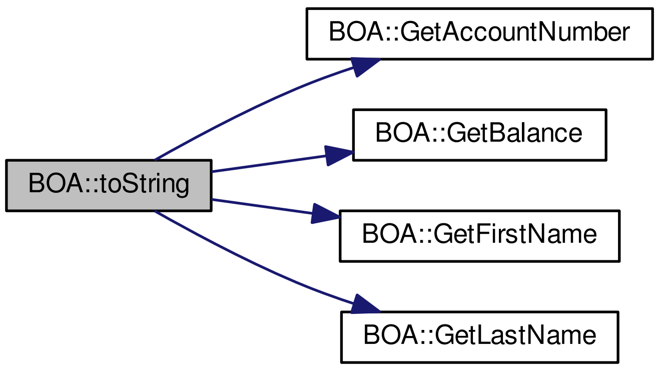
Definition at line 54 of file BOA.cpp.

References GetAccountNumber(), GetBalance(), GetFirstName(), and GetLastName().

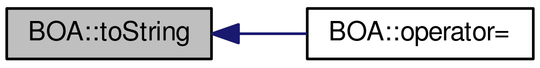
Referenced by operator=().

```
{
   // 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;
     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;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



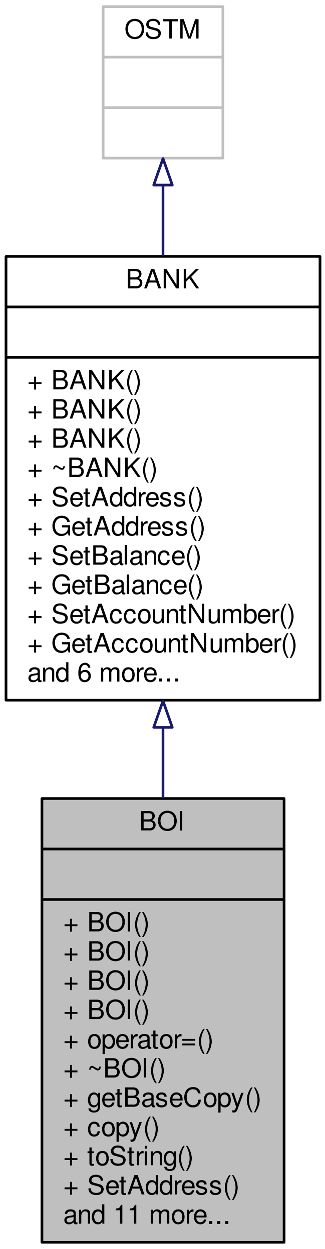
The documentation for this class was generated from the following files:

- BOA.h
- BOA.cpp

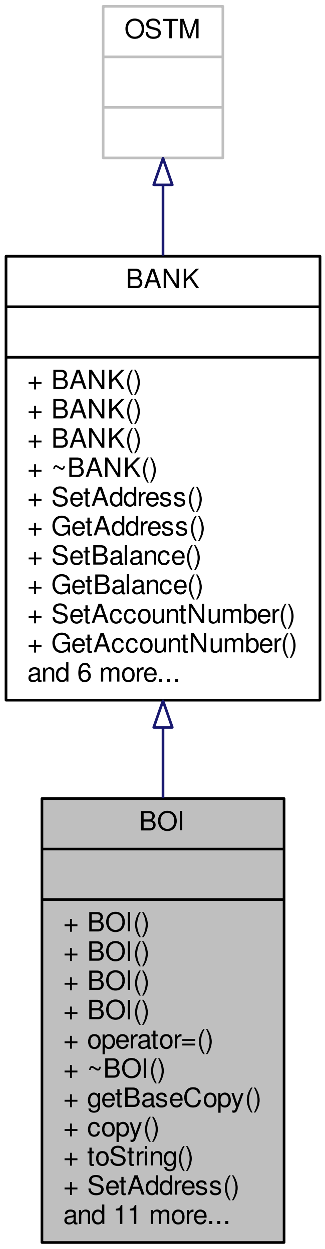
6.4 BOI Class Reference

```
#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]

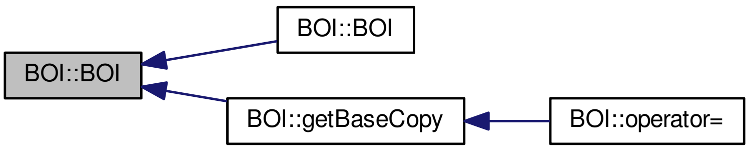
Constructor

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.

```
                                                                                                                            :
      BANK()
          {
              this->accountNumber = accountNumber;
              this->balance = balance;
              this->firstName = firstName;
              this->lastName = lastName;
              this->address = address;
              this->fullname = firstName + " " + lastName;
          };
```

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

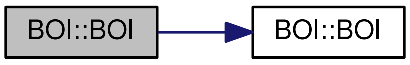
Custom constructor, used by the library for deep copying

Definition at line 49 of file BOI.h.

References BOI().

```
                                                                            : BANK(_version, _unique_id)
          {
              this->accountNumber = obj->GetAccountNumber();
              this->balance = obj->GetBalance();
              this->firstName = obj->GetFirstName();
              this->lastName = obj->GetLastName();
              this->address = obj->GetAddress();
              this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
          };
```

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.

```
                              {
 }
```

6.4.2.5 BOI::∼BOI ( ) `[virtual]`

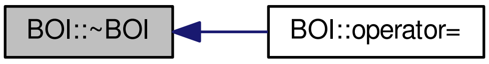
de-constructor

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

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

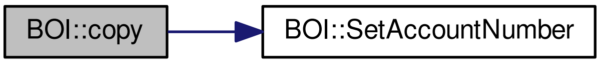
Definition at line 35 of file BOI.cpp.

References SetAccountNumber().

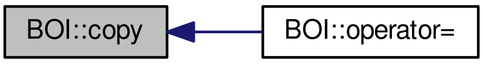
Referenced by operator=().

```
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]

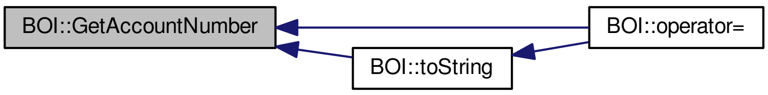
Implements BANK.

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]

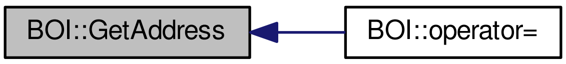
Implements BANK.

Definition at line 62 of file BOI.cpp.

Referenced by operator=().

```
                                   {
    return address;
}
```

Here is the caller graph for this function:



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

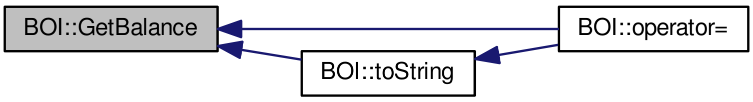
Implements BANK.

Definition at line 70 of file BOI.cpp.

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

```
                                {
    return balance;
}
```

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

objTO	is a BANK type pointer for casting
obj	is a BANK* return type

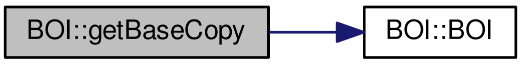
Definition at line 22 of file BOI.cpp.

References BOI().

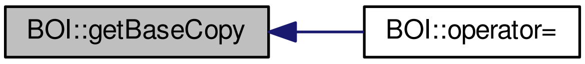
Referenced by operator=().

```
{

    std::shared_ptr<BOI> objTO = std::dynamic_pointer_cast<BOI>(object);
    std::shared_ptr<BOI> obj(new BOI(objTO,object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
```

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]

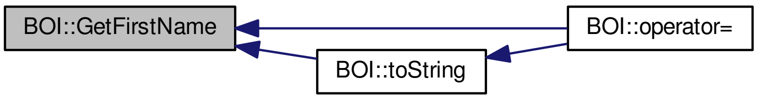
Implements BANK.

Definition at line 94 of file BOI.cpp.

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

```
                                 {
    return firstName;
}
```

Here is the caller graph for this function:



6.4.3.7 std::string BOI::GetFullname ( ) const [virtual]

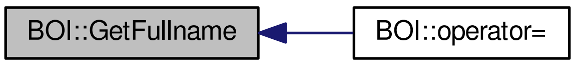
Implements BANK.

Definition at line 102 of file BOI.cpp.

Referenced by operator=().

```
                                {
    return fullname;
}
```

Here is the caller graph for this function:



6.4.3.8 std::string BOI::GetLastName ( ) const [virtual]

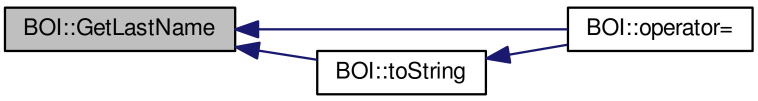
Implements BANK.

Definition at line 86 of file BOI.cpp.

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

```
                                {
    return lastName;
}
```

Here is the caller graph for this function:



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

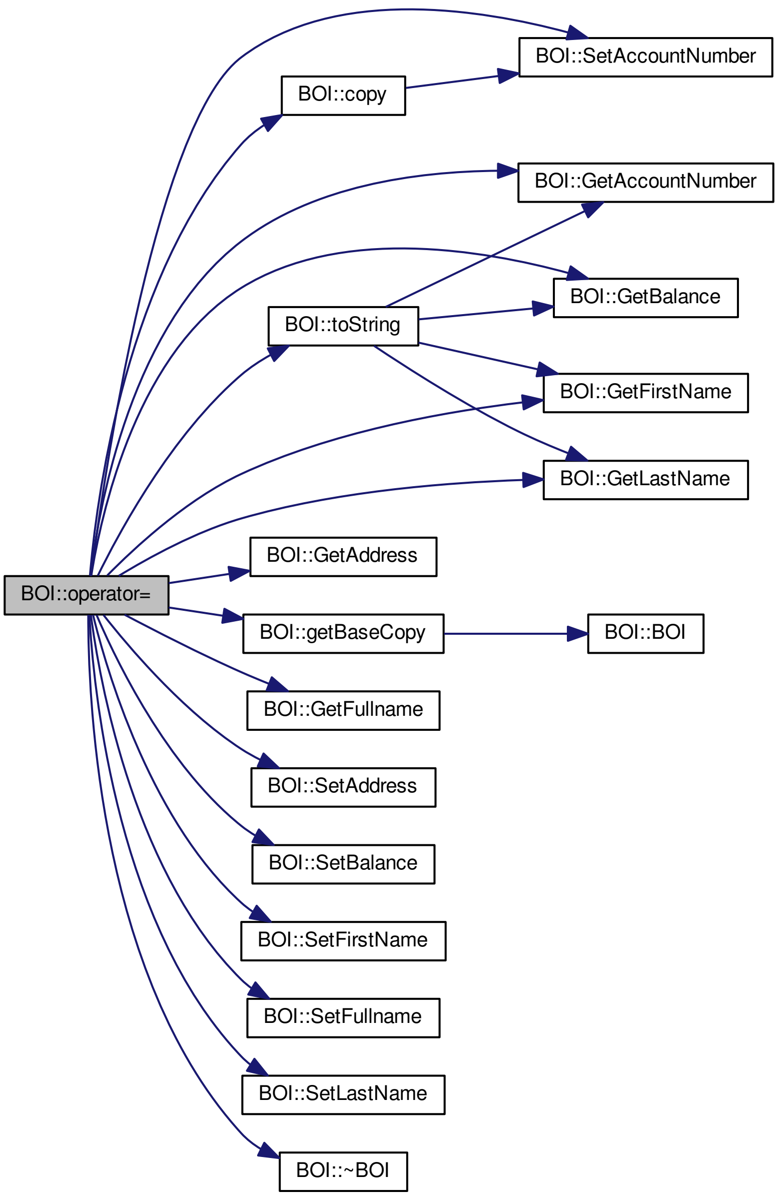
Operator

Definition at line 65 of file BOI.h.

References copy(), GetAccountNumber(), GetAddress(), GetBalance(), getBaseCopy(), GetFirstName(), Get←
Fullname(), 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]

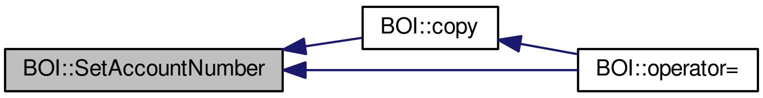
Implements BANK.

Definition at line 74 of file BOI.cpp.

Referenced by copy(), and operator=().

```
                                               {
    this->accountNumber = accountNumber;
}
```

Here is the caller graph for this function:



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

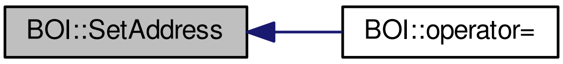
Implements BANK.

Definition at line 58 of file BOI.cpp.

Referenced by operator=().

```
                                         {
    this->address = address;
}
```

Here is the caller graph for this function:



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

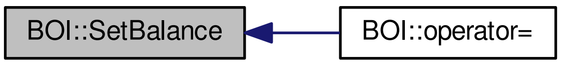
Implements BANK.

Definition at line 66 of file BOI.cpp.

Referenced by operator=().

```
                                {
    this->balance = balance;
}
```

Here is the caller graph for this function:



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

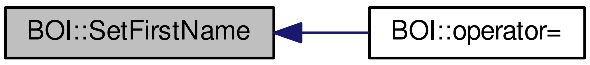
Implements BANK.

Definition at line 90 of file BOI.cpp.

Referenced by operator=().

```
                                    {
    this->firstName = firstName;
}
```

Here is the caller graph for this function:



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

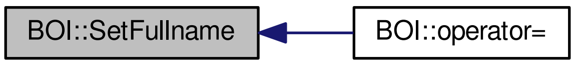
Implements BANK.

Definition at line 98 of file BOI.cpp.

Referenced by operator=().

```
                                        {
    this->fullname = fullname;
}
```

Here is the caller graph for this function:



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

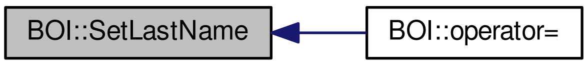
Implements BANK.

Definition at line 82 of file BOI.cpp.

Referenced by operator=().

```
                                        {
    this->lastName = lastName;
}
```

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

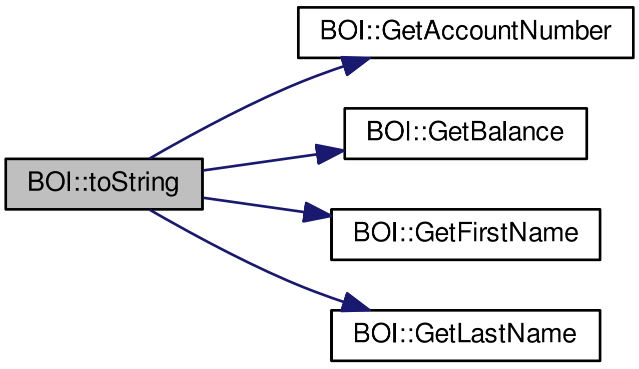
Definition at line 54 of file BOI.cpp.

References GetAccountNumber(), GetBalance(), GetFirstName(), and GetLastName().

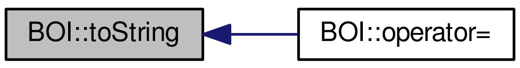
Referenced by operator=().

```
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:



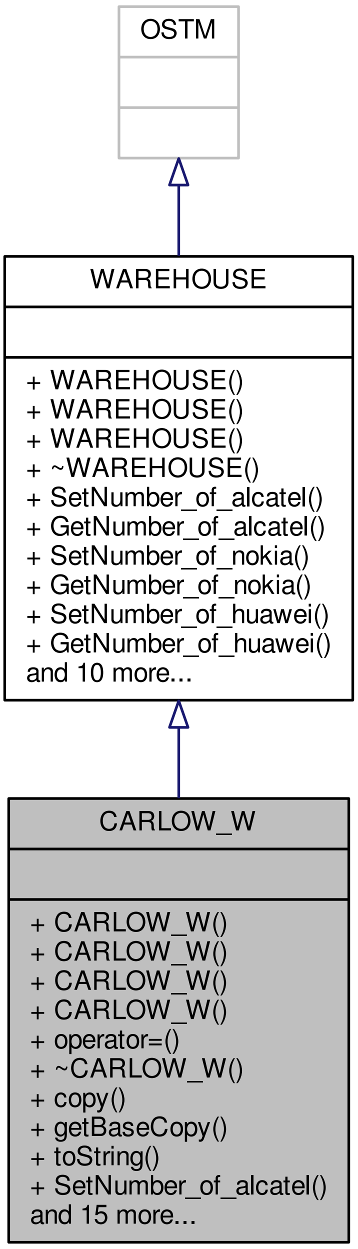
The documentation for this class was generated from the following files:

- BOI.h
- BOI.cpp

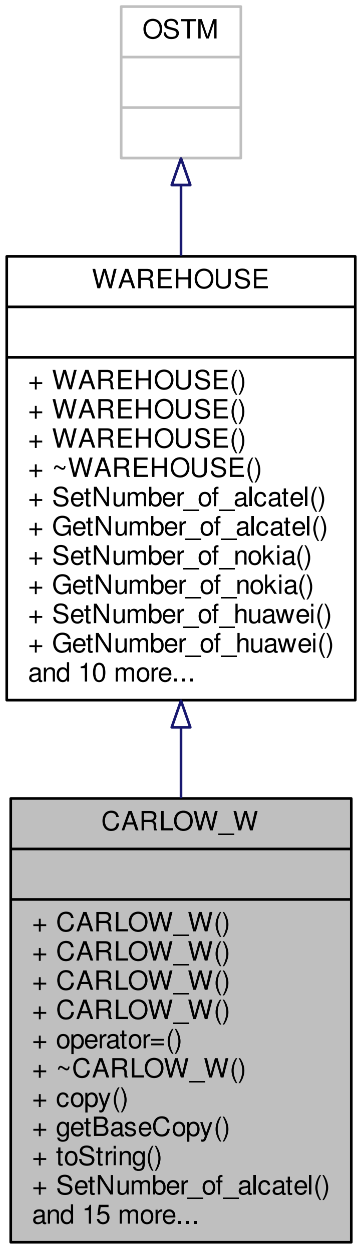
6.5 CARLOW_W Class Reference

```
#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]

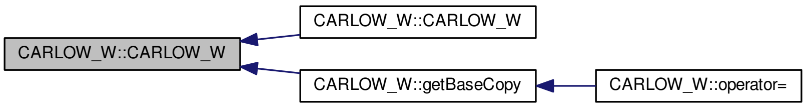
Constructor

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.

```
                        : WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
```

6.5.2.3 CARLOW_W::CARLOW_W (std::shared_ptr< WAREHOUSE > *obj*, int *_version*, int *_unique_id*) [inline]

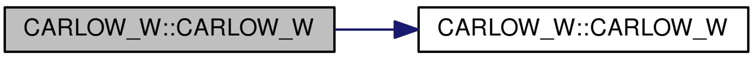
Custom constructor, used by the library for deep copying

Definition at line 55 of file CARLOW_W.h.

References CARLOW_W().

```
                                                                          :
      WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
```

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]

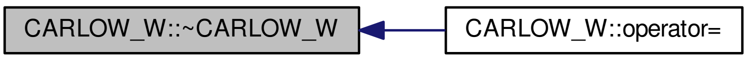
de-constructor

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

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

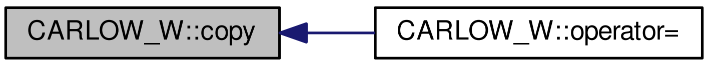
Definition at line 37 of file CARLOW_W.cpp.

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();
```

```
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
```

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

objTO	is a BANK type pointer for casting
obj	is a BANK∗ return type

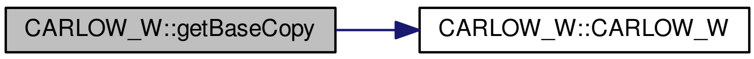
Definition at line 24 of file CARLOW_W.cpp.

References CARLOW_W().

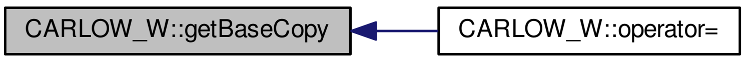
Referenced by operator=().

```
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new CARLOW_W(objTO, object->Get_Version(),object->Get_Unique_ID(
)));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
```

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]

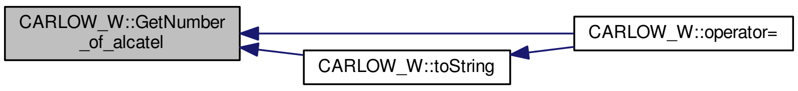
Implements WAREHOUSE.

Definition at line 75 of file CARLOW_W.cpp.

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

```
                                        {
    return _number_of_alcatel;
}
```

Here is the caller graph for this function:



6.5.3.4 int CARLOW_W::GetNumber_of_huawei ( ) [virtual]

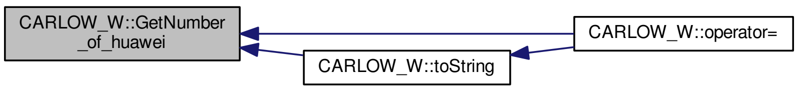
Implements WAREHOUSE.

Definition at line 91 of file CARLOW_W.cpp.

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

```
                                       {
    return _number_of_huawei;
}
```

Here is the caller graph for this function:



6.5.3.5 int CARLOW_W::GetNumber_of_iphones ( ) [virtual]

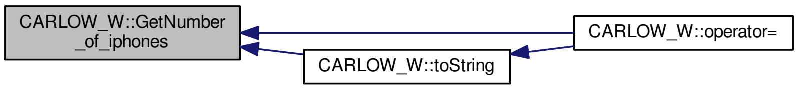
Implements WAREHOUSE.

Definition at line 115 of file CARLOW_W.cpp.

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

```
                                   {
    return _number_of_iphones;
}
```

Here is the caller graph for this function:



6.5.3.6 int CARLOW_W::GetNumber_of_nokia ( ) [virtual]

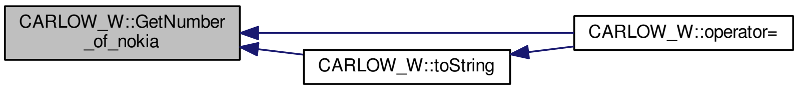
Implements WAREHOUSE.

Definition at line 83 of file CARLOW_W.cpp.

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

```
                                 {
    return _number_of_nokia;
}
```

Here is the caller graph for this function:



6.5.3.7 int CARLOW_W::GetNumber_of_samsung ( ) [virtual]

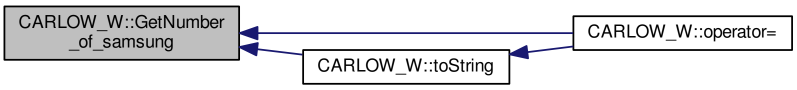
Implements WAREHOUSE.

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]

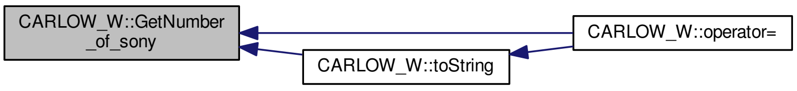
Implements WAREHOUSE.

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]

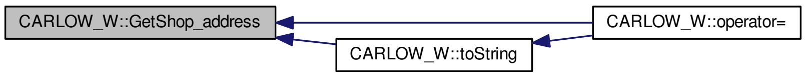
Implements WAREHOUSE.

Definition at line 131 of file CARLOW_W.cpp.

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

```
                                    {
    return _shop_address;
}
```

Here is the caller graph for this function:



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

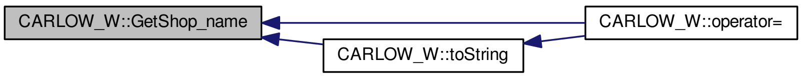
Implements WAREHOUSE.

Definition at line 123 of file CARLOW_W.cpp.

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

```
                                 {
    return _shop_name;
}
```

Here is the caller graph for this function:



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

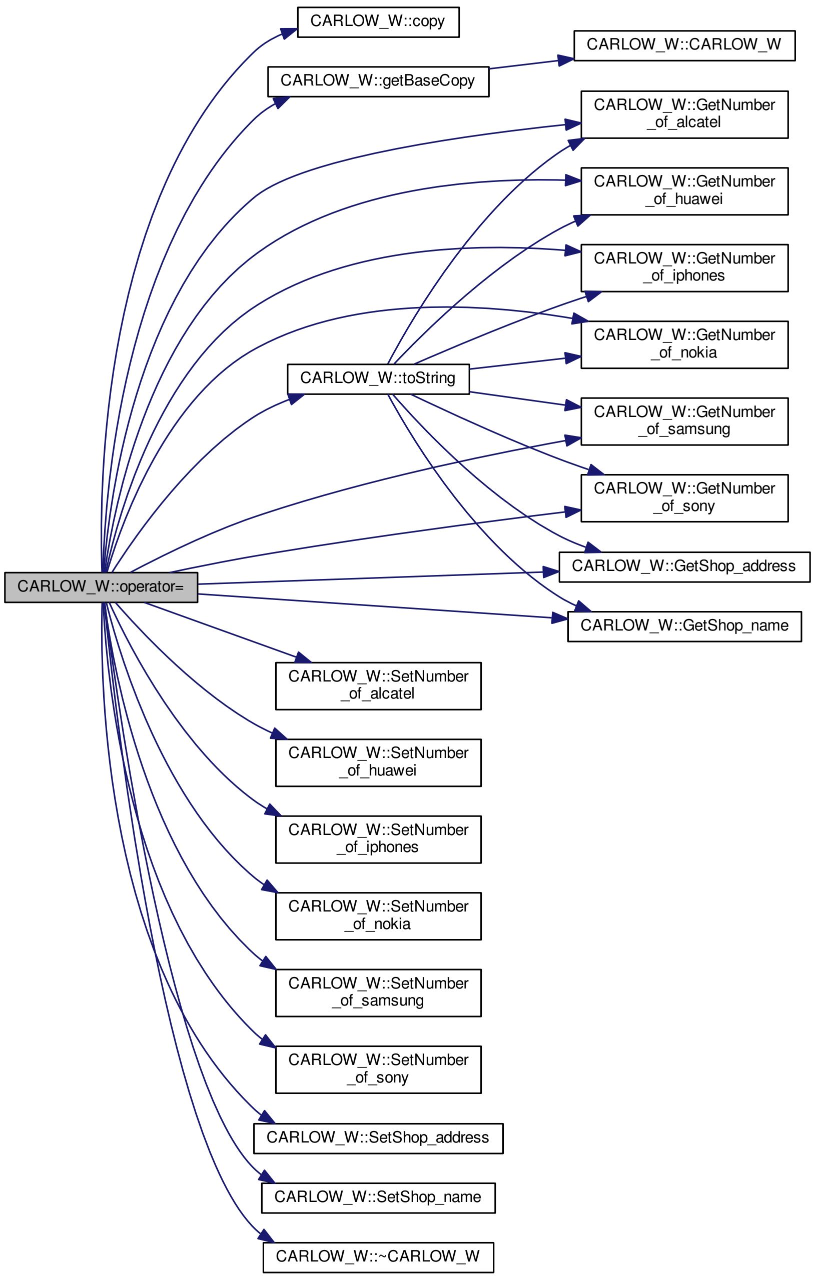
Operator

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:


CARLOW_W::copy
CARLOW_W::CARLOW_W
CARLOW_W::getBaseCopy
CARLOW_W::GetNumber_of_alcatel
CARLOW_W::GetNumber_of_huawei
CARLOW_W::GetNumber_of_iphones
CARLOW_W::GetNumber_of_nokia
CARLOW_W::toString
CARLOW_W::GetNumber_of_samsung
CARLOW_W::GetNumber_of_sony
CARLOW_W::GetShop_address
CARLOW_W::operator=
CARLOW_W::GetShop_name
CARLOW_W::SetNumber_of_alcatel
CARLOW_W::SetNumber_of_huawei
CARLOW_W::SetNumber_of_iphones
CARLOW_W::SetNumber_of_nokia
CARLOW_W::SetNumber_of_samsung
CARLOW_W::SetNumber_of_sony
CARLOW_W::SetShop_address
CARLOW_W::SetShop_name
CARLOW_W::~CARLOW_W

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

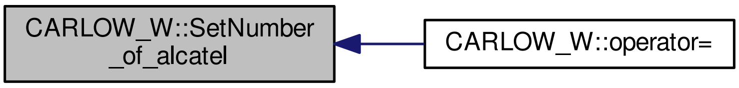
Implements WAREHOUSE.

Definition at line 71 of file CARLOW_W.cpp.

Referenced by operator=().

```
                                                          {
    this->_number_of_alcatel = _number_of_alcatel;
}
```

Here is the caller graph for this function:



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

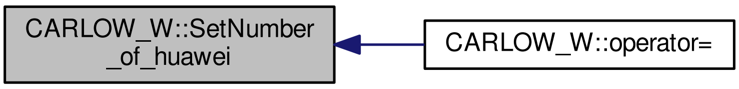
Implements WAREHOUSE.

Definition at line 87 of file CARLOW_W.cpp.

Referenced by operator=().

```
                                                        {
    this->_number_of_huawei = _number_of_huawei;
}
```

Here is the caller graph for this function:



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

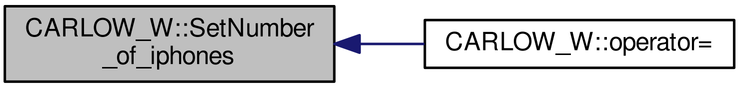
Implements WAREHOUSE.

Definition at line 111 of file CARLOW_W.cpp.

Referenced by operator=().

```
                                                   {
    this->_number_of_iphones = _number_of_iphones;
}
```

Here is the caller graph for this function:



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

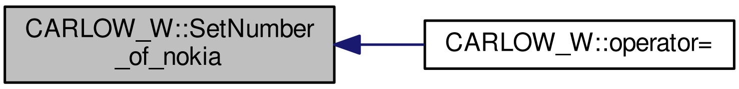
Implements WAREHOUSE.

Definition at line 79 of file CARLOW_W.cpp.

Referenced by operator=().

```
                                               {
    this->_number_of_nokia = _number_of_nokia;
}
```

Here is the caller graph for this function:



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

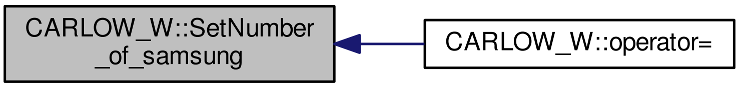
Implements WAREHOUSE.

Definition at line 103 of file CARLOW_W.cpp.

Referenced by operator=().

```
                                                          {
    this->_number_of_samsung = _number_of_samsung;
}
```

Here is the caller graph for this function:



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

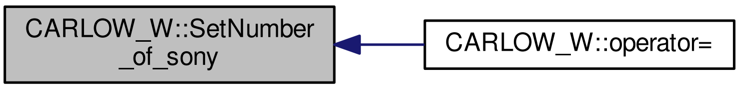
Implements WAREHOUSE.

Definition at line 95 of file CARLOW_W.cpp.

Referenced by operator=().

```
                                                    {
    this->_number_of_sony = _number_of_sony;
}
```

Here is the caller graph for this function:



6.5.3.18 void CARLOW_W::SetShop_address (std::string *_shop_address*) [virtual]

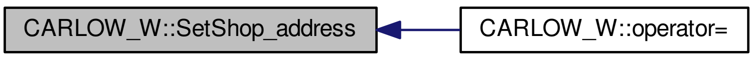
Implements WAREHOUSE.

Definition at line 127 of file CARLOW_W.cpp.

Referenced by operator=().

```
                                                        {
    this->_shop_address = _shop_address;
}
```

Here is the caller graph for this function:



6.5.3.19 void CARLOW_W::SetShop_name (std::string *_shop_name*) [virtual]

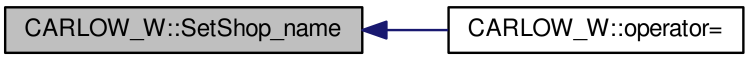
Implements WAREHOUSE.

Definition at line 119 of file CARLOW_W.cpp.

Referenced by operator=().

```
                                                  {
    this->_shop_name = _shop_name;
}
```

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

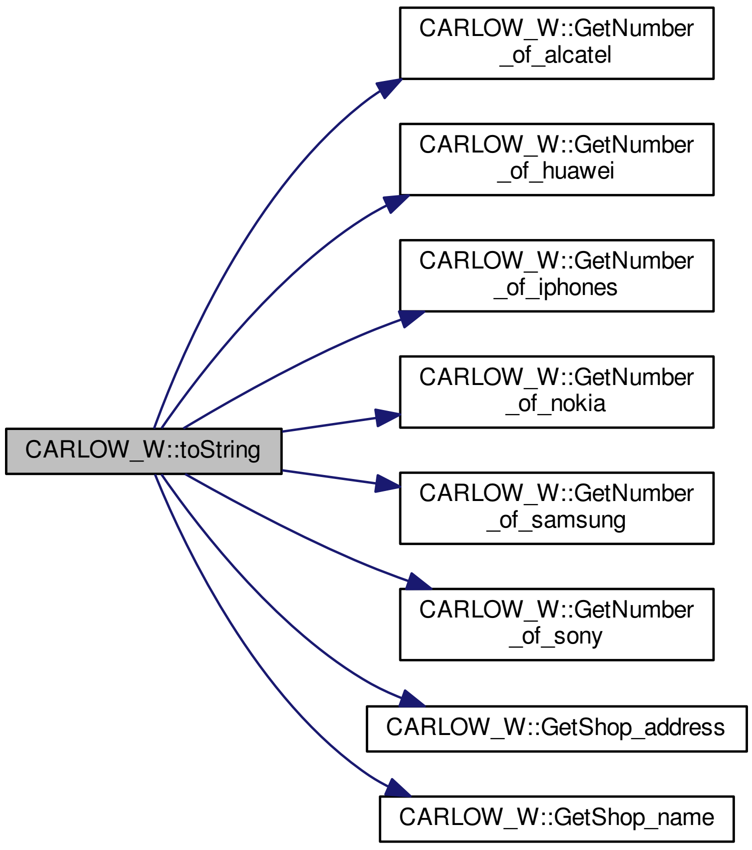
Definition at line 64 of file CARLOW_W.cpp.

References GetNumber_of_alcatel(), GetNumber_of_huawei(), GetNumber_of_iphones(), GetNumber_of_nokia(), GetNumber_of_samsung(), GetNumber_of_sony(), GetShop_address(), and GetShop_name().

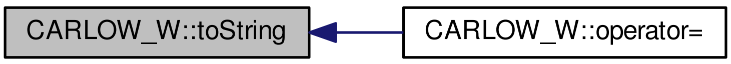
Referenced by operator=().

```
{
   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;
}
```

Here is the call graph for this function:


CARLOW_W::toString
CARLOW_W::GetNumber_of_alcatel
CARLOW_W::GetNumber_of_huawei
CARLOW_W::GetNumber_of_iphones
CARLOW_W::GetNumber_of_nokia
CARLOW_W::GetNumber_of_samsung
CARLOW_W::GetNumber_of_sony
CARLOW_W::GetShop_address
CARLOW_W::GetShop_name

Here is the caller graph for this function:



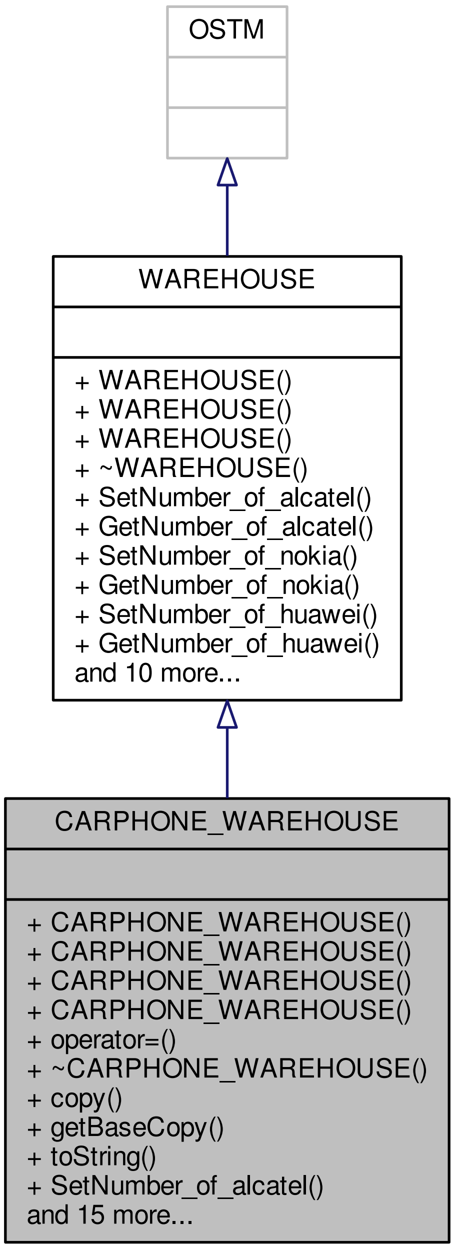
The documentation for this class was generated from the following files:

- CARLOW_W.h
- CARLOW_W.cpp

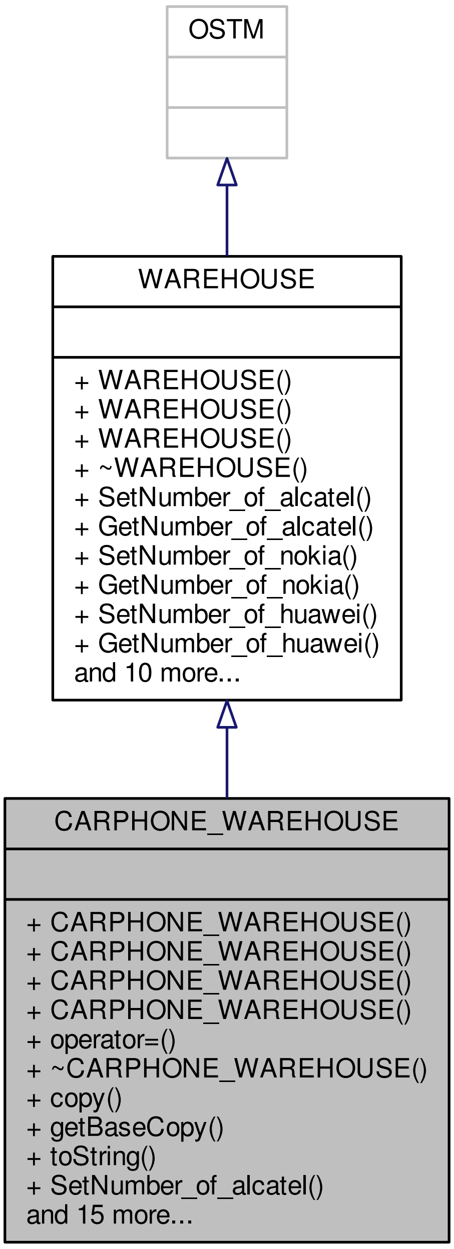
6.6 CARPHONE_WAREHOUSE Class Reference

```
#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]

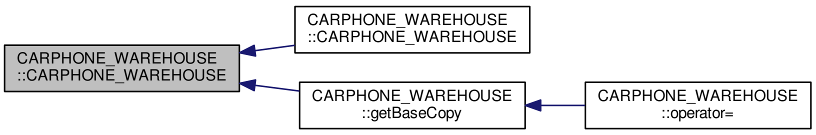
Constructor

Definition at line 24 of file CARPHONE_WAREHOUSE.h.

Referenced by CARPHONE_WAREHOUSE(), and getBaseCopy().

```
                             : WAREHOUSE(){

        this->_shop_address = "DUBLIN XII";
        this->_shop_name = "DISTRIBUTION CENTER";
        this->_number_of_iphones = 10000;
        this->_number_of_samsung = 10000;
        this->_number_of_sony = 10000;
        this->_number_of_huawei = 10000;
        this->_number_of_nokia = 10000;
        this->_number_of_alcatel = 10000;
    };
```

Here is the caller graph for this function:


CARPHONE_WAREHOUSE ::CARPHONE_WAREHOUSE
CARPHONE_WAREHOUSE ::CARPHONE_WAREHOUSE
CARPHONE_WAREHOUSE ::getBaseCopy
CARPHONE_WAREHOUSE ::operator=

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.

```
                                    : WAREHOUSE(){
            /*
             * copy over values
             */
            this->_shop_address = address;
            this->_shop_name = shop_name;
            this->_number_of_iphones = iphone;
            this->_number_of_samsung = samsung;
            this->_number_of_sony = sony;
            this->_number_of_huawei = huawei;
            this->_number_of_nokia = nokia;
            this->_number_of_alcatel = alcatel;

        };
```

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

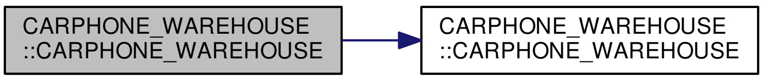
Custom constructor, used by the library for deep copying

Definition at line 55 of file CARPHONE_WAREHOUSE.h.

References CARPHONE_WAREHOUSE().

```
                                                                                :
      WAREHOUSE(_version, _unique_id){
            /*
             * copy over values
             */
            this->_shop_address = obj->GetShop_address();
            this->_shop_name = obj->GetShop_name();
            this->_number_of_iphones = obj->GetNumber_of_iphones();
            this->_number_of_samsung = obj->GetNumber_of_samsung();
            this->_number_of_sony = obj->GetNumber_of_sony();
            this->_number_of_huawei = obj->GetNumber_of_huawei();
            this->_number_of_nokia = obj->GetNumber_of_nokia();
            this->_number_of_alcatel = obj->GetNumber_of_alcatel();
        }
```

Here is the call graph for this function:


CARPHONE_WAREHOUSE ::CARPHONE_WAREHOUSE
CARPHONE_WAREHOUSE ::CARPHONE_WAREHOUSE

6.6.2.4 CARPHONE_WAREHOUSE::CARPHONE_WAREHOUSE (const CARPHONE_WAREHOUSE & *orig*)

Copy constructor

Definition at line 11 of file CARPHONE_WAREHOUSE.cpp.

```
                                                                          {
 }
```

6.6.2.5 CARPHONE_WAREHOUSE::∼CARPHONE_WAREHOUSE ( ) [virtual]

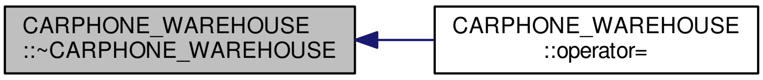
de-constructor

Definition at line 14 of file CARPHONE_WAREHOUSE.cpp.

Referenced by operator=().

```
                                            {
 }
```

Here is the caller graph for this function:


CARPHONE_WAREHOUSE ::~CARPHONE_WAREHOUSE
CARPHONE_WAREHOUSE ::operator=

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

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

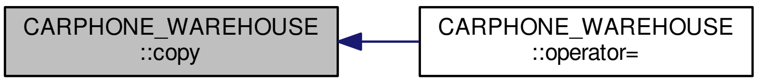
Definition at line 34 of file CARPHONE_WAREHOUSE.cpp.

Referenced by operator=().

```
                                                                                  {

    std::shared_ptr<CARPHONE_WAREHOUSE> objTO = std::dynamic_pointer_cast<
```

```
CARPHONE_WAREHOUSE>(to);
    std::shared_ptr<CARPHONE_WAREHOUSE> objFROM = std::dynamic_pointer_cast<
CARPHONE_WAREHOUSE>(from);
    objTO->_shop_address = objFROM->GetShop_address();
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());

}
```

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

objTO	is a BANK type pointer for casting
obj	is a BANK∗ return type

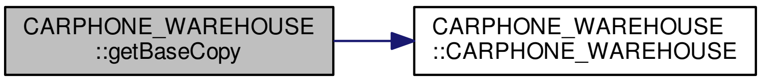
Definition at line 21 of file CARPHONE_WAREHOUSE.cpp.

References CARPHONE_WAREHOUSE().

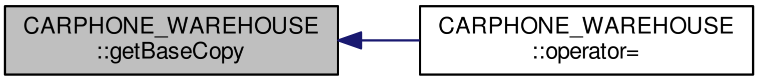
Referenced by operator=().

```
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new CARPHONE_WAREHOUSE(objTO, object->Get_Version(),
object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
```

Here is the call graph for this function:


CARPHONE_WAREHOUSE ::getBaseCopy
CARPHONE_WAREHOUSE ::CARPHONE_WAREHOUSE

Here is the caller graph for this function:


CARPHONE_WAREHOUSE ::getBaseCopy
CARPHONE_WAREHOUSE ::operator=

6.6.3.3 int CARPHONE_WAREHOUSE::GetNumber_of_alcatel ( ) [virtual]

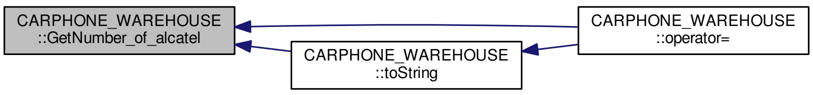
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::GetNumber_of_alcatel
CARPHONE_WAREHOUSE ::toString
CARPHONE_WAREHOUSE ::operator=

6.6.3.4 int CARPHONE_WAREHOUSE::GetNumber_of_huawei ( ) [virtual]

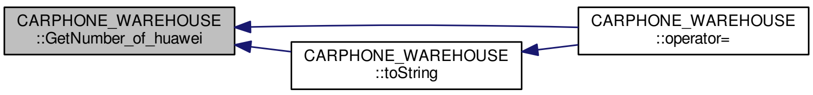
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::GetNumber_of_huawei
CARPHONE_WAREHOUSE ::toString
CARPHONE_WAREHOUSE ::operator=

6.6.3.5 int CARPHONE_WAREHOUSE::GetNumber_of_iphones ( ) [virtual]

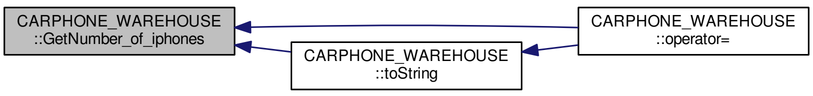
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::GetNumber_of_iphones
CARPHONE_WAREHOUSE ::toString
CARPHONE_WAREHOUSE ::operator=

6.6.3.6 int CARPHONE_WAREHOUSE::GetNumber_of_nokia ( ) [virtual]

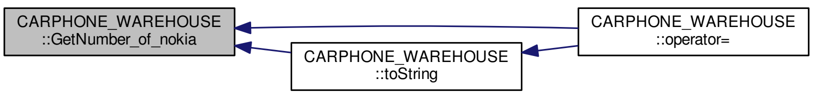
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::GetNumber_of_nokia
CARPHONE_WAREHOUSE ::toString
CARPHONE_WAREHOUSE ::operator=

6.6.3.7 int CARPHONE_WAREHOUSE::GetNumber_of_samsung ( ) [virtual]

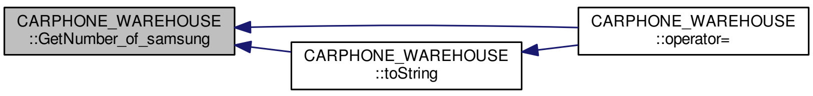
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::GetNumber_of_samsung
CARPHONE_WAREHOUSE ::toString
CARPHONE_WAREHOUSE ::operator=

6.6.3.8 int CARPHONE_WAREHOUSE::GetNumber_of_sony ( ) [virtual]

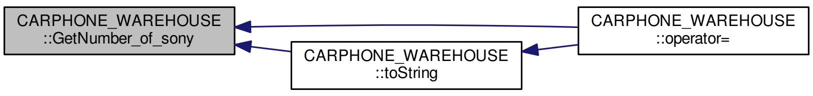
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::GetNumber_of_sony
CARPHONE_WAREHOUSE ::toString
CARPHONE_WAREHOUSE ::operator=

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

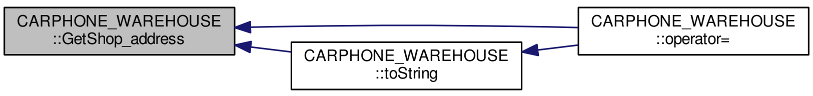
Implements WAREHOUSE.

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]

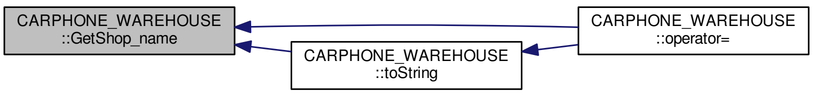
Implements WAREHOUSE.

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]

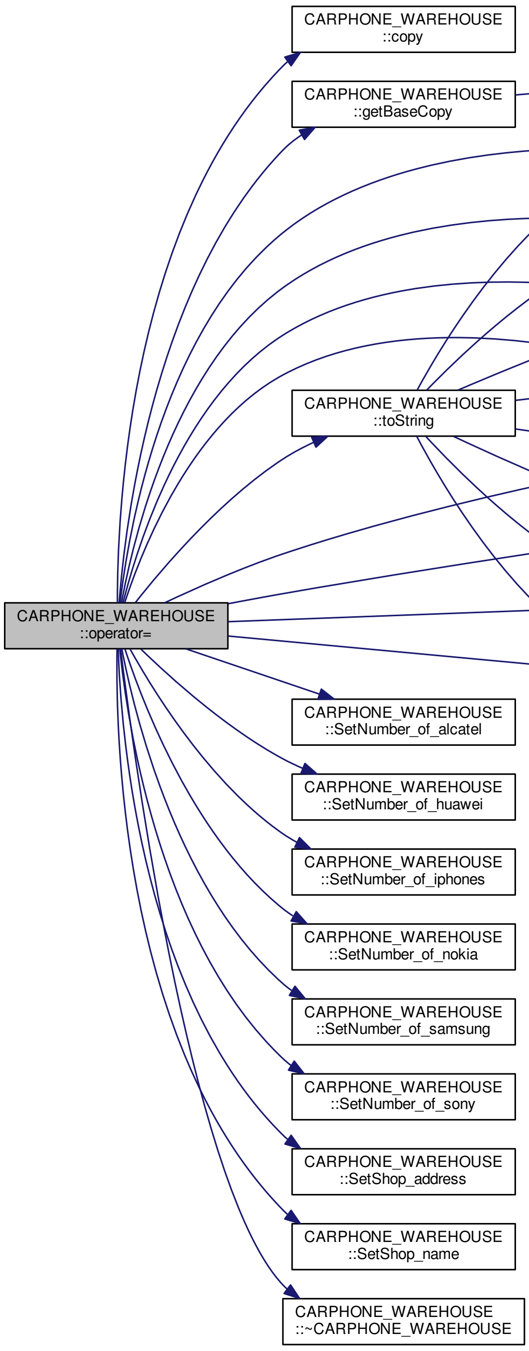
Operator

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 ∼CAR←
PHONE_WAREHOUSE().

```
00075 {};
```

Here is the call graph for this function:


CARPHONE_WAREHOUSE ::copy
CARPHONE_WAREHOUSE ::getBaseCopy
CARPHONE_WAREHOUSE ::CARPHONE_WAREHOUSE
CARPHONE_WAREHOUSE ::GetNumber_of_alcatel
CARPHONE_WAREHOUSE ::GetNumber_of_huawei
CARPHONE_WAREHOUSE ::GetNumber_of_iphones
CARPHONE_WAREHOUSE ::GetNumber_of_nokia
CARPHONE_WAREHOUSE ::toString
CARPHONE_WAREHOUSE ::GetNumber_of_samsung
CARPHONE_WAREHOUSE ::GetNumber_of_sony
CARPHONE_WAREHOUSE ::GetShop_address
CARPHONE_WAREHOUSE ::operator=
CARPHONE_WAREHOUSE ::GetShop_name
CARPHONE_WAREHOUSE ::SetNumber_of_alcatel
CARPHONE_WAREHOUSE ::SetNumber_of_huawei
CARPHONE_WAREHOUSE ::SetNumber_of_iphones
CARPHONE_WAREHOUSE ::SetNumber_of_nokia
CARPHONE_WAREHOUSE ::SetNumber_of_samsung
CARPHONE_WAREHOUSE ::SetNumber_of_sony
CARPHONE_WAREHOUSE ::SetShop_address
CARPHONE_WAREHOUSE ::SetShop_name
CARPHONE_WAREHOUSE ::~CARPHONE_WAREHOUSE

6.6.3.12 void CARPHONE_WAREHOUSE::SetNumber_of_alcatel (int *_number_of_alcatel*) [virtual]

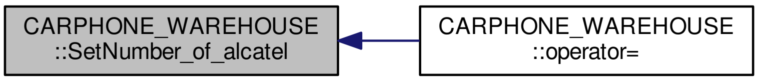
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::SetNumber_of_alcatel
CARPHONE_WAREHOUSE ::operator=

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

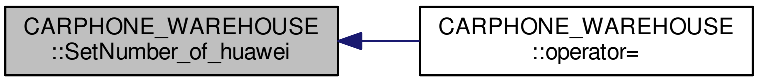
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::SetNumber_of_huawei
CARPHONE_WAREHOUSE ::operator=

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

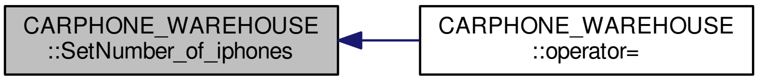
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::SetNumber_of_iphones
CARPHONE_WAREHOUSE ::operator=

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

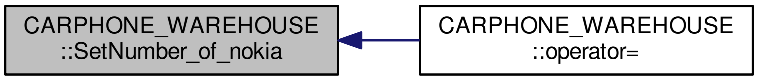
Implements WAREHOUSE.

Definition at line 75 of file CARPHONE_WAREHOUSE.cpp.

Referenced by operator=().

```
                                                                {
    this->_number_of_nokia = _number_of_nokia;
}
```

Here is the caller graph for this function:


CARPHONE_WAREHOUSE ::SetNumber_of_nokia
CARPHONE_WAREHOUSE ::operator=

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

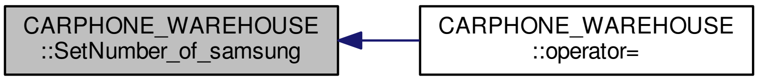
Implements WAREHOUSE.

Definition at line 99 of file CARPHONE_WAREHOUSE.cpp.

Referenced by operator=().

```
                                                                    {
    this->_number_of_samsung = _number_of_samsung;
}
```

Here is the caller graph for this function:


CARPHONE_WAREHOUSE ::SetNumber_of_samsung
CARPHONE_WAREHOUSE ::operator=

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

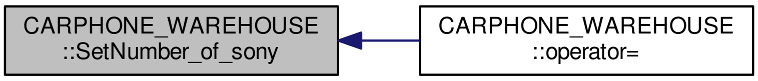
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::SetNumber_of_sony
CARPHONE_WAREHOUSE ::operator=

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

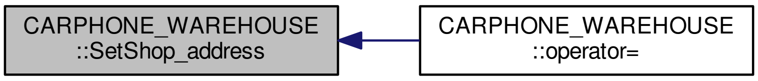
Implements WAREHOUSE.

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:


CARPHONE_WAREHOUSE ::SetShop_address
CARPHONE_WAREHOUSE ::operator=

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

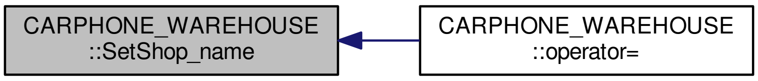
Implements WAREHOUSE.

Definition at line 115 of file CARPHONE_WAREHOUSE.cpp.

Referenced by operator=().

```
                                                        {
    this->_shop_name = _shop_name;
}
```

Here is the caller graph for this function:


CARPHONE_WAREHOUSE ::SetShop_name
CARPHONE_WAREHOUSE ::operator=

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

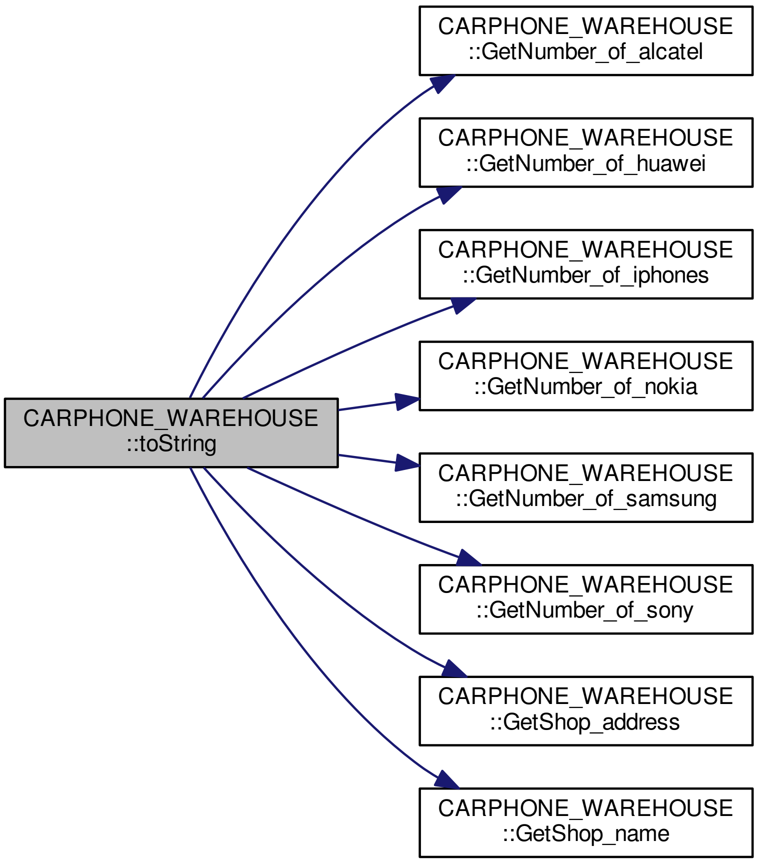
Definition at line 60 of file CARPHONE_WAREHOUSE.cpp.

References GetNumber_of_alcatel(), GetNumber_of_huawei(), GetNumber_of_iphones(), GetNumber_of_nokia(), GetNumber_of_samsung(), GetNumber_of_sony(), GetShop_address(), and GetShop_name().

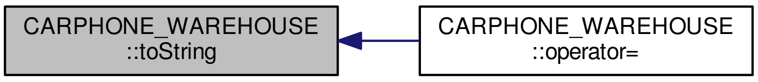
Referenced by operator=().

```
{
   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;
}
```

Here is the call graph for this function:


CARPHONE_WAREHOUSE ::toString
CARPHONE_WAREHOUSE ::GetNumber_of_alcatel
CARPHONE_WAREHOUSE ::GetNumber_of_huawei
CARPHONE_WAREHOUSE ::GetNumber_of_iphones
CARPHONE_WAREHOUSE ::GetNumber_of_nokia
CARPHONE_WAREHOUSE ::GetNumber_of_samsung
CARPHONE_WAREHOUSE ::GetNumber_of_sony
CARPHONE_WAREHOUSE ::GetShop_address
CARPHONE_WAREHOUSE ::GetShop_name

Here is the caller graph for this function:



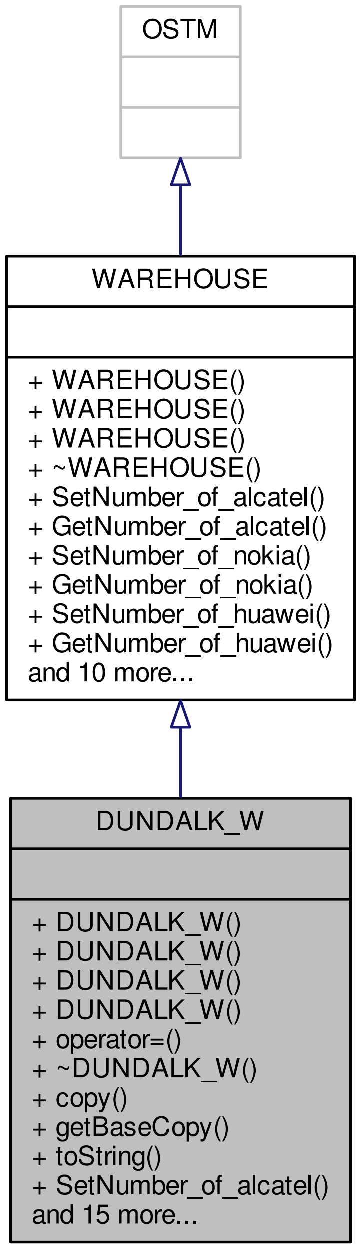
The documentation for this class was generated from the following files:

- CARPHONE_WAREHOUSE.h
- CARPHONE_WAREHOUSE.cpp

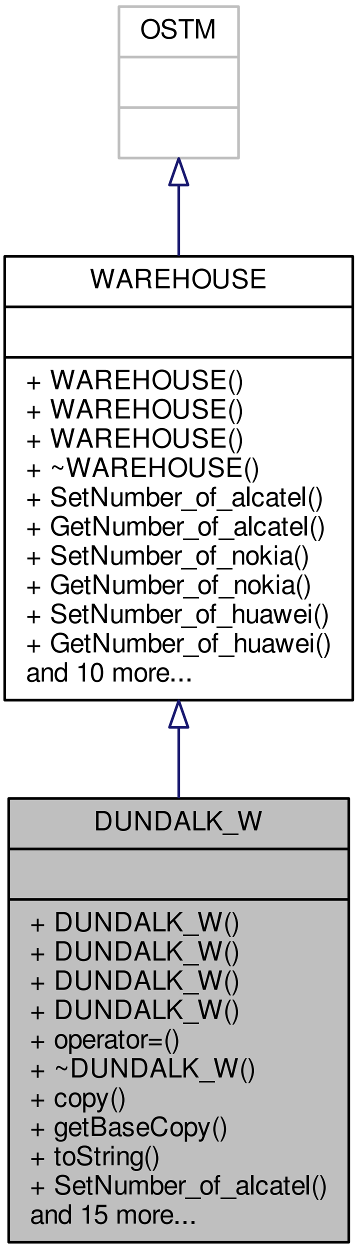
6.7 DUNDALK_W Class Reference

```
#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]

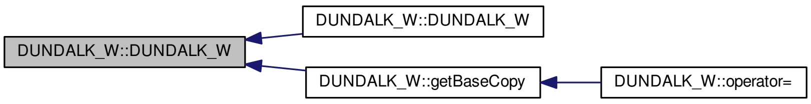
Constructor

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.

```
                      : WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
```

6.7.2.3 DUNDALK_W::DUNDALK_W (std::shared_ptr< WAREHOUSE > *obj*, int *_version*, int *_unique_id*) [inline]

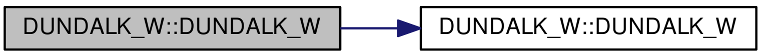
Custom constructor, used by the library for deep copying

Definition at line 55 of file DUNDALK_W.h.

References DUNDALK_W().

```
                                                                          :
      WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
```

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]

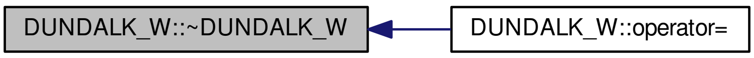
de-constructor

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

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

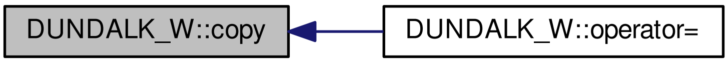
Definition at line 35 of file DUNDALK_W.cpp.

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();
```

```
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
```

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

objTO	is a BANK type pointer for casting
obj	is a BANK∗ return type

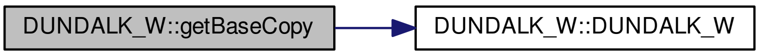
Definition at line 22 of file DUNDALK_W.cpp.

References DUNDALK_W().

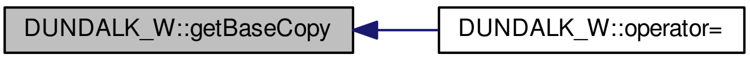
Referenced by operator=().

```
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new DUNDALK_W(objTO, object->Get_Version(),object->
Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
```

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]

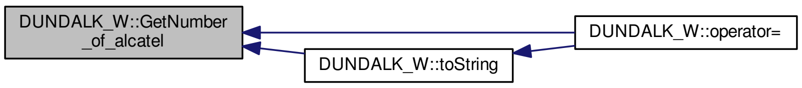
Implements WAREHOUSE.

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]

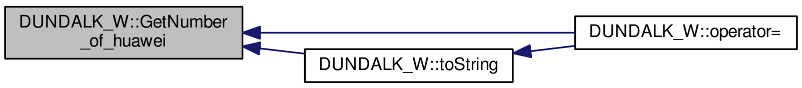
Implements WAREHOUSE.

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]

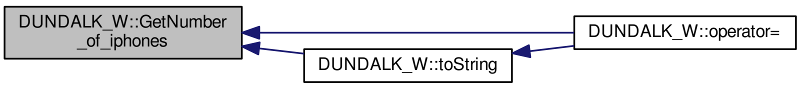
Implements WAREHOUSE.

Definition at line 113 of file DUNDALK_W.cpp.

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

```
                                        {
      return _number_of_iphones;
  }
```

Here is the caller graph for this function:



6.7.3.6 int DUNDALK_W::GetNumber_of_nokia ( ) [virtual]

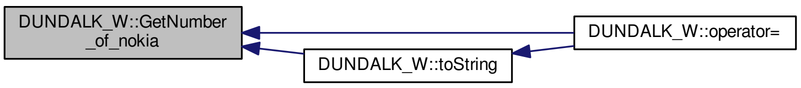
Implements WAREHOUSE.

Definition at line 81 of file DUNDALK_W.cpp.

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

```
                                     {
      return _number_of_nokia;
  }
```

Here is the caller graph for this function:



6.7.3.7 int DUNDALK_W::GetNumber_of_samsung ( ) [virtual]

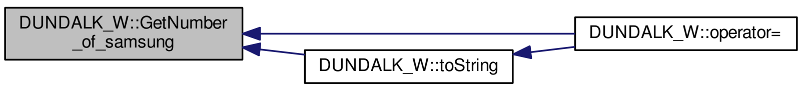
Implements WAREHOUSE.

Definition at line 105 of file DUNDALK_W.cpp.

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

```
                                    {
        return _number_of_samsung;
  }
```

Here is the caller graph for this function:



6.7.3.8 int DUNDALK_W::GetNumber_of_sony ( ) [virtual]

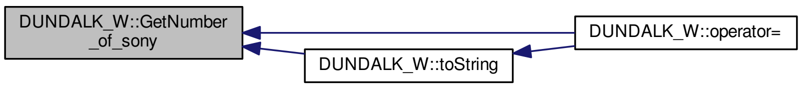
Implements WAREHOUSE.

Definition at line 97 of file DUNDALK_W.cpp.

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

```
                                 {
        return _number_of_sony;
  }
```

Here is the caller graph for this function:



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

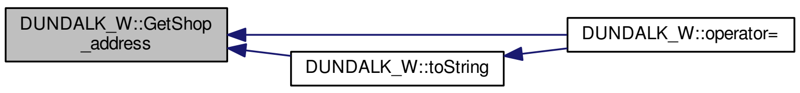
Implements WAREHOUSE.

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]

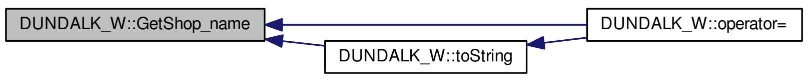
Implements WAREHOUSE.

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]

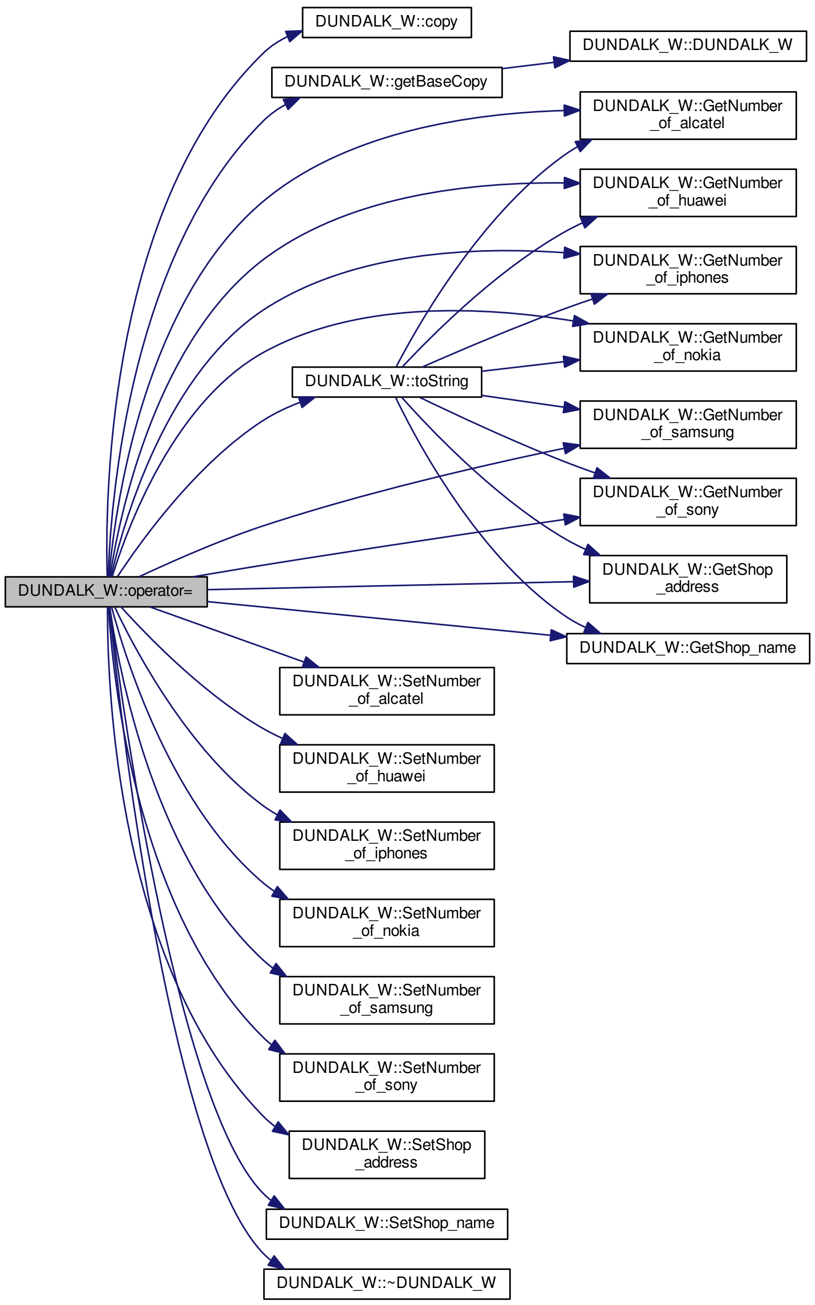
Operator

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:


DUNDALK_W::copy
DUNDALK_W::DUNDALK_W
DUNDALK_W::getBaseCopy
DUNDALK_W::GetNumber_of_alcatel
DUNDALK_W::GetNumber_of_huawei
DUNDALK_W::GetNumber_of_iphones
DUNDALK_W::GetNumber_of_nokia
DUNDALK_W::toString
DUNDALK_W::GetNumber_of_samsung
DUNDALK_W::GetNumber_of_sony
DUNDALK_W::operator=
DUNDALK_W::GetShop_address
DUNDALK_W::GetShop_name
DUNDALK_W::SetNumber_of_alcatel
DUNDALK_W::SetNumber_of_huawei
DUNDALK_W::SetNumber_of_iphones
DUNDALK_W::SetNumber_of_nokia
DUNDALK_W::SetNumber_of_samsung
DUNDALK_W::SetNumber_of_sony
DUNDALK_W::SetShop_address
DUNDALK_W::SetShop_name
DUNDALK_W::~DUNDALK_W

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

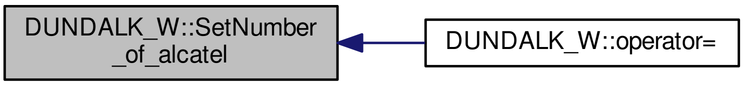
Implements WAREHOUSE.

Definition at line 69 of file DUNDALK_W.cpp.

Referenced by operator=().

```
                                                              {
    this->_number_of_alcatel = _number_of_alcatel;
}
```

Here is the caller graph for this function:



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

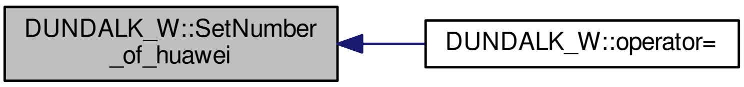
Implements WAREHOUSE.

Definition at line 85 of file DUNDALK_W.cpp.

Referenced by operator=().

```
                                                            {
    this->_number_of_huawei = _number_of_huawei;
}
```

Here is the caller graph for this function:



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

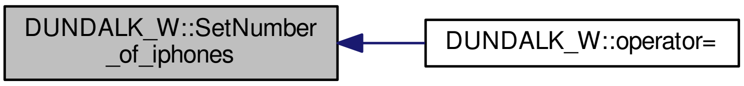
Implements WAREHOUSE.

Definition at line 109 of file DUNDALK_W.cpp.

Referenced by operator=().

```
                                                              {
    this->_number_of_iphones = _number_of_iphones;
}
```

Here is the caller graph for this function:



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

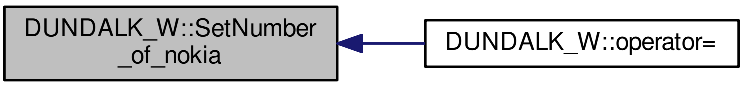
Implements WAREHOUSE.

Definition at line 77 of file DUNDALK_W.cpp.

Referenced by operator=().

```
                                                    {
    this->_number_of_nokia = _number_of_nokia;
}
```

Here is the caller graph for this function:



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

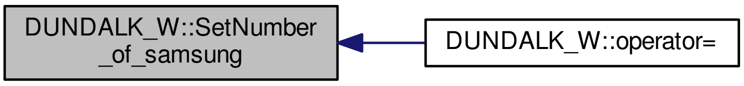
Implements WAREHOUSE.

Definition at line 101 of file DUNDALK_W.cpp.

Referenced by operator=().

```
                                                        {
    this->_number_of_samsung = _number_of_samsung;
}
```

Here is the caller graph for this function:



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

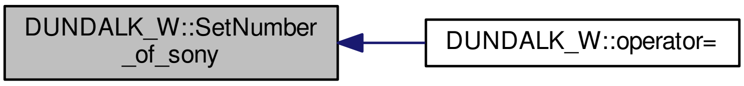
Implements WAREHOUSE.

Definition at line 93 of file DUNDALK_W.cpp.

Referenced by operator=().

```
                                                        {
    this->_number_of_sony = _number_of_sony;
}
```

Here is the caller graph for this function:



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

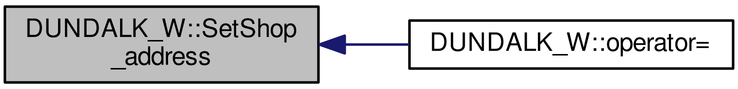
Implements WAREHOUSE.

Definition at line 125 of file DUNDALK_W.cpp.

Referenced by operator=().

```
                                                          {
    this->_shop_address = _shop_address;
}
```

Here is the caller graph for this function:



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

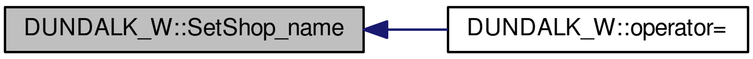
Implements WAREHOUSE.

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

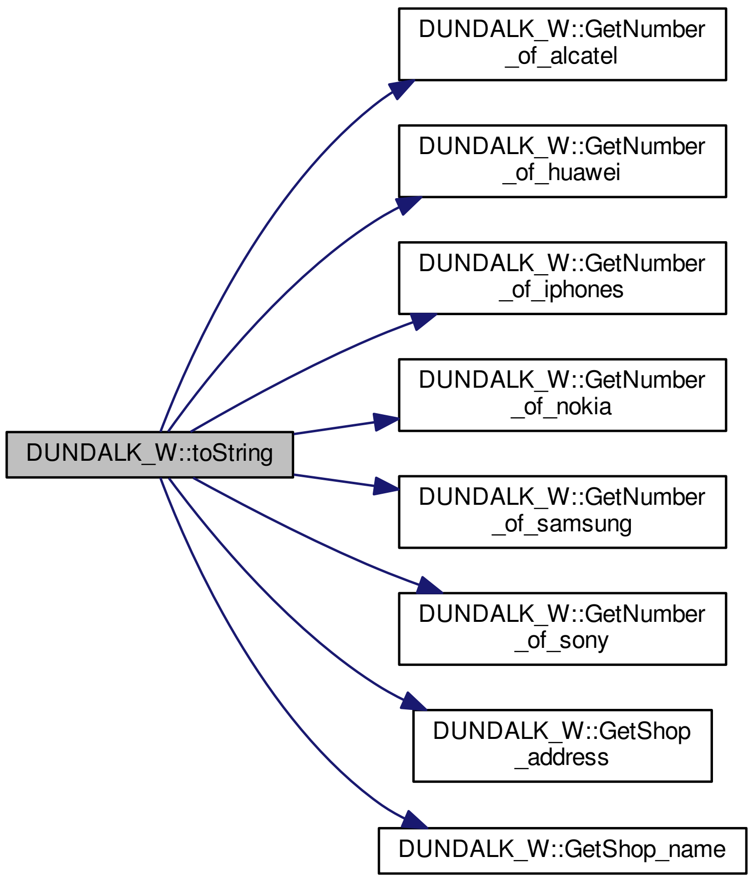
Definition at line 62 of file DUNDALK_W.cpp.

References GetNumber_of_alcatel(), GetNumber_of_huawei(), GetNumber_of_iphones(), GetNumber_of_nokia(), GetNumber_of_samsung(), GetNumber_of_sony(), GetShop_address(), and GetShop_name().

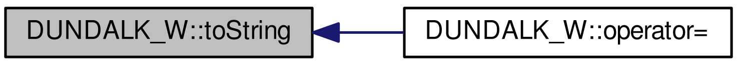
Referenced by operator=().

```
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:


DUNDALK_W::toString
DUNDALK_W::GetNumber _of_alcatel
DUNDALK_W::GetNumber _of_huawei
DUNDALK_W::GetNumber _of_iphones
DUNDALK_W::GetNumber _of_nokia
DUNDALK_W::GetNumber _of_samsung
DUNDALK_W::GetNumber _of_sony
DUNDALK_W::GetShop _address
DUNDALK_W::GetShop_name

Here is the caller graph for this function:



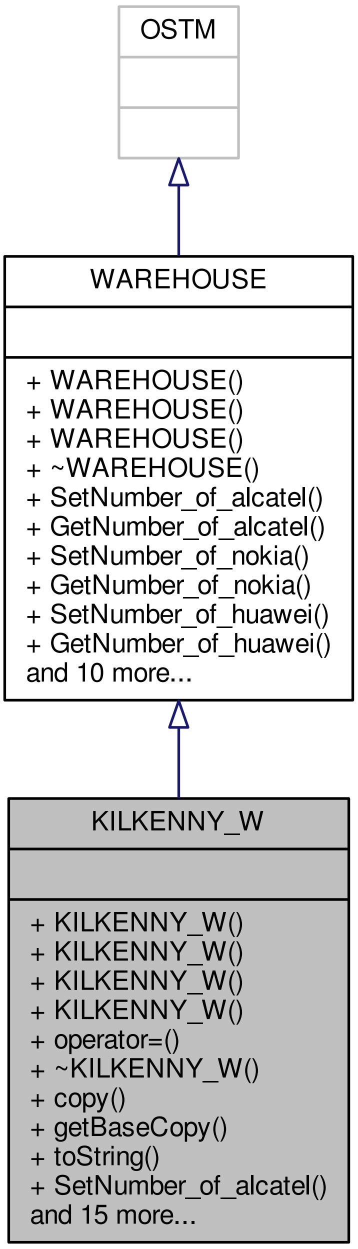
The documentation for this class was generated from the following files:

- DUNDALK_W.h
- DUNDALK_W.cpp

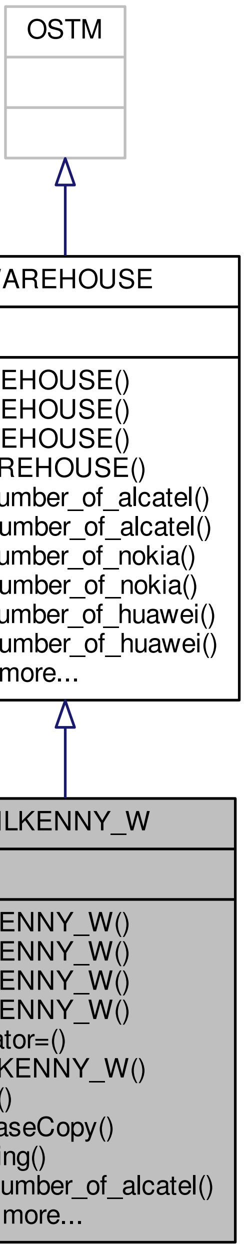
6.8 KILKENNY_W Class Reference

```
#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]

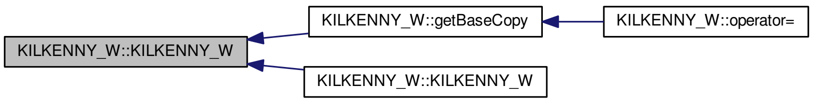
Constructor

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.

```
                            : WAREHOUSE(){
            /*
             * copy over values
             */
            this->_shop_address = address;
            this->_shop_name = shop_name;
            this->_number_of_iphones = iphone;
            this->_number_of_samsung = samsung;
            this->_number_of_sony = sony;
            this->_number_of_huawei = huawei;
            this->_number_of_nokia = nokia;
            this->_number_of_alcatel = alcatel;

        };
```

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

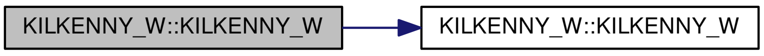
Custom constructor, used by the library for deep copying

Definition at line 55 of file KILKENNY_W.h.

References KILKENNY_W().

```
                                                                                :
     WAREHOUSE(_version, _unique_id){
            /*
             * copy over values
             */
            this->_shop_address = obj->GetShop_address();
            this->_shop_name = obj->GetShop_name();
            this->_number_of_iphones = obj->GetNumber_of_iphones();
            this->_number_of_samsung = obj->GetNumber_of_samsung();
            this->_number_of_sony = obj->GetNumber_of_sony();
            this->_number_of_huawei = obj->GetNumber_of_huawei();
            this->_number_of_nokia = obj->GetNumber_of_nokia();
            this->_number_of_alcatel = obj->GetNumber_of_alcatel();
        }
```

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.

```
{
}
```

6.8.2.5 KILKENNY_W::∼KILKENNY_W ( ) [virtual]

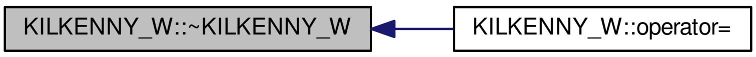
de-constructor

Definition at line 12 of file KILKENNY_W.cpp.

Referenced by operator=().

```
{
}
```

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

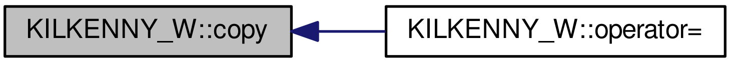
objTO	is a BANK∗ type object casted back from std::shared_ptr<OSTM>
objFROM	is a BANK∗ type object casted back from std::shared_ptr<OSTM>

Definition at line 35 of file KILKENNY_W.cpp.

Referenced by operator=().

```
                                                                   {

    std::shared_ptr<KILKENNY_W> objTO = std::dynamic_pointer_cast<KILKENNY_W>(to);
    std::shared_ptr<KILKENNY_W> objFROM = std::dynamic_pointer_cast<KILKENNY_W>(from);
    objTO->_shop_address = objFROM->GetShop_address();
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
```

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

objTO	is a BANK type pointer for casting
obj	is a BANK∗ return type

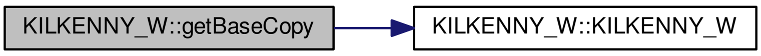
Definition at line 22 of file KILKENNY_W.cpp.

References KILKENNY_W().

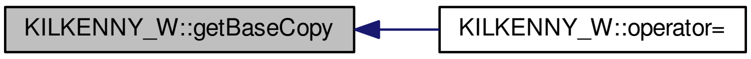
Referenced by operator=().

```
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new KILKENNY_W(objTO, object->Get_Version(),object->
  Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
```

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]

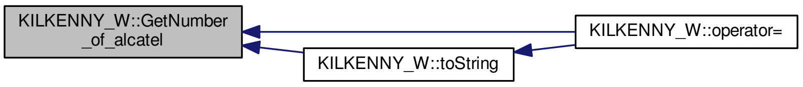
Implements WAREHOUSE.

Definition at line 73 of file KILKENNY_W.cpp.

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

```
                                    {
        return _number_of_alcatel;
    }
```

Here is the caller graph for this function:



6.8.3.4 int KILKENNY_W::GetNumber_of_huawei ( ) [virtual]

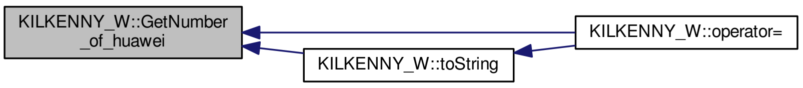
Implements WAREHOUSE.

Definition at line 89 of file KILKENNY_W.cpp.

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

```
                                   {
        return _number_of_huawei;
    }
```

Here is the caller graph for this function:



6.8.3.5 int KILKENNY_W::GetNumber_of_iphones ( ) [virtual]

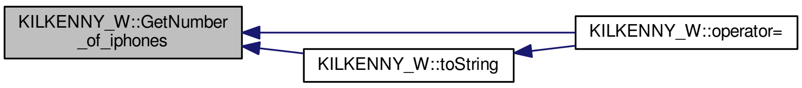
Implements WAREHOUSE.

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]

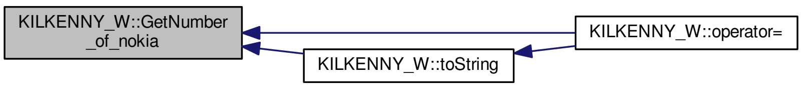
Implements WAREHOUSE.

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]

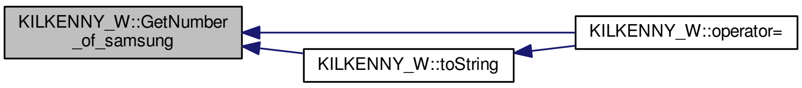
Implements WAREHOUSE.

Definition at line 105 of file KILKENNY_W.cpp.

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

```
                                    {
    return _number_of_samsung;
}
```

Here is the caller graph for this function:



6.8.3.8 int KILKENNY_W::GetNumber_of_sony ( ) [virtual]

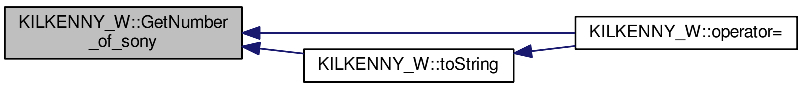
Implements WAREHOUSE.

Definition at line 97 of file KILKENNY_W.cpp.

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

```
                                 {
    return _number_of_sony;
}
```

Here is the caller graph for this function:



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

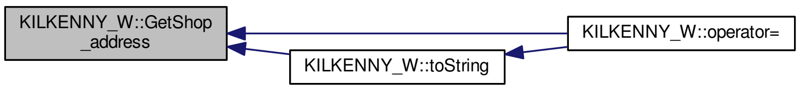
Implements WAREHOUSE.

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]

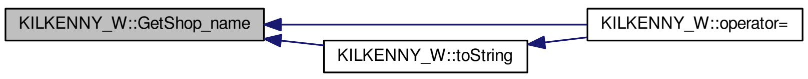
Implements WAREHOUSE.

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]

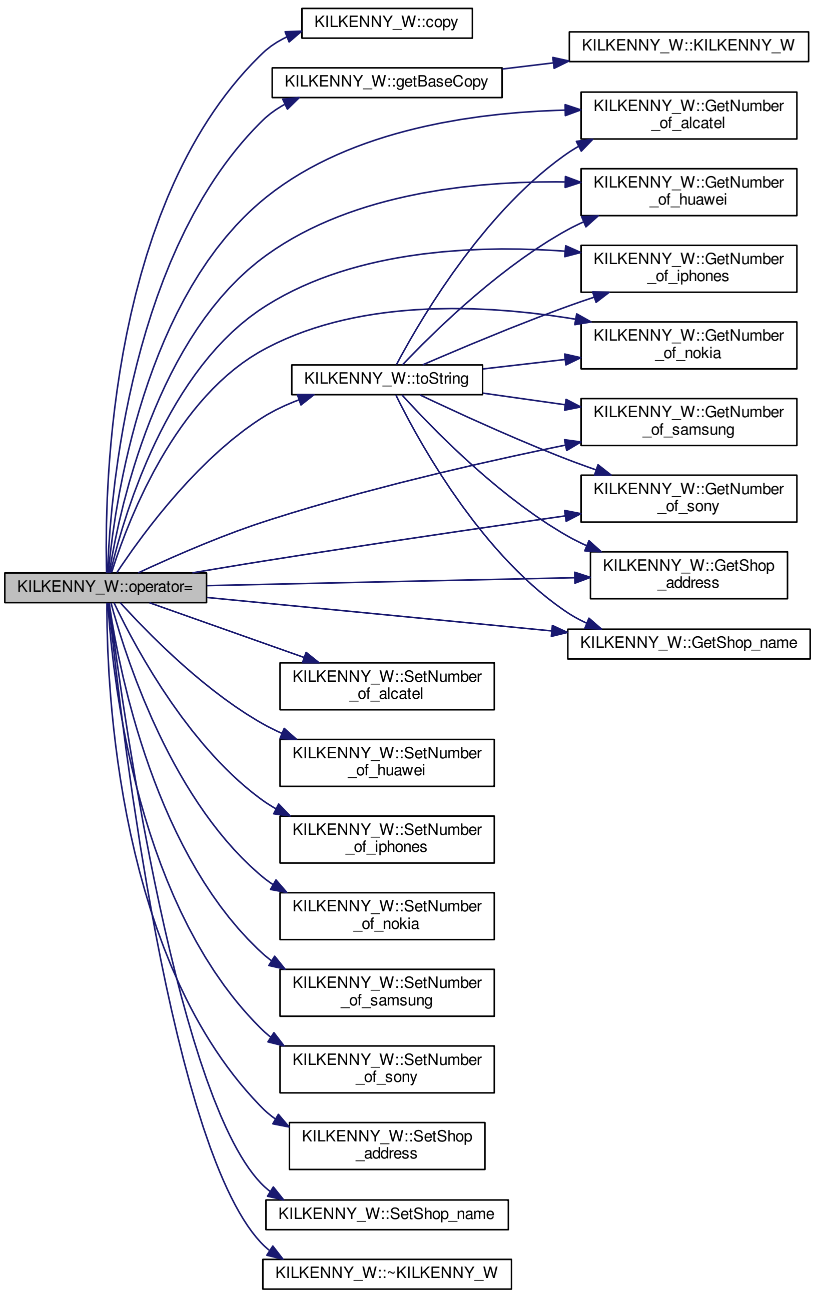
Operator

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 ∼KILK↩ ENNY_W().

```
00075 {};
```

Here is the call graph for this function:


KILKENNY_W::copy
KILKENNY_W::KILKENNY_W
KILKENNY_W::getBaseCopy
KILKENNY_W::GetNumber _of_alcatel
KILKENNY_W::GetNumber _of_huawei
KILKENNY_W::GetNumber _of_iphones
KILKENNY_W::GetNumber _of_nokia
KILKENNY_W::toString
KILKENNY_W::GetNumber _of_samsung
KILKENNY_W::GetNumber _of_sony
KILKENNY_W::GetShop _address
KILKENNY_W::operator=
KILKENNY_W::GetShop_name
KILKENNY_W::SetNumber _of_alcatel
KILKENNY_W::SetNumber _of_huawei
KILKENNY_W::SetNumber _of_iphones
KILKENNY_W::SetNumber _of_nokia
KILKENNY_W::SetNumber _of_samsung
KILKENNY_W::SetNumber _of_sony
KILKENNY_W::SetShop _address
KILKENNY_W::SetShop_name
KILKENNY_W::~KILKENNY_W

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

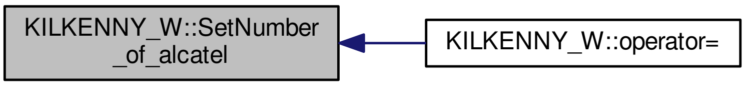
Implements WAREHOUSE.

Definition at line 69 of file KILKENNY_W.cpp.

Referenced by operator=().

```
                                                         {
    this->_number_of_alcatel = _number_of_alcatel;
}
```

Here is the caller graph for this function:



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

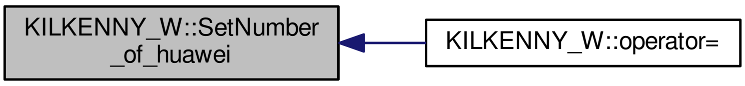
Implements WAREHOUSE.

Definition at line 85 of file KILKENNY_W.cpp.

Referenced by operator=().

```
                                                       {
    this->_number_of_huawei = _number_of_huawei;
}
```

Here is the caller graph for this function:



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

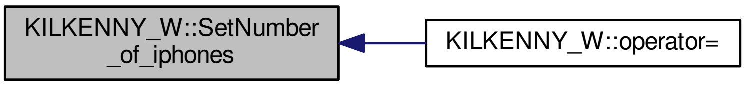
Implements WAREHOUSE.

Definition at line 109 of file KILKENNY_W.cpp.

Referenced by operator=().

```
                                                             {
    this->_number_of_iphones = _number_of_iphones;
}
```

Here is the caller graph for this function:



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

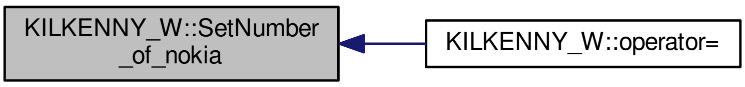
Implements WAREHOUSE.

Definition at line 77 of file KILKENNY_W.cpp.

Referenced by operator=().

```
                                                         {
    this->_number_of_nokia = _number_of_nokia;
}
```

Here is the caller graph for this function:



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

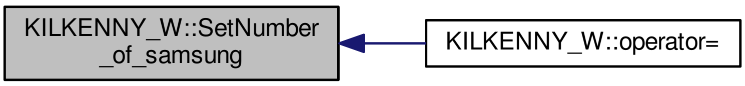
Implements WAREHOUSE.

Definition at line 101 of file KILKENNY_W.cpp.

Referenced by operator=().

```
                                                        {
    this->_number_of_samsung = _number_of_samsung;
}
```

Here is the caller graph for this function:



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

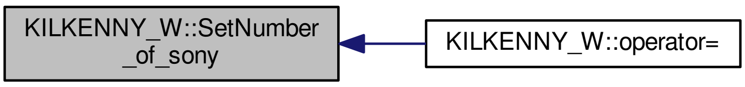
Implements WAREHOUSE.

Definition at line 93 of file KILKENNY_W.cpp.

Referenced by operator=().

```
                                                  {
    this->_number_of_sony = _number_of_sony;
}
```

Here is the caller graph for this function:



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

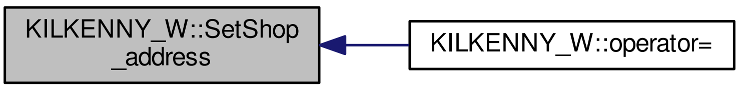
Implements WAREHOUSE.

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]

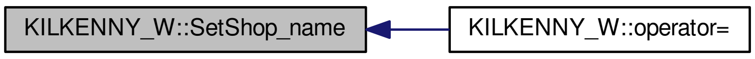
Implements WAREHOUSE.

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

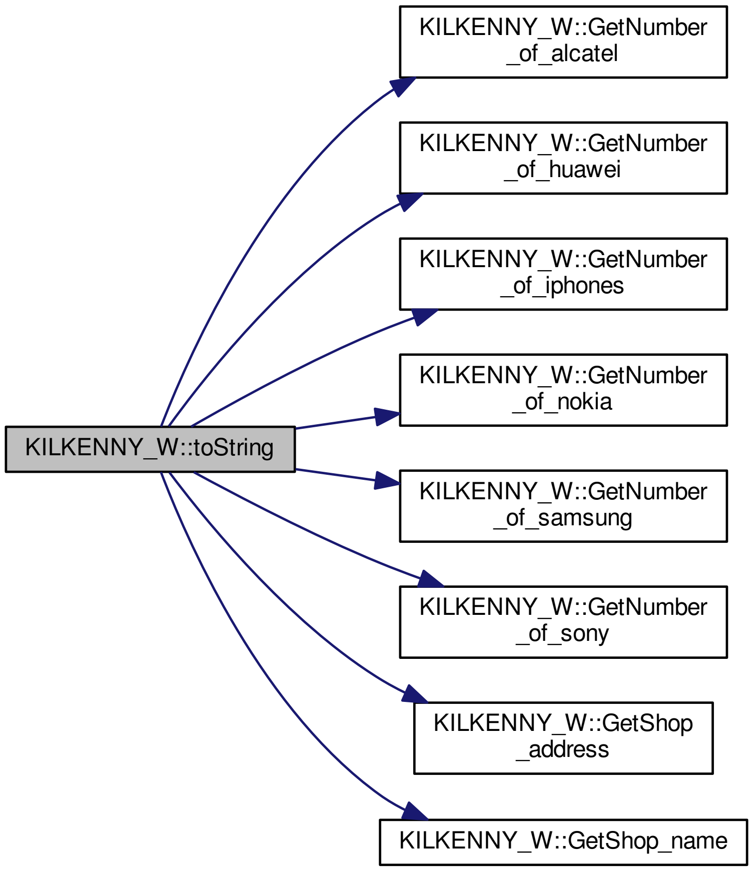
Definition at line 62 of file KILKENNY_W.cpp.

References GetNumber_of_alcatel(), GetNumber_of_huawei(), GetNumber_of_iphones(), GetNumber_of_nokia(), GetNumber_of_samsung(), GetNumber_of_sony(), GetShop_address(), and GetShop_name().

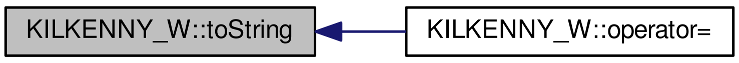
Referenced by operator=().

```
{
   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;
}
```

Here is the call graph for this function:


KILKENNY_W::toString
KILKENNY_W::GetNumber_of_alcatel
KILKENNY_W::GetNumber_of_huawei
KILKENNY_W::GetNumber_of_iphones
KILKENNY_W::GetNumber_of_nokia
KILKENNY_W::GetNumber_of_samsung
KILKENNY_W::GetNumber_of_sony
KILKENNY_W::GetShop_address
KILKENNY_W::GetShop_name

Here is the caller graph for this function:



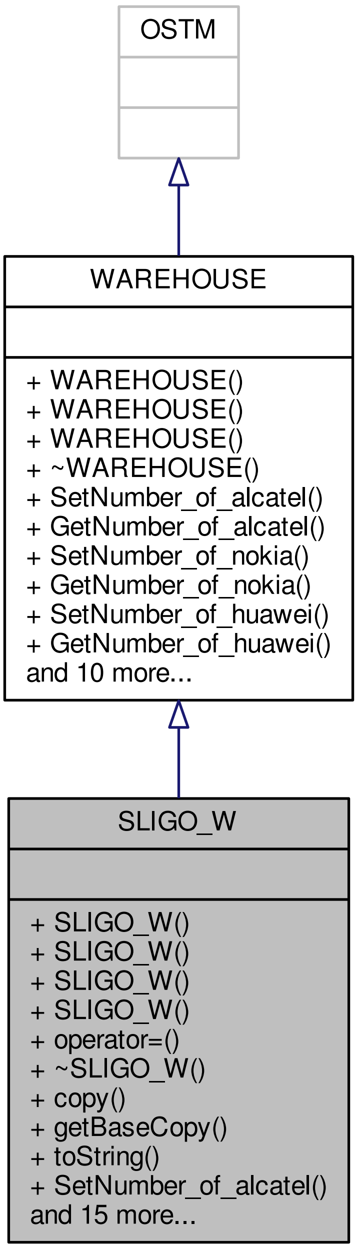
The documentation for this class was generated from the following files:

- KILKENNY_W.h
- KILKENNY_W.cpp

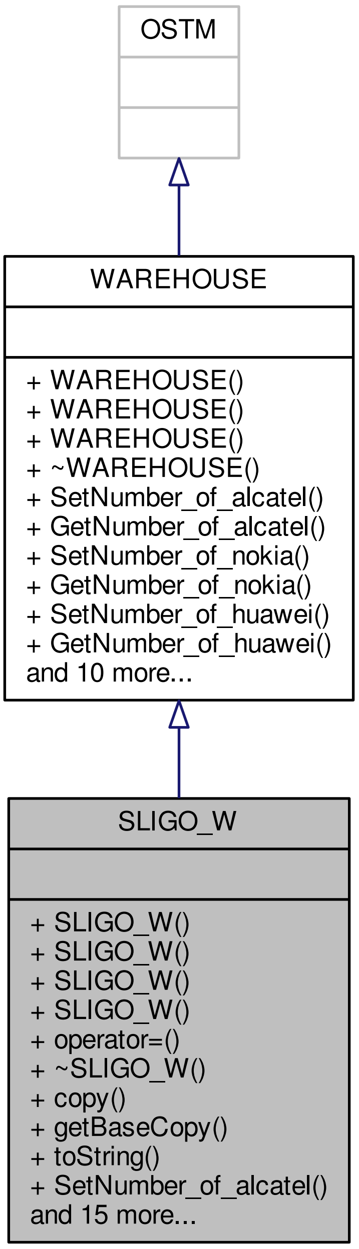
6.9 SLIGO_W Class Reference

```
#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]

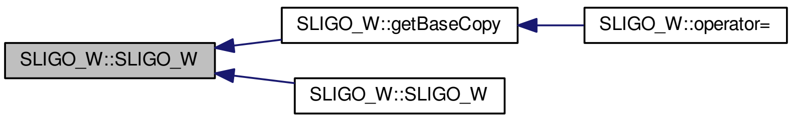
Constructor

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.

```
                 : WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
```

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

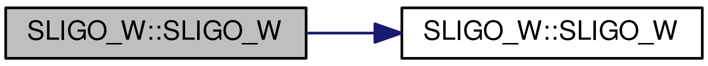
Custom constructor, used by the library for deep copying

Definition at line 55 of file SLIGO_W.h.

References SLIGO_W().

```
                                                                          :
    WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
```

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]

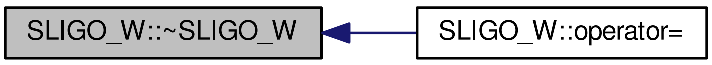
de-constructor

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

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

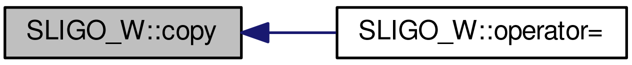
Definition at line 35 of file SLIGO_W.cpp.

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);
```

```
    objTO->_shop_address = objFROM->GetShop_address();
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
```

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

objTO	is a BANK type pointer for casting
obj	is a BANK∗ return type

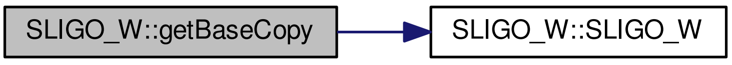
Definition at line 22 of file SLIGO_W.cpp.

References SLIGO_W().

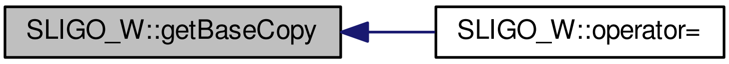
Referenced by operator=().

```
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new SLIGO_W(objTO, object->Get_Version(),object->Get_Unique_ID())
    );
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

    return ostm_obj;
}
```

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]

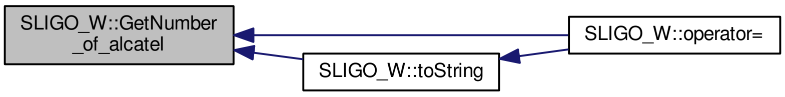
Implements WAREHOUSE.

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]

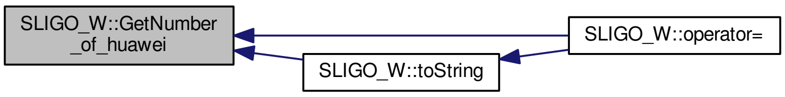
Implements WAREHOUSE.

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]

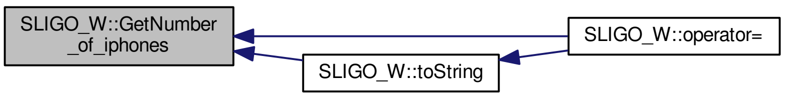
Implements WAREHOUSE.

Definition at line 113 of file SLIGO_W.cpp.

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

```
                                 {
    return _number_of_iphones;
}
```

Here is the caller graph for this function:



6.9.3.6 int SLIGO_W::GetNumber_of_nokia ( ) [virtual]

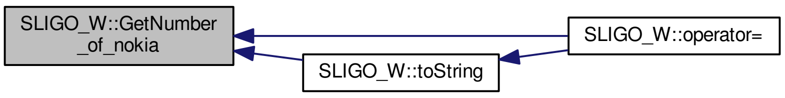
Implements WAREHOUSE.

Definition at line 81 of file SLIGO_W.cpp.

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

```
                               {
    return _number_of_nokia;
}
```

Here is the caller graph for this function:



6.9.3.7 int SLIGO_W::GetNumber_of_samsung ( ) [virtual]

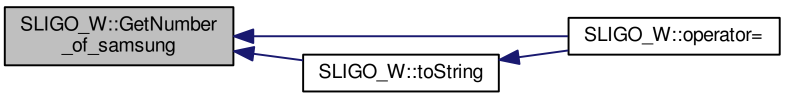
Implements WAREHOUSE.

Definition at line 105 of file SLIGO_W.cpp.

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

```
                                  {
    return _number_of_samsung;
}
```

Here is the caller graph for this function:



6.9.3.8 int SLIGO_W::GetNumber_of_sony ( ) [virtual]

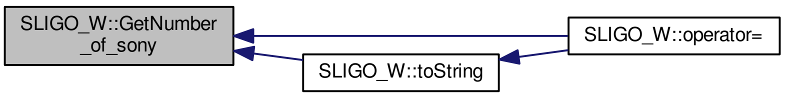
Implements WAREHOUSE.

Definition at line 97 of file SLIGO_W.cpp.

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

```
                               {
    return _number_of_sony;
}
```

Here is the caller graph for this function:



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

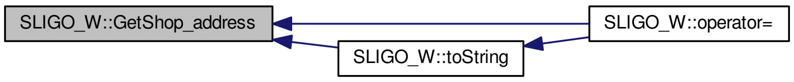
Implements WAREHOUSE.

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]

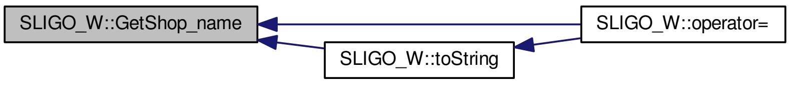
Implements WAREHOUSE.

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]

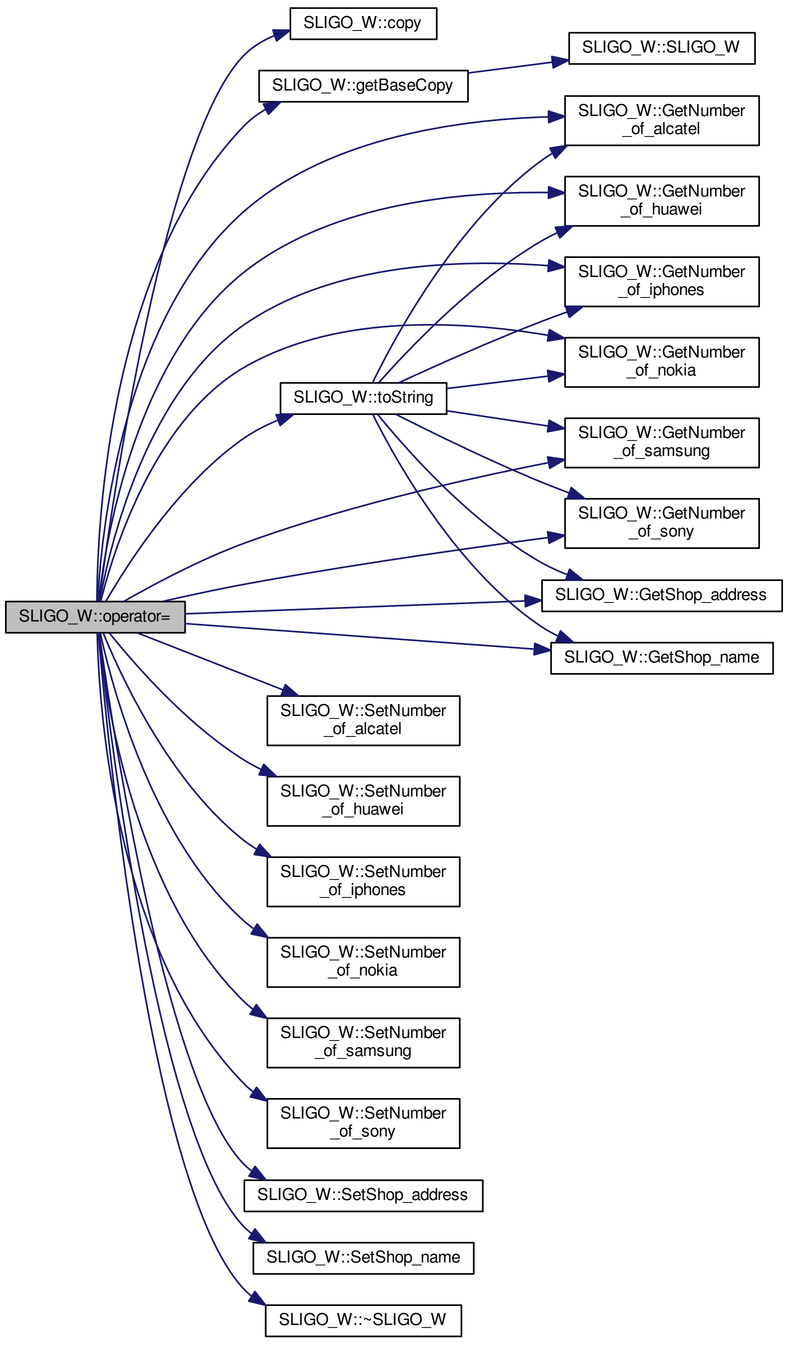
Operator

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 ~SLIG↩ O_W().

```
00075 {};
```

Here is the call graph for this function:


SLIGO_W::operator=
SLIGO_W::copy
SLIGO_W::getBaseCopy
SLIGO_W::SLIGO_W
SLIGO_W::GetNumber_of_alcatel
SLIGO_W::GetNumber_of_huawei
SLIGO_W::GetNumber_of_iphones
SLIGO_W::GetNumber_of_nokia
SLIGO_W::toString
SLIGO_W::GetNumber_of_samsung
SLIGO_W::GetNumber_of_sony
SLIGO_W::GetShop_address
SLIGO_W::GetShop_name
SLIGO_W::SetNumber_of_alcatel
SLIGO_W::SetNumber_of_huawei
SLIGO_W::SetNumber_of_iphones
SLIGO_W::SetNumber_of_nokia
SLIGO_W::SetNumber_of_samsung
SLIGO_W::SetNumber_of_sony
SLIGO_W::SetShop_address
SLIGO_W::SetShop_name
SLIGO_W::~SLIGO_W

6.9.3.12 void SLIGO_W::SetNumber_of_alcatel (int *_number_of_alcatel*) [virtual]

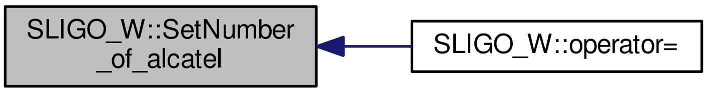
Implements WAREHOUSE.

Definition at line 69 of file SLIGO_W.cpp.

Referenced by operator=().

```
                                                                {
    this->_number_of_alcatel = _number_of_alcatel;
}
```

Here is the caller graph for this function:



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

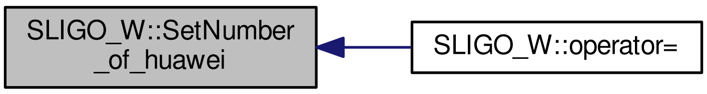
Implements WAREHOUSE.

Definition at line 85 of file SLIGO_W.cpp.

Referenced by operator=().

```
                                                              {
    this->_number_of_huawei = _number_of_huawei;
}
```

Here is the caller graph for this function:



6.9.3.14 void SLIGO_W::SetNumber_of_iphones (int *_number_of_iphones*) [virtual]

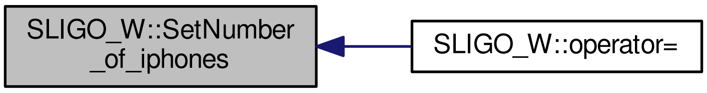
Implements WAREHOUSE.

Definition at line 109 of file SLIGO_W.cpp.

Referenced by operator=().

```
                                                                {
    this->_number_of_iphones = _number_of_iphones;
}
```

Here is the caller graph for this function:



6.9.3.15 void SLIGO_W::SetNumber_of_nokia (int *_number_of_nokia*) `[virtual]`

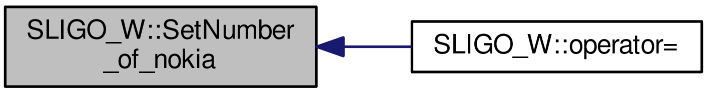
Implements WAREHOUSE.

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]`

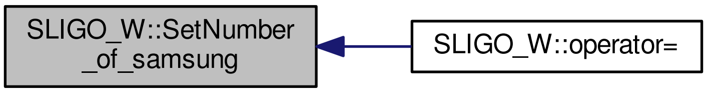
Implements WAREHOUSE.

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]

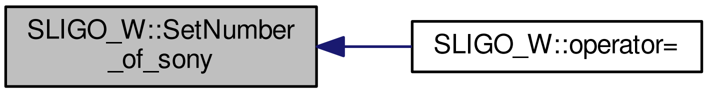
Implements WAREHOUSE.

Definition at line 93 of file SLIGO_W.cpp.

Referenced by operator=().

```
                                                                  {
    this->_number_of_sony = _number_of_sony;
}
```

Here is the caller graph for this function:



6.9.3.18 void SLIGO_W::SetShop_address (std::string *_shop_address*) [virtual]

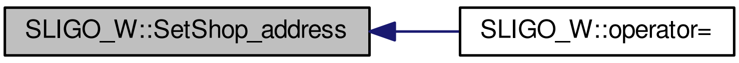
Implements WAREHOUSE.

Definition at line 125 of file SLIGO_W.cpp.

Referenced by operator=().

```
                                                                    {
    this->_shop_address = _shop_address;
}
```

Here is the caller graph for this function:



6.9.3.19 void SLIGO_W::SetShop_name (std::string *_shop_name*) [virtual]

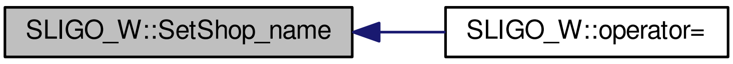
Implements WAREHOUSE.

Definition at line 117 of file SLIGO_W.cpp.

Referenced by operator=().

```
                                                   {
    this->_shop_name = _shop_name;
}
```

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

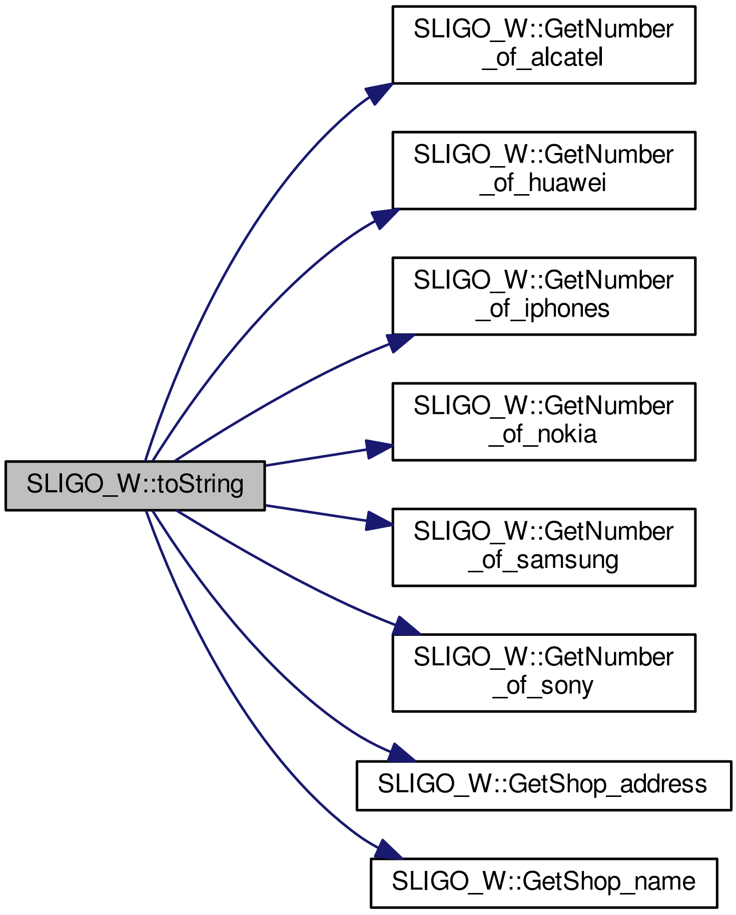
Definition at line 62 of file SLIGO_W.cpp.

References GetNumber_of_alcatel(), GetNumber_of_huawei(), GetNumber_of_iphones(), GetNumber_of_nokia(), GetNumber_of_samsung(), GetNumber_of_sony(), GetShop_address(), and GetShop_name().

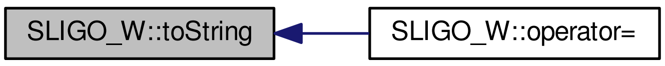
Referenced by operator=().

```
{
    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;
}
```

Here is the call graph for this function:


SLIGO_W::toString
SLIGO_W::GetNumber _of_alcatel
SLIGO_W::GetNumber _of_huawei
SLIGO_W::GetNumber _of_iphones
SLIGO_W::GetNumber _of_nokia
SLIGO_W::GetNumber _of_samsung
SLIGO_W::GetNumber _of_sony
SLIGO_W::GetShop_address
SLIGO_W::GetShop_name

Here is the caller graph for this function:



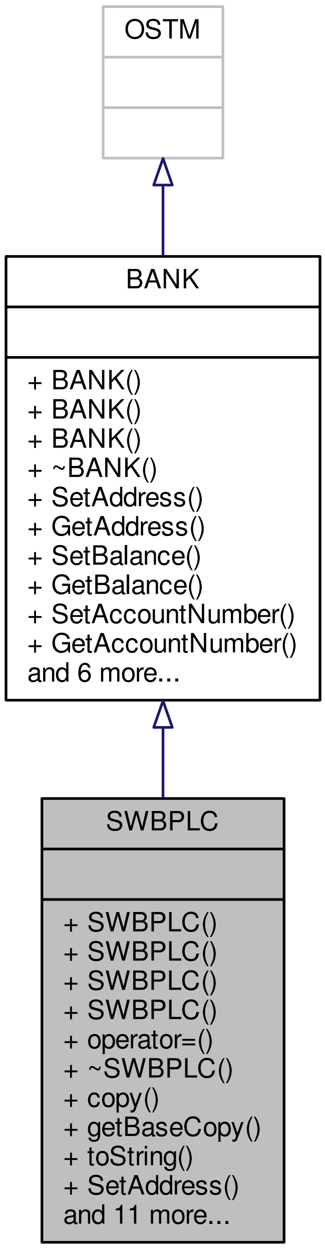
The documentation for this class was generated from the following files:

- SLIGO_W.h
- SLIGO_W.cpp

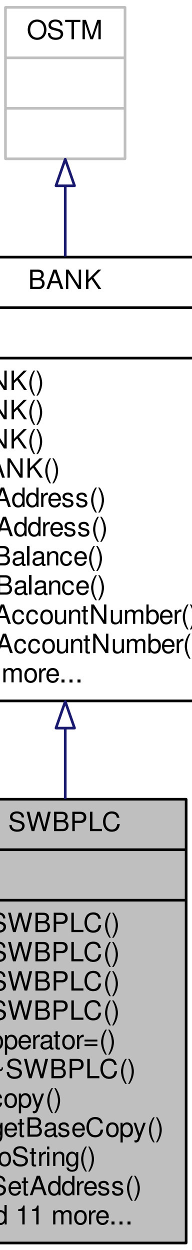
6.10 SWBPLC Class Reference

```
#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]

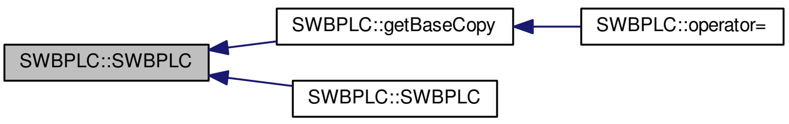
Constructor

Definition at line 24 of file SWBPLC.h.

Referenced by getBaseCopy(), and SWBPLC().

```
            : BANK() {
        this->accountNumber = 0;
        this->balance = 50;
        this->firstName = "Joe";
        this->lastName = "Blog";
        this->address = "High street, Carlow";
        this->fullname = firstName + " " + lastName;
    };
```

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                                                                                                              :
       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     };
```

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

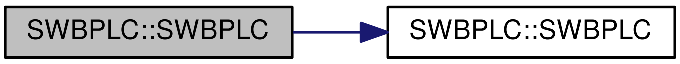
Custom constructor, used by the library for deep copying

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]

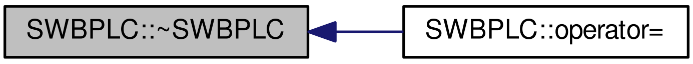
de-constructor

Definition at line 15 of file SWBPLC.cpp.

Referenced by operator=().

```
                 {
}
```

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

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

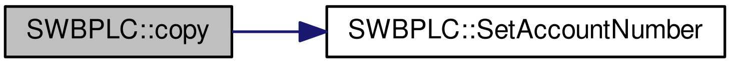
Definition at line 34 of file SWBPLC.cpp.

References SetAccountNumber().

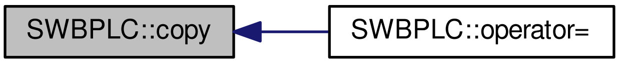
Referenced by operator=().

```
                                                                  {

    std::shared_ptr<SWBPLC> objTO = std::dynamic_pointer_cast<SWBPLC>(to);
    std::shared_ptr<SWBPLC> objFROM = std::dynamic_pointer_cast<SWBPLC>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());


}
```

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]

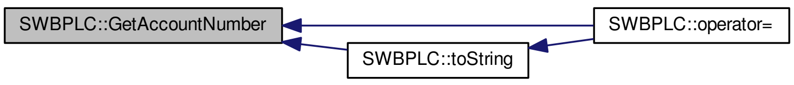
Implements BANK.

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]

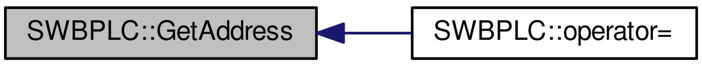
Implements BANK.

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]

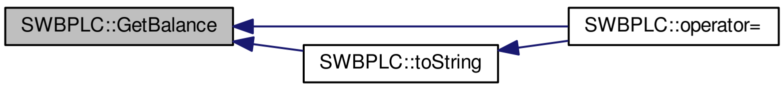
Implements BANK.

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

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

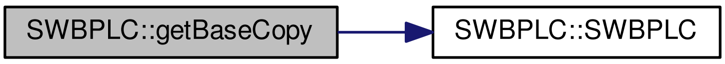
Definition at line 22 of file SWBPLC.cpp.

References SWBPLC().

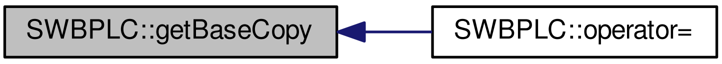
Referenced by operator=().

```
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]

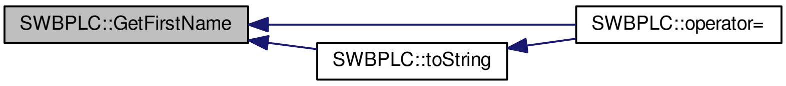
Implements BANK.

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]

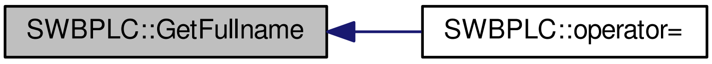
Implements BANK.

Definition at line 104 of file SWBPLC.cpp.

Referenced by operator=().

```
                                      {
    return fullname;
}
```

Here is the caller graph for this function:



6.10.3.8 std::string SWBPLC::GetLastName ( ) const [virtual]

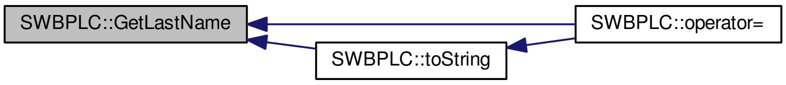
Implements BANK.

Definition at line 88 of file SWBPLC.cpp.

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

```
                                      {
    return lastName;
}
```

Here is the caller graph for this function:



6.10.3.9 SWBPLC SWBPLC::operator= (const SWBPLC & *orig*) [inline]

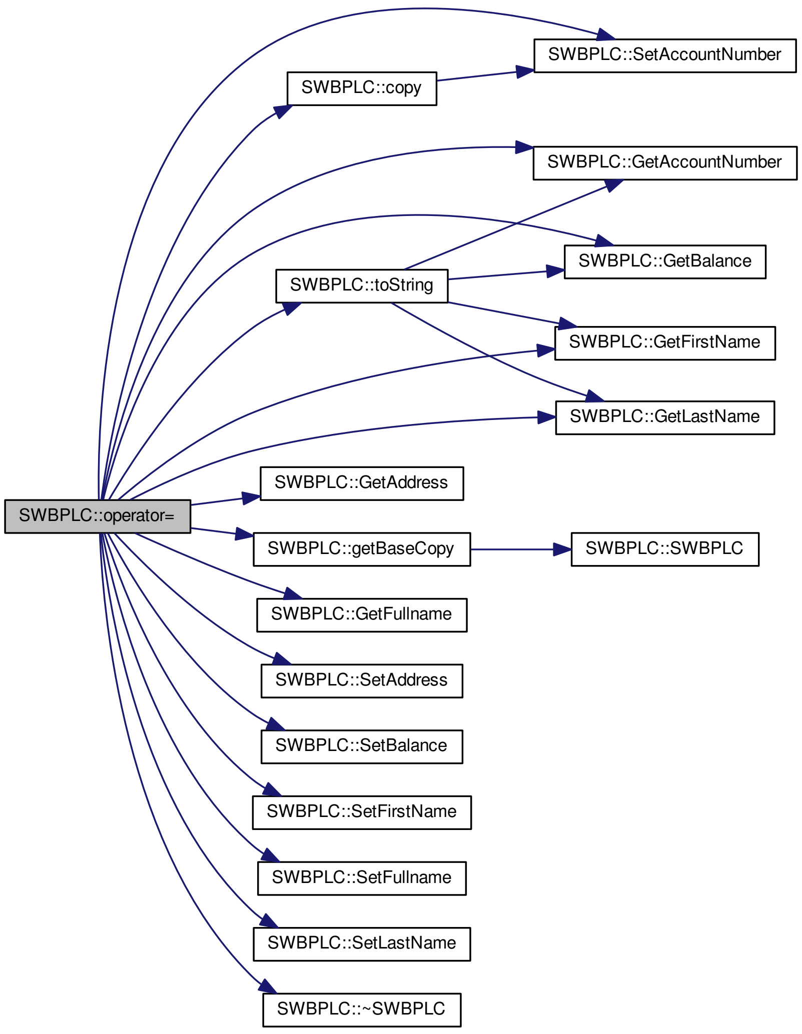
Operator

Definition at line 63 of file SWBPLC.h.

References copy(), GetAccountNumber(), GetAddress(), GetBalance(), getBaseCopy(), GetFirstName(), Get↩ Fullname(), 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]

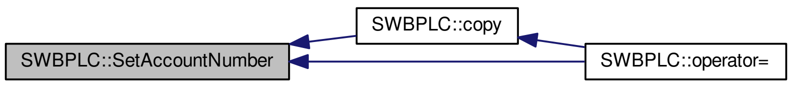
Implements BANK.

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]

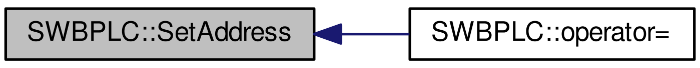
Implements BANK.

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]

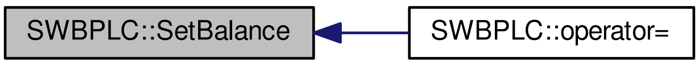
Implements BANK.

Definition at line 68 of file SWBPLC.cpp.

Referenced by operator=().

```
                                                {
    this->balance = balance;
}
```

Here is the caller graph for this function:



6.10.3.13 void SWBPLC::SetFirstName (std::string *firstName*) [virtual]

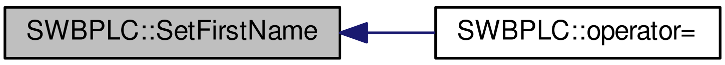
Implements BANK.

Definition at line 92 of file SWBPLC.cpp.

Referenced by operator=().

```
                                                        {
    this->firstName = firstName;
}
```

Here is the caller graph for this function:



6.10.3.14 void SWBPLC::SetFullname (std::string *fullname*) [virtual]

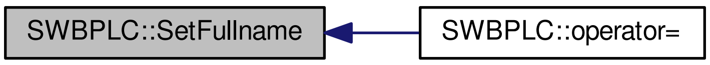
Implements BANK.

Definition at line 100 of file SWBPLC.cpp.

Referenced by operator=().

```
                                        {
        this->fullname = fullname;
    }
```

Here is the caller graph for this function:



6.10.3.15 void SWBPLC::SetLastName (std::string *lastName*) [virtual]

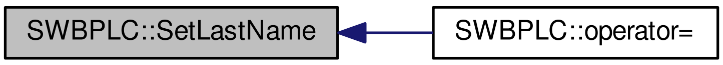
Implements BANK.

Definition at line 84 of file SWBPLC.cpp.

Referenced by operator=().

```
                                        {
        this->lastName = lastName;
    }
```

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

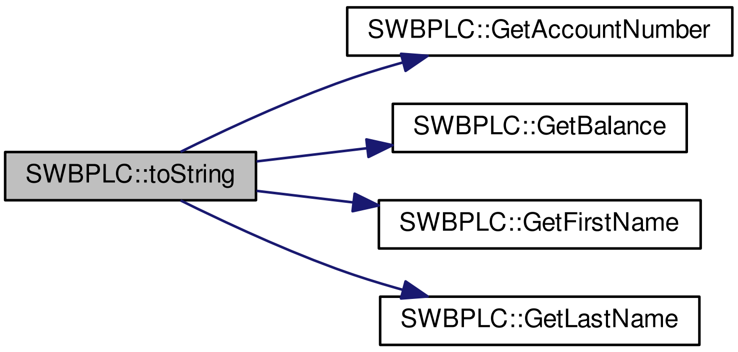
Definition at line 55 of file SWBPLC.cpp.

References GetAccountNumber(), GetBalance(), GetFirstName(), and GetLastName().

Referenced by operator=().

```
{
    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;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

SWBPLC::toString
SWBPLC::operator=

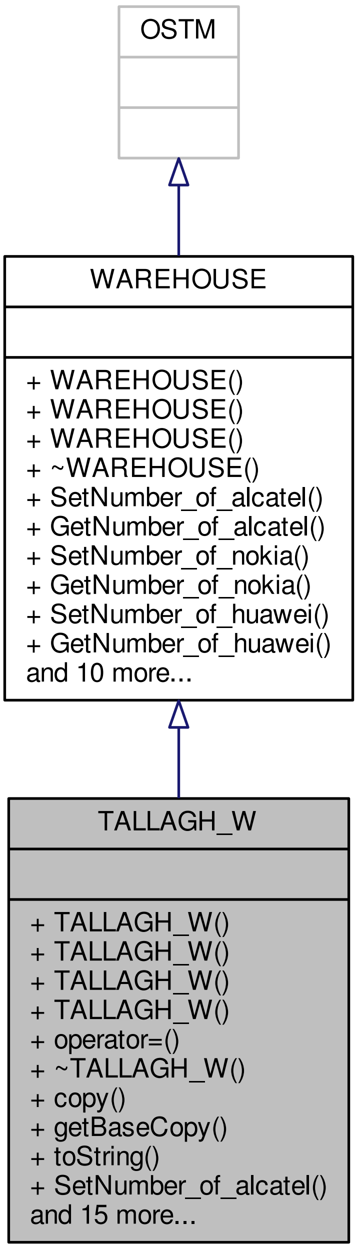
The documentation for this class was generated from the following files:

- SWBPLC.h
- SWBPLC.cpp

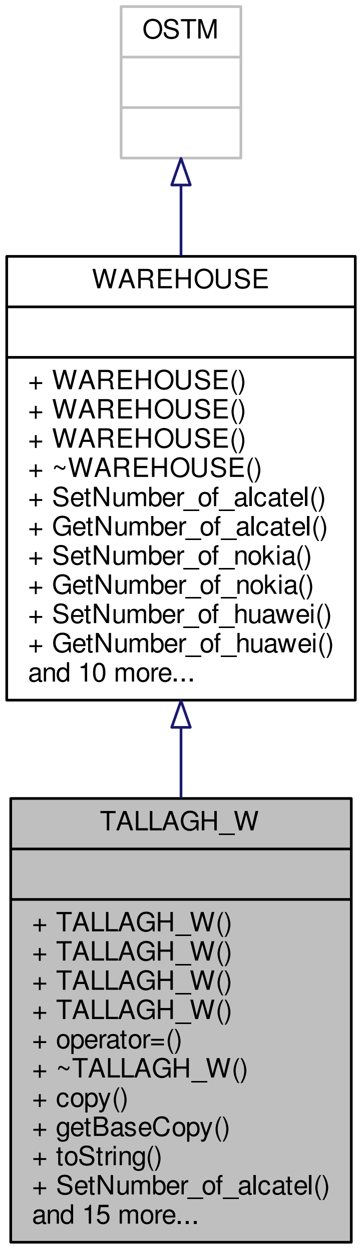
6.11 TALLAGH_W Class Reference

```
#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]

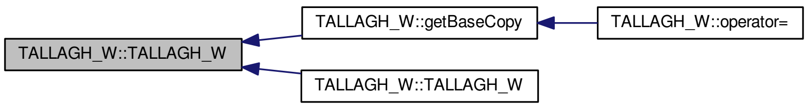
Constructor

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.

```
                       : WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
```

6.11.2.3 TALLAGH_W::TALLAGH_W (std::shared_ptr< WAREHOUSE > *obj*, int *_version*, int *_unique_id*) [inline]

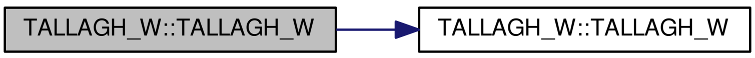
Custom constructor, used by the library for deep copying

Definition at line 55 of file TALLAGH_W.h.

References TALLAGH_W().

```
                                                                          :
      WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
```

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.

```
                                        {
}
```

6.11.2.5 TALLAGH_W::∼TALLAGH_W ( ) [virtual]

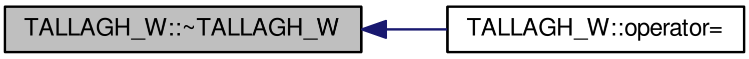
de-constructor

Definition at line 12 of file TALLAGH_W.cpp.

Referenced by operator=().

```
                          {
}
```

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

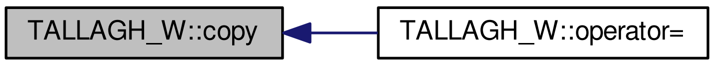
objTO	is a BANK∗ type object casted back from std::shared_ptr<OSTM>
objFROM	is a BANK∗ type object casted back from std::shared_ptr<OSTM>

Definition at line 35 of file TALLAGH_W.cpp.

Referenced by operator=().

```
                                                                     {

    std::shared_ptr<TALLAGH_W> objTO = std::dynamic_pointer_cast<TALLAGH_W>(to);
    std::shared_ptr<TALLAGH_W> objFROM = std::dynamic_pointer_cast<TALLAGH_W>(from);
    objTO->_shop_address = objFROM->GetShop_address();
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
```

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

objTO	is a BANK type pointer for casting
obj	is a BANK∗ return type

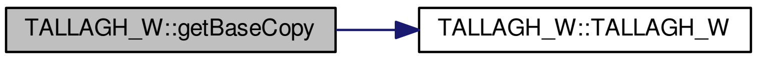
Definition at line 22 of file TALLAGH_W.cpp.

References TALLAGH_W().

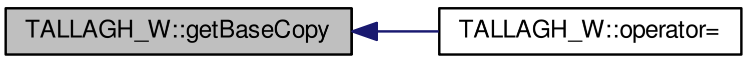
Referenced by operator=().

```
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new TALLAGH_W(objTO, object->Get_Version(),object->
  Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

    return ostm_obj;
}
```

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]

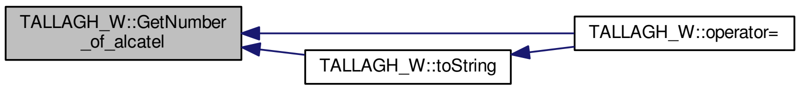
Implements WAREHOUSE.

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]

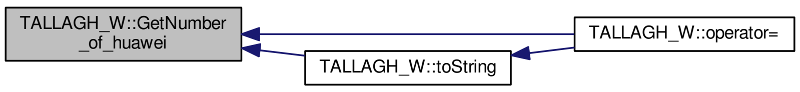
Implements WAREHOUSE.

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]

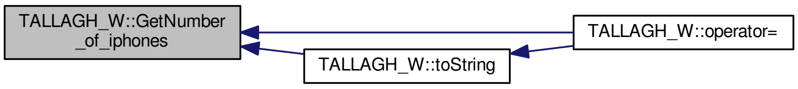
Implements WAREHOUSE.

Definition at line 111 of file TALLAGH_W.cpp.

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

```
                                     {
    return _number_of_iphones;
}
```

Here is the caller graph for this function:



6.11.3.6 int TALLAGH_W::GetNumber_of_nokia ( ) [virtual]

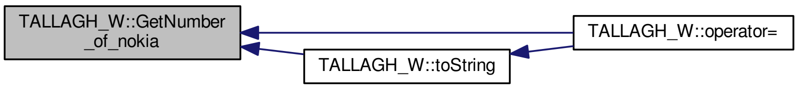
Implements WAREHOUSE.

Definition at line 79 of file TALLAGH_W.cpp.

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

```
                                   {
    return _number_of_nokia;
}
```

Here is the caller graph for this function:



6.11.3.7 int TALLAGH_W::GetNumber_of_samsung ( ) [virtual]

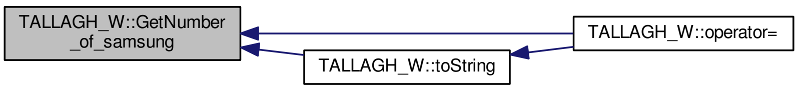
Implements WAREHOUSE.

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]

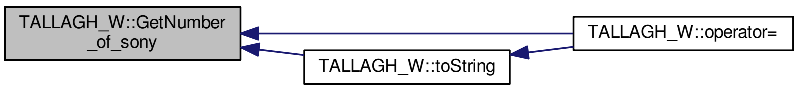
Implements WAREHOUSE.

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]

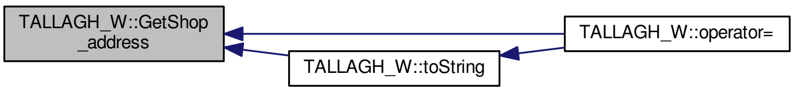
Implements WAREHOUSE.

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]

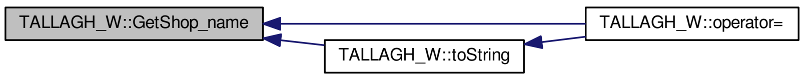
Implements WAREHOUSE.

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]

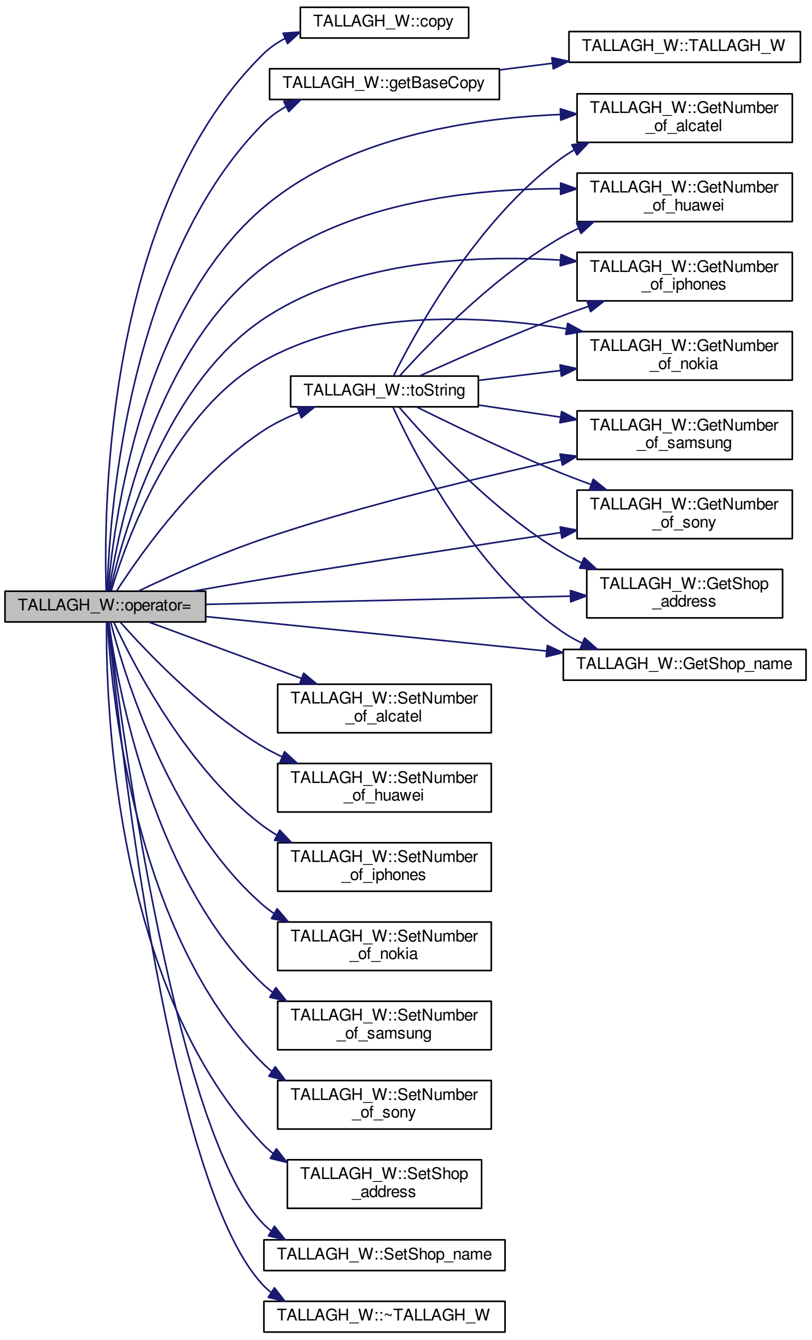
Operator

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 ∼TAL↵
LAGH_W().

```
00075 {};
```

Here is the call graph for this function:


TALLAGH_W::operator=
TALLAGH_W::copy
TALLAGH_W::getBaseCopy
TALLAGH_W::TALLAGH_W
TALLAGH_W::GetNumber_of_alcatel
TALLAGH_W::GetNumber_of_huawei
TALLAGH_W::GetNumber_of_iphones
TALLAGH_W::GetNumber_of_nokia
TALLAGH_W::toString
TALLAGH_W::GetNumber_of_samsung
TALLAGH_W::GetNumber_of_sony
TALLAGH_W::GetShop_address
TALLAGH_W::GetShop_name
TALLAGH_W::SetNumber_of_alcatel
TALLAGH_W::SetNumber_of_huawei
TALLAGH_W::SetNumber_of_iphones
TALLAGH_W::SetNumber_of_nokia
TALLAGH_W::SetNumber_of_samsung
TALLAGH_W::SetNumber_of_sony
TALLAGH_W::SetShop_address
TALLAGH_W::SetShop_name
TALLAGH_W::~TALLAGH_W

6.11.3.12 void TALLAGH_W::SetNumber_of_alcatel (int *_number_of_alcatel*) [virtual]

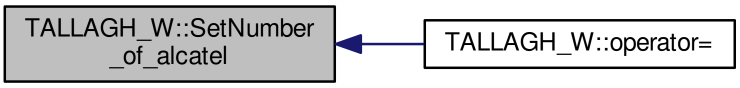
Implements WAREHOUSE.

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]

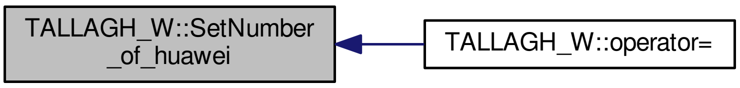
Implements WAREHOUSE.

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]

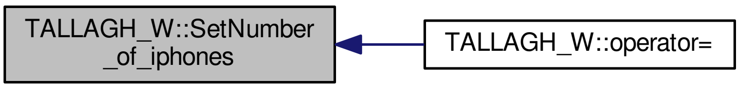
Implements WAREHOUSE.

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]

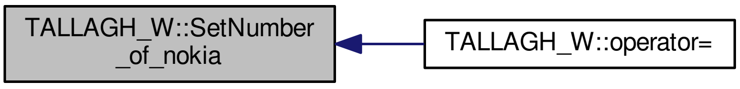
Implements WAREHOUSE.

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]

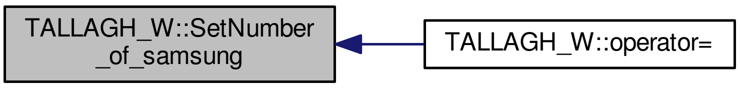
Implements WAREHOUSE.

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]

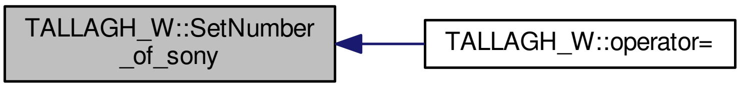
Implements WAREHOUSE.

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]

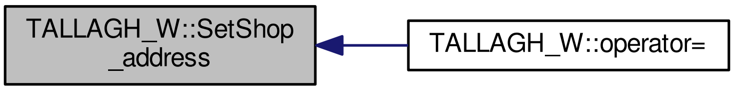
Implements WAREHOUSE.

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]

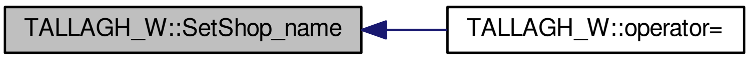
Implements WAREHOUSE.

Definition at line 115 of file TALLAGH_W.cpp.

Referenced by operator=().

```
                                                  {
    this->_shop_name = _shop_name;
}
```

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

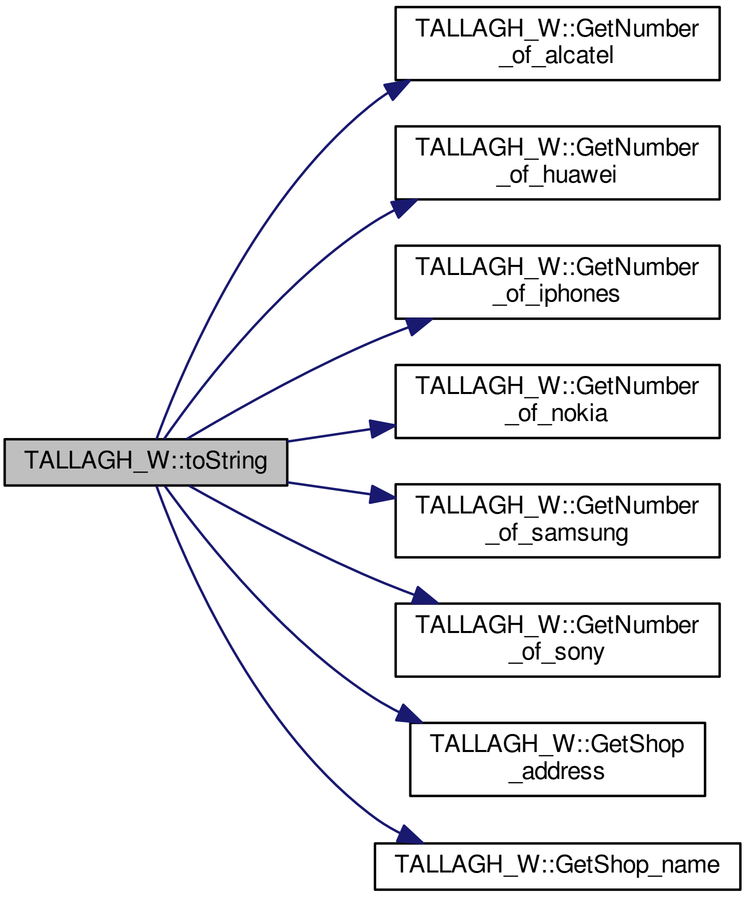
Definition at line 62 of file TALLAGH_W.cpp.

References GetNumber_of_alcatel(), GetNumber_of_huawei(), GetNumber_of_iphones(), GetNumber_of_nokia(), GetNumber_of_samsung(), GetNumber_of_sony(), GetShop_address(), and GetShop_name().

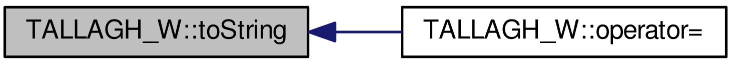
Referenced by operator=().

```
{
    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;
}
```

Here is the call graph for this function:


TALLAGH_W::toString
TALLAGH_W::GetNumber_of_alcatel
TALLAGH_W::GetNumber_of_huawei
TALLAGH_W::GetNumber_of_iphones
TALLAGH_W::GetNumber_of_nokia
TALLAGH_W::GetNumber_of_samsung
TALLAGH_W::GetNumber_of_sony
TALLAGH_W::GetShop_address
TALLAGH_W::GetShop_name

Here is the caller graph for this function:



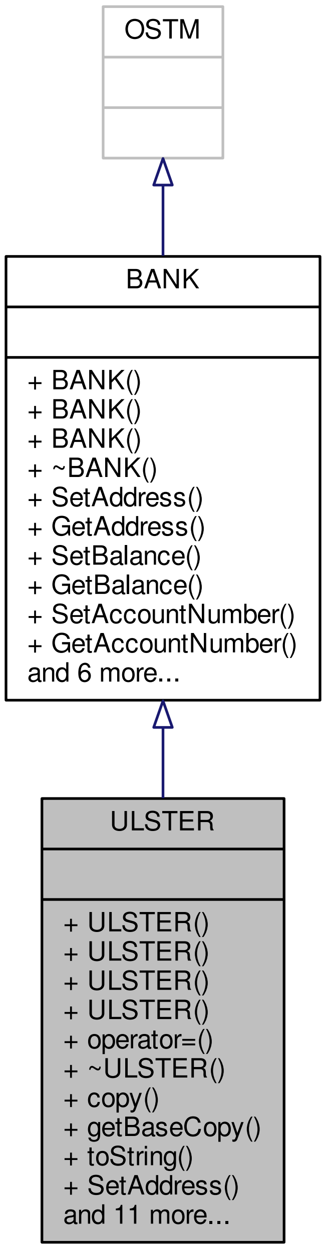
The documentation for this class was generated from the following files:

- TALLAGH_W.h
- TALLAGH_W.cpp

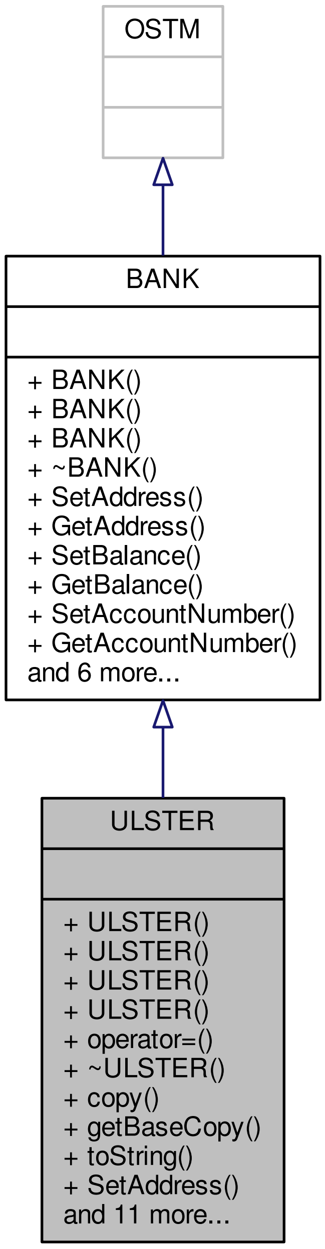
6.12 ULSTER Class Reference

```
#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]

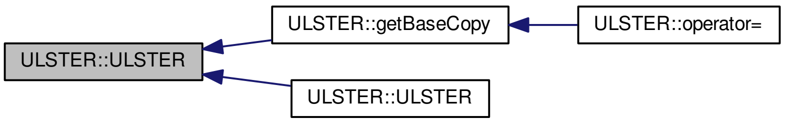
Constructor

Definition at line 24 of file ULSTER.h.

Referenced by getBaseCopy(), and ULSTER().

```
            : BANK() {
        this->accountNumber = 0;
        this->balance = 50;
        this->firstName = "Joe";
        this->lastName = "Blog";
        this->address = "High street, Carlow";
        this->fullname = firstName + " " + lastName;
    };
```

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.

```
                                                                                               :
       BANK() {
              this->accountNumber = accountNumber;
              this->balance = balance;
              this->firstName = firstName;
              this->lastName = lastName;
              this->address = address;
              this->fullname = firstName + " " + lastName;
          };
```

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

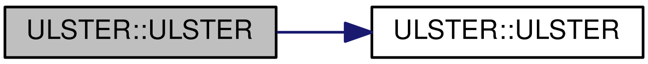
Custom constructor, used by the library for deep copying

Definition at line 46 of file ULSTER.h.

References ULSTER().

```
                                                                    : BANK(_version, _unique_id) {

              this->accountNumber = obj->GetAccountNumber();
              this->balance = obj->GetBalance();
              this->firstName = obj->GetFirstName();
              this->lastName = obj->GetLastName();
              this->address = obj->GetAddress();
              this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
          };
```

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.

```
                                  {
 }
```

6.12.2.5 ULSTER::∼ULSTER ( ) [virtual]

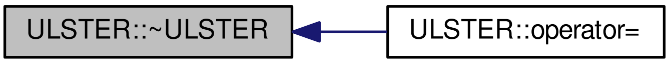
de-constructor

Definition at line 18 of file ULSTER.cpp.

Referenced by operator=().

```
                  {
}
```

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

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

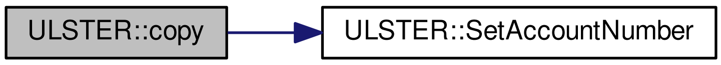
Definition at line 37 of file ULSTER.cpp.

References SetAccountNumber().

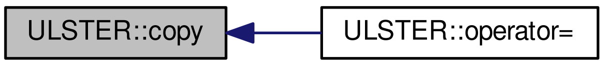
Referenced by operator=().

```
                                                               {

    std::shared_ptr<ULSTER> objTO = std::dynamic_pointer_cast<ULSTER>(to);
    std::shared_ptr<ULSTER> objFROM = std::dynamic_pointer_cast<ULSTER>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());


}
```

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]

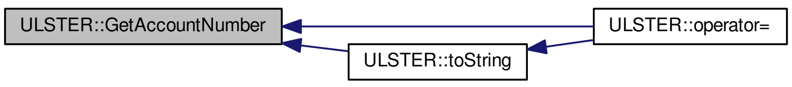
Implements BANK.

Definition at line 83 of file ULSTER.cpp.

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

```
                                        {
    return accountNumber;
}
```

Here is the caller graph for this function:



6.12.3.3 std::string ULSTER::GetAddress ( ) const [virtual]

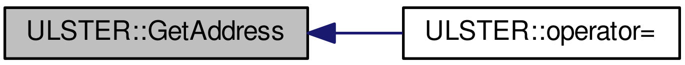
Implements BANK.

Definition at line 67 of file ULSTER.cpp.

Referenced by operator=().

```
                                  {
    return address;
}
```

Here is the caller graph for this function:



6.12.3.4 double ULSTER::GetBalance ( ) const [virtual]

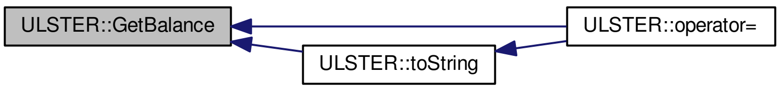
Implements BANK.

Definition at line 75 of file ULSTER.cpp.

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

```
                               {
    return balance;
}
```

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

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

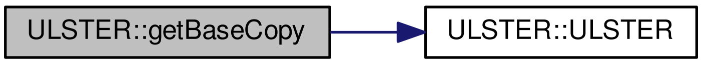
Definition at line 25 of file ULSTER.cpp.

References ULSTER().

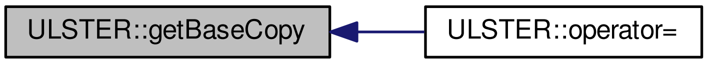
Referenced by operator=().

```
{
    std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
    std::shared_ptr<BANK> obj(new ULSTER(objTO,object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

    return ostm_obj;
}
```

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]

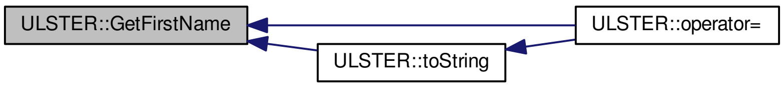
Implements BANK.

Definition at line 99 of file ULSTER.cpp.

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

```
                                    {
    return firstName;
}
```

Here is the caller graph for this function:



6.12.3.7 std::string ULSTER::GetFullname ( ) const [virtual]

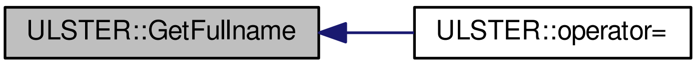
Implements BANK.

Definition at line 107 of file ULSTER.cpp.

Referenced by operator=().

```
                                           {
    return fullname;
}
```

Here is the caller graph for this function:



6.12.3.8 std::string ULSTER::GetLastName ( ) const [virtual]

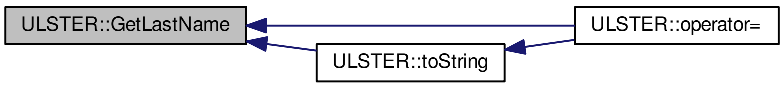
Implements BANK.

Definition at line 91 of file ULSTER.cpp.

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

```
                                           {
    return lastName;
}
```

Here is the caller graph for this function:



6.12.3.9 ULSTER ULSTER::operator= (const ULSTER & *orig*) `[inline]`

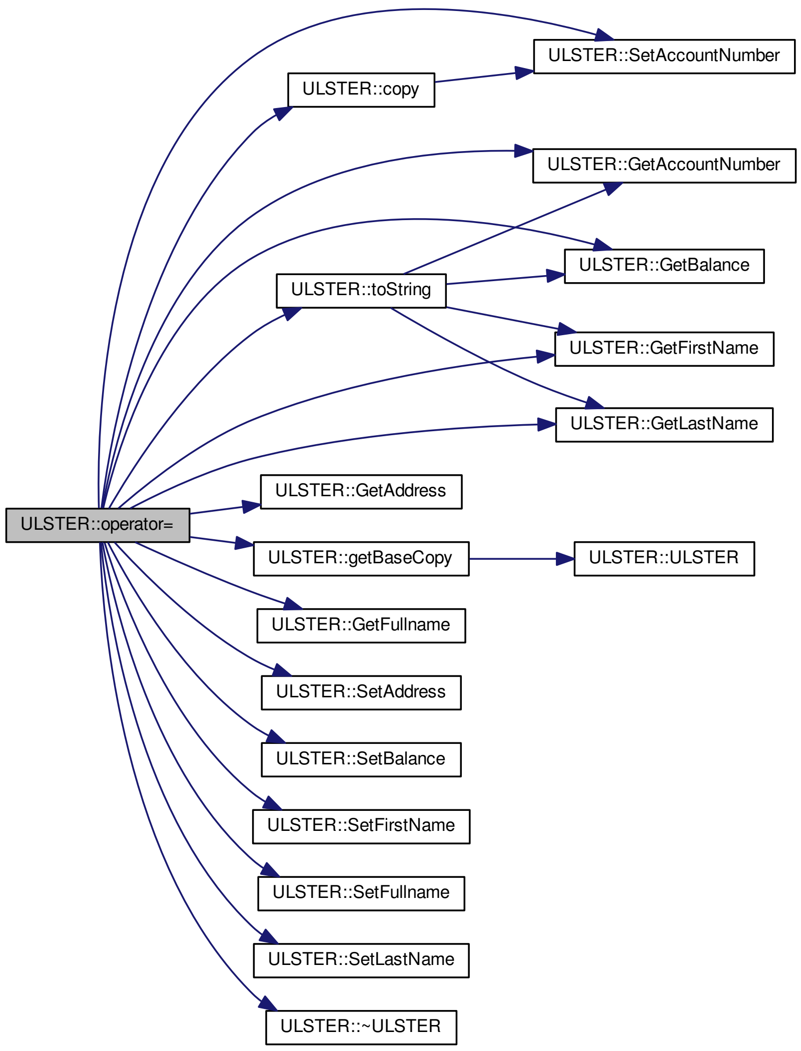
Operator

Definition at line 62 of file ULSTER.h.

References copy(), GetAccountNumber(), GetAddress(), GetBalance(), getBaseCopy(), GetFirstName(), Get←
Fullname(), 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]

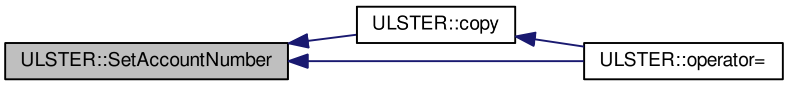
Implements BANK.

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]

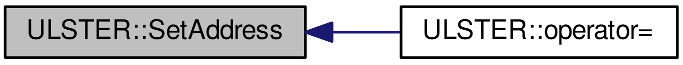
Implements BANK.

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]

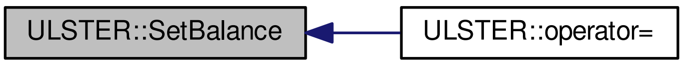
Implements BANK.

Definition at line 71 of file ULSTER.cpp.

Referenced by operator=().

```
                                       {
    this->balance = balance;
}
```

Here is the caller graph for this function:



6.12.3.13 void ULSTER::SetFirstName (std::string *firstName*) [virtual]

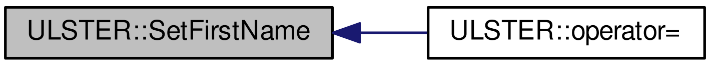
Implements BANK.

Definition at line 95 of file ULSTER.cpp.

Referenced by operator=().

```
                                              {
    this->firstName = firstName;
}
```

Here is the caller graph for this function:



6.12.3.14 void ULSTER::SetFullname (std::string *fullname*) [virtual]

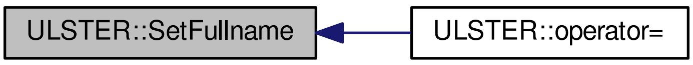
Implements BANK.

Definition at line 103 of file ULSTER.cpp.

Referenced by operator=().

```
                                       {
    this->fullname = fullname;
}
```

Here is the caller graph for this function:



6.12.3.15 void ULSTER::SetLastName (std::string *lastName*) [virtual]

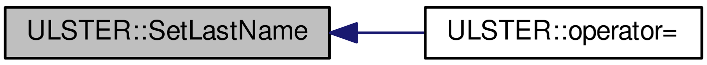
Implements BANK.

Definition at line 87 of file ULSTER.cpp.

Referenced by operator=().

```
                                       {
    this->lastName = lastName;
}
```

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

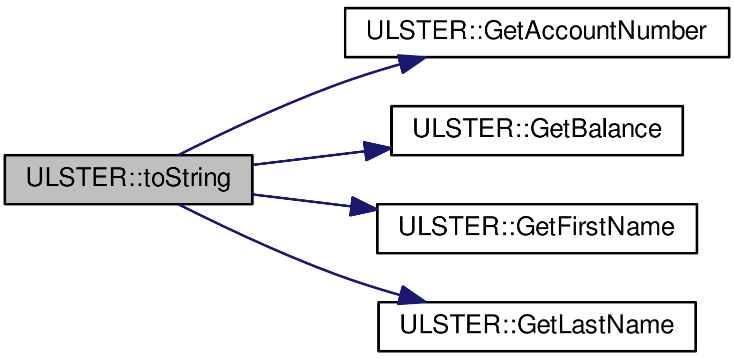
Definition at line 58 of file ULSTER.cpp.

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:

ULSTER::toString
ULSTER::operator=

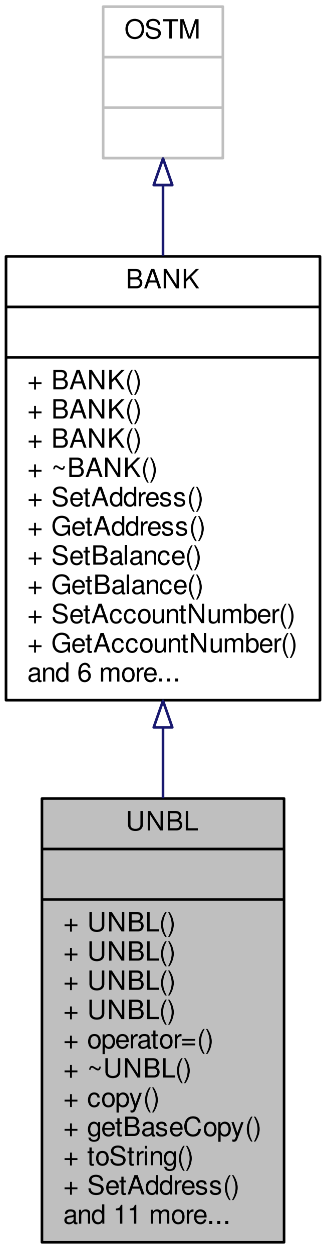
The documentation for this class was generated from the following files:

- ULSTER.h
- ULSTER.cpp

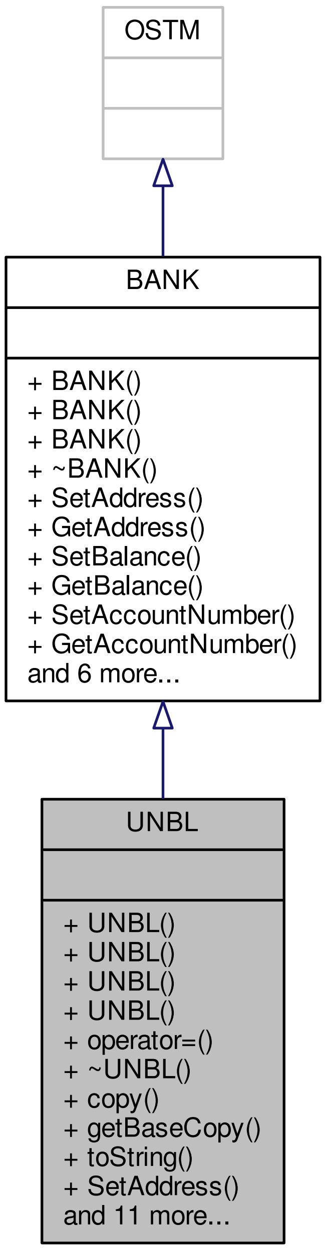
6.13 UNBL Class Reference

```
#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]

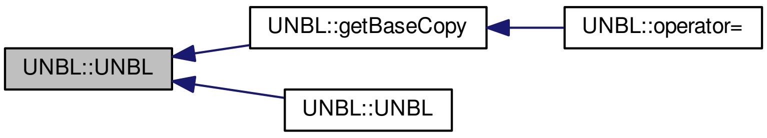
Constructor

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.

```
                                                                                                    :
      BANK() {
            this->accountNumber = accountNumber;
            this->balance = balance;
            this->firstName = firstName;
            this->lastName = lastName;
            this->address = address;
            this->fullname = firstName + " " + lastName;
        };
```

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

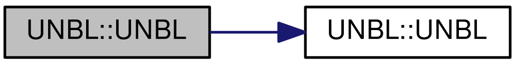
Custom constructor, used by the library for deep copying

Definition at line 46 of file UNBL.h.

References UNBL().

```
                                                                : BANK(_version, _unique_id) {

            this->accountNumber = obj->GetAccountNumber();
            this->balance = obj->GetBalance();
            this->firstName = obj->GetFirstName();
            this->lastName = obj->GetLastName();
            this->address = obj->GetAddress();
            this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
        };
```

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.

```
                           {
}
```

6.13.2.5 UNBL::∼UNBL ( ) [virtual]

de-constructor

Definition at line 14 of file UNBL.cpp.

Referenced by operator=().

```
{
}
```

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

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

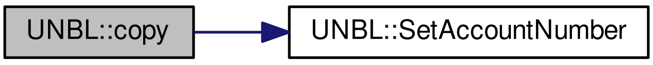
Definition at line 33 of file UNBL.cpp.

References SetAccountNumber().

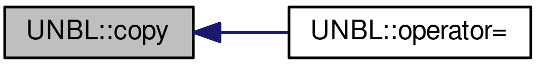
Referenced by operator=().

```
                                                            {

    std::shared_ptr<UNBL> objTO = std::dynamic_pointer_cast<UNBL>(to);
    std::shared_ptr<UNBL> objFROM = std::dynamic_pointer_cast<UNBL>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());

}
```

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]

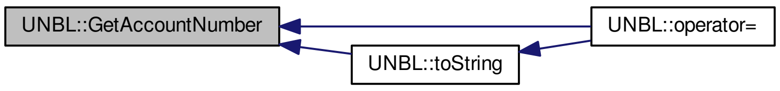
Implements BANK.

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]

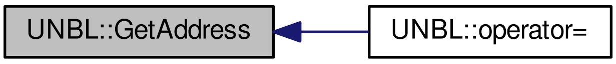
Implements BANK.

Definition at line 62 of file UNBL.cpp.

Referenced by operator=().

```
{
    return address;
}
```

Here is the caller graph for this function:



6.13.3.4 double UNBL::GetBalance ( ) const [virtual]

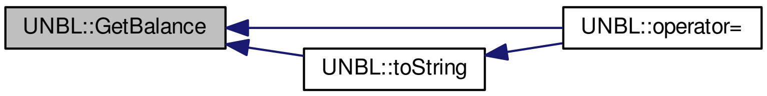
Implements BANK.

Definition at line 70 of file UNBL.cpp.

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

```
{
    return balance;
}
```

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

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

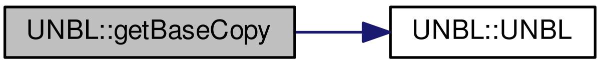
Definition at line 21 of file UNBL.cpp.

References UNBL().

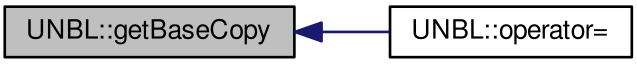
Referenced by operator=().

```
{
    std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
    std::shared_ptr<BANK> obj(new UNBL(objTO,object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

    return ostm_obj;
}
```

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]

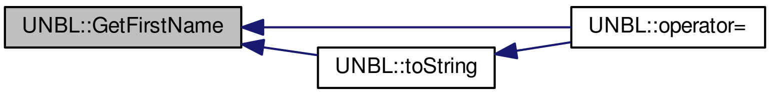
Implements BANK.

Definition at line 94 of file UNBL.cpp.

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

```
                                {
    return firstName;
}
```

Here is the caller graph for this function:



6.13.3.7 std::string UNBL::GetFullname ( ) const [virtual]

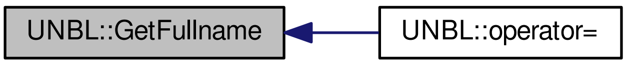
Implements BANK.

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]

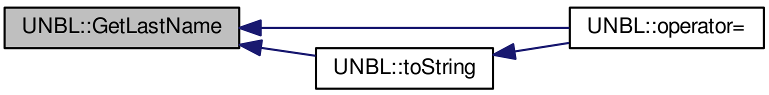
Implements BANK.

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]

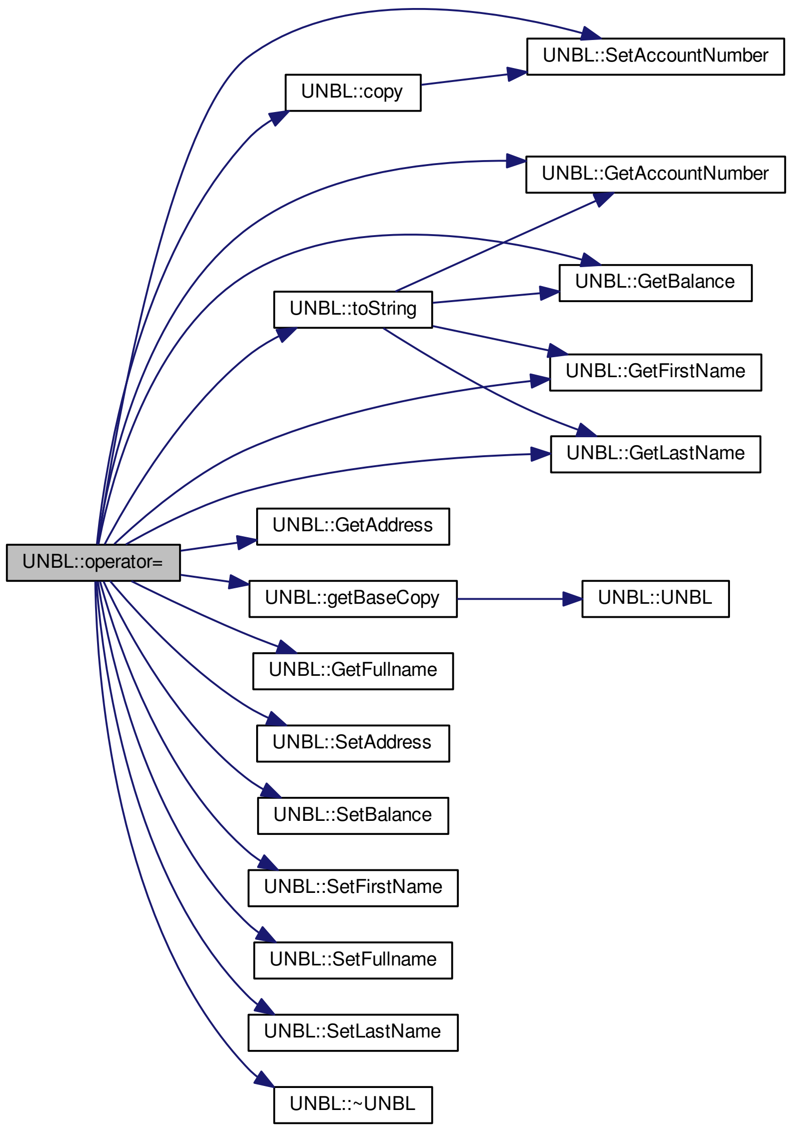
Operator

Definition at line 62 of file UNBL.h.

References copy(), GetAccountNumber(), GetAddress(), GetBalance(), getBaseCopy(), GetFirstName(), Get←
Fullname(), 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]

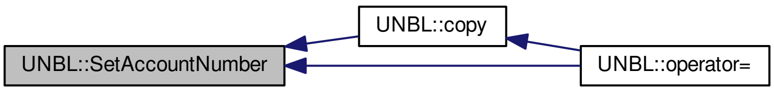
Implements BANK.

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]

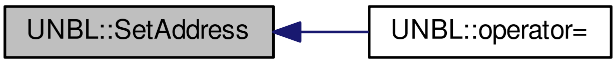
Implements BANK.

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]

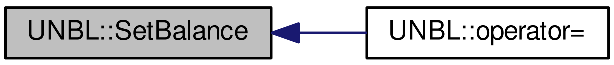
Implements BANK.

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]

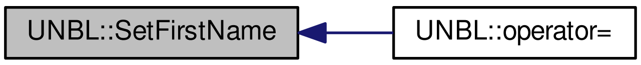
Implements BANK.

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]

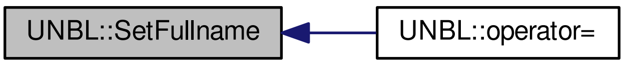
Implements BANK.

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]

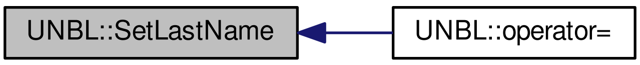
Implements BANK.

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

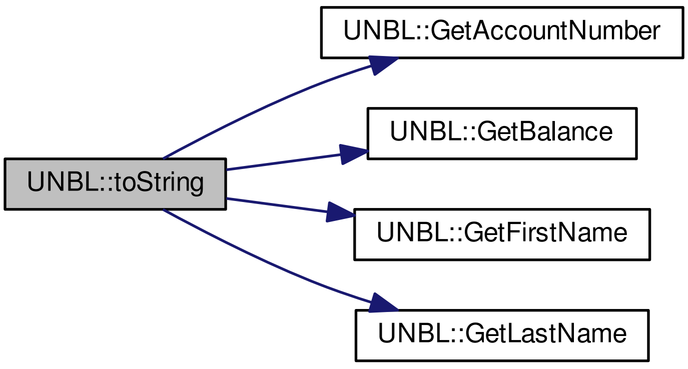
Definition at line 53 of file UNBL.cpp.

References GetAccountNumber(), GetBalance(), GetFirstName(), and GetLastName().

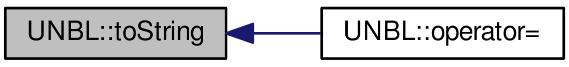
Referenced by operator=().

```
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:



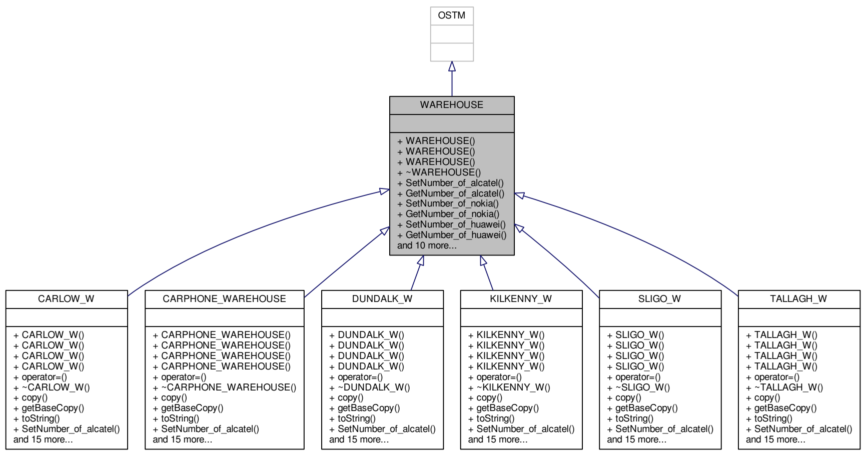
The documentation for this class was generated from the following files:

- UNBL.h
- UNBL.cpp

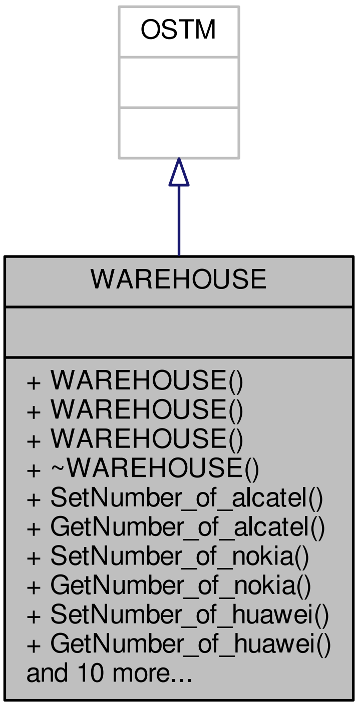
6.14 WAREHOUSE Class Reference

```
#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]

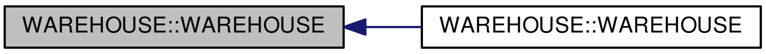
Constructor

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]

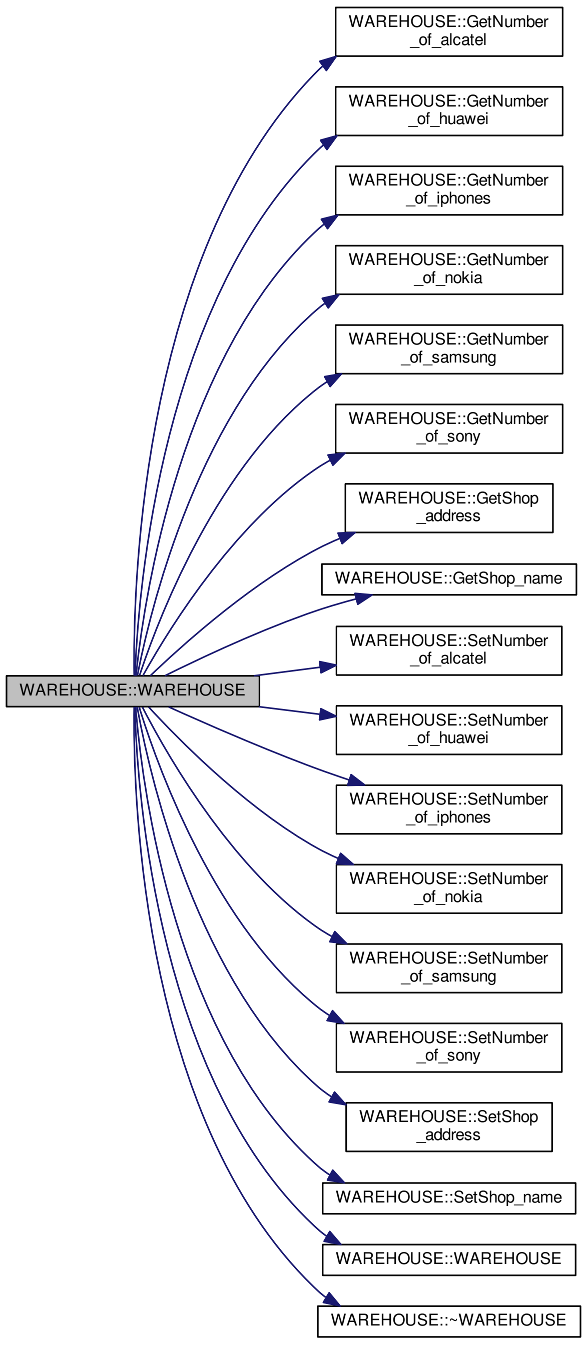
Custom Constructor

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().

```
                                              : OSTM(_version, _unique_id){

    };
```

Here is the call graph for this function:


WAREHOUSE::WAREHOUSE
WAREHOUSE::GetNumber_of_alcatel
WAREHOUSE::GetNumber_of_huawei
WAREHOUSE::GetNumber_of_iphones
WAREHOUSE::GetNumber_of_nokia
WAREHOUSE::GetNumber_of_samsung
WAREHOUSE::GetNumber_of_sony
WAREHOUSE::GetShop_address
WAREHOUSE::GetShop_name
WAREHOUSE::SetNumber_of_alcatel
WAREHOUSE::SetNumber_of_huawei
WAREHOUSE::SetNumber_of_iphones
WAREHOUSE::SetNumber_of_nokia
WAREHOUSE::SetNumber_of_samsung
WAREHOUSE::SetNumber_of_sony
WAREHOUSE::SetShop_address
WAREHOUSE::SetShop_name
WAREHOUSE::WAREHOUSE
WAREHOUSE::~WAREHOUSE

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]

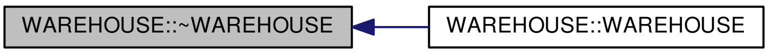
de-constructor

Definition at line 15 of file WAREHOUSE.cpp.

Referenced by WAREHOUSE().

```
00015                          {
00016 }
```

Here is the caller graph for this function:



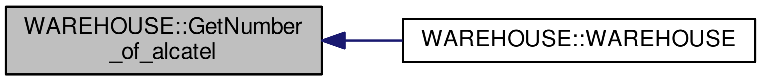
6.14.3 Member Function Documentation

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 KILK↩ ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

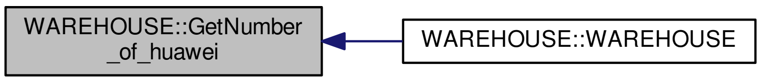


6.14.3.2 virtual int WAREHOUSE::GetNumber_of_huawei ( ) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩ ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

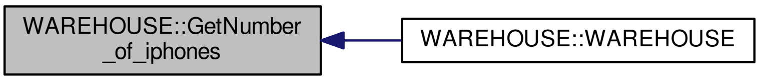


6.14.3.3 virtual int WAREHOUSE::GetNumber_of_iphones ( ) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩
ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

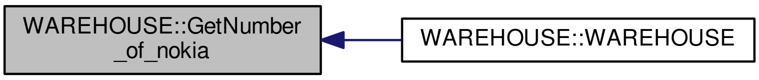


6.14.3.4 virtual int WAREHOUSE::GetNumber_of_nokia ( ) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩
ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

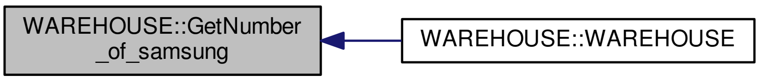


6.14.3.5 virtual int WAREHOUSE::GetNumber_of_samsung ( ) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩
ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

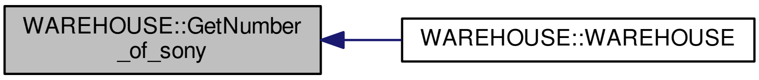


6.14.3.6 virtual int WAREHOUSE::GetNumber_of_sony ( ) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩ ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

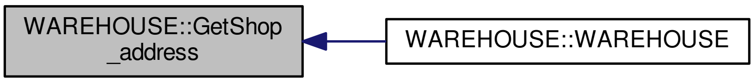


6.14.3.7 virtual std::string WAREHOUSE::GetShop_address ( ) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩ ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

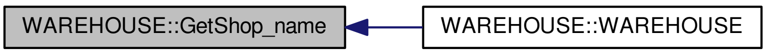


6.14.3.8 virtual std::string WAREHOUSE::GetShop_name ( ) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩ ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

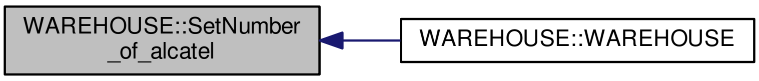


6.14.3.9 virtual void WAREHOUSE::SetNumber_of_alcatel (int *_number_of_alcatel*) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩ ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

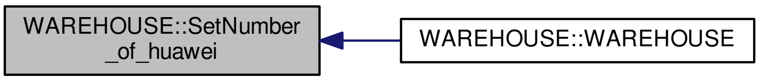


6.14.3.10 virtual void WAREHOUSE::SetNumber_of_huawei (int *_number_of_huawei*) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩ ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

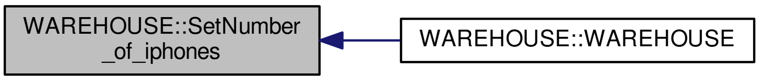


6.14.3.11 virtual void WAREHOUSE::SetNumber_of_iphones (int *_number_of_iphones*) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩ ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

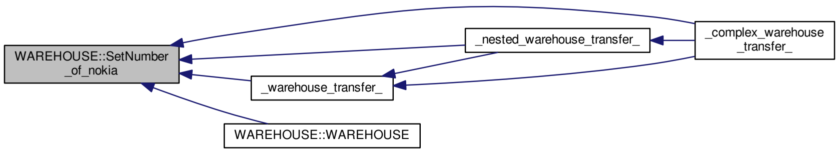


6.14.3.12 virtual void WAREHOUSE::SetNumber_of_nokia (int *_number_of_nokia*) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK←
ENNY_W.

Referenced by _complex_warehouse_transfer_(), _nested_warehouse_transfer_(), _warehouse_transfer_(), and WAREHOUSE().

Here is the caller graph for this function:

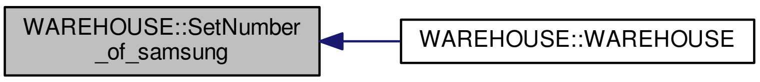

WAREHOUSE::SetNumber _of_nokia
_nested_warehouse_transfer_
_complex_warehouse _transfer_
_warehouse_transfer_
WAREHOUSE::WAREHOUSE

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 KILK←
ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

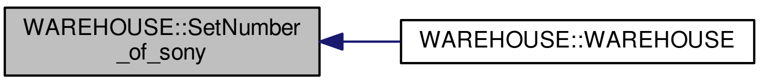


6.14.3.14 virtual void WAREHOUSE::SetNumber_of_sony (int *_number_of_sony*) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK←
ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

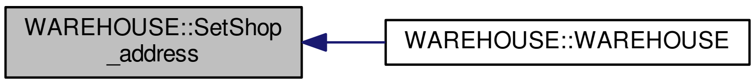


6.14.3.15 virtual void WAREHOUSE::SetShop_address (std::string *_shop_address*) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩
ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:

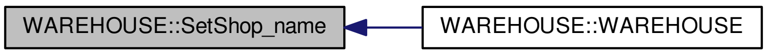


6.14.3.16 virtual void WAREHOUSE::SetShop_name (std::string *_shop_name*) [pure virtual]

Implemented in CARPHONE_WAREHOUSE, SLIGO_W, TALLAGH_W, CARLOW_W, DUNDALK_W, and KILK↩
ENNY_W.

Referenced by WAREHOUSE().

Here is the caller graph for this function:



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

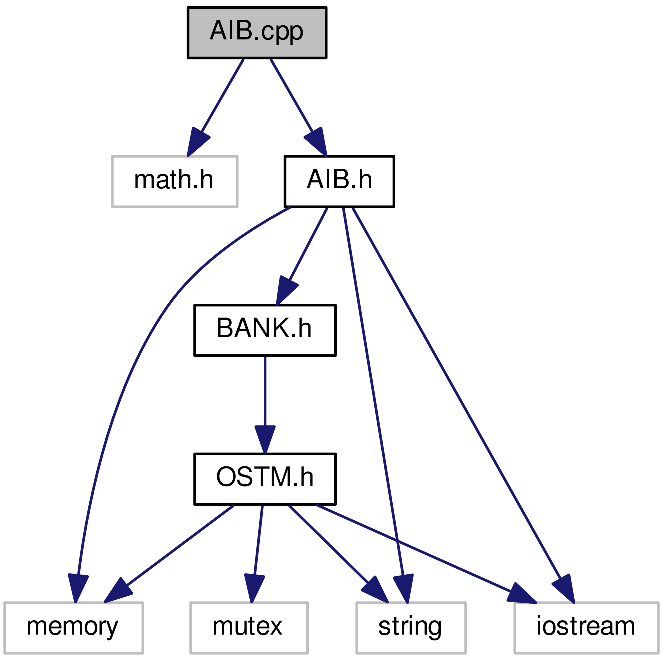
- WAREHOUSE.h
- WAREHOUSE.cpp

7 File Documentation

7.1 AIB.cpp File Reference

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

Include dependency graph for AIB.cpp:



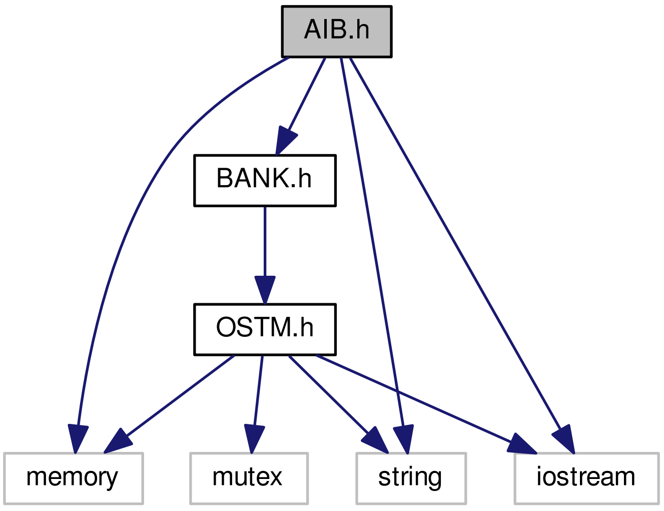
7.2 AIB.cpp

```
/*
 * File:   AIB.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include <math.h>

#include "AIB.h"


AIB::AIB(const AIB& orig) {
}

AIB::~AIB() {
}
std::shared_ptr<OSTM> AIB::getBaseCopy(std::shared_ptr<OSTM> object)
{

    std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
    std::shared_ptr<BANK> obj(new AIB(objTO, object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
void AIB::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<AIB> objTO = std::dynamic_pointer_cast<AIB>(to);
    std::shared_ptr<AIB> objFROM = std::dynamic_pointer_cast<AIB>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());
}
//std::shared_ptr<AIB> AIB::_cast(std::shared_ptr<OSTM> _object){
//
//    return std::static_pointer_cast<AIB>(_object);
//}
void AIB::toString()
{
    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;
}

void AIB::SetAddress(std::string address) {
    this->address = address;
}

std::string AIB::GetAddress() const {
    return address;
}

void AIB::SetBalance(double balance) {
    this->balance = balance;
}

double AIB::GetBalance() const {
    return balance;
}

void AIB::SetAccountNumber(int accountNumber) {
    this->accountNumber = accountNumber;
}

int AIB::GetAccountNumber() const {
    return accountNumber;
}

void AIB::SetLastName(std::string lastName) {
    this->lastName = lastName;
}

std::string AIB::GetLastName() const {
    return lastName;
}

void AIB::SetFirstName(std::string firstName) {
    this->firstName = firstName;
}

std::string AIB::GetFirstName() const {
    return firstName;
}

void AIB::SetFullname(std::string fullname) {
    this->fullname = fullname;
}

std::string AIB::GetFullname() const {
    return fullname;
}
```

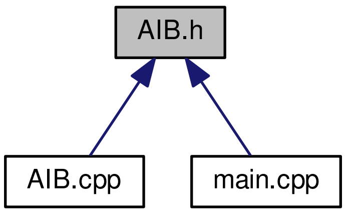
7.3 AIB.h File Reference

```
#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

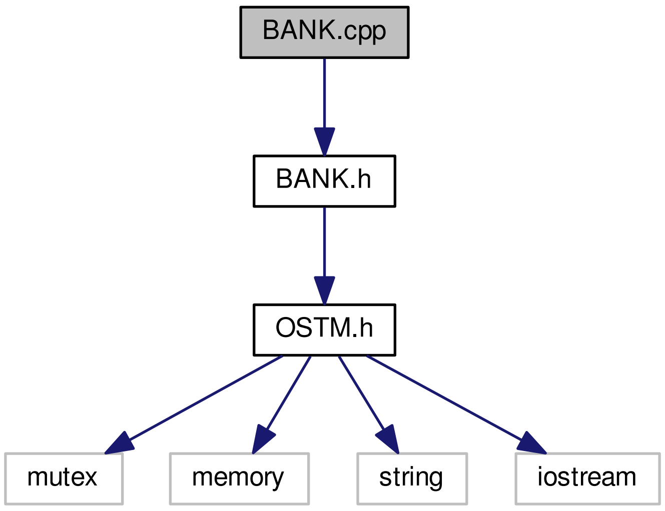
```
/*
 * File:   AIB.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */
```

```
#ifndef AIB_H
#define AIB_H
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>
class AIB : public BANK {
public:
    AIB(): BANK()
    {
        this->accountNumber = 0;
        this->balance = 50;
        this->firstName = "Joe";
        this->lastName = "Blog";
        this->address = "High street, Carlow";
        this->fullname = firstName + " " + lastName;

    };
    AIB(int accountNumber, double balance, std::string firstName, std::string lastName, std::string
address): BANK()
    {
        this->accountNumber = accountNumber;
        this->balance = balance;
        this->firstName = firstName;
        this->lastName = lastName;
        this->address = address;
        this->fullname = firstName + " " + lastName;
    };
    AIB(std::shared_ptr<BANK> obj, int _version, int _unique_id): BANK(_version, _unique_id)
    {

        this->accountNumber = obj->GetAccountNumber();
        this->balance = obj->GetBalance();
        this->firstName = obj->GetFirstName();
        this->lastName = obj->GetLastName();
        this->address = obj->GetAddress();
        this->fullname = obj->GetFirstName() + " " + obj->GetLastName();

    };
    AIB(const AIB& orig);
    AIB operator=(const AIB& orig){};
    virtual ~AIB();

    /*
     * Implement OSTM virtual methods
     */
   // virtual std::shared_ptr<AIB> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();

    /*
     * Implement BANK virtual methods
     */
    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;

private:
    std::string fullname;
    std::string firstName;
    std::string lastName;
    int accountNumber;
    double balance;
    std::string address;


};

#endif /* AIB_H */
```

7.5 BANK.cpp File Reference

```
#include "BANK.h"
```

Include dependency graph for BANK.cpp:



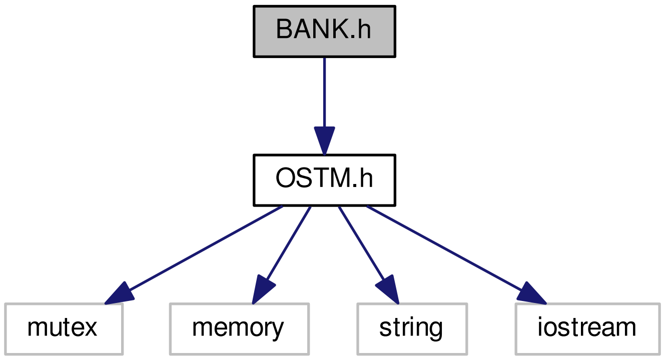
7.6 BANK.cpp

```
/*
 * File:   BANK.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "BANK.h"

BANK::BANK(const BANK& orig) {
}

BANK::~BANK() {
}

```

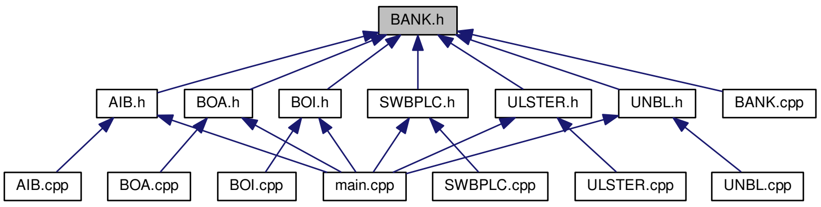
7.7 BANK.h File Reference

```
#include "OSTM.h"
```

Include dependency graph for BANK.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BANK

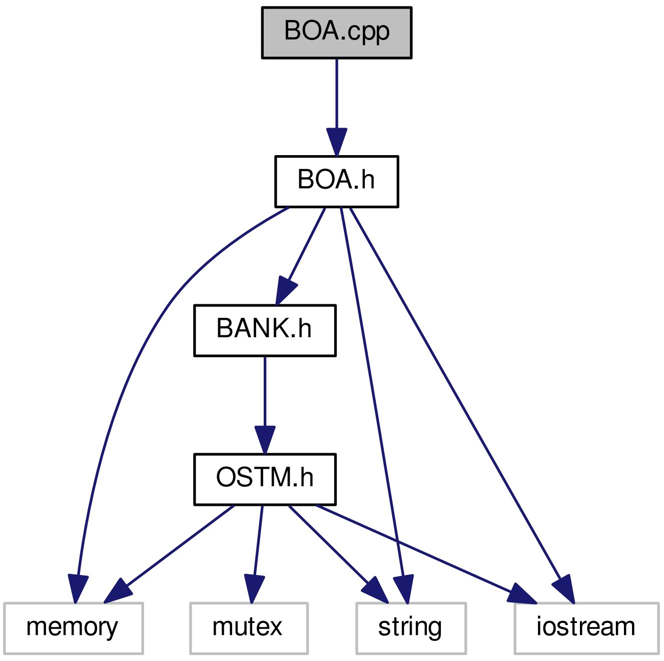
7.8 BANK.h

```
/*
 * File:   BANK.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef BANK_H
#define BANK_H
#include "OSTM.h"
class BANK : public OSTM {


public:
    BANK(): OSTM(){

    };
    BANK(int _version, int _unique_id) : OSTM(_version, _unique_id){

    };
    BANK(const BANK& orig);
    virtual ~BANK();

    /*
     * Bank specific virtual functions
     */
    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;

private:

};

#endif /* BANK_H */

```

7.9 BOA.cpp File Reference

`#include "BOA.h"`

Include dependency graph for BOA.cpp:



7.10 BOA.cpp

```
/*
```

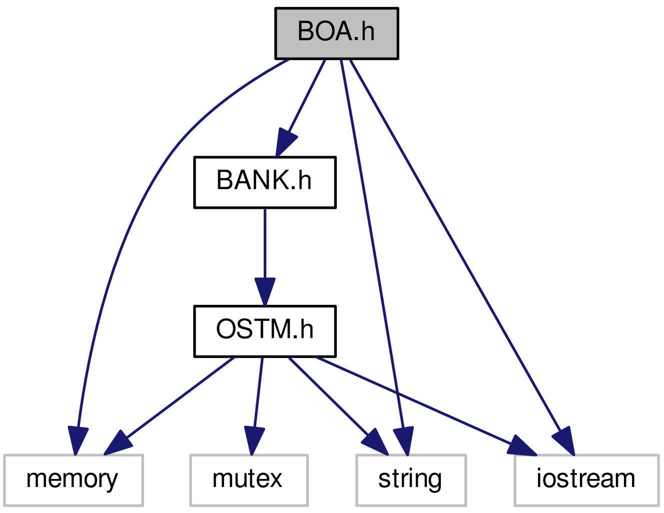
```cpp
 * File:   BOA.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "BOA.h"


BOA::BOA(const BOA& orig) {
}

BOA::~BOA() {
}
std::shared_ptr<OSTM> BOA::getBaseCopy(std::shared_ptr<OSTM> object)
{
        std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
    std::shared_ptr<BANK> obj(new BOA(objTO,object->Get_Version(),object->Get_Unique_ID()));
        std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
void BOA::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<BOA> objTO = std::dynamic_pointer_cast<BOA>(to);
    std::shared_ptr<BOA> objFROM = std::dynamic_pointer_cast<BOA>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());

}
//std::shared_ptr<BOA> BOA::_cast(std::shared_ptr<OSTM> _object){
//
//    return std::static_pointer_cast<BOA>(_object);
//}
void BOA::toString()
{
   // 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;
     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;
}

void BOA::SetAddress(std::string address) {
    this->address = address;
}

std::string BOA::GetAddress() const {
    return address;
}

void BOA::SetBalance(double balance) {
    this->balance = balance;
}

double BOA::GetBalance() const {
    return balance;
}

void BOA::SetAccountNumber(int accountNumber) {
    this->accountNumber = accountNumber;
}

int BOA::GetAccountNumber() const {
    return accountNumber;
}

void BOA::SetLastName(std::string lastName) {
    this->lastName = lastName;
}

std::string BOA::GetLastName() const {
    return lastName;
}

void BOA::SetFirstName(std::string firstName) {
    this->firstName = firstName;
}

std::string BOA::GetFirstName() const {
    return firstName;
}
```

```

void BOA::SetFullname(std::string fullname) {
    this->fullname = fullname;
}

std::string BOA::GetFullname() const {
    return fullname;
}

```

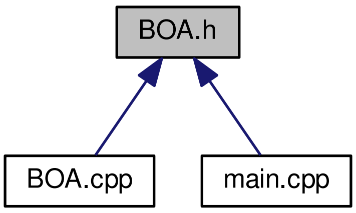
7.11 BOA.h File Reference

```
#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

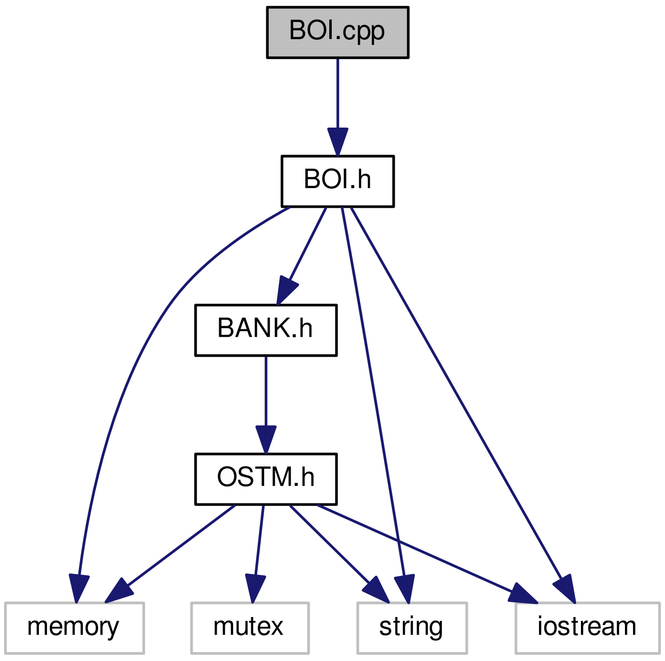
7.12 BOA.h

```
/*
 * File:   BOA.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef BOA_H
#define BOA_H
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>
class BOA : public BANK {
public:

    BOA() : BANK() {
        this->accountNumber = 0;
        this->balance = 50;
        this->firstName = "Joe";
        this->lastName = "Blog";
        this->address = "High street, Carlow";
        this->fullname = firstName + " " + lastName;
    };
    BOA(int accountNumber, double balance, std::string firstName, std::string lastName, std::string
address) : BANK() {
        this->accountNumber = accountNumber;
        this->balance = balance;
        this->firstName = firstName;
        this->lastName = lastName;
        this->address = address;
        this->fullname = firstName + " " + lastName;
    };
    BOA(std::shared_ptr<BANK> obj, int _version, int _unique_id) : BANK(_version, _unique_id) {

        this->accountNumber = obj->GetAccountNumber();
        this->balance = obj->GetBalance();
        this->firstName = obj->GetFirstName();
        this->lastName = obj->GetLastName();
        this->address = obj->GetAddress();
        this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
    };


    BOA(const BOA& orig);
    BOA operator=(const BOA& orig) {
    };
    virtual ~BOA();

    /*
     * Implement OSTM virtual methods
     */
    //virtual std::shared_ptr<BOA> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();

    /*
     * Implement BANK virtual methods
     */
    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;
private:
    std::string fullname;
    std::string firstName;
    std::string lastName;
    int accountNumber;
    double balance;
    std::string address;

};

#endif /* BOA_H */

```

7.13 BOI.cpp File Reference

`#include "BOI.h"`

Include dependency graph for BOI.cpp:



7.14 BOI.cpp

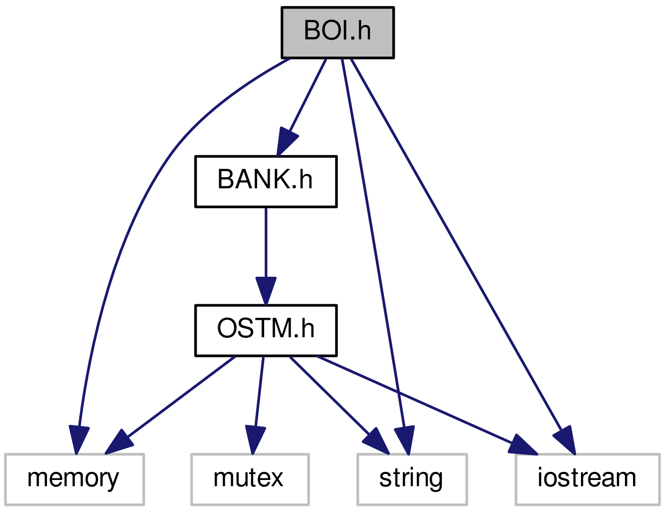
```

/*
 * File:   BOI.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "BOI.h"

BOI::~BOI() {
}

BOI::BOI(const BOI& orig) {
}
std::shared_ptr<OSTM> BOI::getBaseCopy(std::shared_ptr<OSTM> object)
```

```
{

    std::shared_ptr<BOI> objTO = std::dynamic_pointer_cast<BOI>(object);
    std::shared_ptr<BOI> obj(new BOI(objTO,object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
void BOI::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<BOI> objTO = std::dynamic_pointer_cast<BOI>(to);
    std::shared_ptr<BOI> objFROM = std::dynamic_pointer_cast<BOI>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());
}
//std::shared_ptr<BOI> BOI::_cast(std::shared_ptr<OSTM> _object){
//
//    return std::static_pointer_cast<BOI>(_object);
//}
void BOI::toString()
{
   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;
}
void BOI::SetAddress(std::string address) {
    this->address = address;
}

std::string BOI::GetAddress() const {
    return address;
}

void BOI::SetBalance(double balance) {
    this->balance = balance;
}

double BOI::GetBalance() const {
    return balance;
}

void BOI::SetAccountNumber(int accountNumber) {
    this->accountNumber = accountNumber;
}

int BOI::GetAccountNumber() const {
    return accountNumber;
}

void BOI::SetLastName(std::string lastName) {
    this->lastName = lastName;
}

std::string BOI::GetLastName() const {
    return lastName;
}

void BOI::SetFirstName(std::string firstName) {
    this->firstName = firstName;
}

std::string BOI::GetFirstName() const {
    return firstName;
}

void BOI::SetFullname(std::string fullname) {
    this->fullname = fullname;
}

std::string BOI::GetFullname() const {
    return fullname;
}

```

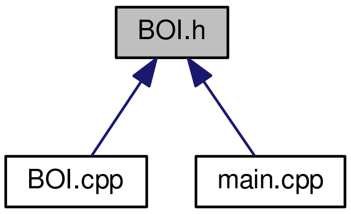
7.15 BOI.h File Reference

```
#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

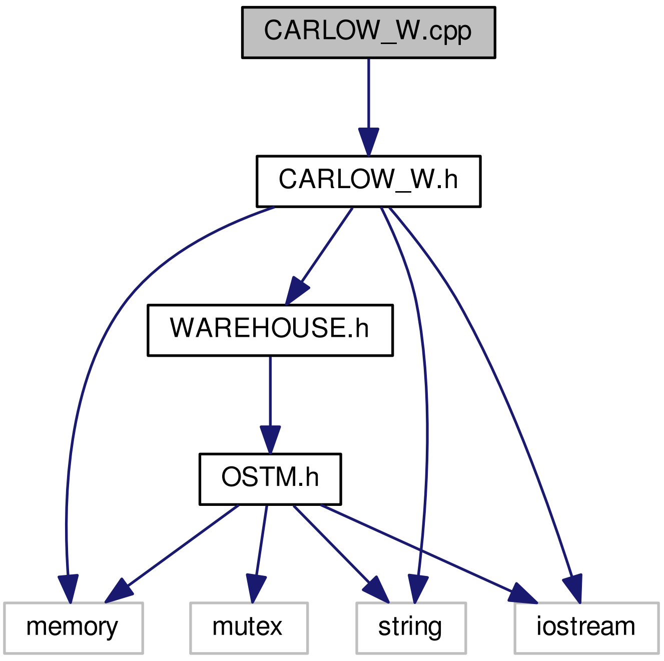
```
00001
00002 /*
00003  * File:   BOI.h
00004  * Author: Zoltan Fuzesi
00005  * IT Carlow : C00197361
00006  *
00007  * Created on January 17, 2018, 8:02 PM
```

```
 */

#ifndef BOI_H
#define BOI_H
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>
class BOI: public BANK {
public:
    BOI(): BANK()
    {
        this->accountNumber = 0;
        this->balance = 50;
        this->firstName = "Joe";
        this->lastName = "Blog";
        this->address = "High street, Carlow";
        this->fullname = firstName + " " + lastName;

    }
    BOI(int accountNumber, double balance, std::string firstName, std::string lastName, std::string
address): BANK()
    {
        this->accountNumber = accountNumber;
        this->balance = balance;
        this->firstName = firstName;
        this->lastName = lastName;
        this->address = address;
        this->fullname = firstName + " " + lastName;
    };
    BOI(std::shared_ptr<BOI> obj, int _version, int _unique_id): BANK(_version, _unique_id)
    {
        this->accountNumber = obj->GetAccountNumber();
        this->balance = obj->GetBalance();
        this->firstName = obj->GetFirstName();
        this->lastName = obj->GetLastName();
        this->address = obj->GetAddress();
        this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
    };
    BOI(const BOI& orig);
    BOI operator=(const BOI& orig){};
    virtual ~BOI();

    /*
     * Implement OSTM virtual methods
     */
   // virtual std::shared_ptr<BOI> _cast(std::shared_ptr<OSTM> _object);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual void toString();

    /*
     * Implement BANK virtual methods
     */
    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;

private:
    std::string fullname;
    std::string firstName;
    std::string lastName;
    int accountNumber;
    double balance;
    std::string address;

};

#endif /* BOI_H */


    //virtual std::string get_class();

   //
    //virtual bool get(std::shared_ptr<OSTM> object);

// * To change this license header, choose License Headers in Project Properties.
// * To change this template file, choose Tools | Templates
// * and open the template in the editor.
// */
//
// * File:   BOI.h
// * Author: zoltan
// *
// * Created on January 19, 2018, 6:37 PM
// */
//
//#ifndef BOI_H
//#define BOI_H
//#include "OSTM.h"
//#include <string>
//#include <memory>
//#include <iostream>
//
//class BOI: public OSTM {
//public:
//    BOI(): OSTM()
//    {
//        this->accountNumber = 0;
//        this->balance = 50;
//        this->firstName = "Joe";
//        this->lastName = "Blog";
//        this->address = "High street, Carlow";
//        this->fullname = firstName + " " + lastName;
//
//    }
//    BOI(int accountNumber, double balance, std::string firstName, std::string lastName, std::string address): OSTM()
//    {
//        this->accountNumber = accountNumber;
//        this->balance = balance;
//        this->firstName = firstName;
//        this->lastName = lastName;
//        this->address = address;
//        this->fullname = firstName + " " + lastName;
//    };
//
//     BOI(OSTM& obj, int _version, int _unique_id): OSTM(_version, _unique_id)
//    {
//          this->accountNumber = obj.GetAccountNumber();
//        this->balance = obj.GetBalance();
//        this->firstName = obj.GetFirstName();
//        this->lastName = obj.GetLastName();
//        this->address = obj.GetAddress();
//        this->fullname = obj.GetFirstName() + " " + obj.GetLastName();
//    };
//
//     BOI(std::shared_ptr<OSTM> obj, int _version, int _unique_id): OSTM(_version, _unique_id)
//    {
//        this->accountNumber = obj->GetAccountNumber();
//        this->balance = obj->GetBalance();
//        this->firstName = obj->GetFirstName();
//        this->lastName = obj->GetLastName();
//        this->address = obj->GetAddress();
//        this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
//    };
//
//
//    BOI(const BOI& orig);
//    BOI operator=(const BOI& orig){};
//    virtual ~BOI();
//
//    virtual std::shared_ptr<BOI> _cast(std::shared_ptr<OSTM> _object);
//    virtual std::shared_ptr<BOI> getBaseCopy(OSTM& object);
//    virtual std::shared_ptr<BOI> getBaseCopy(std::shared_ptr<OSTM> object);
//    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
//    virtual void toString();
//    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;
//
//private:
//    std::string fullname;
//    std::string firstName;
//    std::string lastName;
//    int accountNumber;
//    double balance;
//    std::string address;
//
//};
//
//#endif /* BOI_H */
//
//
//    //virtual std::string get_class();
//
//   //
//    //virtual bool get(std::shared_ptr<OSTM> object);
//
```

7.17 CARLOW_W.cpp File Reference

`#include "CARLOW_W.h"`

Include dependency graph for CARLOW_W.cpp:



7.18 CARLOW_W.cpp

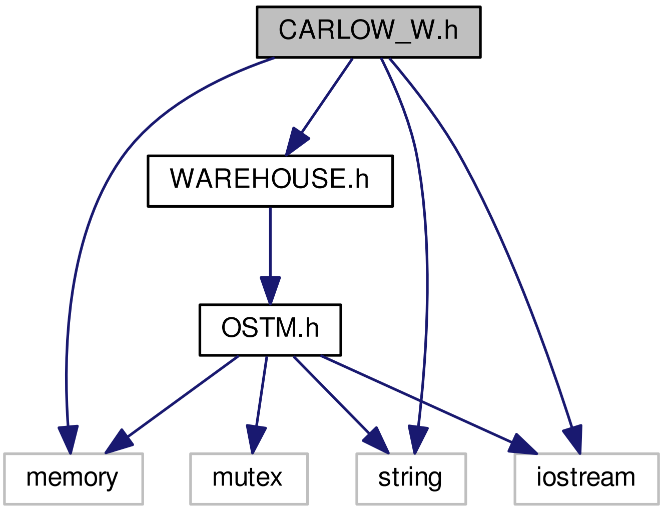
```

/*
 * File:   CARLOW_W.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */
#include "CARLOW_W.h"


//CARLOW_W::CARLOW_W() {
//}
```

```
CARLOW_W::~CARLOW_W() {
}

CARLOW_W::CARLOW_W(const CARLOW_W& orig) {
}
std::shared_ptr<OSTM> CARLOW_W::getBaseCopy(std::shared_ptr<OSTM> object)
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new CARLOW_W(objTO, object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
void CARLOW_W::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<CARLOW_W> objTO = std::dynamic_pointer_cast<CARLOW_W>(to);
    std::shared_ptr<CARLOW_W> objFROM = std::dynamic_pointer_cast<CARLOW_W>(from);
    objTO->_shop_address = objFROM->GetShop_address();
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
//std::shared_ptr<CARLOW_W> CARLOW_W::_cast(std::shared_ptr<OSTM> _object){
//
//    return std::static_pointer_cast<CARLOW_W>(_object);
//}
void CARLOW_W::toString()
{
   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;
}



void CARLOW_W::SetNumber_of_alcatel(int _number_of_alcatel) {
    this->_number_of_alcatel = _number_of_alcatel;
}

int CARLOW_W::GetNumber_of_alcatel(){
    return _number_of_alcatel;
}

void CARLOW_W::SetNumber_of_nokia(int _number_of_nokia) {
    this->_number_of_nokia = _number_of_nokia;
}

int CARLOW_W::GetNumber_of_nokia(){
    return _number_of_nokia;
}

void CARLOW_W::SetNumber_of_huawei(int _number_of_huawei) {
    this->_number_of_huawei = _number_of_huawei;
}

int CARLOW_W::GetNumber_of_huawei(){
    return _number_of_huawei;
}

void CARLOW_W::SetNumber_of_sony(int _number_of_sony) {
    this->_number_of_sony = _number_of_sony;
}

int CARLOW_W::GetNumber_of_sony(){
    return _number_of_sony;
}

void CARLOW_W::SetNumber_of_samsung(int _number_of_samsung) {
    this->_number_of_samsung = _number_of_samsung;
}

int CARLOW_W::GetNumber_of_samsung(){
    return _number_of_samsung;
}

void CARLOW_W::SetNumber_of_iphones(int _number_of_iphones) {
    this->_number_of_iphones = _number_of_iphones;
}

int CARLOW_W::GetNumber_of_iphones(){
    return _number_of_iphones;
}

void CARLOW_W::SetShop_name(std::string _shop_name) {
    this->_shop_name = _shop_name;
}

std::string CARLOW_W::GetShop_name(){
    return _shop_name;
}

void CARLOW_W::SetShop_address(std::string _shop_address) {
    this->_shop_address = _shop_address;
}

std::string CARLOW_W::GetShop_address(){
    return _shop_address;
}



```

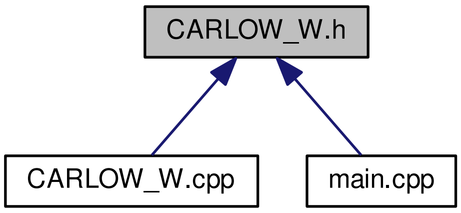
7.19 CARLOW_W.h File Reference

```
#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

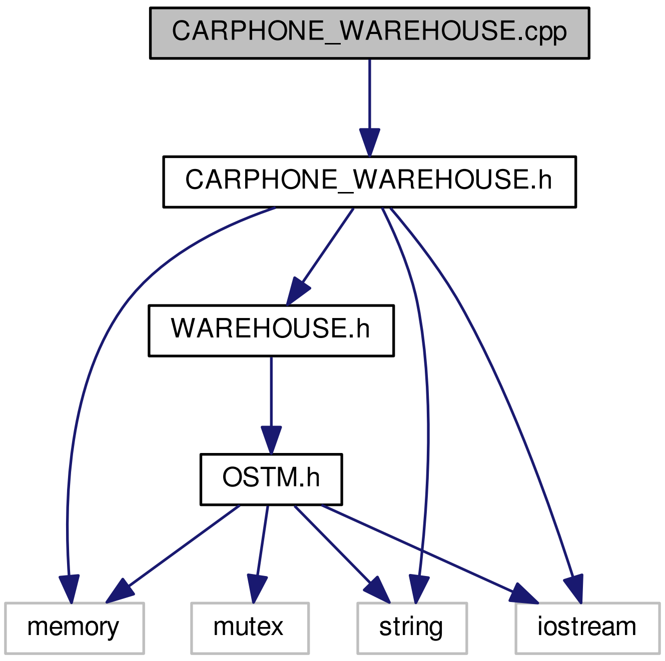
7.20 CARLOW_W.h

```

/*
 * File:   CARLOW_W.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef CARLOW_W_H
#define CARLOW_W_H
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>
class CARLOW_W :public WAREHOUSE {
public:
    CARLOW_W() : WAREHOUSE(){

        this->_shop_address = "Carlow potato street";
        this->_shop_name = "CARLOW C_WAREHOUSE";
        this->_number_of_iphones = 200;
        this->_number_of_samsung = 200;
        this->_number_of_sony = 200;
        this->_number_of_huawei = 200;
        this->_number_of_nokia = 200;
        this->_number_of_alcatel = 200;
    };
    CARLOW_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
huawei, int nokia, int alcatel): WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
    CARLOW_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
    CARLOW_W(const CARLOW_W& orig);
    CARLOW_W operator=(const CARLOW_W& orig){};
    virtual ~CARLOW_W();
    /*
     * Implement OSTM virtual methods
     */
   // virtual std::shared_ptr<CARLOW_W> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();
    /*
     * Implement Warehouse methods
     */
    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();


private:
    std::string _shop_address;
    std::string _shop_name;
    int _number_of_iphones;
    int _number_of_samsung;
    int _number_of_sony;
    int _number_of_huawei;
    int _number_of_nokia;
    int _number_of_alcatel;

};

#endif /* CARLOW_W_H */

```

7.21 CARPHONE_WAREHOUSE.cpp File Reference

```
#include "CARPHONE_WAREHOUSE.h"
```

Include dependency graph for CARPHONE_WAREHOUSE.cpp:



7.22 CARPHONE_WAREHOUSE.cpp

```

/*
 * File:   CARPHONE_WAREHOUSE.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */
#include "CARPHONE_WAREHOUSE.h"

CARPHONE_WAREHOUSE::CARPHONE_WAREHOUSE(const CARPHONE_WAREHOUSE& orig) {
}

CARPHONE_WAREHOUSE::~CARPHONE_WAREHOUSE() {
}
std::shared_ptr<OSTM> CARPHONE_WAREHOUSE::getBaseCopy(std::shared_ptr<OSTM> object)
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new CARPHONE_WAREHOUSE(objTO, object->Get_Version(), object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
void CARPHONE_WAREHOUSE::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from)
{

    std::shared_ptr<CARPHONE_WAREHOUSE> objTO = std::dynamic_pointer_cast<CARPHONE_WAREHOUSE>(to);
    std::shared_ptr<CARPHONE_WAREHOUSE> objFROM = std::dynamic_pointer_cast<CARPHONE_WAREHOUSE>(from);
    objTO->_shop_address = objFROM->GetShop_address();
    objTO->_shop_name = objFROM->GetShop_name();
```

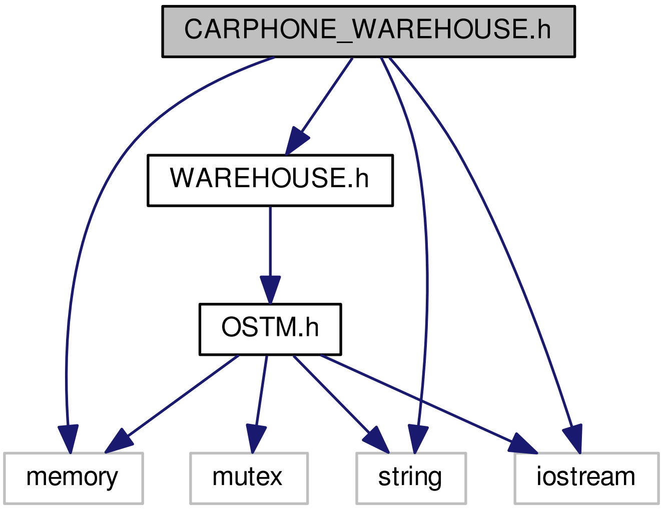
```cpp
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());

}
//std::shared_ptr<CARPHONE_WAREHOUSE> CARPHONE_WAREHOUSE::_cast(std::shared_ptr<OSTM> _object){
//
//    return std::static_pointer_cast<CARPHONE_WAREHOUSE>(_object);
//}
void CARPHONE_WAREHOUSE::toString()
{
    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;
}



void CARPHONE_WAREHOUSE::SetNumber_of_alcatel(int _number_of_alcatel) {
    this->_number_of_alcatel = _number_of_alcatel;
}

int CARPHONE_WAREHOUSE::GetNumber_of_alcatel(){
    return _number_of_alcatel;
}

void CARPHONE_WAREHOUSE::SetNumber_of_nokia(int _number_of_nokia) {
    this->_number_of_nokia = _number_of_nokia;
}

int CARPHONE_WAREHOUSE::GetNumber_of_nokia(){
    return _number_of_nokia;
}

void CARPHONE_WAREHOUSE::SetNumber_of_huawei(int _number_of_huawei) {
    this->_number_of_huawei = _number_of_huawei;
}

int CARPHONE_WAREHOUSE::GetNumber_of_huawei(){
    return _number_of_huawei;
}

void CARPHONE_WAREHOUSE::SetNumber_of_sony(int _number_of_sony) {
    this->_number_of_sony = _number_of_sony;
}

int CARPHONE_WAREHOUSE::GetNumber_of_sony(){
    return _number_of_sony;
}

void CARPHONE_WAREHOUSE::SetNumber_of_samsung(int _number_of_samsung) {
    this->_number_of_samsung = _number_of_samsung;
}

int CARPHONE_WAREHOUSE::GetNumber_of_samsung(){
    return _number_of_samsung;
}

void CARPHONE_WAREHOUSE::SetNumber_of_iphones(int _number_of_iphones) {
    this->_number_of_iphones = _number_of_iphones;
}

int CARPHONE_WAREHOUSE::GetNumber_of_iphones(){
    return _number_of_iphones;
}

void CARPHONE_WAREHOUSE::SetShop_name(std::string _shop_name) {
    this->_shop_name = _shop_name;
}

std::string CARPHONE_WAREHOUSE::GetShop_name(){
    return _shop_name;
```

```
}

void CARPHONE_WAREHOUSE::SetShop_address(std::string _shop_address) {
    this->_shop_address = _shop_address;
}

std::string CARPHONE_WAREHOUSE::GetShop_address(){
    return _shop_address;
}



```

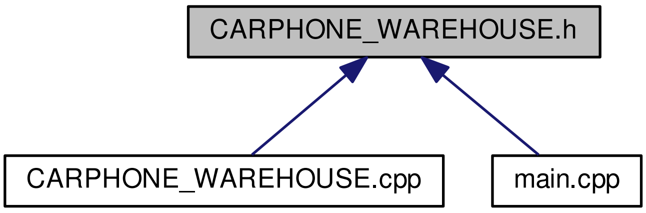
7.23 CARPHONE_WAREHOUSE.h File Reference

```
#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

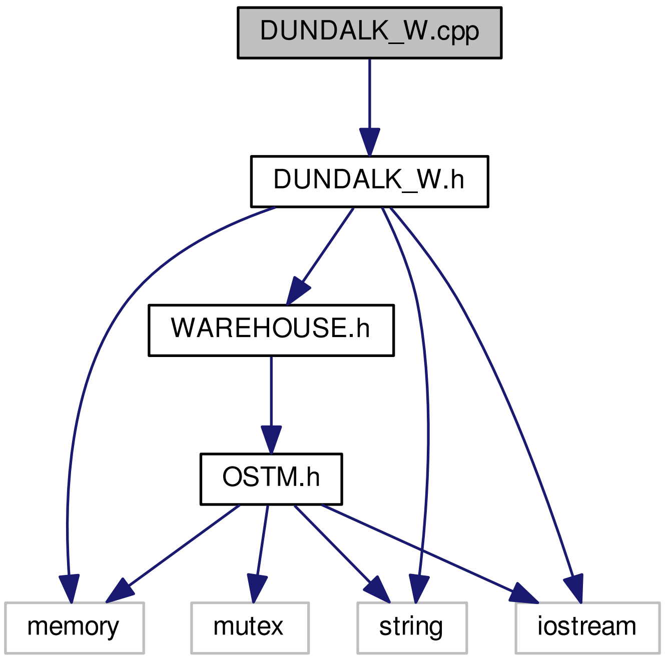
7.24 CARPHONE_WAREHOUSE.h

```cpp

/*
 * File:   CARPHONE_WAREHOUSE.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef CARPHONE_WAREHOUSE_H
#define CARPHONE_WAREHOUSE_H
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>
class CARPHONE_WAREHOUSE :public WAREHOUSE {
public:
    CARPHONE_WAREHOUSE(): WAREHOUSE(){

        this->_shop_address = "DUBLIN XII";
        this->_shop_name = "DISTRIBUTION CENTER";
        this->_number_of_iphones = 10000;
        this->_number_of_samsung = 10000;
        this->_number_of_sony = 10000;
        this->_number_of_huawei = 10000;
        this->_number_of_nokia = 10000;
        this->_number_of_alcatel = 10000;
    };
    CARPHONE_WAREHOUSE(std::string address, std::string shop_name, int iphone, int
samsung, int sony, int huawei, int nokia, int alcatel): WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
    CARPHONE_WAREHOUSE(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
    CARPHONE_WAREHOUSE(const CARPHONE_WAREHOUSE& orig);
    CARPHONE_WAREHOUSE operator=(const
CARPHONE_WAREHOUSE& orig){};
    virtual ~CARPHONE_WAREHOUSE();
    /*
     * Implement OSTM virtual methods
     */
    //virtual std::shared_ptr<CARPHONE_WAREHOUSE> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);



    virtual void toString();
    /*
     * Implement Warehouse methods
     */
    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();

private:
    std::string _shop_address;
    std::string _shop_name;
    int _number_of_iphones;
    int _number_of_samsung;
    int _number_of_sony;
    int _number_of_huawei;
    int _number_of_nokia;
    int _number_of_alcatel;

};

#endif /* CARPHONE_WAREHOUSE_H */

```

7.25 DUNDALK_W.cpp File Reference

```
#include "DUNDALK_W.h"
```

Include dependency graph for DUNDALK_W.cpp:



7.26 DUNDALK_W.cpp

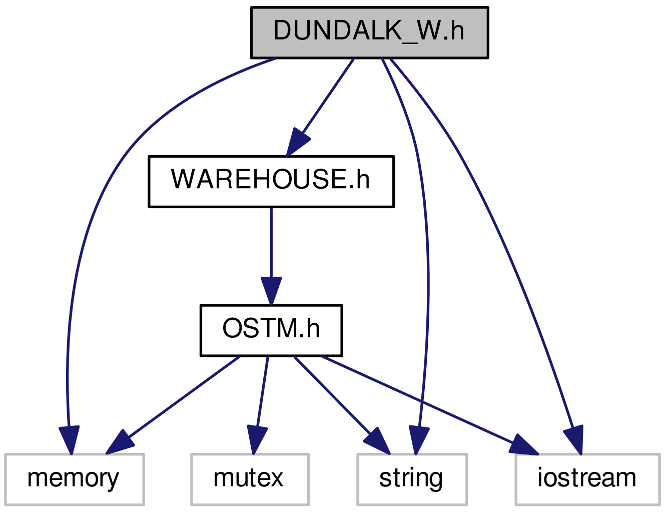
```

/*
 * File:   DUNDALK_W.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "DUNDALK_W.h"

DUNDALK_W::~DUNDALK_W() {
}

DUNDALK_W::DUNDALK_W(const DUNDALK_W& orig) {
}
std::shared_ptr<OSTM> DUNDALK_W::getBaseCopy(std::shared_ptr<OSTM> object)
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new DUNDALK_W(objTO, object->Get_Version(),object->
Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
void DUNDALK_W::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<DUNDALK_W> objTO = std::dynamic_pointer_cast<DUNDALK_W>(to);
    std::shared_ptr<DUNDALK_W> objFROM = std::dynamic_pointer_cast<DUNDALK_W>(from);
    objTO->_shop_address = objFROM->GetShop_address();
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
//std::shared_ptr<DUNDALK_W> DUNDALK_W::_cast(std::shared_ptr<OSTM> _object){
//
//    return std::static_pointer_cast<DUNDALK_W>(_object);
//}
void DUNDALK_W::toString()
{
   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;
}



void DUNDALK_W::SetNumber_of_alcatel(int _number_of_alcatel) {
    this->_number_of_alcatel = _number_of_alcatel;
}

int DUNDALK_W::GetNumber_of_alcatel(){
    return _number_of_alcatel;
}

void DUNDALK_W::SetNumber_of_nokia(int _number_of_nokia) {
    this->_number_of_nokia = _number_of_nokia;
}

int DUNDALK_W::GetNumber_of_nokia(){
    return _number_of_nokia;
}

void DUNDALK_W::SetNumber_of_huawei(int _number_of_huawei) {
    this->_number_of_huawei = _number_of_huawei;
}

int DUNDALK_W::GetNumber_of_huawei(){
    return _number_of_huawei;
}
```

```

void DUNDALK_W::SetNumber_of_sony(int _number_of_sony) {
    this->_number_of_sony = _number_of_sony;
}

int DUNDALK_W::GetNumber_of_sony(){
    return _number_of_sony;
}

void DUNDALK_W::SetNumber_of_samsung(int _number_of_samsung) {
    this->_number_of_samsung = _number_of_samsung;
}

int DUNDALK_W::GetNumber_of_samsung(){
    return _number_of_samsung;
}

void DUNDALK_W::SetNumber_of_iphones(int _number_of_iphones) {
    this->_number_of_iphones = _number_of_iphones;
}

int DUNDALK_W::GetNumber_of_iphones(){
    return _number_of_iphones;
}

void DUNDALK_W::SetShop_name(std::string _shop_name) {
    this->_shop_name = _shop_name;
}

std::string DUNDALK_W::GetShop_name(){
    return _shop_name;
}

void DUNDALK_W::SetShop_address(std::string _shop_address) {
    this->_shop_address = _shop_address;
}

std::string DUNDALK_W::GetShop_address(){
    return _shop_address;
}



```

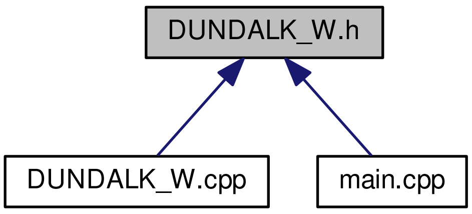
7.27 DUNDALK_W.h File Reference

```
#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

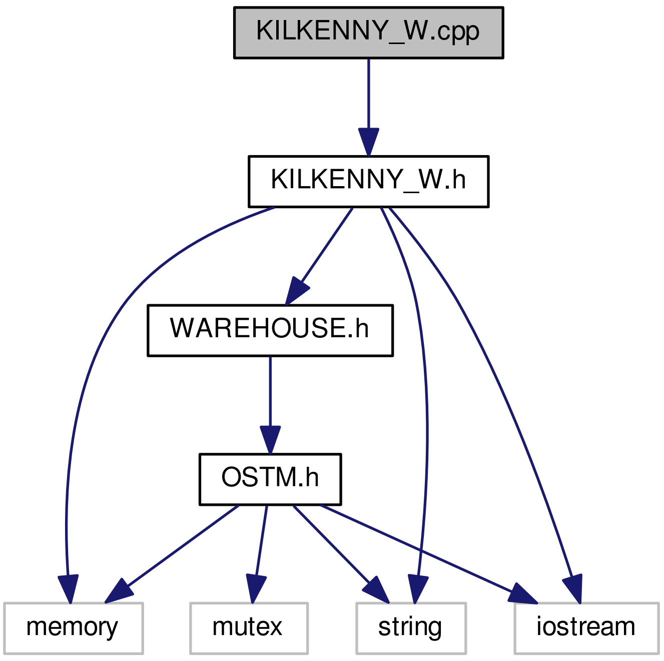
7.28 DUNDALK_W.h

```

/*
 * File:   DUNDALK_W.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef DUNDALK_W_H
#define DUNDALK_W_H
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>
class DUNDALK_W :public WAREHOUSE {
public:
    DUNDALK_W() : WAREHOUSE(){

        this->_shop_address = "Dundalk Busy Street";
        this->_shop_name = "DUNDALK D_WAREHOUSE";
        this->_number_of_iphones = 200;
        this->_number_of_samsung = 200;
        this->_number_of_sony = 200;
        this->_number_of_huawei = 200;
        this->_number_of_nokia = 200;
        this->_number_of_alcatel = 200;
    };
    DUNDALK_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
huawei, int nokia, int alcatel): WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
    DUNDALK_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
    DUNDALK_W(const DUNDALK_W& orig);
    DUNDALK_W operator=(const DUNDALK_W& orig){};
    virtual ~DUNDALK_W();
    /*
     * Implement OSTM virtual methods
     */
    //virtual std::shared_ptr<DUNDALK_W> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();
    /*
     * Implement Warehouse methods
     */
    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();


private:
    std::string _shop_address;
    std::string _shop_name;
    int _number_of_iphones;
    int _number_of_samsung;
    int _number_of_sony;
    int _number_of_huawei;
    int _number_of_nokia;
    int _number_of_alcatel;

};

#endif /* DUNDALK_W_H */

```

7.29 KILKENNY_W.cpp File Reference

`#include "KILKENNY_W.h"`

Include dependency graph for KILKENNY_W.cpp:



7.30 KILKENNY_W.cpp

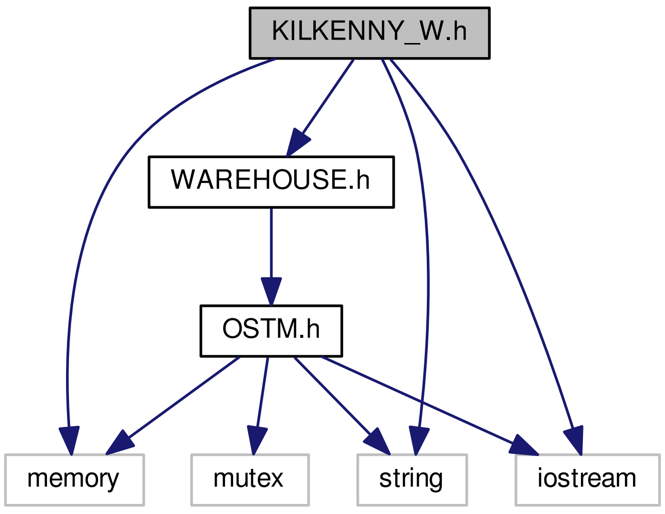
```

/*
 * File:   KILKENNY_W.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "KILKENNY_W.h"

KILKENNY_W::~KILKENNY_W() {
}

KILKENNY_W::KILKENNY_W(const KILKENNY_W& orig) {
}
std::shared_ptr<OSTM> KILKENNY_W::getBaseCopy(std::shared_ptr<OSTM> object)
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new KILKENNY_W(objTO, object->Get_Version(),object->
Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);
    return ostm_obj;
}
void KILKENNY_W::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<KILKENNY_W> objTO = std::dynamic_pointer_cast<KILKENNY_W>(to);
    std::shared_ptr<KILKENNY_W> objFROM = std::dynamic_pointer_cast<KILKENNY_W>(from);
    objTO->_shop_address = objFROM->GetShop_address();
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
//std::shared_ptr<KILKENNY_W> KILKENNY_W::_cast(std::shared_ptr<OSTM> _object){
//
//    return static_cast<std::shared_ptr<KILKENNY_W>>(_object);
//}
void KILKENNY_W::toString()
{
   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;
}



void KILKENNY_W::SetNumber_of_alcatel(int _number_of_alcatel) {
    this->_number_of_alcatel = _number_of_alcatel;
}

int KILKENNY_W::GetNumber_of_alcatel(){
    return _number_of_alcatel;
}

void KILKENNY_W::SetNumber_of_nokia(int _number_of_nokia) {
    this->_number_of_nokia = _number_of_nokia;
}

int KILKENNY_W::GetNumber_of_nokia(){
    return _number_of_nokia;
}

void KILKENNY_W::SetNumber_of_huawei(int _number_of_huawei) {
    this->_number_of_huawei = _number_of_huawei;
}

int KILKENNY_W::GetNumber_of_huawei(){
    return _number_of_huawei;
}

void KILKENNY_W::SetNumber_of_sony(int _number_of_sony) {
    this->_number_of_sony = _number_of_sony;
}

int KILKENNY_W::GetNumber_of_sony(){
    return _number_of_sony;
}

void KILKENNY_W::SetNumber_of_samsung(int _number_of_samsung) {
    this->_number_of_samsung = _number_of_samsung;
}

int KILKENNY_W::GetNumber_of_samsung(){
    return _number_of_samsung;
}

void KILKENNY_W::SetNumber_of_iphones(int _number_of_iphones) {
    this->_number_of_iphones = _number_of_iphones;
}

```

```
int KILKENNY_W::GetNumber_of_iphones(){
    return _number_of_iphones;
}

void KILKENNY_W::SetShop_name(std::string _shop_name) {
    this->_shop_name = _shop_name;
}

std::string KILKENNY_W::GetShop_name(){
    return _shop_name;
}

void KILKENNY_W::SetShop_address(std::string _shop_address) {
    this->_shop_address = _shop_address;
}

std::string KILKENNY_W::GetShop_address(){
    return _shop_address;
}

```

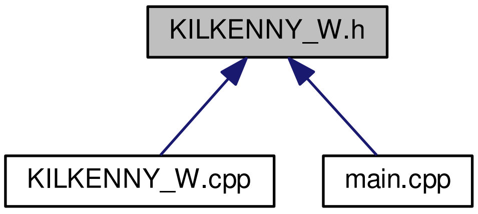
7.31 KILKENNY_W.h File Reference

```
#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

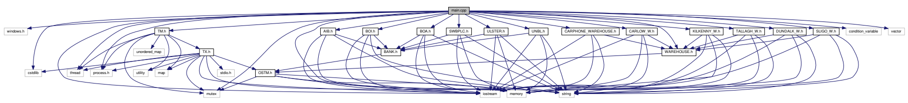
```

/*
 * File:   KILKENNY_W.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef KILKENNY_W_H
#define KILKENNY_W_H
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>
class KILKENNY_W : public WAREHOUSE {
public:
    KILKENNY_W(): WAREHOUSE(){

        this->_shop_address = "Kilkenny High Street";
        this->_shop_name = "KILKENNY K_WAREHOUSE";
        this->_number_of_iphones = 200;
        this->_number_of_samsung = 200;
        this->_number_of_sony = 200;
        this->_number_of_huawei = 200;
        this->_number_of_nokia = 200;
        this->_number_of_alcatel = 200;
    };
    KILKENNY_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
 huawei, int nokia, int alcatel): WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
    KILKENNY_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
    KILKENNY_W(const KILKENNY_W& orig);
    KILKENNY_W operator=(const KILKENNY_W& orig){};
    virtual ~KILKENNY_W();
    /*
     * Implement OSTM virtual methods
     */
    //virtual std::shared_ptr<KILKENNY_W> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();
    /*
     * Implement Warehouse methods
     */
    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();


private:
    std::string _shop_address;
    std::string _shop_name;
    int _number_of_iphones;
    int _number_of_samsung;
    int _number_of_sony;
    int _number_of_huawei;
    int _number_of_nokia;
    int _number_of_alcatel;

};

#endif /* KILKENNY_W_H */

```

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

std::shared_ptr<TX>	tx, Transaction Object
std::shared_ptr<BANK>	type, *FROM* & *FROM_TWO* & *TO*
std::shared_ptr<OSTM>	type, *FROM_OSTM_ONE* & *FROM_OSTM_TWO* & *TO_OSTM*

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

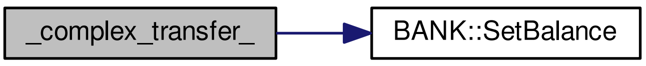
Commit changes

Definition at line 296 of file main.cpp.

References BANK::SetBalance().

```
                                                                                                                    {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_from_);
    tx->_register(_from_two_);
    std::shared_ptr<OSTM> _FROM_OSTM_ONE_, _FROM_OSTM_TWO_, _TO_OSTM_;
    std::shared_ptr<BANK> _FROM_, _FROM_TWO_, _TO_;

    bool done = false;
    try {
        while (!done) {
            // for (int i = 0; i < vector_number; ++i) {
            for (auto&& obj : _customer_vec) {
                // auto&& obj = _customer_vec.at(i);
                tx->_register(obj);
                _FROM_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
                _FROM_TWO_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_two_));
                _TO_ = std::dynamic_pointer_cast<BANK> (tx->load(obj));
                _FROM_->SetBalance(_FROM_->GetBalance() - _amount);
                _FROM_TWO_->SetBalance(_FROM_TWO_->GetBalance() - _amount);
                _TO_->SetBalance(_TO_->GetBalance() + (_amount * 2));
                _FROM_OSTM_ONE_ = std::dynamic_pointer_cast<OSTM> (_FROM_);
                _FROM_OSTM_TWO_ = std::dynamic_pointer_cast<OSTM> (_FROM_TWO_);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_);
                tx->store(_FROM_OSTM_ONE_);
                tx->store(_FROM_OSTM_TWO_);
                tx->store(_TO_OSTM_);
            }
            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}
```

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

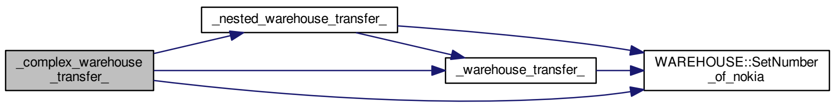
Commit changes

Definition at line 520 of file main.cpp.

References _nested_warehouse_transfer_(), _warehouse_transfer_(), and WAREHOUSE::SetNumber_of_nokia().

```
 {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_to_two);
    tx->_register(_to_three);
    tx->_register(_from_);
    std::shared_ptr<WAREHOUSE> _TO_SHOP_, _TO_SHOP_TWO, _TO_SHOP_VEC, _FROM_DIST_;
    std::shared_ptr<OSTM> _TO_OSTM_, _TO_OSTM_TWO, _TO_OSTM_VEC, _FROM_OSTM_;

    bool done = false;
    try {
        while (!done) {

            // for (int i = 0; i < vector_number; ++i) {
            for (auto&& obj : _warehouse_vec) {
                //auto&& obj = _warehouse_vec.at(i);
                tx->_register(obj);
                _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
                _TO_SHOP_TWO = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_two));
                _TO_SHOP_VEC = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(obj));
                _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));

                _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
                _TO_SHOP_TWO->SetNumber_of_nokia(_TO_SHOP_TWO->GetNumber_of_nokia() + _amount);
                _TO_SHOP_VEC->SetNumber_of_nokia(_TO_SHOP_VEC->GetNumber_of_nokia() + _amount);
                _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - (_amount * 3));

                _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
                _TO_SHOP_TWO->SetNumber_of_samsung(_TO_SHOP_TWO->GetNumber_of_samsung() + _amount);
                _TO_SHOP_VEC->SetNumber_of_samsung(_TO_SHOP_VEC->GetNumber_of_samsung() + _amount);
                _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - (_amount * 3));

                _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
                _TO_SHOP_TWO->SetNumber_of_iphones(_TO_SHOP_TWO->GetNumber_of_iphones() + _amount);
                _TO_SHOP_VEC->SetNumber_of_iphones(_TO_SHOP_VEC->GetNumber_of_iphones() + _amount);
                _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - (_amount * 3));

                _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
                _TO_SHOP_TWO->SetNumber_of_sony(_TO_SHOP_TWO->GetNumber_of_sony() + _amount);
                _TO_SHOP_VEC->SetNumber_of_sony(_TO_SHOP_VEC->GetNumber_of_sony() + _amount);
                _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - (_amount * 3));

                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
                _TO_OSTM_TWO = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_TWO);
                _TO_OSTM_VEC = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_VEC);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
                tx->store(_TO_OSTM_);
                tx->store(_TO_SHOP_TWO);
                tx->store(_TO_SHOP_VEC);
                tx->store(_FROM_OSTM_);



            }
            std::shared_ptr<TX> txTwo = _tm._get_tx();
            bool nestedDone = false;
            while (!nestedDone) {
                _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_to_two));
                _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_from_));
                _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
                _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);

                _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
                _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);

                _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
                _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);

                _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
                _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
                txTwo->store(_TO_OSTM_);
                txTwo->store(_FROM_OSTM_);

                /*
                 * NESTED TRANSACTION TEST _to_three
                 */
                _warehouse_transfer_(_to_three, _from_, _tm, _amount);
                _nested_warehouse_transfer_(_to_, _to_two, _to_three, _from_,
      _tm, _amount);

                nestedDone = tx->commit();
            }

            done = tx->commit();

        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}
```

Here is the call graph for this function:


_nested_warehouse_transfer_
_complex_warehouse_transfer_
_warehouse_transfer_
WAREHOUSE::SetNumber_of_nokia

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

std::shared_ptr<TX>	tx, Transaction Object
std::shared_ptr<WAREHOUSE>	type, *TO_SHOP* & *FROM_DIST*
std::shared_ptr<OSTM>	type, *TO_OSTM* & *FROM_OSTM*

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

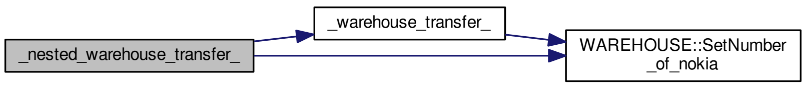
Definition at line 421 of file main.cpp.

References _warehouse_transfer_(), and WAREHOUSE::SetNumber_of_nokia().

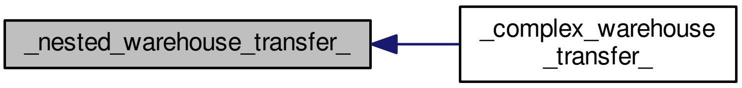
Referenced by _complex_warehouse_transfer_().

```
                                                                {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_to_two);
    tx->_register(_to_three);
    tx->_register(_from_);
    std::shared_ptr<WAREHOUSE> _TO_SHOP_, _FROM_DIST_;
    std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;

    bool done = false;
    try {
        while (!done) {
            _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
            _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));
            _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
            _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);

            _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
            _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);

            _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
            _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);

            _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
            _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
            _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
            _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
            tx->store(_TO_OSTM_);
            tx->store(_FROM_OSTM_);

            std::shared_ptr<TX> txTwo = _tm._get_tx();
            bool nestedDone = false;
            while (!nestedDone) {
                _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_to_two));
                _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_from_));
                _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
                _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);

                _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
                _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);

                _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
                _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);

                _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
                _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
                txTwo->store(_TO_OSTM_);
                txTwo->store(_FROM_OSTM_);

                /*
                 * NESTED TRANSACTION TEST _to_three
                 */
                _warehouse_transfer_(_to_three, _from_, _tm, _amount);


                nestedDone = tx->commit();
            }
            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}
```

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

std::shared_ptr<TX>	tx, Transaction Object
std::shared_ptr<BANK>	type, *TO_BANK* & *FROM_BANK*
std::shared_ptr<OSTM>	type, *TO_OSTM* & *FROM_OSTM*

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

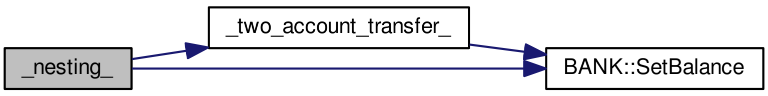
Definition at line 208 of file main.cpp.

References _two_account_transfer_(), and BANK::SetBalance().

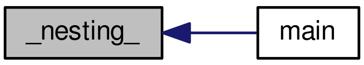
Referenced by main().

```
                                                                                             {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_from_);
    std::shared_ptr<BANK> _TO_BANK_, _FROM_BANK_;
    std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;


    bool done = false;
    try {
        while (!done) {
            _TO_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
            _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
            _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
            _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
            _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
            _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
            tx->store(_TO_OSTM_);
            tx->store(_FROM_OSTM_);

            std::shared_ptr<TX> txTwo = _tm._get_tx();

            bool nestedDone = false;
            while (!nestedDone) {
                _TO_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_to_));
                _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_from_));
                _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
                _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
                txTwo->store(_TO_OSTM_);
                txTwo->store(_FROM_OSTM_);
                _two_account_transfer_(_to_, _from_, _tm, _amount);

                nestedDone = txTwo->commit();
            }

            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}
```

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

std::shared_ptr<TX>	tx, Transaction Object
std::shared_ptr<BANK>	type, *TO* & *FROM_ONE* & *FROM_TWO* & *FROM_THREE* & *FROM_FOUR* & *FROM_FIVE*
std::shared_ptr<OSTM>	type, _TO_OSTM & _FROM_ONE_OSTM & _FROM_TWO_OSTM & _FROM_THREE_OSTM & _FROM_FOUR_OSTM & _FROM_FIVE_OSTM

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

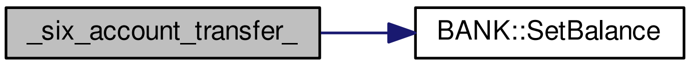
Commit changes

Definition at line 53 of file main.cpp.

References BANK::SetBalance().

```
                                  {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_from_one_);
    tx->_register(_from_two_);
    tx->_register(_from_three_);
    tx->_register(_from_four_);
    tx->_register(_from_five_);

    std::shared_ptr<OSTM> _TO_OSTM, _FROM_ONE_OSTM, _FROM_TWO_OSTM, _FROM_THREE_OSTM, _FROM_FOUR_OSTM,
 _FROM_FIVE_OSTM;
    std::shared_ptr<BANK> _TO_, _FROM_ONE_, _FROM_TWO_, _FROM_THREE_, _FROM_FOUR_, _FROM_FIVE_;
    try {
        bool done = false;
        while (!done) {
            _TO_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
            _FROM_ONE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_one_));
            _FROM_TWO_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_two_));
            _FROM_THREE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_three_));
            _FROM_FOUR_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_four_));
            _FROM_FIVE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_five_));
            _TO_->SetBalance(_TO_->GetBalance() + (_amount * 5));
            _FROM_ONE_->SetBalance(_FROM_ONE_->GetBalance() - _amount);
            _FROM_TWO_->SetBalance(_FROM_TWO_->GetBalance() - _amount);
            _FROM_THREE_->SetBalance(_FROM_THREE_->GetBalance() - _amount);
            _FROM_FOUR_->SetBalance(_FROM_FOUR_->GetBalance() - _amount);
            _FROM_FIVE_->SetBalance(_FROM_FIVE_->GetBalance() - _amount);
            _TO_OSTM = std::dynamic_pointer_cast<OSTM> (_TO_);
            _FROM_ONE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_ONE_);
            _FROM_TWO_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_TWO_);
            _FROM_THREE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_THREE_);
            _FROM_FOUR_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_FOUR_);
            _FROM_FIVE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_FIVE_);
            tx->store(_TO_OSTM);
            tx->store(_FROM_ONE_OSTM);
            tx->store(_FROM_TWO_OSTM);
            tx->store(_FROM_THREE_OSTM);
            tx->store(_FROM_FOUR_OSTM);
            tx->store(_FROM_FIVE_OSTM);
            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}
```

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

std::shared_ptr<TX>	tx, Transaction Object
std::shared_ptr<BANK>	type, *TO_BANK* & *FROM_BANK*
std::shared_ptr<OSTM>	type, *TO_OSTM* & *FROM_OSTM*

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

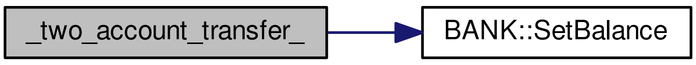
Definition at line 125 of file main.cpp.

References BANK::SetBalance().

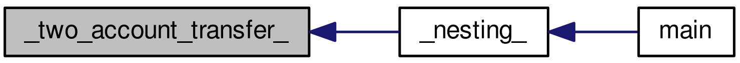
Referenced by _nesting_().

```
{
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_from_);
    std::shared_ptr<BANK> _TO_BANK_, _FROM_BANK_;
    std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;

    bool done = false;
    try {
        while (!done) {
            _TO_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
            _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
            _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
            _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
            _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
            _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
            tx->store(_TO_OSTM_);
            tx->store(_FROM_OSTM_);

            std::shared_ptr<TX> txTwo = _tm._get_tx();

            bool nestedDone = false;
            while (!nestedDone) {
                _TO_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_to_));
                _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_from_));
                _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
                _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
                txTwo->store(_TO_OSTM_);
                txTwo->store(_FROM_OSTM_);
                nestedDone = txTwo->commit();
            }
            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}
```

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

std::shared_ptr<TX>	tx, Transaction Object
std::shared_ptr<WAREHOUSE>	type, *TO_SHOP* & *FROM_DIST*
std::shared_ptr<OSTM>	type, *TO_OSTM* & *FROM_OSTM*

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

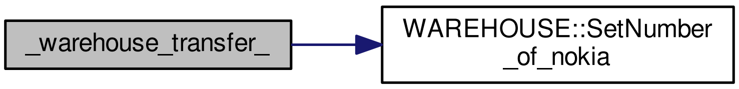
Definition at line 360 of file main.cpp.

References WAREHOUSE::SetNumber_of_nokia().

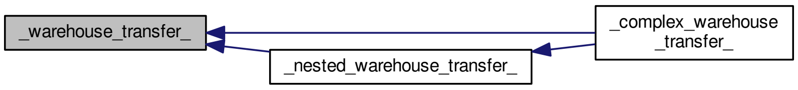
Referenced by _complex_warehouse_transfer_(), and _nested_warehouse_transfer_().

```
00360                                                                                                                   {
00361     std::shared_ptr<TX> tx = _tm._get_tx();
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 }
```

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

transferAmount	in the transaction, control the value in the transaction between objetcs
threadArraySize	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 threads^n -> 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();

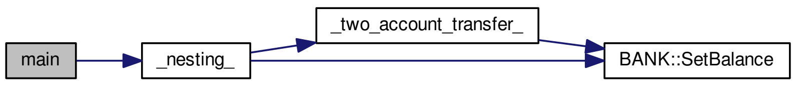
Definition at line 651 of file main.cpp.

References _nesting_().

```
00651                 {
00656     TM& tm = TM::Instance();
00657
00664     std::vector<std::shared_ptr < OSTM>>_customer_vec; //(vector_number);
00665     std::vector<std::shared_ptr < OSTM>>_warehouse_vec; //(vector_number);
00666
00676     std::shared_ptr<OSTM> aib_ptr(new AIB(100, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny")
      );
00677     std::shared_ptr<OSTM> boi_ptr(new BOI(200, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny")
      );
00678     std::shared_ptr<OSTM> boa_ptr(new BOA(300, 500, "Joe", "Blog", "High street, Kilkenny, Co.Kilkenny")
      );
00679     std::shared_ptr<OSTM> swplc_ptr(new SWBPLC(400, 500, "Joe", "Blog", "High street, Kilkenny,
       Co.Kilkenny"));
00680     std::shared_ptr<OSTM> ulster_ptr(new ULSTER(500, 500, "Joe", "Blog", "High street, Kilkenny,
       Co.Kilkenny"));
00681     std::shared_ptr<OSTM> unbl_ptr(new UNBL(600, 500, "Joe", "Blog", "High street, Kilkenny,
       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));
```

```
        } else if (i % 4 == 0) {
            std::shared_ptr<OSTM> sharedptr(new BOA(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        } else if (i % 3 == 0) {
            std::shared_ptr<OSTM> sharedptr(new SWBPLC(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        } else if (i % 2 == 0) {
            std::shared_ptr<OSTM> sharedptr(new ULSTER(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        } else if (i % 1 == 0) {
            std::shared_ptr<OSTM> sharedptr(new UNBL(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        }
    }

    //    w_dist->toString();
    //    c_shop->toString();
    //    k_shop->toString();
    //    t_shop->toString();
    //    d_shop->toString();
    //    s_shop->toString();

    /*
     * TEST 1 : object requirements
     */
    aib_ptr->toString();
    boi_ptr->toString();
    boa_ptr->toString();
    swplc_ptr->toString();
    ulster_ptr->toString();
    unbl_ptr->toString();

    /*
     * TEST 2 : object requirements
     */
    //    aib_ptr->toString();
    //    boi_ptr->toString();
    //    boa_ptr->toString();
    //    swplc_ptr->toString();
    //    ulster_ptr->toString();
    //    unbl_ptr->toString();
    //    for(int i=0; i<vector_number; ++i){
    //        _customer_vec[i]->toString();
    //    }

    /*
     * TEST 3 : object requirements
     */
    //    w_dist->toString();
    //    c_shop->toString();
    //    k_shop->toString();
    //    t_shop->toString();

    /*
     * TEST 4 : objects requirements
     */
    //        w_dist->toString();
    //        c_shop->toString();
    //        k_shop->toString();
    //        t_shop->toString();
    //        d_shop->toString();
    //        s_shop->toString();


    /*
     * TEST 5 : objects requirements
     */
    //        w_dist->toString();
    //        c_shop->toString();
    //        k_shop->toString();
    //        t_shop->toString();
    //        d_shop->toString();
    //        s_shop->toString();

    //        for(auto&& elem: _warehouse_vec){
    //            elem->toString(); // virtual dispatch
    //
    //        }



    int transferAmount = 1;
    int threadArraySize = 300;

    std::thread thArray[300];

    for (int i = 0; i < threadArraySize; ++i) {

        //thArray[i] = std::thread(_nesting_, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
      if (i % 3 == 0)
          thArray[i] = std::thread(_nesting_, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
      else if (i % 2 == 0)
          thArray[i] = std::thread(_nesting_, boa_ptr, swplc_ptr, std::ref(tm), transferAmount);
      else if (i % 1 == 0)
          thArray[i] = std::thread(_nesting_, ulster_ptr, unbl_ptr, std::ref(tm), transferAmount);

        //    if (i % 3 == 0)
        //        thArray[i] = std::thread(_two_account_transfer_, aib_ptr, boi_ptr, std::ref(tm),
transferAmount);
        //    else if (i % 2 == 0)
        //        thArray[i] = std::thread(_six_account_transfer_, boi_ptr, boa_ptr, swplc_ptr, ulster_ptr,
aib_ptr, unbl_ptr, std::ref(tm), transferAmount);
        //    else if (i % 1 == 0)
        //        thArray[i] = std::thread(_complex_transfer_, aib_ptr, boi_ptr, std::ref(_customer_vec),
std::ref(tm), transferAmount);


        //        if (i % 3 == 0)
        //            thArray[i] = std::thread(_warehouse_transfer_, c_shop, w_dist, std::ref(tm),
transferAmount);
        //        else if (i % 2 == 0)
        //            thArray[i] = std::thread(_warehouse_transfer_, k_shop, w_dist, std::ref(tm),
transferAmount);
        //        else if (i % 1 == 0)
        //            thArray[i] = std::thread(_warehouse_transfer_, t_shop, w_dist, std::ref(tm),
transferAmount);

        //        if (i % 3 == 0)
        //            thArray[i] = std::thread(_nested_warehouse_transfer_, c_shop, d_shop, k_shop, w_dist,
std::ref(tm), transferAmount);
        //        else if (i % 2 == 0)
        //            thArray[i] = std::thread(_nested_warehouse_transfer_, k_shop, s_shop, t_shop, w_dist,
std::ref(tm), transferAmount);
        //        else if (i % 1 == 0)
        //            thArray[i] = std::thread(_nested_warehouse_transfer_, t_shop, c_shop, s_shop, w_dist,
std::ref(tm), transferAmount);

        //        if (i % 3 == 0)
        //            thArray[i] = std::thread(_warehouse_transfer_, c_shop, w_dist, std::ref(tm),
transferAmount);
        //        else if (i % 2 == 0)
        //            thArray[i] = std::thread(_nested_warehouse_transfer_, k_shop, s_shop, t_shop, w_dist,
std::ref(tm), transferAmount);
        //        else if (i % 1 == 0)
        //            thArray[i] = std::thread(_complex_warehouse_transfer_, d_shop, s_shop, c_shop,
std::ref(_warehouse_vec), w_dist, std::ref(tm), transferAmount);


    }
    /*
     * Join threads^n -> threadArraySize<br>
     * thArray[i].join();
     */
    for (int i = 0; i < threadArraySize; ++i) {
        thArray[i].join();
    }


    std::cout << "\nMain process print " << std::endl;
    /*
     * TEST 1 : object requirements
     */
    aib_ptr->toString();
    boi_ptr->toString();
    boa_ptr->toString();
    swplc_ptr->toString();
    ulster_ptr->toString();
    unbl_ptr->toString();

    /*
     * TEST 2 : object requirements
     */
    //    aib_ptr->toString();
    //    boi_ptr->toString();
    //    boa_ptr->toString();
    //    swplc_ptr->toString();
    //    ulster_ptr->toString();
    //    unbl_ptr->toString();
    //    for(int i=0; i<vector_number; ++i){
    //        _customer_vec[i]->toString();
    //    }

    /*
     * TEST 3 : object requirements
     */
    //            w_dist->toString();
    //            c_shop->toString();
    //            k_shop->toString();
    //            t_shop->toString();

    /*
     * TEST 4 : objects requirements
     */
    //        w_dist->toString();
    //        c_shop->toString();
    //        k_shop->toString();
    //        t_shop->toString();
    //        d_shop->toString();
    //        s_shop->toString();

    /*
     * TEST 5 : objects requirements
     */
    //        w_dist->toString();
    //        c_shop->toString();
    //        k_shop->toString();
    //        t_shop->toString();
    //        d_shop->toString();
    //        s_shop->toString();

    //        for(auto&& elem: _warehouse_vec){
    //            elem->toString(); // virtual dispatch
    //
    //        }

    /* TEST 5 FINISH */


    std::cout << "\nMAIN PROCESS EXIT !!!! " << std::endl;
    std::shared_ptr<TX> tx = tm._get_tx();

    std::cout << "Rollback counter is : " << tx->getTest_counter() << std::endl;
    //    std::cout << "[vector_number]" << std::endl;
    //    for (int i = 0; i < vector_number; ++i) {
    //        //_customer_vec[i]->toString();
    //        auto&& os = _customer_vec.at(i);
    //        os->toString();
    //    }
    //    std::cout << "[_warehouse_vec]" << std::endl;
    //    for(auto&& elem: _warehouse_vec){
    //        elem->toString(); // virtual dispatch
    //
    //    }
    //_customer_vec[10]->toString();

    tm._TX_EXIT();
    std::cout << "\nPRINT ALL FROM TM !!!! SHOULD BE EMPTY AFTER _TX_EXIT() !!" << std::endl;
    tm.print_all();
    int t = 0;
    std::cin >> t;
    return 0;
}
```

Here is the call graph for this function:



7.34 main.cpp

```
/*
```

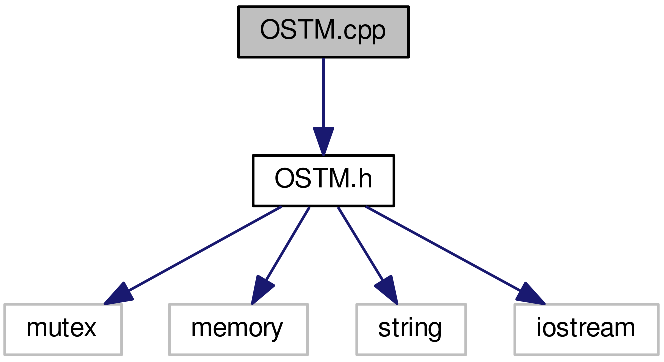
```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   main.cpp
 * Author: zoltan
 *
 * Created on November 27, 2017, 9:26 PM
 */
#include <windows.h>
#include <cstdlib>
#include <iostream>
#include <thread>
#include <process.h>

//#include <unistd.h>//used for pid_t

//STM library requirement
#include "TM.h"
#include "AIB.h"    //Bank Account
#include "BOI.h"    //Bank Account
#include "BOA.h"    //Bank Account
#include "SWBPLC.h" //Bank Account
#include "ULSTER.h" //Bank Account
#include "UNBL.h"   //Bank Account
#include "WAREHOUSE.h"   //WAREHOUSE
#include "CARPHONE_WAREHOUSE.h" //WAREHOUSE
#include "CARLOW_W.h"    //WAREHOUSE
#include "KILKENNY_W.h"  //WAREHOUSE
#include "TALLAGH_W.h"   //WAREHOUSE
#include "DUNDALK_W.h"   //WAREHOUSE
#include "SLIGO_W.h"     //WAREHOUSE
#include <mutex>
#include <memory>
#include <condition_variable>
#include <vector>


static int vector_number = 600;

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) {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_from_one_);
    tx->_register(_from_two_);
    tx->_register(_from_three_);
    tx->_register(_from_four_);
    tx->_register(_from_five_);

    std::shared_ptr<OSTM> _TO_OSTM, _FROM_ONE_OSTM, _FROM_TWO_OSTM, _FROM_THREE_OSTM, _FROM_FOUR_OSTM, _FROM_FIVE_OSTM;
    std::shared_ptr<BANK> _TO_, _FROM_ONE_, _FROM_TWO_, _FROM_THREE_, _FROM_FOUR_, _FROM_FIVE_;
    try {
        bool done = false;
        while (!done) {
            _TO_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
            _FROM_ONE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_one_));
            _FROM_TWO_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_two_));
            _FROM_THREE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_three_));
            _FROM_FOUR_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_four_));
            _FROM_FIVE_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_five_));
            _TO_->SetBalance(_TO_->GetBalance() + (_amount * 5));
            _FROM_ONE_->SetBalance(_FROM_ONE_->GetBalance() - _amount);
            _FROM_TWO_->SetBalance(_FROM_TWO_->GetBalance() - _amount);
            _FROM_THREE_->SetBalance(_FROM_THREE_->GetBalance() - _amount);
            _FROM_FOUR_->SetBalance(_FROM_FOUR_->GetBalance() - _amount);
            _FROM_FIVE_->SetBalance(_FROM_FIVE_->GetBalance() - _amount);
            _TO_OSTM = std::dynamic_pointer_cast<OSTM> (_TO_);
            _FROM_ONE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_ONE_);
            _FROM_TWO_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_TWO_);
            _FROM_THREE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_THREE_);
            _FROM_FOUR_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_FOUR_);
            _FROM_FIVE_OSTM = std::dynamic_pointer_cast<OSTM> (_FROM_FIVE_);
            tx->store(_TO_OSTM);
            tx->store(_FROM_ONE_OSTM);
            tx->store(_FROM_TWO_OSTM);
            tx->store(_FROM_THREE_OSTM);
            tx->store(_FROM_FOUR_OSTM);
            tx->store(_FROM_FIVE_OSTM);
            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}

void _two_account_transfer_(std::shared_ptr<OSTM> _to_, std::shared_ptr<OSTM> _from_,
 TM& _tm, double _amount) {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_from_);
    std::shared_ptr<BANK> _TO_BANK_, _FROM_BANK_;
    std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;

    bool done = false;
    try {
        while (!done) {
            _TO_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
            _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
            _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
            _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
            _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
            _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
            tx->store(_TO_OSTM_);
            tx->store(_FROM_OSTM_);

            std::shared_ptr<TX> txTwo = _tm._get_tx();

            bool nestedDone = false;
            while (!nestedDone) {
                _TO_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_to_));
                _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_from_));
                _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
                _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
                txTwo->store(_TO_OSTM_);
                txTwo->store(_FROM_OSTM_);
                nestedDone = txTwo->commit();
            }
            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}

void _nesting_(std::shared_ptr<OSTM> _to_, std::shared_ptr<OSTM> _from_, TM& _tm, double _amount)
{
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_from_);
    std::shared_ptr<BANK> _TO_BANK_, _FROM_BANK_;
    std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;


    bool done = false;
    try {
        while (!done) {
            _TO_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_to_));
            _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
            _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
            _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
            _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
            _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
            tx->store(_TO_OSTM_);
            tx->store(_FROM_OSTM_);

            std::shared_ptr<TX> txTwo = _tm._get_tx();

            bool nestedDone = false;
            while (!nestedDone) {
                _TO_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_to_));
                _FROM_BANK_ = std::dynamic_pointer_cast<BANK> (txTwo->load(_from_));
                _TO_BANK_->SetBalance(_TO_BANK_->GetBalance() + _amount);
                _FROM_BANK_->SetBalance(_FROM_BANK_->GetBalance() - _amount);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_BANK_);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_BANK_);
                txTwo->store(_TO_OSTM_);
                txTwo->store(_FROM_OSTM_);
                _two_account_transfer_(_to_, _from_, _tm, _amount);

                nestedDone = txTwo->commit();
            }

            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
```

```
}

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) {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_from_);
    tx->_register(_from_two_);
    std::shared_ptr<OSTM> _FROM_OSTM_ONE_, _FROM_OSTM_TWO_, _TO_OSTM_;
    std::shared_ptr<BANK> _FROM_, _FROM_TWO_, _TO_;

    bool done = false;
    try {
        while (!done) {
            // for (int i = 0; i < vector_number; ++i) {
            for (auto&& obj : _customer_vec) {
                // auto&& obj = _customer_vec.at(i);
                tx->_register(obj);
                _FROM_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_));
                _FROM_TWO_ = std::dynamic_pointer_cast<BANK> (tx->load(_from_two_));
                _TO_ = std::dynamic_pointer_cast<BANK> (tx->load(obj));
                _FROM_->SetBalance(_FROM_->GetBalance() - _amount);
                _FROM_TWO_->SetBalance(_FROM_TWO_->GetBalance() - _amount);
                _TO_->SetBalance(_TO_->GetBalance() + (_amount * 2));
                _FROM_OSTM_ONE_ = std::dynamic_pointer_cast<OSTM> (_FROM_);
                _FROM_OSTM_TWO_ = std::dynamic_pointer_cast<OSTM> (_FROM_TWO_);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_);
                tx->store(_FROM_OSTM_ONE_);
                tx->store(_FROM_OSTM_TWO_);
                tx->store(_TO_OSTM_);
            }
            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}

void _warehouse_transfer_(std::shared_ptr<OSTM> _to_, std::shared_ptr<OSTM> _from_, TM&
 _tm, double _amount) {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_from_);
    std::shared_ptr<WAREHOUSE> _TO_SHOP_, _FROM_DIST_;
    std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;

    bool done = false;
    try {
        while (!done) {
            _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
            _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));
            _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
            _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);

            _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
            _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);

            _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
            _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);

            _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
            _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
            _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
            _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
            tx->store(_TO_OSTM_);
            tx->store(_FROM_OSTM_);
            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}

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)
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_to_two);
    tx->_register(_to_three);
    tx->_register(_from_);
    std::shared_ptr<WAREHOUSE> _TO_SHOP_, _FROM_DIST_;
    std::shared_ptr<OSTM> _TO_OSTM_, _FROM_OSTM_;

    bool done = false;
    try {
        while (!done) {
            _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
            _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));
            _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
            _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);

            _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
            _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);

            _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
            _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);

            _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
            _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
            _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
            _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
            tx->store(_TO_OSTM_);
            tx->store(_FROM_OSTM_);

            std::shared_ptr<TX> txTwo = _tm._get_tx();
            bool nestedDone = false;
            while (!nestedDone) {
                _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_to_two));
                _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_from_));
                _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
                _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);

                _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
                _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);

                _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
                _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);

                _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
                _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
                txTwo->store(_TO_OSTM_);
                txTwo->store(_FROM_OSTM_);

                /*
                 * NESTED TRANSACTION TEST _to_three
                 */
                _warehouse_transfer_(_to_three, _from_, _tm, _amount);


                nestedDone = tx->commit();
            }
            done = tx->commit();
        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}

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) {
    std::shared_ptr<TX> tx = _tm._get_tx();
    tx->_register(_to_);
    tx->_register(_to_two);
    tx->_register(_to_three);
    tx->_register(_from_);
    std::shared_ptr<WAREHOUSE> _TO_SHOP_, _TO_SHOP_TWO, _TO_SHOP_VEC, _FROM_DIST_;
    std::shared_ptr<OSTM> _TO_OSTM_, _TO_OSTM_TWO, _TO_OSTM_VEC, _FROM_OSTM_;

    bool done = false;
    try {
        while (!done) {

            // for (int i = 0; i < vector_number; ++i) {
            for (auto&& obj : _warehouse_vec) {
                //auto&& obj = _warehouse_vec.at(i);
                tx->_register(obj);
                _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_));
                _TO_SHOP_TWO = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_to_two));
                _TO_SHOP_VEC = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(obj));
                _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (tx->load(_from_));

                _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
                _TO_SHOP_TWO->SetNumber_of_nokia(_TO_SHOP_TWO->GetNumber_of_nokia() + _amount);
                _TO_SHOP_VEC->SetNumber_of_nokia(_TO_SHOP_VEC->GetNumber_of_nokia() + _amount);
                _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - (_amount * 3));

                _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
                _TO_SHOP_TWO->SetNumber_of_samsung(_TO_SHOP_TWO->GetNumber_of_samsung() + _amount);
                _TO_SHOP_VEC->SetNumber_of_samsung(_TO_SHOP_VEC->GetNumber_of_samsung() + _amount);
                _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - (_amount * 3));

                _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
                _TO_SHOP_TWO->SetNumber_of_iphones(_TO_SHOP_TWO->GetNumber_of_iphones() + _amount);
                _TO_SHOP_VEC->SetNumber_of_iphones(_TO_SHOP_VEC->GetNumber_of_iphones() + _amount);
                _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - (_amount * 3));

                _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
                _TO_SHOP_TWO->SetNumber_of_sony(_TO_SHOP_TWO->GetNumber_of_sony() + _amount);
                _TO_SHOP_VEC->SetNumber_of_sony(_TO_SHOP_VEC->GetNumber_of_sony() + _amount);
                _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - (_amount * 3));

                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
                _TO_OSTM_TWO = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_TWO);
                _TO_OSTM_VEC = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_VEC);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
                tx->store(_TO_OSTM_);
                tx->store(_TO_SHOP_TWO);
                tx->store(_TO_SHOP_VEC);
                tx->store(_FROM_OSTM_);



            }
            std::shared_ptr<TX> txTwo = _tm._get_tx();
            bool nestedDone = false;
            while (!nestedDone) {
                _TO_SHOP_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_to_two));
                _FROM_DIST_ = std::dynamic_pointer_cast<WAREHOUSE> (txTwo->load(_from_));
                _TO_SHOP_->SetNumber_of_nokia(_TO_SHOP_->GetNumber_of_nokia() + _amount);
                _FROM_DIST_->SetNumber_of_nokia(_FROM_DIST_->GetNumber_of_nokia() - _amount);

                _TO_SHOP_->SetNumber_of_samsung(_TO_SHOP_->GetNumber_of_samsung() + _amount);
                _FROM_DIST_->SetNumber_of_samsung(_FROM_DIST_->GetNumber_of_samsung() - _amount);

                _TO_SHOP_->SetNumber_of_iphones(_TO_SHOP_->GetNumber_of_iphones() + _amount);
                _FROM_DIST_->SetNumber_of_iphones(_FROM_DIST_->GetNumber_of_iphones() - _amount);

                _TO_SHOP_->SetNumber_of_sony(_TO_SHOP_->GetNumber_of_sony() + _amount);
                _FROM_DIST_->SetNumber_of_sony(_FROM_DIST_->GetNumber_of_sony() - _amount);
                _TO_OSTM_ = std::dynamic_pointer_cast<OSTM> (_TO_SHOP_);
                _FROM_OSTM_ = std::dynamic_pointer_cast<OSTM> (_FROM_DIST_);
                txTwo->store(_TO_OSTM_);
                txTwo->store(_FROM_OSTM_);

                /*
                 * NESTED TRANSACTION TEST _to_three
                 */
                _warehouse_transfer_(_to_three, _from_, _tm, _amount);
                _nested_warehouse_transfer_(_to_, _to_two, _to_three, _from_, _tm, _amount);

                nestedDone = tx->commit();
            }

            done = tx->commit();

        }
    } catch (std::runtime_error& e) {
        std::cout << e.what() << std::endl;
    }
}

int main(void) {
    TM& tm = TM::Instance();

    std::vector<std::shared_ptr < OSTM>>_customer_vec; //(vector_number);
    std::vector<std::shared_ptr < OSTM>>_warehouse_vec; //(vector_number);

    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"));

    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());

    for (int i = 0; i < vector_number; ++i) {
        if (i % 5 == 0) {
            std::shared_ptr<OSTM> sharedptr(new CARLOW_W());
            _warehouse_vec.push_back(std::move(sharedptr));
        } else if (i % 4 == 0) {
            std::shared_ptr<OSTM> sharedptr(new KILKENNY_W());
            _warehouse_vec.push_back(std::move(sharedptr));
        } else if (i % 3 == 0) {
            std::shared_ptr<OSTM> sharedptr(new TALLAGH_W());
            _warehouse_vec.push_back(std::move(sharedptr));
        } else if (i % 2 == 0) {
            std::shared_ptr<OSTM> sharedptr(new DUNDALK_W());
            _warehouse_vec.push_back(std::move(sharedptr));
        } else if (i % 1 == 0) {
            std::shared_ptr<OSTM> sharedptr(new SLIGO_W());
            _warehouse_vec.push_back(std::move(sharedptr));
        }
    }

    for (int i = 0; i < vector_number; ++i) {
        if (i % 6 == 0) {
            std::shared_ptr<OSTM> sharedptr(new AIB(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        } else if (i % 5 == 0) {
            std::shared_ptr<OSTM> sharedptr(new BOI(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        } else if (i % 4 == 0) {
            std::shared_ptr<OSTM> sharedptr(new BOA(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        } else if (i % 3 == 0) {
            std::shared_ptr<OSTM> sharedptr(new SWBPLC(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        } else if (i % 2 == 0) {
            std::shared_ptr<OSTM> sharedptr(new ULSTER(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        } else if (i % 1 == 0) {
            std::shared_ptr<OSTM> sharedptr(new UNBL(i, 50, "Joe", "Blog", "High street, Kilkenny,
Co.Kilkenny"));
            _customer_vec.push_back(std::move(sharedptr));
        }
    }

    //    w_dist->toString();
    //    c_shop->toString();
    //    k_shop->toString();
    //    t_shop->toString();
    //    d_shop->toString();
    //    s_shop->toString();

    /*
     * TEST 1 : object requirements
     */
    aib_ptr->toString();
    boi_ptr->toString();
    boa_ptr->toString();
    swplc_ptr->toString();
    ulster_ptr->toString();
    unbl_ptr->toString();

    /*
     * TEST 2 : object requirements
     */
    //    aib_ptr->toString();
    //    boi_ptr->toString();
    //    boa_ptr->toString();
    //    swplc_ptr->toString();
    //    ulster_ptr->toString();
    //    unbl_ptr->toString();
    //    for(int i=0; i<vector_number; ++i){
    //        _customer_vec[i]->toString();
    //    }

    /*
     * TEST 3 : object requirements
     */
    //    w_dist->toString();
    //    c_shop->toString();
    //    k_shop->toString();
    //    t_shop->toString();

    /*
     * TEST 4 : objects requirements
     */
    //        w_dist->toString();
    //        c_shop->toString();
    //        k_shop->toString();
    //        t_shop->toString();
    //        d_shop->toString();
    //        s_shop->toString();


    /*
     * TEST 5 : objects requirements
     */
    //        w_dist->toString();
    //        c_shop->toString();
    //        k_shop->toString();
    //        t_shop->toString();
    //        d_shop->toString();
    //        s_shop->toString();

    //        for(auto&& elem: _warehouse_vec){
    //            elem->toString(); // virtual dispatch
    //
    //        }



    int transferAmount = 1;
    int threadArraySize = 300;

    std::thread thArray[300];

    for (int i = 0; i < threadArraySize; ++i) {

        //thArray[i] = std::thread(_nesting_, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
      if (i % 3 == 0)
          thArray[i] = std::thread(_nesting_, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
      else if (i % 2 == 0)
          thArray[i] = std::thread(_nesting_, boa_ptr, swplc_ptr, std::ref(tm), transferAmount);
      else if (i % 1 == 0)
          thArray[i] = std::thread(_nesting_, ulster_ptr, unbl_ptr, std::ref(tm), transferAmount);

        //    if (i % 3 == 0)
        //        thArray[i] = std::thread(_two_account_transfer_, aib_ptr, boi_ptr, std::ref(tm), transferAmount);
        //    else if (i % 2 == 0)
        //        thArray[i] = std::thread(_six_account_transfer_, boi_ptr, boa_ptr, swplc_ptr, ulster_ptr, aib_ptr, unbl_ptr, std::ref(tm), transferAmount);
        //    else if (i % 1 == 0)
        //        thArray[i] = std::thread(_complex_transfer_, aib_ptr, boi_ptr, std::ref(_customer_vec), std::ref(tm), transferAmount);


        //        if (i % 3 == 0)
        //            thArray[i] = std::thread(_warehouse_transfer_, c_shop, w_dist, std::ref(tm), transferAmount);
        //        else if (i % 2 == 0)
        //            thArray[i] = std::thread(_warehouse_transfer_, k_shop, w_dist, std::ref(tm), transferAmount);
        //        else if (i % 1 == 0)
        //            thArray[i] = std::thread(_warehouse_transfer_, t_shop, w_dist, std::ref(tm), transferAmount);

        //        if (i % 3 == 0)
        //            thArray[i] = std::thread(_nested_warehouse_transfer_, c_shop, d_shop, k_shop, w_dist, std::ref(tm), transferAmount);
        //        else if (i % 2 == 0)
        //            thArray[i] = std::thread(_nested_warehouse_transfer_, k_shop, s_shop, t_shop, w_dist, std::ref(tm), transferAmount);
        //        else if (i % 1 == 0)
        //            thArray[i] = std::thread(_nested_warehouse_transfer_, t_shop, c_shop, s_shop, w_dist, std::ref(tm), transferAmount);

        //        if (i % 3 == 0)
        //            thArray[i] = std::thread(_warehouse_transfer_, c_shop, w_dist, std::ref(tm), transferAmount);
        //        else if (i % 2 == 0)
        //            thArray[i] = std::thread(_nested_warehouse_transfer_, k_shop, s_shop, t_shop, w_dist, std::ref(tm), transferAmount);
        //        else if (i % 1 == 0)
        //            thArray[i] = std::thread(_complex_warehouse_transfer_, d_shop, s_shop, c_shop, std::ref(_warehouse_vec), w_dist, std::ref(tm), transferAmount);


    }
    /*
     * Join threads^n -> threadArraySize<br>
     * thArray[i].join();
     */
    for (int i = 0; i < threadArraySize; ++i) {
        thArray[i].join();
    }


    std::cout << "\nMain process print " << std::endl;
    /*
     * TEST 1 : object requirements
     */
    aib_ptr->toString();
    boi_ptr->toString();
    boa_ptr->toString();
    swplc_ptr->toString();
    ulster_ptr->toString();
    unbl_ptr->toString();

    /*
     * TEST 2 : object requirements
     */
    //    aib_ptr->toString();
    //    boi_ptr->toString();
    //    boa_ptr->toString();
    //    swplc_ptr->toString();
    //    ulster_ptr->toString();
    //    unbl_ptr->toString();
    //    for(int i=0; i<vector_number; ++i){
    //        _customer_vec[i]->toString();
    //    }

    /*
     * TEST 3 : object requirements
     */
    //            w_dist->toString();
    //            c_shop->toString();
    //            k_shop->toString();
    //            t_shop->toString();

    /*
     * TEST 4 : objects requirements
     */
    //        w_dist->toString();
    //        c_shop->toString();
    //        k_shop->toString();
    //        t_shop->toString();
    //        d_shop->toString();
    //        s_shop->toString();

    /*
     * TEST 5 : objects requirements
     */
    //        w_dist->toString();
    //        c_shop->toString();
    //        k_shop->toString();
    //        t_shop->toString();
    //        d_shop->toString();
    //        s_shop->toString();

    //        for(auto&& elem: _warehouse_vec){
    //            elem->toString(); // virtual dispatch
    //
    //        }

    /* TEST 5 FINISH */


    std::cout << "\nMAIN PROCESS EXIT !!!! " << std::endl;
    std::shared_ptr<TX> tx = tm._get_tx();

    std::cout << "Rollback counter is : " << tx->getTest_counter() << std::endl;
    //    std::cout << "[vector_number]" << std::endl;
    //    for (int i = 0; i < vector_number; ++i) {
    //        //_customer_vec[i]->toString();
    //        auto&& os = _customer_vec.at(i);
    //        os->toString();
    //    }
    //    std::cout << "[_warehouse_vec]" << std::endl;
    //    for(auto&& elem: _warehouse_vec){
    //        elem->toString(); // virtual dispatch
    //
    //    }
    //_customer_vec[10]->toString();

    tm._TX_EXIT();
    std::cout << "\nPRINT ALL FROM TM !!!! SHOULD BE EMPTY AFTER _TX_EXIT() !!" << std::endl;
    tm.print_all();
    int t = 0;
    std::cin >> t;
    return 0;
}
```

7.35 OSTM.cpp File Reference

`#include "OSTM.h"`

Include dependency graph for OSTM.cpp:



7.36 OSTM.cpp

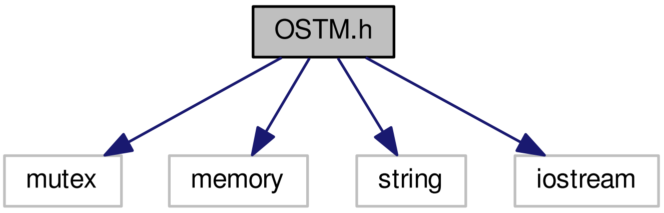
```
/*
 * File:   OSTM.cpp
 * Author: Zoltan Fuzesi
 *
 * Created on December 18, 2017, 2:09 PM
 * OSTM cpp file methods implementations
 */

#include "OSTM.h"

int OSTM::global_Unique_ID_Number = 0;

OSTM::OSTM()
{
    this->version = ZERO;
    this->uniqueID = Get_global_Unique_ID_Number(); //++global_Unique_ID_Number;
    this->canCommit = true;
    this->abort_Transaction = false;
}


OSTM::OSTM(int _version_number_, int _unique_id_)
{
   // std::cout << "OSTM COPY CONSTRUCTOR" << global_Unique_ID_Number << std::endl;
    this->uniqueID = _unique_id_;
    this->version = _version_number_;
    this->canCommit = true;
    this->abort_Transaction = false;
}

OSTM::~OSTM() {
    //std::cout << "[OSTM DELETE]" << std::endl;
}
int OSTM::Get_global_Unique_ID_Number() {
    if(global_Unique_ID_Number > 10000000)
        global_Unique_ID_Number = 0;
    return ++global_Unique_ID_Number;
}
```

```

void OSTM::Set_Unique_ID(int uniqueID) {
    this->uniqueID = uniqueID;
}
int OSTM::Get_Unique_ID() const
{
    return uniqueID;
}
void OSTM::Set_Version(int version)
{
    this->version = version;
}
int OSTM::Get_Version() const
{
    return version;
}
void OSTM::increase_VersionNumber()
{
    this->version += 1;
}
void OSTM::Set_Can_Commit(bool canCommit) {
    this->canCommit = canCommit;
}
bool OSTM::Is_Can_Commit() const {
    return canCommit;
}
void OSTM::Set_Abort_Transaction(bool abortTransaction) {
    this->abort_Transaction = abortTransaction;
}
bool OSTM::Is_Abort_Transaction() const {
    return abort_Transaction;
}
void OSTM::lock_Mutex() {
    this->mutex.lock();
}
void OSTM::unlock_Mutex() {
    this->mutex.unlock();
}
bool OSTM::is_Locked(){
    return this->mutex.try_lock();
}
```

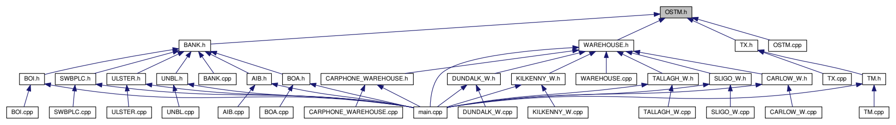
7.37 OSTM.h File Reference

```
#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.

```
                                  {
public:
    OSTM();
    OSTM(int _version_number_, int _unique_id_);
    virtual ~OSTM();
    virtual void copy(std::shared_ptr<OSTM> from, std::shared_ptr<OSTM> to){};
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object) = 0;//std::cout << "[OSTM GETBASECOPY]" << std::endl;};
    virtual void toString(){};
    void Set_Unique_ID(int uniqueID);
    int Get_Unique_ID() const;
    void Set_Version(int version);
    int Get_Version() const;
    void increase_VersionNumber();
    bool Is_Can_Commit() const;
    void Set_Can_Commit(bool canCommit);
    void Set_Abort_Transaction(bool abortTransaction);
    bool Is_Abort_Transaction() const;
    void lock_Mutex();
    void unlock_Mutex();
    bool is_Locked();

private:
    /*
     * \brief Object version number
     */
    int version;
    /*
     * \brief Object unique identifier
     */
    int uniqueID;
    /*
     * \brief Boolean value to check any other thread failed to commit
     */
    bool canCommit;
    /*
     * \brief Abort the transaction
     */
    bool abort_Transaction;
    static int global_Unique_ID_Number;
    const int ZERO = 0;
    std::mutex mutex;
    int Get_global_Unique_ID_Number();

};
```

7.38 OSTM.h

```
/*
 * File:   OSTM.h
 * Author: Zoltan FUzesi
 *
 * Created on December 18, 2017, 2:09 PM
 * OSTM header file fields and  methods declarations
 */

#ifndef OSTM_H
#define OSTM_H
#include <mutex>
#include <memory>
#include <string>
#include <iostream>
#include <string>

class __declspec(dllexport) OSTM {
public:
    OSTM();
    OSTM(int _version_number_, int _unique_id_);
    virtual ~OSTM();
    virtual void copy(std::shared_ptr<OSTM> from, std::shared_ptr<OSTM> to){};
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object) = 0;//std::cout << "[OSTM GETBASECOPY]" << std::endl;};
    virtual void toString(){};
    void Set_Unique_ID(int uniqueID);
    int Get_Unique_ID() const;
    void Set_Version(int version);
    int Get_Version() const;
    void increase_VersionNumber();
    bool Is_Can_Commit() const;
    void Set_Can_Commit(bool canCommit);
    void Set_Abort_Transaction(bool abortTransaction);
    bool Is_Abort_Transaction() const;
    void lock_Mutex();
    void unlock_Mutex();
    bool is_Locked();

private:
    /*
     * \brief Object version number
     */
    int version;
    /*
     * \brief Object unique identifier
     */
    int uniqueID;
    /*
     * \brief Boolean value to check any other thread failed to commit
     */
    bool canCommit;
    /*
     * \brief Abort the transaction
     */
    bool abort_Transaction;
    static int global_Unique_ID_Number;
    const int ZERO = 0;
    std::mutex mutex;
    int Get_global_Unique_ID_Number();

};

#endif /* OSTM_H */
```

7.39 README.md File Reference

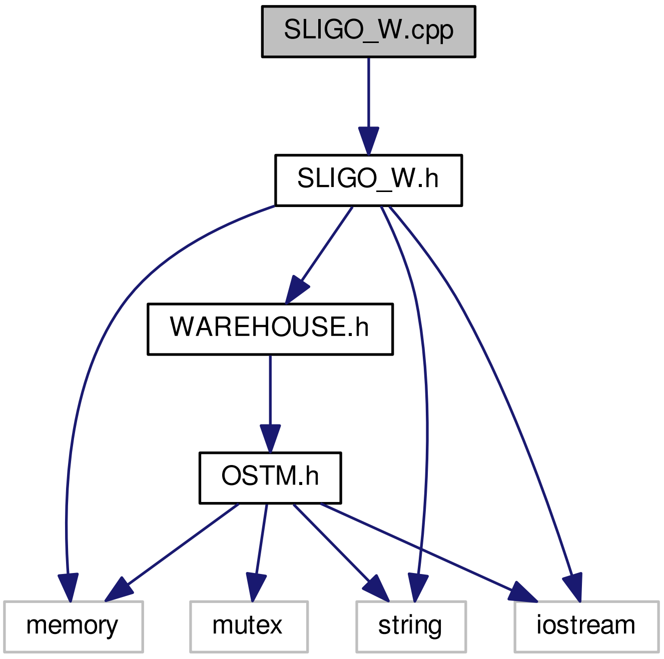
7.40 README.md

```
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 demostrate 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 speciffic behaviours.





```

7.41 SLIGO_W.cpp File Reference

```
#include "SLIGO_W.h"
```

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     std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

00028     return ostm_obj;
00029 }
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();
```

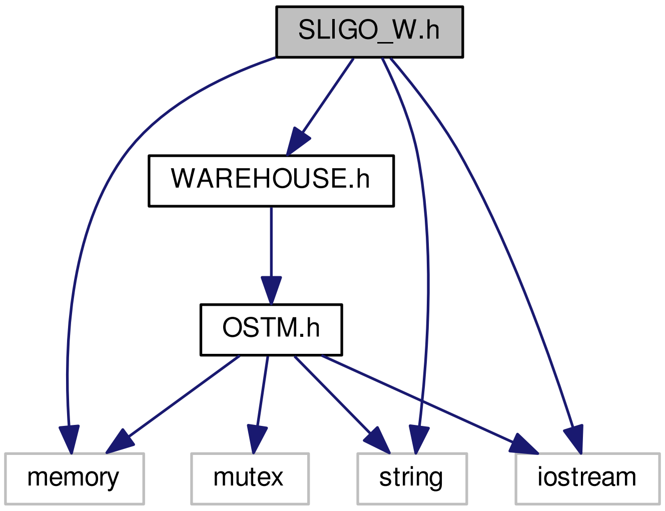
```
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
//std::shared_ptr<SLIGO_W> SLIGO_W::_cast(std::shared_ptr<OSTM> _object){
//
//    return static_cast<std::shared_ptr<SLIGO_W>>(_object);
//}
void SLIGO_W::toString()
{
    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;
}



void SLIGO_W::SetNumber_of_alcatel(int _number_of_alcatel) {
    this->_number_of_alcatel = _number_of_alcatel;
}

int SLIGO_W::GetNumber_of_alcatel(){
    return _number_of_alcatel;
}

void SLIGO_W::SetNumber_of_nokia(int _number_of_nokia) {
    this->_number_of_nokia = _number_of_nokia;
}

int SLIGO_W::GetNumber_of_nokia(){
    return _number_of_nokia;
}

void SLIGO_W::SetNumber_of_huawei(int _number_of_huawei) {
    this->_number_of_huawei = _number_of_huawei;
}

int SLIGO_W::GetNumber_of_huawei(){
    return _number_of_huawei;
}

void SLIGO_W::SetNumber_of_sony(int _number_of_sony) {
    this->_number_of_sony = _number_of_sony;
}

int SLIGO_W::GetNumber_of_sony(){
    return _number_of_sony;
}

void SLIGO_W::SetNumber_of_samsung(int _number_of_samsung) {
    this->_number_of_samsung = _number_of_samsung;
}

int SLIGO_W::GetNumber_of_samsung(){
    return _number_of_samsung;
}

void SLIGO_W::SetNumber_of_iphones(int _number_of_iphones) {
    this->_number_of_iphones = _number_of_iphones;
}

int SLIGO_W::GetNumber_of_iphones(){
    return _number_of_iphones;
}

void SLIGO_W::SetShop_name(std::string _shop_name) {
    this->_shop_name = _shop_name;
}

std::string SLIGO_W::GetShop_name(){
    return _shop_name;
}

void SLIGO_W::SetShop_address(std::string _shop_address) {
    this->_shop_address = _shop_address;
}

```

```
std::string SLIGO_W::GetShop_address(){
    return _shop_address;
}



```

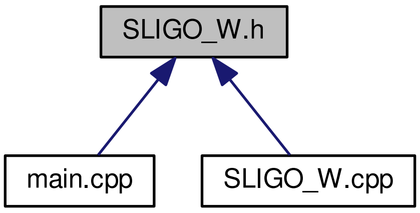
7.43 SLIGO_W.h File Reference

```
#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

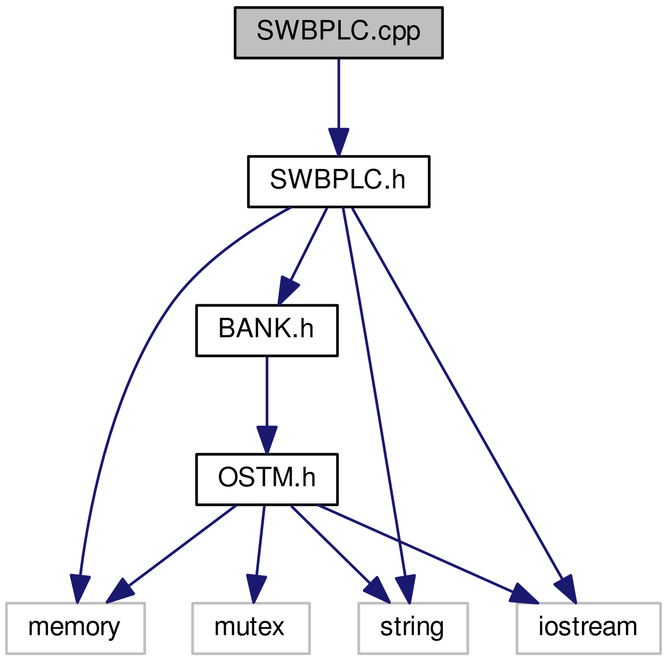
7.44 SLIGO_W.h

```

/*
 * File:   SLIGO_W.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef SLIGO_W_H
#define SLIGO_W_H
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>
class SLIGO_W :public WAREHOUSE {
public:
    SLIGO_W() : WAREHOUSE(){

        this->_shop_address = "Sligo River Street";
        this->_shop_name = "SLIGO S_WAREHOUSE";
        this->_number_of_iphones = 200;
        this->_number_of_samsung = 200;
        this->_number_of_sony = 200;
        this->_number_of_huawei = 200;
        this->_number_of_nokia = 200;
        this->_number_of_alcatel = 200;
    };
    SLIGO_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
huawei, int nokia, int alcatel): WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
    SLIGO_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
    SLIGO_W(const SLIGO_W& orig);
    SLIGO_W operator=(const SLIGO_W& orig){};
    virtual ~SLIGO_W();

    /*
     * Implement OSTM virtual methods
     */
   // virtual std::shared_ptr<SLIGO_W> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();
    /*
     * Implement Warehouse methods
     */
    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();


private:
    std::string _shop_address;
    std::string _shop_name;
    int _number_of_iphones;
    int _number_of_samsung;
    int _number_of_sony;
    int _number_of_huawei;
    int _number_of_nokia;
    int _number_of_alcatel;

};

#endif /* SLIGO_W_H */

```

7.45 SWBPLC.cpp File Reference

```
#include "SWBPLC.h"
```

Include dependency graph for SWBPLC.cpp:



7.46 SWBPLC.cpp

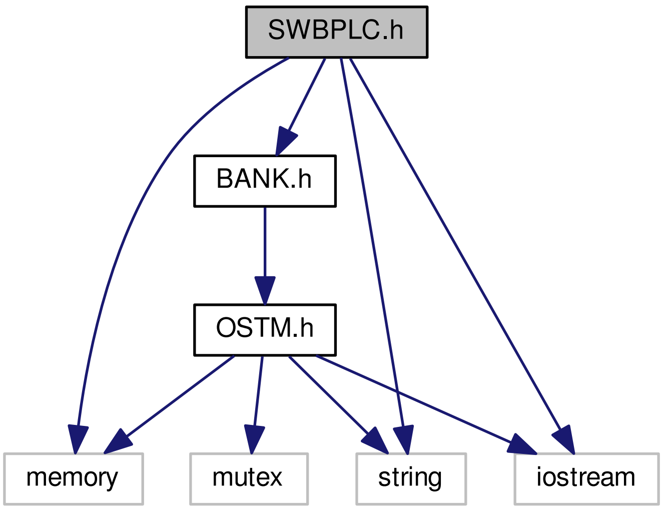
```

```

```
/*
 * File:   SWBPLC.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "SWBPLC.h"

SWBPLC::SWBPLC(const SWBPLC& orig) {
}

SWBPLC::~SWBPLC() {
}
std::shared_ptr<OSTM> SWBPLC::getBaseCopy(std::shared_ptr<OSTM> object)
{
    std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
    std::shared_ptr<BANK> obj(new SWBPLC(objTO,object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

    return ostm_obj;
}
void SWBPLC::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<SWBPLC> objTO = std::dynamic_pointer_cast<SWBPLC>(to);
    std::shared_ptr<SWBPLC> objFROM = std::dynamic_pointer_cast<SWBPLC>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());


}
//std::shared_ptr<SWBPLC> SWBPLC::_cast(std::shared_ptr<OSTM> _object){
//
//    return static_cast<std::shared_ptr<SWBPLC>>(_object);
//}
void SWBPLC::toString()
{
     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;
}

void SWBPLC::SetAddress(std::string address) {
    this->address = address;
}

std::string SWBPLC::GetAddress() const {
    return address;
}

void SWBPLC::SetBalance(double balance) {
    this->balance = balance;
}

double SWBPLC::GetBalance() const {
    return balance;
}

void SWBPLC::SetAccountNumber(int accountNumber) {
    this->accountNumber = accountNumber;
}

int SWBPLC::GetAccountNumber() const {
    return accountNumber;
}

void SWBPLC::SetLastName(std::string lastName) {
    this->lastName = lastName;
}

std::string SWBPLC::GetLastName() const {
    return lastName;
}

void SWBPLC::SetFirstName(std::string firstName) {
    this->firstName = firstName;
}

std::string SWBPLC::GetFirstName() const {
    return firstName;
}

void SWBPLC::SetFullname(std::string fullname) {
    this->fullname = fullname;
}

std::string SWBPLC::GetFullname() const {
    return fullname;
}

```

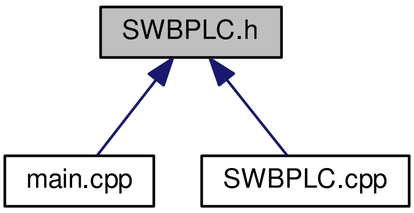
7.47 SWBPLC.h File Reference

```
#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

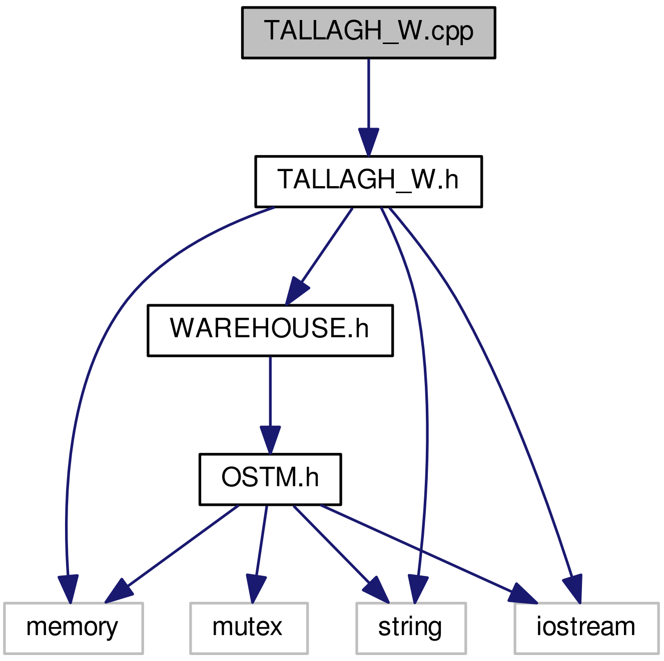
- class SWBPLC

7.48 SWBPLC.h

```

/*
 * File:   SWBPLC.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef SWBPLC_H
#define SWBPLC_H
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>
class SWBPLC : public BANK {
public:
    SWBPLC() : BANK() {
        this->accountNumber = 0;
        this->balance = 50;
        this->firstName = "Joe";
        this->lastName = "Blog";
        this->address = "High street, Carlow";
        this->fullname = firstName + " " + lastName;
    };
    SWBPLC(int accountNumber, double balance, std::string firstName, std::string lastName,
std::string address) : BANK() {
        this->accountNumber = accountNumber;
        this->balance = balance;
        this->firstName = firstName;
        this->lastName = lastName;
        this->address = address;
        this->fullname = firstName + " " + lastName;
    };
    SWBPLC(std::shared_ptr<BANK> obj, int _version, int _unique_id) : BANK(_version, _unique_id)
{

        this->accountNumber = obj->GetAccountNumber();
        this->balance = obj->GetBalance();
        this->firstName = obj->GetFirstName();
        this->lastName = obj->GetLastName();
        this->address = obj->GetAddress();
        this->fullname = obj->GetFirstName() + " " + obj->GetLastName();

    };
    SWBPLC(const SWBPLC& orig);
    SWBPLC operator=(const SWBPLC& orig) {};
    virtual ~SWBPLC();

    /*
     * Implement OSTM virtual methods
     */
    //virtual std::shared_ptr<SWBPLC> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();

    /*
     * Implement BANK virtual methods
     */
    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;
private:
    std::string fullname;
    std::string firstName;
    std::string lastName;
    int accountNumber;
    double balance;
    std::string address;

};

#endif /* SWBPLC_H */

```

7.49 TALLAGH_W.cpp File Reference

`#include "TALLAGH_W.h"`
Include dependency graph for TALLAGH_W.cpp:



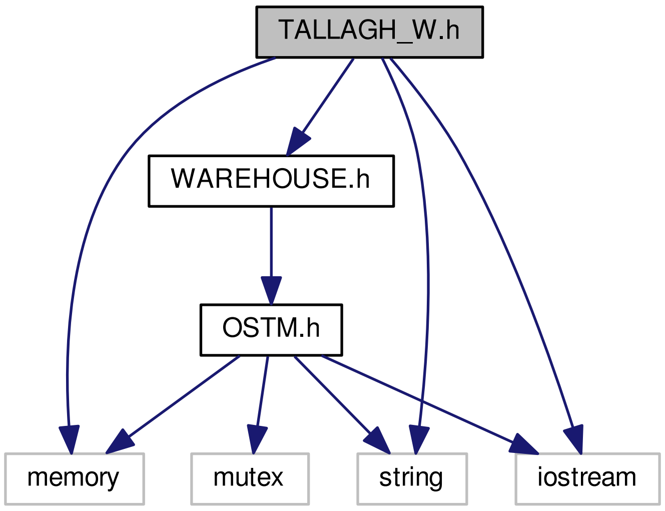
7.50 TALLAGH_W.cpp

```

/*
 * File:   TALLAGH_W.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "TALLAGH_W.h"

TALLAGH_W::~TALLAGH_W() {
}

TALLAGH_W::TALLAGH_W(const TALLAGH_W& orig) {
}
std::shared_ptr<OSTM> TALLAGH_W::getBaseCopy(std::shared_ptr<OSTM> object)
{

    std::shared_ptr<WAREHOUSE> objTO = std::dynamic_pointer_cast<WAREHOUSE>(object);
    std::shared_ptr<WAREHOUSE> obj(new TALLAGH_W(objTO, object->Get_Version(),object->
Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

    return ostm_obj;
}
void TALLAGH_W::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<TALLAGH_W> objTO = std::dynamic_pointer_cast<TALLAGH_W>(to);
    std::shared_ptr<TALLAGH_W> objFROM = std::dynamic_pointer_cast<TALLAGH_W>(from);
    objTO->_shop_address = objFROM->GetShop_address();
    objTO->_shop_name = objFROM->GetShop_name();
    objTO->_number_of_iphones = objFROM->GetNumber_of_iphones();
    objTO->_number_of_samsung = objFROM->GetNumber_of_samsung();
    objTO->_number_of_sony = objFROM->GetNumber_of_sony();
    objTO->_number_of_huawei = objFROM->GetNumber_of_huawei();
    objTO->_number_of_nokia = objFROM->GetNumber_of_nokia();
    objTO->_number_of_alcatel = objFROM->GetNumber_of_alcatel();
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());


}
//std::shared_ptr<TALLAGH_W> TALLAGH_W::_cast(std::shared_ptr<OSTM> _object){
//
//    return static_cast<std::shared_ptr<TALLAGH_W>>(_object);
//}
void TALLAGH_W::toString()
{
    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;
}

void TALLAGH_W::SetNumber_of_alcatel(int _number_of_alcatel) {
    this->_number_of_alcatel = _number_of_alcatel;
}

int TALLAGH_W::GetNumber_of_alcatel(){
    return _number_of_alcatel;
}

void TALLAGH_W::SetNumber_of_nokia(int _number_of_nokia) {
    this->_number_of_nokia = _number_of_nokia;
}

int TALLAGH_W::GetNumber_of_nokia(){
    return _number_of_nokia;
}

void TALLAGH_W::SetNumber_of_huawei(int _number_of_huawei) {
    this->_number_of_huawei = _number_of_huawei;
}

int TALLAGH_W::GetNumber_of_huawei(){
    return _number_of_huawei;
}

void TALLAGH_W::SetNumber_of_sony(int _number_of_sony) {
    this->_number_of_sony = _number_of_sony;
}

int TALLAGH_W::GetNumber_of_sony(){
    return _number_of_sony;
}

void TALLAGH_W::SetNumber_of_samsung(int _number_of_samsung) {
    this->_number_of_samsung = _number_of_samsung;
}

int TALLAGH_W::GetNumber_of_samsung(){
    return _number_of_samsung;
}

void TALLAGH_W::SetNumber_of_iphones(int _number_of_iphones) {
    this->_number_of_iphones = _number_of_iphones;
}

int TALLAGH_W::GetNumber_of_iphones(){
    return _number_of_iphones;
}

void TALLAGH_W::SetShop_name(std::string _shop_name) {
    this->_shop_name = _shop_name;
}

std::string TALLAGH_W::GetShop_name(){
    return _shop_name;
}

void TALLAGH_W::SetShop_address(std::string _shop_address) {
    this->_shop_address = _shop_address;
}

std::string TALLAGH_W::GetShop_address(){
    return _shop_address;
}
```

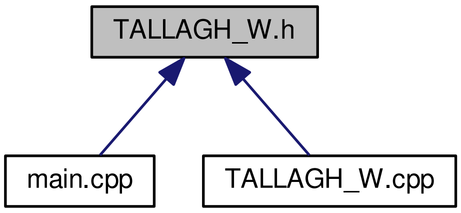
7.51 TALLAGH_W.h File Reference

```
#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

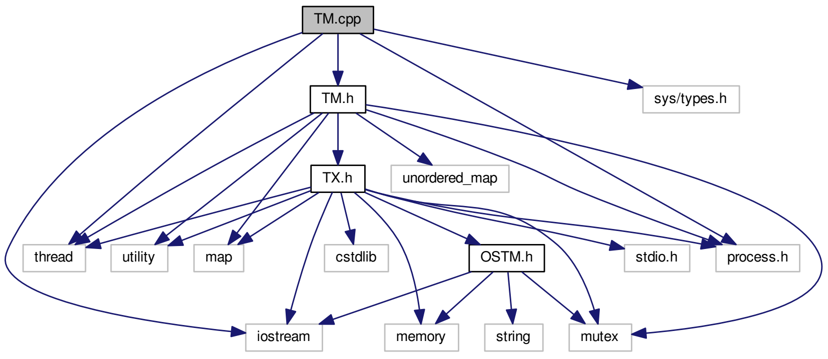
7.52 TALLAGH_W.h

```

/*
 * File:   TALLAGH_W.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef TALLAGH_W_H
#define TALLAGH_W_H
#include "WAREHOUSE.h"
#include <string>
#include <memory>
#include <iostream>
class TALLAGH_W :public WAREHOUSE {
public:
    TALLAGH_W() : WAREHOUSE(){

        this->_shop_address = "Tallagh Low street";
        this->_shop_name = "TALLAGH T_WAREHOUSE";
        this->_number_of_iphones = 200;
        this->_number_of_samsung = 200;
        this->_number_of_sony = 200;
        this->_number_of_huawei = 200;
        this->_number_of_nokia = 200;
        this->_number_of_alcatel = 200;
    };
    TALLAGH_W(std::string address, std::string shop_name, int iphone, int samsung, int sony, int
huawei, int nokia, int alcatel): WAREHOUSE(){
        /*
         * copy over values
         */
        this->_shop_address = address;
        this->_shop_name = shop_name;
        this->_number_of_iphones = iphone;
        this->_number_of_samsung = samsung;
        this->_number_of_sony = sony;
        this->_number_of_huawei = huawei;
        this->_number_of_nokia = nokia;
        this->_number_of_alcatel = alcatel;

    };
    TALLAGH_W(std::shared_ptr<WAREHOUSE> obj, int _version, int _unique_id):
WAREHOUSE(_version, _unique_id){
        /*
         * copy over values
         */
        this->_shop_address = obj->GetShop_address();
        this->_shop_name = obj->GetShop_name();
        this->_number_of_iphones = obj->GetNumber_of_iphones();
        this->_number_of_samsung = obj->GetNumber_of_samsung();
        this->_number_of_sony = obj->GetNumber_of_sony();
        this->_number_of_huawei = obj->GetNumber_of_huawei();
        this->_number_of_nokia = obj->GetNumber_of_nokia();
        this->_number_of_alcatel = obj->GetNumber_of_alcatel();
    }
    TALLAGH_W(const TALLAGH_W& orig);
    TALLAGH_W operator=(const TALLAGH_W& orig){};
    virtual ~TALLAGH_W();

    /*
     * Implement OSTM virtual methods
     */
    //virtual std::shared_ptr<TALLAGH_W> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();
    /*
     * Implement Warehouse methods
     */
    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();


private:
    std::string _shop_address;
    std::string _shop_name;
    int _number_of_iphones;
    int _number_of_samsung;
    int _number_of_sony;
    int _number_of_huawei;
    int _number_of_nokia;
    int _number_of_alcatel;

};

#endif /* TALLAGH_W_H */

```

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:



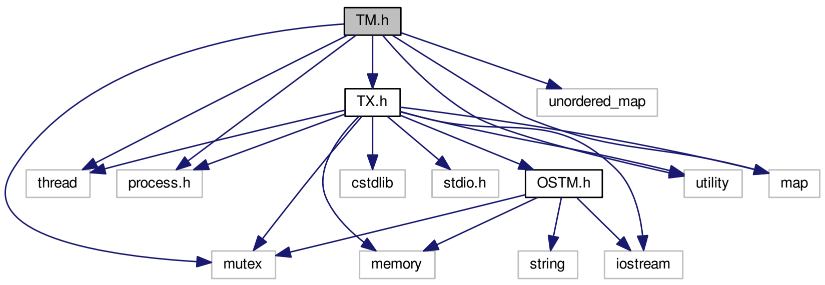
7.54 TM.cpp

```
/*
 * File:   TM.cpp
 * Author: Zoltan Fuzesi
 *
 * Created on December 18, 2017, 2:09 PM
 * Transaction Manager class methods implementation
 */
#include "TM.h"
#include <thread>
//#include <unistd.h>
#include <process.h>
#include <sys/types.h>
#include <iostream>

int TM::_tm_id;
std::map<int, std::map< std::thread::id, int >> TM::process_map_collection;
TM& TM::Instance() {
    static TM _instance;
    _instance._tm_id = _getpid();

    return _instance;
}

//TM Transaction managger checking the Process ID existence in the map
//If not in the map then register
void TM::registerTX()
{
    std::lock_guard<std::mutex> guard(register_Lock);
    int ppid = _getpid();
    std::map<int, std::map< std::thread::id, int >>::iterator process_map_collection_Iterator =
TM::process_map_collection.find(ppid);
    if (process_map_collection_Iterator == TM::process_map_collection.end()) {
        /*
         * Register main process/application to the global map
         */
        std::map< std::thread::id, int >map = get_thread_Map();
        TM::process_map_collection.insert({ppid, map});

    }
    std::map<std::thread::id, std::shared_ptr < TX>>::iterator it = txMap.find(std::this_thread::get_id());
    if (it == txMap.end()) {
        std::shared_ptr<TX> _transaction_object(new TX(std::this_thread::get_id()));
        txMap.insert({std::this_thread::get_id(), _transaction_object});
        /*
         * Get the map if registered first time
         */
        process_map_collection_Iterator = TM::process_map_collection.find(ppid);
        /*
         * Insert to the GLOBAL MAP as a helper to clean up at end of main process
         */
        process_map_collection_Iterator->second.insert({std::this_thread::get_id(), 1});

    }

}

std::shared_ptr<TX>const TM::_get_tx()
{
    std::lock_guard<std::mutex> guard(get_Lock);

    std::map<std::thread::id, std::shared_ptr<TX>>::iterator it = txMap.find(std::this_thread::get_id());
    if(it == txMap.end())
    {
       registerTX();
       it = txMap.find(std::this_thread::get_id());

    } else {
        it->second->_increase_tx_nesting();
    }
    //it = txMap.find(std::this_thread::get_id());


    return it->second;

}
void TM::_TX_EXIT(){
    TX tx(std::this_thread::get_id());
    int ppid = _getpid();
    std::map<int, std::map< std::thread::id, int >>::iterator process_map_collection_Iterator = TM::process_map_collection.find(ppid);
    if (process_map_collection_Iterator != TM::process_map_collection.end()) {

        for (auto current = process_map_collection_Iterator->second.begin(); current != process_map_collection_Iterator->second.end(); ++current) {
            /*
             * Delete all transaction associated with the actual main process
             */
            txMap.erase(current->first);
        }
        TM::process_map_collection.erase(ppid);

    }
    tx.ostm_exit();
}
void TM::print_all(){
    get_Lock.lock();
    for (auto current = txMap.begin(); current != txMap.end(); ++current) {
        std::cout << "KEY : " << current->first << std::endl;
    }
    get_Lock.unlock();
}

std::map< std::thread::id, int > TM::get_thread_Map() {
    std::map< std::thread::id, int > thread_Map;
    return thread_Map;
}
```

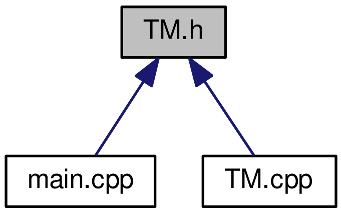
7.55 TM.h File Reference

```
#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

txMap	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

process_map_collection	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

register_Lock	std::mutex, used in the registerTX function
register_Lock	std::mutex, used in the _get_tx function
_tm_id	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

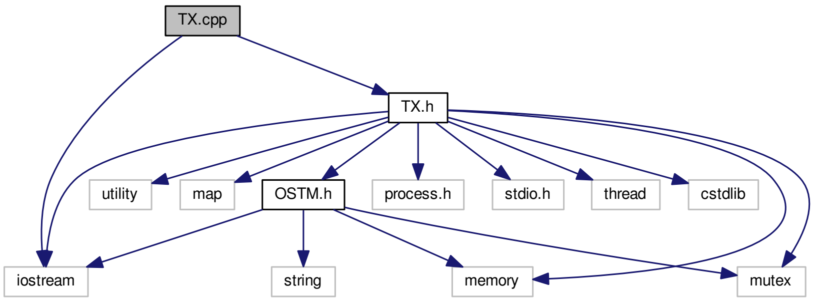
```
00001 /*
00002  * File:   TM.h
00003  * Author: Zoltan Fuzesi
00004  *
00005  * Created on December 18, 2017, 2:09 PM
00006  * Transaction Manager class fields and methods declarations
00007  */
00037 #ifndef TM_H
00038 #define TM_H
00039 
00040 #include <thread>
```

```
#include <process.h>
#include <mutex>
#include <unordered_map>
#include <utility>
#include <map>
#include "TX.h"

class __declspec(dllexport) TM {
private:
    TM() = default;
    ~TM() = default;
    TM(const TM&) = delete;
    TM& operator=(const TM&) = delete;
    std::map<std::thread::id, std::shared_ptr<TX>>txMap;
    static std::map<int, std::map< std::thread::id, int >> process_map_collection;
    std::map< std::thread::id, int > get_thread_Map();
    void registerTX();
    std::mutex register_Lock;
    std::mutex get_Lock;
    static int _tm_id;

public:

    static TM& Instance();
    std::shared_ptr<TX>const _get_tx();
    void _TX_EXIT();
    void print_all();


};


#endif // TM_H
```

7.57 TX.cpp File Reference

```
#include "TX.h"
#include <iostream>
```

Include dependency graph for TX.cpp:



7.58 TX.cpp

```
/*
 * File:   TX.cpp
 * Author: Zoltan Fuzesi
 *
 * Created on December 18, 2017, 2:09 PM
 * TX cpp file methods implementations
 */
#include "TX.h"
#include <iostream>
std::map<int, std::shared_ptr<OSTM> >TX::main_Process_Map_collection;
std::map<int, std::map< int, int >> TX::process_map_collection;
std::mutex TX::register_Lock;
```

```
int TX::test_counter = 0;
TX::TX(std::thread::id id) {
    this->transaction_Number = id;
    this->_tx_nesting_level = 0;
}
TX::~TX() {

}
TX::TX(const TX& orig) {

}

void TX::th_exit() {

    if (this->_tx_nesting_level > 0) {
        /*
         * Active nested transactions running in background, do not delete anything yet
         */
    } else {
        /*
         * Remove all elements map entries from transaction and clear the map
         */
        working_Map_collection.clear();
    }
}

void TX::ostm_exit() {
    std::map<int, std::shared_ptr<OSTM>>::iterator main_Process_Map_collection_Iterator;

    int ppid = _getpid();
    std::map<int, std::map< int, int >>::iterator process_map_collection_Iterator =
TX::process_map_collection.find(ppid);
    if (process_map_collection_Iterator != TX::process_map_collection.end()) {

        for (auto current = process_map_collection_Iterator->second.begin(); current !=
process_map_collection_Iterator->second.end(); ++current) {
            main_Process_Map_collection_Iterator = TX::main_Process_Map_collection.find(current->first);

            if (main_Process_Map_collection_Iterator != TX::main_Process_Map_collection.end()){
                /*
                 * Delete element from shared main_Process_Map_collection by object unique key value,
 shared_ptr will destroy automatically
                 */
                TX::main_Process_Map_collection.erase(main_Process_Map_collection_Iterator->first);
            }
        }
        /*
         * Delete from Process_map_collection, Main process exits delete association with library
         */
        TX::process_map_collection.erase(process_map_collection_Iterator->first);
    }
}

void TX::_register(std::shared_ptr<OSTM> object) {
    /*
     * MUST USE SHARED LOCK TO PROTECT SHARED GLOBAL MAP/COLLECTION
     */
    std::lock_guard<std::mutex> guard(TX::register_Lock);

    /*
     * Check for null pointer !
     * Null pointer can cause segmentation fault!!!
     */
    if(object == nullptr){
        throw std::runtime_error(std::string("[RUNTIME ERROR : NULL POINTER IN REGISTER FUNCTION]") );
    }

    int ppid = _getpid();
    std::map<int, std::map< int, int >>::iterator process_map_collection_Iterator =
TX::process_map_collection.find(ppid);
    if (process_map_collection_Iterator == TX::process_map_collection.end()) {
        /*
         * Register main process/application to the global map
         */
        std::map< int, int >map =  get_thread_Map();
        TX::process_map_collection.insert({ppid, map});
        /*
         * Get the map if registered first time
         */
        process_map_collection_Iterator = TX::process_map_collection.find(ppid);
    }
    std::map<int, std::shared_ptr<OSTM>>::iterator main_Process_Map_collection_Iterator =
TX::main_Process_Map_collection.find(object->Get_Unique_ID());
    if (main_Process_Map_collection_Iterator == TX::main_Process_Map_collection.end()) {
        /*
         * Insert to the GLOBAL MAP
         */
        TX::main_Process_Map_collection.insert({object->Get_Unique_ID(), object});
        /*
         * Insert to the GLOBAL MAP as a helper to clean up at end of main process
         */
        process_map_collection_Iterator->second.insert({object->Get_Unique_ID(), 1});
    }


    std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator = working_Map_collection.find(object->Get_Unique_ID());
    if (working_Map_collection_Object_Shared_Pointer_Iterator == working_Map_collection.end()) {

        working_Map_collection.insert({object->Get_Unique_ID(), object->getBaseCopy(object)});
    }

}
std::shared_ptr<OSTM> TX::load(std::shared_ptr<OSTM> object) {

    std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator;
    /*
     * Check for null pointer !
     * Null pointer can cause segmentation fault!!!
     */
    if(object == nullptr){
        throw std::runtime_error(std::string("[RUNTIME ERROR : NULL POINTER IN LOAD FUNCTION]") );
    }

        working_Map_collection_Object_Shared_Pointer_Iterator = working_Map_collection.find(object->Get_Unique_ID());

    if (working_Map_collection_Object_Shared_Pointer_Iterator != working_Map_collection.end()) {

        return working_Map_collection_Object_Shared_Pointer_Iterator->second->getBaseCopy(working_Map_collection_Object_Shared_Pointer_Iterator->second);

    } else { throw std::runtime_error(std::string("[RUNTIME ERROR : NO OBJECT FOUND LOAD FUNCTION]") );}
}
void TX::store(std::shared_ptr<OSTM> object) {
    /*
     * Check for null pointer !
     * Null pointer can cause segmentation fault!!!
     */
    if(object == nullptr){
        throw std::runtime_error(std::string("[RUNTIME ERROR : NULL POINTER IN STORE FUNCTION]") );
    }

    std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator;

    working_Map_collection_Object_Shared_Pointer_Iterator = working_Map_collection.find(object->Get_Unique_ID());
    if (working_Map_collection_Object_Shared_Pointer_Iterator != working_Map_collection.end()) {

        working_Map_collection_Object_Shared_Pointer_Iterator->second = object;

    } else { std::cout << "[ERROR STORE]" << std::endl; }
}
bool TX::commit() {

    bool can_Commit = true;

    /*
     * Dealing with nested transactions first
     */
    if (this->_tx_nesting_level > 0) {
        _decrease_tx_nesting();
        return true;
    }

    std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator;

    std::map<int, std::shared_ptr<OSTM>>::iterator main_Process_Map_collection_Iterator;
    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++) {

            main_Process_Map_collection_Iterator = TX::main_Process_Map_collection.find(working_Map_collection_Object_Shared_Pointer_Iterator->second->Get_Unique_ID());
            /*
             * Throws runtime error if object can not find
             */
            if(main_Process_Map_collection_Iterator == TX::main_Process_Map_collection.end())
            {
                throw std::runtime_error(std::string("[RUNTIME ERROR : CAN'T FIND OBJECT COMMIT FUNCTION]") );
            }

        /*
```

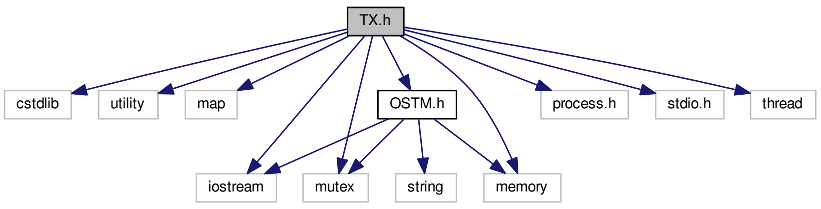
```
         * Busy wait WHILE object locked by other thread
         */
        while(!(main_Process_Map_collection_Iterator->second)->is_Locked());

        if (main_Process_Map_collection_Iterator->second->Get_Version() > working_Map_collection_Object_Shared_Pointer_Iterator->second->Get_Version()) {

            working_Map_collection_Object_Shared_Pointer_Iterator->second->Set_Can_Commit(false);
            can_Commit = false;
            break;
        } else {

            working_Map_collection_Object_Shared_Pointer_Iterator->second->Set_Can_Commit(true);
        }
    }
    if (!can_Commit) {
        TX::test_counter += 1;
        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++) {

            main_Process_Map_collection_Iterator  = TX::main_Process_Map_collection.find( working_Map_collection_Object_Shared_Pointer_Iterator->second->Get_Unique_ID());
            (working_Map_collection_Object_Shared_Pointer_Iterator->second)->copy( working_Map_collection_Object_Shared_Pointer_Iterator->second, main_Process_Map_collection_Iterator->second);

        }

        _release_object_lock();

        return false;
    } else {
        /*
         * Commit changes
         */
        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++) {

                main_Process_Map_collection_Iterator = TX::main_Process_Map_collection.find(( working_Map_collection_Object_Shared_Pointer_Iterator->second)->Get_Unique_ID());
                if (main_Process_Map_collection_Iterator != TX::main_Process_Map_collection.end()) {

                    (main_Process_Map_collection_Iterator->second)->copy( main_Process_Map_collection_Iterator->second, working_Map_collection_Object_Shared_Pointer_Iterator->second);
                    main_Process_Map_collection_Iterator->second->increase_VersionNumber();


                } else {
                    throw std::runtime_error(std::string("[RUNTIME ERROR : CAN'T FIND OBJECT COMMIT FUNCTION]"));

                }
        }


        _release_object_lock();
        this->th_exit();
        return true;
    }
}//Commit finish

void TX::_release_object_lock(){

    std::map< int, std::shared_ptr<OSTM> >::iterator working_Map_collection_Object_Shared_Pointer_Iterator;
    std::map<int, std::shared_ptr<OSTM>>::iterator main_Process_Map_collection_Iterator;
    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++) {

            main_Process_Map_collection_Iterator = TX::main_Process_Map_collection.find(( working_Map_collection_Object_Shared_Pointer_Iterator->second)->Get_Unique_ID());
            if (main_Process_Map_collection_Iterator != TX::main_Process_Map_collection.end()) {
                /*
                 * Release object lock
                 */
                (main_Process_Map_collection_Iterator)->second->unlock_Mutex();

            }
        }
}

void TX::_increase_tx_nesting() {

    this->_tx_nesting_level += 1;
    // std::cout << "[this->_tx_nesting_level] = " << this->_tx_nesting_level << std::endl;
}
```

```
void TX::_decrease_tx_nesting() {
   // std::cout << "[this->_tx_nesting_level] = " << this->_tx_nesting_level << std::endl;
    this->_tx_nesting_level -= 1;
;
}
int TX::getTest_counter() {
    return TX::test_counter;
}
const std::thread::id TX::_get_tx_number() const {
    return transaction_Number;
}
std::map< int, int > TX::get_thread_Map() {
    std::map< int, int > thread_Map;
    return thread_Map;
}

void TX::_print_all_tx() {

    std::cout << "[PRINTALLTHREAD]" << std::endl;
    std::map< int, std::shared_ptr<OSTM> >::iterator it;
    /*
     * All registered thread id in the TX global
     */
     int ppid = _getpid();
    std::map<int, std::map< int, int >>::iterator process_map_collection_Iterator =
TX::process_map_collection.find(ppid);
    if (process_map_collection_Iterator != TX::process_map_collection.end()) {

        for (auto current = process_map_collection_Iterator->second.begin(); current !=
process_map_collection_Iterator->second.end(); ++current) {
            it = working_Map_collection.find(current->first);
            if(it != working_Map_collection.end()){
                std::cout << "[Unique number ] : " <<it->second->Get_Unique_ID() << std::endl;
            }


        }

    }
}
```

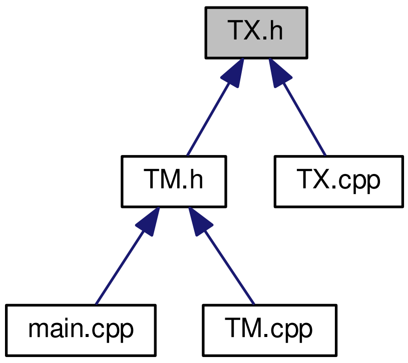
7.59 TX.h File Reference

```
#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

test_counter	int ONLY FOR TESTING!!!

MAP Collection to store OSTM∗ parent based pointers to make invisible changes during isolated transaction

Parameters

working_Map_collection	std::map

Returning the transaction number

Parameters

transaction_Number	std::thread::id NOT USED YET
_tx_nesting_level	int

STATIC GLOBAL MAP Collection to store OSTM∗ parent based pointers to control/lock and compare objects version number within transactions

Parameters

main_Process_Map_collection	std::map

STATIC GLOBAL MAP Collection to store all process associated keys to find when deleting transactions

Parameters

process_map_collection	std::map

get_thread_Map returning and map to insert to the process_map_collection as an inner value

Parameters

register_Lock	std::mutex to control shared access on MAIN MAP

_get_tx_number returning the transaction uniqe 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.

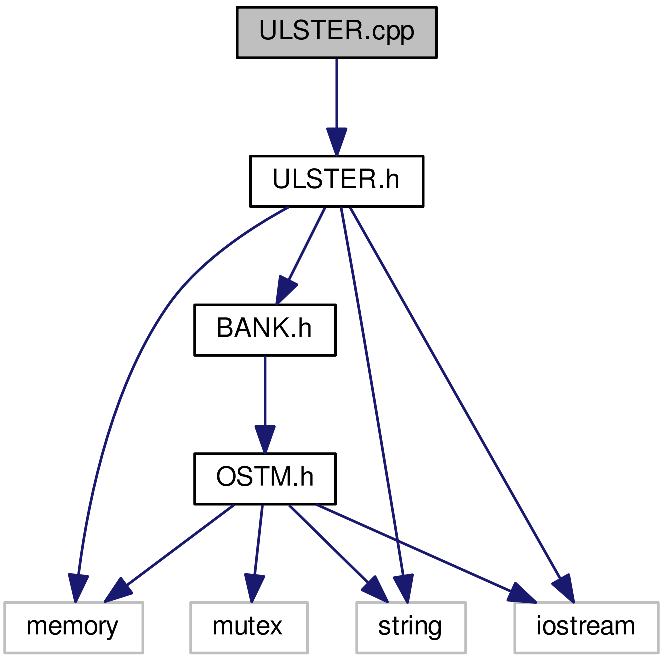
```
                                {
public:
    TX(std::thread::id id);
    ~TX();
    TX(const TX& orig);
    void ostm_exit();

    void _register(std::shared_ptr<OSTM>  object);
    std::shared_ptr<OSTM>  load(std::shared_ptr<OSTM>  object);
    void store(std::shared_ptr<OSTM>  object);
    bool commit();
    void _increase_tx_nesting();
    void _decrease_tx_nesting();
    friend class TM;
    /*
     * \brief ONLY FOR TESTING!!! returning the number of rollback happened during transactions
     */
    int getTest_counter();
    static int test_counter;
    /*
     * TESTING ONLY
     */
    void _print_all_tx() ;


private:
    std::map< int, std::shared_ptr<OSTM>  > working_Map_collection;
    std::thread::id transaction_Number;
    int _tx_nesting_level;

    static std::map<int, std::shared_ptr<OSTM>  >main_Process_Map_collection;
    static std::map<int, std::map< int, int >> process_map_collection;
    //static std::map<pid_t, std::map< int, std::pair<ppid, int>  >> process_map_collection;
    std::map< int , int > get_thread_Map();
    static std::mutex register_Lock;
    const std::thread::id _get_tx_number() const;

    void _release_object_lock();
    void th_exit();



};
```

7.60 TX.h

```
/*
 * File:   TX.h
 * Author: Zoltan Fuzesi
 *
 * Created on December 18, 2017, 2:09 PM
 * Transaction class fields and methods declarations
 */

#ifndef TX_H
#define TX_H
#include <cstdlib>
#include <utility>
#include <map>
#include <iostream>
#include <mutex>
#include <process.h>
#include <memory>
#include <stdio.h>
#include <thread>
#include "OSTM.h"

class TM;

class __declspec(dllexport) TX {
public:
    TX(std::thread::id id);
    ~TX();
    TX(const TX& orig);
    void ostm_exit();

    void _register(std::shared_ptr<OSTM>  object);
    std::shared_ptr<OSTM>  load(std::shared_ptr<OSTM>  object);
    void store(std::shared_ptr<OSTM>  object);
    bool commit();
    void _increase_tx_nesting();
    void _decrease_tx_nesting();
    friend class TM;
    /*
     * \brief ONLY FOR TESTING!!! returning the number of rollback happened during transactions
     */
    int getTest_counter();
    static int test_counter;
    /*
     * TESTING ONLY
     */
    void _print_all_tx() ;


private:
    std::map< int, std::shared_ptr<OSTM>  > working_Map_collection;
    std::thread::id transaction_Number;
    int _tx_nesting_level;

    static std::map<int, std::shared_ptr<OSTM>  >main_Process_Map_collection;
    static std::map<int, std::map< int, int >> process_map_collection;
    //static std::map<pid_t, std::map< int, std::pair<ppid, int>  >> process_map_collection;
    std::map< int , int > get_thread_Map();
    static std::mutex register_Lock;
    const std::thread::id _get_tx_number() const;

    void _release_object_lock();
    void th_exit();



};
#endif // _TX_H_
```

7.61 ULSTER.cpp File Reference

`#include "ULSTER.h"`
Include dependency graph for ULSTER.cpp:



7.62 ULSTER.cpp

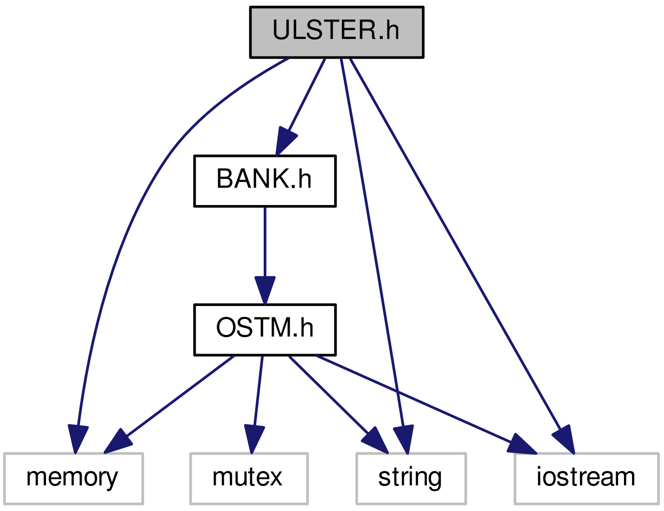
```

/*
 * File:   ULSTER.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "ULSTER.h"

//ULSTER::ULSTER() {
```

```
//}

ULSTER::ULSTER(const ULSTER& orig) {
}

ULSTER::~ULSTER() {
}
std::shared_ptr<OSTM> ULSTER::getBaseCopy(std::shared_ptr<OSTM> object)
{
    std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
    std::shared_ptr<BANK> obj(new ULSTER(objTO,object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

    return ostm_obj;
}
void ULSTER::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<ULSTER> objTO = std::dynamic_pointer_cast<ULSTER>(to);
    std::shared_ptr<ULSTER> objFROM = std::dynamic_pointer_cast<ULSTER>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());


}
//std::shared_ptr<ULSTER> ULSTER::_cast(std::shared_ptr<OSTM> _object){
//
//    return static_cast<std::shared_ptr<ULSTER>>(_object);
//}
void ULSTER::toString()
{
   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;
}

void ULSTER::SetAddress(std::string address) {
    this->address = address;
}

std::string ULSTER::GetAddress() const {
    return address;
}

void ULSTER::SetBalance(double balance) {
    this->balance = balance;
}

double ULSTER::GetBalance() const {
    return balance;
}

void ULSTER::SetAccountNumber(int accountNumber) {
    this->accountNumber = accountNumber;
}

int ULSTER::GetAccountNumber() const {
    return accountNumber;
}

void ULSTER::SetLastName(std::string lastName) {
    this->lastName = lastName;
}

std::string ULSTER::GetLastName() const {
    return lastName;
}

void ULSTER::SetFirstName(std::string firstName) {
    this->firstName = firstName;
}

std::string ULSTER::GetFirstName() const {
    return firstName;
}

void ULSTER::SetFullname(std::string fullname) {
    this->fullname = fullname;
}

std::string ULSTER::GetFullname() const {
    return fullname;
}

```

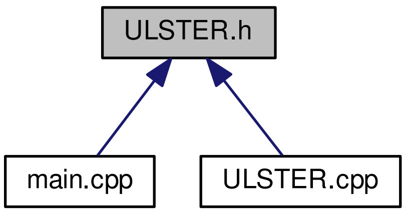
7.63 ULSTER.h File Reference

```
#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

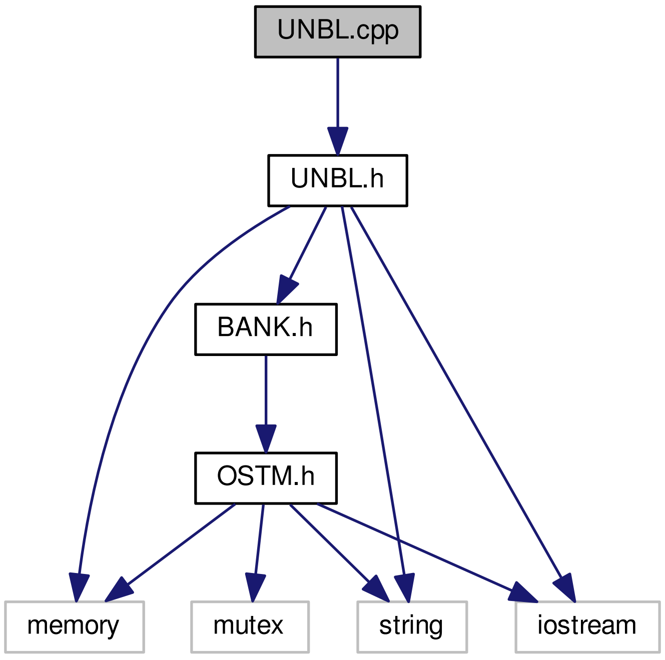
7.64 ULSTER.h

```cpp

/*
 * File:   ULSTER.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef ULSTER_H
#define ULSTER_H
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>
class ULSTER : public BANK {
public:
    ULSTER() : BANK() {
        this->accountNumber = 0;
        this->balance = 50;
        this->firstName = "Joe";
        this->lastName = "Blog";
        this->address = "High street, Carlow";
        this->fullname = firstName + " " + lastName;
    };
    ULSTER(int accountNumber, double balance, std::string firstName, std::string lastName,
std::string address) : BANK() {
        this->accountNumber = accountNumber;
        this->balance = balance;
        this->firstName = firstName;
        this->lastName = lastName;
        this->address = address;
        this->fullname = firstName + " " + lastName;
    };
    ULSTER(std::shared_ptr<BANK> obj, int _version, int _unique_id) : BANK(_version, _unique_id)
{

        this->accountNumber = obj->GetAccountNumber();
        this->balance = obj->GetBalance();
        this->firstName = obj->GetFirstName();
        this->lastName = obj->GetLastName();
        this->address = obj->GetAddress();
        this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
    };
    ULSTER(const ULSTER& orig);
    ULSTER operator=(const ULSTER& orig) {};
    virtual ~ULSTER();

    /*
     * Implement OSTM virtual methods
     */
    //virtual std::shared_ptr<ULSTER> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();

    /*
     * Implement BANK virtual methods
     */
    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;
private:
    std::string fullname;
    std::string firstName;
    std::string lastName;
    int accountNumber;
    double balance;
    std::string address;

};

#endif /* ULSTER_H */

```

7.65 UNBL.cpp File Reference

`#include "UNBL.h"`

Include dependency graph for UNBL.cpp:



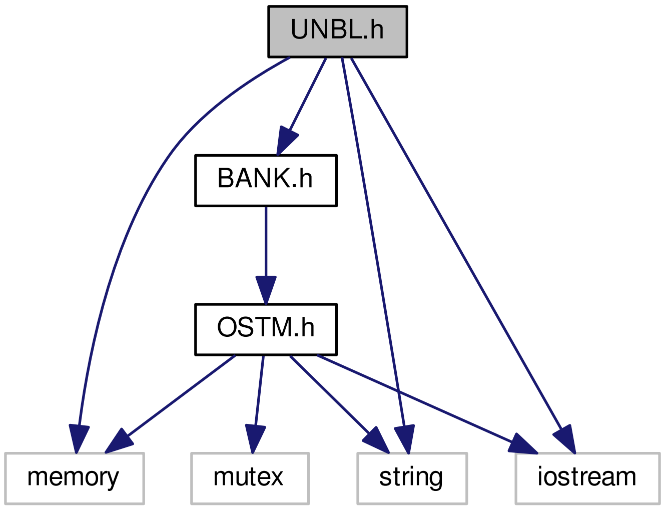
7.66 UNBL.cpp

```
/*
 * File:   UNBL.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "UNBL.h"

UNBL::UNBL(const UNBL& orig) {
}

UNBL::~UNBL() {
}
std::shared_ptr<OSTM> UNBL::getBaseCopy(std::shared_ptr<OSTM> object)
{
    std::shared_ptr<BANK> objTO = std::dynamic_pointer_cast<BANK>(object);
    std::shared_ptr<BANK> obj(new UNBL(objTO,object->Get_Version(),object->Get_Unique_ID()));
    std::shared_ptr<OSTM> ostm_obj = std::dynamic_pointer_cast<OSTM>(obj);

    return ostm_obj;
}
void UNBL::copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from){

    std::shared_ptr<UNBL> objTO = std::dynamic_pointer_cast<UNBL>(to);
    std::shared_ptr<UNBL> objFROM = std::dynamic_pointer_cast<UNBL>(from);
    objTO->Set_Unique_ID(objFROM->Get_Unique_ID());
    objTO->Set_Version(objFROM->Get_Version());
    objTO->SetAccountNumber(objFROM->GetAccountNumber());
    objTO->SetBalance(objFROM->GetBalance());

}
//std::shared_ptr<UNBL> UNBL::_cast(std::shared_ptr<OSTM> _object){
//
//    return static_cast<std::shared_ptr<UNBL>>(_object);
//}
void UNBL::toString()
{
   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;
}

void UNBL::SetAddress(std::string address) {
    this->address = address;
}

std::string UNBL::GetAddress() const {
    return address;
}

void UNBL::SetBalance(double balance) {
    this->balance = balance;
}

double UNBL::GetBalance() const {
    return balance;
}

void UNBL::SetAccountNumber(int accountNumber) {
    this->accountNumber = accountNumber;
}

int UNBL::GetAccountNumber() const {
    return accountNumber;
}

void UNBL::SetLastName(std::string lastName) {
    this->lastName = lastName;
}

std::string UNBL::GetLastName() const {
    return lastName;
}

void UNBL::SetFirstName(std::string firstName) {
    this->firstName = firstName;
}

std::string UNBL::GetFirstName() const {
    return firstName;
}

void UNBL::SetFullname(std::string fullname) {
    this->fullname = fullname;
}

std::string UNBL::GetFullname() const {
    return fullname;
}

```

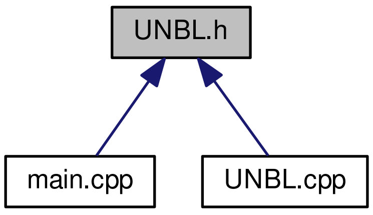
7.67 UNBL.h File Reference

```
#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

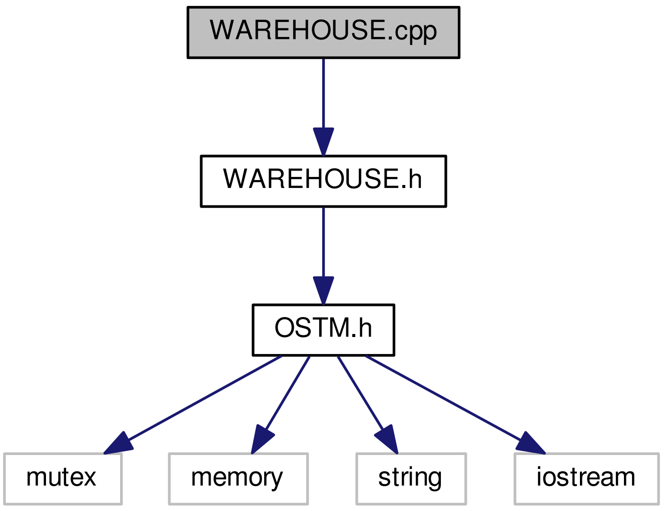
7.68 UNBL.h

```

/*
 * File:   UNBL.h
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef UNBL_H
#define UNBL_H
#include "BANK.h"
#include <string>
#include <memory>
#include <iostream>
class UNBL : public BANK {
public:
    UNBL() : BANK() {
        this->accountNumber = 0;
        this->balance = 50;
        this->firstName = "Joe";
        this->lastName = "Blog";
        this->address = "High street, Carlow";
        this->fullname = firstName + " " + lastName;
    };
    UNBL(int accountNumber, double balance, std::string firstName, std::string lastName, std::string
address) : BANK() {
        this->accountNumber = accountNumber;
        this->balance = balance;
        this->firstName = firstName;
        this->lastName = lastName;
        this->address = address;
        this->fullname = firstName + " " + lastName;
    };
    UNBL(std::shared_ptr<BANK> obj, int _version, int _unique_id) : BANK(_version, _unique_id) {

        this->accountNumber = obj->GetAccountNumber();
        this->balance = obj->GetBalance();
        this->firstName = obj->GetFirstName();
        this->lastName = obj->GetLastName();
        this->address = obj->GetAddress();
        this->fullname = obj->GetFirstName() + " " + obj->GetLastName();
    };
    UNBL(const UNBL& orig);
    UNBL operator=(const UNBL& orig) {};
    virtual ~UNBL();

    /*
     * Implement OSTM virtual methods
     */
    //virtual std::shared_ptr<UNBL> _cast(std::shared_ptr<OSTM> _object);
    virtual void copy(std::shared_ptr<OSTM> to, std::shared_ptr<OSTM> from);
    virtual std::shared_ptr<OSTM> getBaseCopy(std::shared_ptr<OSTM> object);
    virtual void toString();

    /*
     * Implement BANK virtual methods
     */
    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;
private:
    std::string fullname;
    std::string firstName;
    std::string lastName;
    int accountNumber;
    double balance;
    std::string address;

};

#endif /* UNBL_H */

```

7.69 WAREHOUSE.cpp File Reference

```
#include "WAREHOUSE.h"
```

Include dependency graph for WAREHOUSE.cpp:



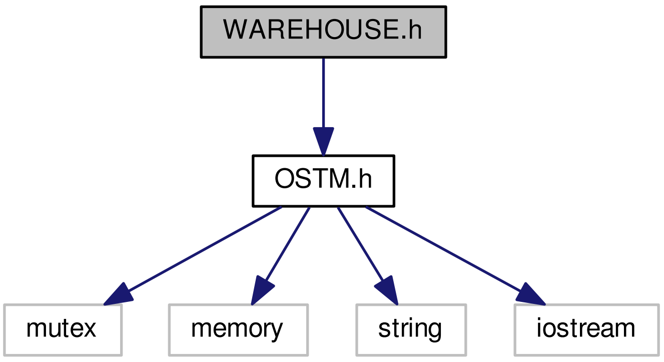
7.70 WAREHOUSE.cpp

```

/*
 * File:   WAREHOUSE.cpp
 * Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#include "WAREHOUSE.h"

WAREHOUSE::WAREHOUSE(const WAREHOUSE& orig) {
}

WAREHOUSE::~WAREHOUSE() {
}

```

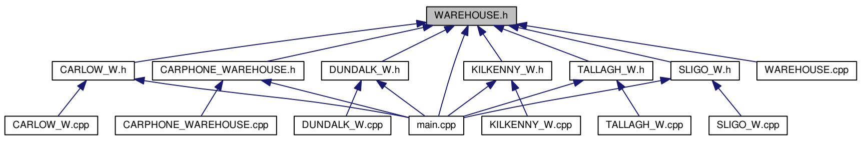
7.71 WAREHOUSE.h File Reference

```
#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

```
/*
 * File:   WAREHOUSE.h
* Author: Zoltan Fuzesi
 * IT Carlow : C00197361
 *
 * Created on January 17, 2018, 8:02 PM
 */

#ifndef WAREHOUSE_H
#define WAREHOUSE_H
#include "OSTM.h"
class WAREHOUSE : public OSTM {
public:
    WAREHOUSE():OSTM(){

    };
    WAREHOUSE(int _version, int _unique_id) : OSTM(_version, _unique_id){

    };
    WAREHOUSE(const WAREHOUSE& orig);
    virtual ~WAREHOUSE();

    /*
```

```
     * WAREHOUSE BASE METHODS
     */

    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;

private:

};

#endif /* WAREHOUSE_H */

```