

StockMarket

C++ Console Application

Developed by André Cruz, Edgar Carneiro and João Conde

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	BuyOrder Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Constructor & Destructor Documentation	6
3.1.2.1	BuyOrder() [1/2]	6
3.1.2.2	BuyOrder() [2/2]	6
3.1.3	Member Function Documentation	6
3.1.3.1	getClientNIF()	6
3.1.3.2	operator>()	7
3.1.3.3	saveChanges()	7
3.1.4	Member Data Documentation	7
3.1.4.1	buyerNIF	7
3.2	Client Class Reference	8
3.2.1	Detailed Description	8
3.2.2	Constructor & Destructor Documentation	8
3.2.2.1	Client() [1/3]	8
3.2.2.2	Client() [2/3]	8
3.2.2.3	Client() [3/3]	9

3.2.3	Member Function Documentation	9
3.2.3.1	getName()	9
3.2.3.2	getNIF()	9
3.2.3.3	saveChanges()	9
3.2.4	Member Data Documentation	10
3.2.4.1	name	10
3.2.4.2	nif	10
3.3	Company Class Reference	10
3.3.1	Constructor & Destructor Documentation	11
3.3.1.1	Company() [1/3]	11
3.3.1.2	Company() [2/3]	11
3.3.1.3	Company() [3/3]	11
3.3.2	Member Function Documentation	12
3.3.2.1	getArea()	12
3.3.2.2	getName()	12
3.3.2.3	getValue()	12
3.3.2.4	saveChanges()	12
3.3.2.5	setValue()	12
3.3.3	Friends And Related Function Documentation	12
3.3.3.1	operator<	13
3.3.3.2	operator<<	13
3.3.4	Member Data Documentation	13
3.3.4.1	business_area	13
3.3.4.2	max_transaction_value	14
3.3.4.3	name	14
3.3.4.4	NIF	14
3.4	Date Class Reference	14
3.4.1	Detailed Description	15
3.4.2	Constructor & Destructor Documentation	15
3.4.2.1	Date() [1/3]	15

3.4.2.2	Date() [2/3]	15
3.4.2.3	Date() [3/3]	15
3.4.3	Member Function Documentation	15
3.4.3.1	get_day()	16
3.4.3.2	get_month()	16
3.4.3.3	get_year()	16
3.4.4	Friends And Related Function Documentation	16
3.4.4.1	operator<	16
3.4.4.2	operator<<	17
3.4.4.3	operator<=	17
3.4.4.4	operator==	18
3.4.4.5	operator>>	18
3.4.5	Member Data Documentation	18
3.4.5.1	day	18
3.4.5.2	month	19
3.4.5.3	year	19
3.5	Client::InvalidNIF Class Reference	19
3.5.1	Detailed Description	19
3.5.2	Constructor & Destructor Documentation	19
3.5.2.1	InvalidNIF()	19
3.5.3	Member Function Documentation	20
3.5.3.1	getNIF()	20
3.6	Order::InvalidValue Class Reference	20
3.6.1	Detailed Description	20
3.6.2	Constructor & Destructor Documentation	20
3.6.2.1	InvalidValue()	20
3.6.3	Member Data Documentation	21
3.6.3.1	value	21
3.7	Investor Class Reference	21
3.7.1	Constructor & Destructor Documentation	22

3.7.1.1	Investor() [1/3]	22
3.7.1.2	Investor() [2/3]	22
3.7.1.3	Investor() [3/3]	22
3.7.2	Member Function Documentation	22
3.7.2.1	addBudget()	22
3.7.2.2	debitInvest()	23
3.7.2.3	getBudget()	23
3.7.2.4	getMaxInv()	23
3.7.2.5	getPhoneNumber()	24
3.7.2.6	updatePhoneN()	24
3.7.3	Friends And Related Function Documentation	24
3.7.3.1	operator<	24
3.7.3.2	operator<<	25
3.7.3.3	operator==	25
3.7.4	Member Data Documentation	25
3.7.4.1	availableBudget	25
3.7.4.2	maxInvestment	26
3.7.4.3	name	26
3.7.4.4	phone	26
3.8	investorPtrHash Struct Reference	26
3.8.1	Detailed Description	26
3.8.2	Member Function Documentation	26
3.8.2.1	operator>() [1/2]	26
3.8.2.2	operator>() [2/2]	27
3.9	Market Class Reference	27
3.9.1	Detailed Description	29
3.9.2	Constructor & Destructor Documentation	29
3.9.2.1	Market()	29
3.9.2.2	~Market()	29
3.9.3	Member Function Documentation	29

3.9.3.1	changeCompany()	29
3.9.3.2	changeInvestorContact()	29
3.9.3.3	clientHistory()	29
3.9.3.4	deleteCompany()	30
3.9.3.5	eraseClientOrder()	30
3.9.3.6	getCurrentNIF()	30
3.9.3.7	insertCompany()	31
3.9.3.8	instance()	31
3.9.3.9	listBuyOrders()	31
3.9.3.10	listCompanies() [1/2]	31
3.9.3.11	listCompanies() [2/2]	31
3.9.3.12	listInactiveInvestors()	31
3.9.3.13	listInvestors()	32
3.9.3.14	listInvestorsB()	32
3.9.3.15	listInvestorsI()	32
3.9.3.16	listSellOrders()	32
3.9.3.17	placeOrder()	32
3.9.3.18	printTransactions() [1/4]	32
3.9.3.19	printTransactions() [2/4]	33
3.9.3.20	printTransactions() [3/4]	33
3.9.3.21	printTransactions() [4/4]	33
3.9.3.22	recreditInvestor()	33
3.9.3.23	requestInvestement()	34
3.9.3.24	saveChanges()	34
3.9.3.25	showClientHistory()	34
3.9.3.26	showClientInfo()	34
3.9.3.27	showClientOrders()	34
3.9.3.28	signIn()	34
3.9.3.29	signOut()	35
3.9.3.30	signUp()	35

3.9.4	Friends And Related Function Documentation	35
3.9.4.1	operator<<	35
3.9.5	Member Data Documentation	36
3.9.5.1	clients	36
3.9.5.2	clientsChanged	36
3.9.5.3	clientsFile	36
3.9.5.4	companys	36
3.9.5.5	companysChanged	36
3.9.5.6	companysFile	36
3.9.5.7	currentNIF	37
3.9.5.8	inactive_investors	37
3.9.5.9	investors	37
3.9.5.10	investorsChanged	37
3.9.5.11	investorsFile	37
3.9.5.12	ordersChanged	37
3.9.5.13	ordersFile	37
3.9.5.14	singleton_instance	37
3.9.5.15	transactions	38
3.9.5.16	transactionsChanged	38
3.9.5.17	transactionsFile	38
3.9.5.18	unfulfilled_orders	38
3.10	Order Class Reference	38
3.10.1	Detailed Description	39
3.10.2	Constructor & Destructor Documentation	39
3.10.2.1	Order() [1/2]	39
3.10.2.2	Order() [2/2]	39
3.10.2.3	~Order()	40
3.10.3	Member Function Documentation	40
3.10.3.1	getClientNIF()	40
3.10.3.2	getDatePlaced()	40

3.10.3.3	getQuantity()	41
3.10.3.4	getStock()	41
3.10.3.5	getValue()	41
3.10.3.6	operator()()	41
3.10.3.7	printInfo()	42
3.10.3.8	saveChanges()	42
3.10.4	Member Data Documentation	42
3.10.4.1	datePlaced	42
3.10.4.2	quantity	42
3.10.4.3	stock	42
3.10.4.4	valuePerStock	43
3.11	SellOrder Class Reference	43
3.11.1	Detailed Description	43
3.11.2	Constructor & Destructor Documentation	43
3.11.2.1	SellOrder() [1/2]	43
3.11.2.2	SellOrder() [2/2]	44
3.11.3	Member Function Documentation	44
3.11.3.1	getClientNIF()	44
3.11.3.2	operator()()	44
3.11.3.3	saveChanges()	45
3.11.4	Member Data Documentation	45
3.11.4.1	sellerNIF	45
3.12	Transaction Class Reference	45
3.12.1	Detailed Description	46
3.12.2	Constructor & Destructor Documentation	46
3.12.2.1	Transaction() [1/3]	46
3.12.2.2	Transaction() [2/3]	46
3.12.2.3	Transaction() [3/3]	47
3.12.3	Member Function Documentation	47
3.12.3.1	getBuyerNIF()	47

3.12.3.2	getDate()	47
3.12.3.3	getQuantity()	48
3.12.3.4	getSellerNIF()	48
3.12.3.5	getStock()	48
3.12.3.6	getValue()	48
3.12.3.7	saveChanges()	48
3.12.4	Friends And Related Function Documentation	49
3.12.4.1	operator<<	49
3.12.5	Member Data Documentation	49
3.12.5.1	buyerNIF	49
3.12.5.2	quantity	49
3.12.5.3	sellerNIF	49
3.12.5.4	stock	50
3.12.5.5	time_stamp	50
3.12.5.6	value	50
Index		51

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Client	8
Company	10
Date	14
Client::InvalidNIF	19
Order::InvalidValue	20
Investor	21
investorPtrHash	26
Market	27
Order	38
BuyOrder	5
SellOrder	43
Transaction	45

Chapter 2

Class Index

2.1 Class List

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

BuyOrder	5
Client	8
Company	10
Date	14
Client::InvalidNIF	19
Order::InvalidValue	20
Investor	21
investorPtrHash	26
Market	27
Order	38
SellOrder	43
Transaction	45

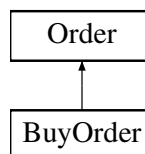
Chapter 3

Class Documentation

3.1 BuyOrder Class Reference

```
#include <Order.h>
```

Inheritance diagram for BuyOrder:



Public Member Functions

- [BuyOrder](#) (ifstream &in)
///
- [BuyOrder](#) (string [stock](#), double val, unsigned [quantity](#), nif_t [buyerNIF](#))
- nif_t [getClientNIF](#) () const
- [Transaction](#) * [operator\(\)](#) ([Order](#) *o)
- void [saveChanges](#) (ofstream &out) const

Private Attributes

- friend **SellOrder**
- nif_t [buyerNIF](#)

Additional Inherited Members

3.1.1 Detailed Description

Class used to represent a buy order. Derives from [Order](#).

3.1.2 Constructor & Destructor Documentation

3.1.2.1 `BuyOrder()` [1/2]

```
BuyOrder::BuyOrder (
    ifstream & in )
```

```
///
```

A constructor. The constructor creates a [BuyOrder](#) object, reading the data from the input stream passed as argument.

Parameters

<i>in</i>	The input stream to read from in order to build the BuyOrder object.
-----------	--

3.1.2.2 `BuyOrder()` [2/2]

```
BuyOrder::BuyOrder (
    string stock,
    double val,
    unsigned quantity,
    nif_t buyerNIF )
```

A constructor. The constructor creates a [BuyOrder](#) object using the data passed as arguments.

Parameters

<i>stock</i>	A string with the stock name.
<i>val</i>	A double with the value per stock.
<i>quantity</i>	An unsigned with the stock quantity.
<i>buyer↔ NIF</i>	The buyer's NIF.

3.1.3 Member Function Documentation

3.1.3.1 `getClientNIF()`

```
nif_t BuyOrder::getClientNIF ( ) const [virtual]
```

A const member function that returns the NIF of the client associated with this [BuyOrder](#).

Returns

The NIF of the Buyer associated with this [Order](#).

Implements [Order](#).

3.1.3.2 operator()

```
Transaction * BuyOrder::operator() (
    Order * o ) [virtual]
```

Overload of operator() for class [Order](#).

Parameters

<i>o</i>	Pointer of an object Order .
----------	--

Returns

A transaction type object.

Implements [Order](#).

3.1.3.3 saveChanges()

```
void BuyOrder::saveChanges (
    ofstream & out ) const [virtual]
```

A const memeber function to save changes in an output stream.

Parameters

<i>out</i>	The outstream file to write to.
------------	---------------------------------

Reimplemented from [Order](#).

3.1.4 Member Data Documentation**3.1.4.1 buyerNIF**

```
nif_t BuyOrder::buyerNIF [private]
```

nif_t buyerNIF. The NIF of the buyer associated with this [Order](#).

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

- D:/ProjetosC++/AEDAv2/StockMarket/Order.h
- D:/ProjetosC++/AEDAv2/StockMarket/Order.cpp

3.2 Client Class Reference

```
#include <Client.h>
```

Classes

- class [InvalidNIF](#)

Public Member Functions

- [Client](#) ()=default
- [Client](#) (ifstream &in)
- [Client](#) (string [name](#), nif_t [nif](#))
- string [getName](#) () const
- nif_t [getNIF](#) () const
- void [saveChanges](#) (ofstream &out) const

Private Attributes

- string [name](#)
- nif_t [nif](#)

3.2.1 Detailed Description

A class used to represent a client. Each client object has a name and a nif (from the portuguese "Numero de Identificação Fiscal").

3.2.2 Constructor & Destructor Documentation

3.2.2.1 [Client\(\)](#) [1/3]

```
Client::Client ( ) [default]
```

A default constructor.

3.2.2.2 [Client\(\)](#) [2/3]

```
Client::Client (
    ifstream & in )
```

A constructor. The construtor creates a [Client](#) object, reading the data from the input stream passed as argument.

Parameters

<i>in</i>	The input stream to read from in order to build the client object.
-----------	--

3.2.2.3 Client() [3/3]

```
Client::Client (
    string name,
    nif_t nif )
```

A constructor. The constructor creates a client object with the data passed as arguments.

Parameters

<i>name</i>	The client's name.
<i>nif</i>	The client's NIF.

3.2.3 Member Function Documentation**3.2.3.1 getName()**

```
string Client::getName ( ) const
```

A const member function with no arguments to get the client's name.

Returns

A string, the client's name.

3.2.3.2 getNIF()

```
nif_t Client::getNIF ( ) const
```

A const member function with no arguments to get the client's NIF.

Returns

A `nif_t`, the client's NIF.

3.2.3.3 saveChanges()

```
void Client::saveChanges (
    ofstream & out ) const
```

A const member function that writes the client's info to the output stream. Generally used to save the client's attributes to a file.

Parameters

<i>out</i>	The output stream to write the client's information.
------------	--

3.2.4 Member Data Documentation

3.2.4.1 name

```
string Client::name [private]
```

string name. The client's name.

3.2.4.2 nif

```
nif_t Client::nif [private]
```

nif_t nif. The client's NIF.

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

- D:/ProjetosC++/AEDAv2/StockMarket/Client.h
- D:/ProjetosC++/AEDAv2/StockMarket/Client.cpp

3.3 Company Class Reference

Public Member Functions

- [Company](#) ()=default
- [Company](#) (ifstream &in)
- [Company](#) (string [name](#), string activity, nif_t [NIF](#), double max_transaction)
- string [getName](#) () const
- string [getArea](#) () const
- double [getValue](#) () const
- void [setValue](#) (double value)
- void [saveChanges](#) (ofstream &out) const

Private Attributes

- string [name](#)
- string [business_area](#)
- nif_t [NIF](#)
- double [max_transaction_value](#)

Friends

- ostream & operator<< (ostream &, const Company &)
- bool operator< (const Company &c1, const Company &c2)

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Company() [1/3]

```
Company::Company ( ) [default]
```

A default constructor.

3.3.1.2 Company() [2/3]

```
Company::Company (
    ifstream & in )
```

A constructor. The constructor creates a company object, reading the data from the input stream passed as argument.

Parameters

<i>in</i>	The input stream to read from in order to build the company object.
-----------	---

3.3.1.3 Company() [3/3]

```
Company::Company (
    string name,
    string activity,
    nif_t NIF,
    double max_transaction )
```

A constructor. The constructor creates a company object using the data passed as arguments.

Parameters

<i>name</i>	The company name.
<i>activity</i>	The company business area.
<i>NIF</i>	The company NIF.
<i>max_transaction</i>	The highest value transaction ever made by that company.

3.3.2 Member Function Documentation

3.3.2.1 `getArea()`

```
string Company::getArea ( ) const
```

A const member function that returns the business area of a certain company.

3.3.2.2 `getName()`

```
string Company::getName ( ) const
```

A const member function that returns the name of the company.

3.3.2.3 `getValue()`

```
double Company::getValue ( ) const
```

A const member function that returns the maximum transaction value.

3.3.2.4 `saveChanges()`

```
void Company::saveChanges (
    ofstream & out ) const
```

A const member function to write the company to a save file.

Parameters

<i>out</i>	The outputstream file to write to.
------------	------------------------------------

3.3.2.5 `setValue()`

```
void Company::setValue (
    double value )
```

A member function that changes the maximum transaction value.

3.3.3 Friends And Related Function Documentation

3.3.3.1 operator<

```
bool operator< (
    const Company & c1,
    const Company & c2 ) [friend]
```

Overload of Operator < for class [Company](#). Compares 2 company's.

Parameters

<i>c1</i>	Left side Company .
<i>c2</i>	Right side Company .

Returns

Returns true if c1 has an alphabetically smaller business area than c2. If equal returns true if c1 has an alphabetically smaller name than c2.

3.3.3.2 operator<<

```
ostream& operator<< (
    ostream & out,
    const Company & c ) [friend]
```

Overload of Operator << for class [Company](#). Prints the company in a human friendly way.

Parameters

<i>out</i>	The ostream to write to.
<i>c</i>	The company to be written.

Returns

Returns the output stream to allow chaining

3.3.4 Member Data Documentation

3.3.4.1 business_area

```
string Company::business_area [private]
```

string business_area. The company's business area.

3.3.4.2 max_transaction_value

```
double Company::max_transaction_value [private]
```

double max_transaction_value. The company's maximum transaction value up to this date.

3.3.4.3 name

```
string Company::name [private]
```

string name. The company's name.

3.3.4.4 NIF

```
nif_t Company::NIF [private]
```

nif_t. The company's NIF.

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

- D:/ProjetosC++/AEDAv2/StockMarket/Company.h
- D:/ProjetosC++/AEDAv2/StockMarket/Company.cpp

3.4 Date Class Reference

```
#include <Date.h>
```

Public Member Functions

- [Date](#) ()
- [Date](#) (string data)
- [Date](#) (int [day](#), int [month](#), int [year](#))
- int [get_day](#) () const
- int [get_month](#) () const
- int [get_year](#) () const

Private Attributes

- unsigned [day](#)
- unsigned [month](#)
- unsigned [year](#)

Friends

- ostream & [operator<<](#) (ostream &out, const [Date](#) &date)
- istream & [operator>>](#) (istream &in, [Date](#) &date)
- bool [operator<=](#) (const [Date](#) &d1, const [Date](#) &d2)
- bool [operator<](#) (const [Date](#) &d1, const [Date](#) &d2)
- bool [operator==](#) (const [Date](#) &d1, const [Date](#) &d2)

3.4.1 Detailed Description

A class used to represent a date. Each date object contains 3 integers, representing day, month and year.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Date() [1/3]

```
Date::Date ( )
```

A default constructor. The default constructor creates a date object with the current system date.

3.4.2.2 Date() [2/3]

```
Date::Date (
    string data )
```

A constructor. The constructor creates a date object with the specified date string specified as argument.

Parameters

<i>data</i>	A string with the date in the DD/MM/YY format
-------------	---

3.4.2.3 Date() [3/3]

```
Date::Date (
    int day,
    int month,
    int year )
```

A constructor. The constructor creates a date object with the specified day, month and year passed as arguments.

Parameters

<i>day</i>	A unsigned short representing the day
<i>month</i>	A unsigned short representing the month
<i>year</i>	A unsigned short representing the year

3.4.3 Member Function Documentation

3.4.3.1 `get_day()`

```
int Date::get_day ( ) const
```

A member function with no arguments to get the date's day.

Returns

An integer, the date's day

3.4.3.2 `get_month()`

```
int Date::get_month ( ) const
```

A member function with no arguments to get the date's month.

Returns

An integer, the date's month

3.4.3.3 `get_year()`

```
int Date::get_year ( ) const
```

A member function with no arguments to get the date's year.

Returns

An integer, the date's year

3.4.4 Friends And Related Function Documentation

3.4.4.1 `operator<`

```
bool operator< (
    const Date & d1,
    const Date & d2 ) [friend]
```

Overload of Operator < for class [Date](#).

Parameters

<i>d1</i>	First date
<i>d2</i>	Second date

Returns

Returns a boolean value, true if $d1 < d2$

3.4.4.2 operator<<

```
ostream& operator<< (
    ostream & out,
    const Date & date ) [friend]
```

Operator << for class [Date](#). Prints the specified [Date](#) as 2nd argument in the outstream passed as 1st argument.

Parameters

<i>out</i>	The outstream to write to.
<i>date</i>	The date to be written.

Returns

Returns the output stream to allow chaining

3.4.4.3 operator<=

```
bool operator<= (
    const Date & d1,
    const Date & d2 ) [friend]
```

Overload of Operator <= for class [Date](#).

Parameters

<i>d1</i>	First date
<i>d2</i>	Second date

Returns

Returns a boolean value, true if $d1 \leq d2$

3.4.4.4 operator==

```
bool operator== (
    const Date & d1,
    const Date & d2 ) [friend]
```

Overload of Operator == for class [Date](#).

Parameters

<i>d1</i>	First date
<i>d2</i>	Second date

Returns

Returns a boolean value, true if d1 equals d2

3.4.4.5 operator>>

```
istream& operator>> (
    istream & in,
    Date & date ) [friend]
```

Operator >> for [Date](#) class. Reads the date from the input stream to change the date object.

Parameters

<i>in</i>	Input stream where to read the date from
<i>date</i>	Date object to be changed

Returns

Returns the input stream to allow chaining

3.4.5 Member Data Documentation

3.4.5.1 day

```
unsigned Date::day [private]
```

unsigned day. Unsigned Integer representing the date day.

3.4.5.2 month

```
unsigned Date::month [private]
```

unsigned month. Unsigned Integer representing the date month.

3.4.5.3 year

```
unsigned Date::year [private]
```

unsigned year. Unsigned Integer representing the date year.

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

- D:/ProjetosC++/AEDAv2/StockMarket/Date.h
- D:/ProjetosC++/AEDAv2/StockMarket/Date.cpp

3.5 Client::InvalidNIF Class Reference

```
#include <Client.h>
```

Public Member Functions

- [InvalidNIF](#) (nif_t nif)
- nif_t [getNIF](#) () const

Private Attributes

- nif_t nif

3.5.1 Detailed Description

A class used to represent an exception. The exception object contains the invalid NIF

3.5.2 Constructor & Destructor Documentation

3.5.2.1 InvalidNIF()

```
Client::InvalidNIF::InvalidNIF (  
    nif_t nif ) [inline]
```

A constructor. The constructor creates an [InvalidNIF](#) object with the supplied NIF.

Parameters

<i>nif</i>	The nif in question.
------------	----------------------

3.5.3 Member Function Documentation

3.5.3.1 getNIF()

```
nif_t Client::InvalidNIF::getNIF ( ) const [inline]
```

A const member function with no arguments to get the object's NIF.

Returns

A `nif_t`, the NIF that originated the creation of this object.

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

- D:/ProjetosC++/AEDAv2/StockMarket/Client.h

3.6 Order::InvalidValue Class Reference

```
#include <Order.h>
```

Public Member Functions

- [InvalidValue](#) (double [value](#))
- double **getValue** () const

Private Attributes

- double [value](#)

3.6.1 Detailed Description

A class used to represent an exception, [InvalidValue](#). The object from [InvalidValue](#) class contains the invalid value stored.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 InvalidValue()

```
Order::InvalidValue::InvalidValue (
    double value ) [inline]
```

A constructor.

Parameters

<i>value</i>	The invalid value in question.
--------------	--------------------------------

3.6.3 Member Data Documentation

3.6.3.1 value

```
double Order::InvalidValue::value [private]
```

double value. The invalid value, for exception handling.

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

- D:/ProjetosC++/AEDAv2/StockMarket/Order.h

3.7 Investor Class Reference

Public Member Functions

- [Investor](#) ()=default
- [Investor](#) (ifstream &in)
- [Investor](#) (string [name](#), tlmv_t [phone](#), double maxInvest=0, double budget=0)
- double [getBudget](#) () const
- double [getMaxInv](#) () const
- tlmv_t [getPhoneNumber](#) () const
- void [debitInvest](#) (double value)
- void [addBudget](#) (double loan)
- void [updatePhoneN](#) (tlmv_t [phone](#))

Private Attributes

- string [name](#)
- tlmv_t [phone](#)
- double [maxInvestment](#)
- double [availableBudget](#)

Friends

- ostream & [operator<<](#) (ostream &out, const [Investor](#) &i)
- bool [operator==](#) (const [Investor](#) &i1, const [Investor](#) &i2)
- bool [operator<](#) (const [Investor](#) &i1, const [Investor](#) &i2)

3.7.1 Constructor & Destructor Documentation

3.7.1.1 `Investor()` [1/3]

```
Investor::Investor ( ) [default]
```

A default constructor.

3.7.1.2 `Investor()` [2/3]

```
Investor::Investor (
    ifstream & in )
```

A constructor. The constructor creates a [Investor](#) object, reading the data from the input stream passed as argument.

Parameters

<i>in</i>	The input stream to read from in order to build the Investor object.
-----------	--

3.7.1.3 `Investor()` [3/3]

```
Investor::Investor (
    string name,
    tlmv_t phone,
    double maxInvest = 0,
    double budget = 0 )
```

A constructor. The constructor creates a [Investor](#) object using the data passed as arguments.

Parameters

<i>name</i>	The Investor name.
-------------	------------------------------------

3.7.2 Member Function Documentation

3.7.2.1 `addBudget()`

```
void Investor::addBudget (
    double loan )
```

Adds capital to the investor.

Parameters

<i>loan</i>	The value of the loan to the investor.
-------------	--

3.7.2.2 debitInvest()

```
void Investor::debitInvest (
    double value )
```

Debits the investor budget with an amount invested.

Parameters

<i>value</i>	The value of the investment.
--------------	------------------------------

3.7.2.3 getBudget()

```
double Investor::getBudget ( ) const
```

A getter method returning investor's budget.

Returns

Returns the investor's budget.

3.7.2.4 getMaxInv()

```
double Investor::getMaxInv ( ) const
```

A getter method returning investor's maximum investment.

Returns

Returns the investor's maximum investment.

3.7.2.5 getPhoneNumber()

```
tlmv_t Investor::getPhoneNumber ( ) const
```

A getter method returning investor's phone number.

Returns

Returns the investor's phone number.

3.7.2.6 updatePhoneN()

```
void Investor::updatePhoneN (
    tlmv_t phone )
```

Updates the investor phone number with a new one.

Parameters

<i>phone</i>	The new investor's phone number.
--------------	----------------------------------

3.7.3 Friends And Related Function Documentation

3.7.3.1 operator<

```
bool operator< (
    const Investor & i1,
    const Investor & i2 ) [friend]
```

Overload of Operator < for class [Investor](#). Compares 2 investor's.

Parameters

<i>i1</i>	Left side Investor .
<i>i2</i>	Right side Investor .

Returns

Returns true if i1 has a greater or equal budget than i2. False otherwise.

3.7.3.2 operator<<

```
ostream& operator<< (
    ostream & out,
    const Investor & i ) [friend]
```

Overload of Operator << for class [Investor](#). Prints the investor in a human friendly way.

Parameters

<i>out</i>	The outstream to write to.
<i>i</i>	The investor to be written.

Returns

Returns the output stream to allow chaining

3.7.3.3 operator==

```
bool operator== (
    const Investor & i1,
    const Investor & i2 ) [friend]
```

Overload of Operator == for class [Investor](#). Compares 2 investor's.

Parameters

<i>i1</i>	Left side Investor .
<i>i2</i>	Right side Investor .

Returns

Returns true if i1 equals i2 meaning same name and phone number (unique qualifiers). False otherwise.

3.7.4 Member Data Documentation

3.7.4.1 availableBudget

```
double Investor::availableBudget [private]
```

double availableBudget. The investor's budget used to finance.

3.7.4.2 maxInvestment

```
double Investor::maxInvestment [private]
```

double maxInvestment. The maximum value the investors is willing to spend.

3.7.4.3 name

```
string Investor::name [private]
```

string name. The investors's name.

3.7.4.4 phone

```
tlmv_t Investor::phone [private]
```

tlmv_t phone. The investor's phone number.

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

- D:/ProjetosC++/AEDAv2/StockMarket/Investor.h
- D:/ProjetosC++/AEDAv2/StockMarket/Investor.cpp

3.8 investorPtrHash Struct Reference

```
#include <Investor.h>
```

Public Member Functions

- int [operator\(\)](#) (const [Investor](#) *i) const
- bool [operator\(\)](#) (const [Investor](#) *i1, const [Investor](#) *i2) const

3.8.1 Detailed Description

A structure to encapsulate the Hash and Comparison functions of [Investor](#) Pointers.

3.8.2 Member Function Documentation

3.8.2.1 [operator\(\)](#) [1/2]

```
int investorPtrHash::operator() (
    const Investor * i ) const [inline]
```

Hash Function for Investor*

Parameters

<i>i</i>	Pointer to an Investor object.
----------	--

Returns

hash value.

3.8.2.2 `operator()` [2/2]

```
bool investorPtrHash::operator() (
    const Investor * i1,
    const Investor * i2 ) const [inline]
```

Comparison Function for Investor*

Parameters

<i>i1</i>	Pointer to an Investor object.
<i>i2</i>	Pointer to an Investor object.

Returns

true if Investors pointed by i1 and i2 are the same, false otherwise.

The documentation for this struct was generated from the following file:

- D:/ProjetosC++/AEDAv2/StockMarket/Investor.h

3.9 Market Class Reference

```
#include <Market.h>
```

Public Member Functions

- bool [signIn](#) (string name, nif_t nif)
- void [signOut](#) ()
- bool [signUp](#) (string name, nif_t nif)
- nif_t [getCurrentNIF](#) () const
- void [showClientInfo](#) () const
- void [showClientHistory](#) () const
- void [showClientOrders](#) () const
- bool [eraseClientOrder](#) (unsigned choice)
- void [listBuyOrders](#) () const
- void [listSellOrders](#) () const

- void `listCompanyys` () const
- void `listCompanyys` (string business) const
- void `insertCompany` (Company c)
- void `deleteCompany` (string name)
- void `changeCompany` (string name, double value)
- void `listInvestors` ()
- void `listInvestorsB` (double budget)
- void `listInvestorsI` (double maxInvest)
- void `requestInvestement` (double requestValue)
- void `listInactiveInvestors` ()
- void `recreditInvestor` (double loan, Investor *investor)
- void `changeInvestorContact` (tlmv_t newPhone_n, Investor *investor)
- vector< Transaction * > `clientHistory` (Client *c) const
- void `printTransactions` () const
- void `printTransactions` (string stock) const
- void `printTransactions` (Date day1, Date day2) const
- void `printTransactions` (Date d) const
- pair< vector< Transaction * >::iterator, vector< Transaction * >::iterator > `placeOrder` (Order *o)
- void `saveChanges` () const

Static Public Member Functions

- static Market * `instance` ()

Private Member Functions

- Market ()
- ~Market ()

Private Attributes

- nif_t `currentNIF`
- map< nif_t, Client * > `clients`
- vector< Transaction * > `transactions`
- vector< Order * > `unfulfilled_orders`
- set< Company > `companyys`
- priority_queue< Investor > `investors`
- unordered_set< Investor *, investorPtrHash, investorPtrHash > `inactive_investors`
- string `clientsFile`
- string `ordersFile`
- string `transactionsFile`
- string `companyysFile`
- string `investorsFile`
- bool `clientsChanged`
- bool `transactionsChanged`
- bool `ordersChanged`
- bool `companyysChanged`
- bool `investorsChanged`

Static Private Attributes

- static Market * `singleton_instance` = NULL

Friends

- ostream & operator<< (ostream &out, const Market &m)

3.9.1 Detailed Description

Singleton Class to implement most of the logic behind the StockMarket

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Market()

```
Market::Market ( ) [private]
```

A default constructor.

3.9.2.2 ~Market()

```
Market::~~Market ( ) [private]
```

A destructor. Deletes all dynamically allocated memory.

3.9.3 Member Function Documentation

3.9.3.1 changeCompany()

```
void Market::changeCompany (
    string name,
    double value )
```

A member function that changes the maximum transaction value of a company in the BST.

3.9.3.2 changeInvestorContact()

```
void Market::changeInvestorContact (
    tlmv_t newPhone_n,
    Investor * investor )
```

A member function that changes the phone number of an investor in the inactive_investors hash table.

3.9.3.3 clientHistory()

```
vector< Transaction * > Market::clientHistory (
    Client * c ) const
```

A const member function used to get the client's history of transactions.

Parameters

<i>c</i>	A client pointer.
----------	-------------------

Returns

A vector of the client's transactions.

3.9.3.4 deleteCompany()

```
void Market::deleteCompany (
    string name )
```

A member function that deletes a company from the BST companys.

3.9.3.5 eraseClientOrder()

```
bool Market::eraseClientOrder (
    unsigned choice )
```

A member function that erases a client's unfulfilled order.

Parameters

<i>The</i>	number corresponding to the order to be erased (sorted by date placed).
------------	---

Returns

A boolean, true if deletion of the order was done successfully.

3.9.3.6 getCurrentNIF()

```
nif_t Market::getCurrentNIF ( ) const
```

A member function that returns the current user's NIF.

Returns

The current user's nif.

3.9.3.7 insertCompany()

```
void Market::insertCompany (
    Company c )
```

A member function that inserts into company BST a new company.

3.9.3.8 instance()

```
Market * Market::instance ( ) [static]
```

A member function returning the one and only instance of [Market](#) (creates it if one doesn't exist).

Returns

A pointer to the singleton instance of [Market](#).

3.9.3.9 listBuyOrders()

```
void Market::listBuyOrders ( ) const
```

A const member function that displays the buy orders.

3.9.3.10 listCompanyys() [1/2]

```
void Market::listCompanyys ( ) const
```

A const member function that displays all the companyys.

3.9.3.11 listCompanyys() [2/2]

```
void Market::listCompanyys (
    string business ) const
```

A const member function that displays all the companyys from one sector of activity.

3.9.3.12 listInactiveInvestors()

```
void Market::listInactiveInvestors ( )
```

A member function that lists all inactive investors in the inactive-investors hash table.

3.9.3.13 listInvestors()

```
void Market::listInvestors ( )
```

A member function that lists all investors in the priority queue.

3.9.3.14 listInvestorsB()

```
void Market::listInvestorsB (
    double budget )
```

A member function that lists all investors in the priority queue with equal or greater budget than specified.

3.9.3.15 listInvestorsI()

```
void Market::listInvestorsI (
    double maxInvest )
```

A member function that lists all investors in the priority queue with equal or greater maximum transaction value than specified.

3.9.3.16 listSellOrders()

```
void Market::listSellOrders ( ) const
```

A const member function that displays the sell orders.

3.9.3.17 placeOrder()

```
pair< vector< Transaction * >::iterator, vector< Transaction * >::iterator > Market::place←
Order (
    Order * o )
```

A member function that adds an order to the unfulfilledOrders vector. Can be from Sell or Buy type.

Parameters

<i>o</i>	A pointer to the order.
----------	-------------------------

3.9.3.18 printTransactions() [1/4]

```
void Market::printTransactions ( ) const
```

A const member function that prints the transactions to the COUT output stream.

3.9.3.19 printTransactions() [2/4]

```
void Market::printTransactions (
    string stock ) const
```

Overload of member function that prints the transactions of a given Stock.

Parameters

<i>stock</i>	The Stock's name.
--------------	-------------------

3.9.3.20 printTransactions() [3/4]

```
void Market::printTransactions (
    Date day1,
    Date day2 ) const
```

Overload of member function that prints the transactions between 2 days.

Parameters

<i>day1</i>	The first day of the interval.
<i>day2</i>	The last day of the interval.

3.9.3.21 printTransactions() [4/4]

```
void Market::printTransactions (
    Date d ) const
```

Overload of member function that prints the daily transactions.

Parameters

<i>d</i>	The day whose transactions will be shown.
----------	---

3.9.3.22 recreditInvestor()

```
void Market::recreditInvestor (
    double loan,
    Investor * investor )
```

A member function that gives capital to an investor making him active, withdrawing him from inactive_investors hash table and placing him in the queue investors.

3.9.3.23 requestInvestement()

```
void Market::requestInvestement (
    double requestValue )
```

A member function that requests a value from an investor to the client. Withdraws from the investor with the smallest budget that covers the client needs.

3.9.3.24 saveChanges()

```
void Market::saveChanges ( ) const
```

A const member function that saves ALL information to the files.

3.9.3.25 showClientHistory()

```
void Market::showClientHistory ( ) const
```

A const member function that displays the client's history of transactions.

3.9.3.26 showClientInfo()

```
void Market::showClientInfo ( ) const
```

A const member function that displays the client's information.

3.9.3.27 showClientOrders()

```
void Market::showClientOrders ( ) const
```

A const member function that displays the client's unfulfilled orders.

3.9.3.28 signIn()

```
bool Market::signIn (
    string name,
    nif_t nif )
```

A member function that signs in the user.

Parameters

<i>name</i>	Name of the client/user
<i>nif</i>	NIF othe client/user

Returns

A boolean, true if signing in was done successfully.

3.9.3.29 `signOut()`

```
void Market::signOut ( )
```

A member function that signs out the user.

3.9.3.30 `signUp()`

```
bool Market::signUp (
    string name,
    nif_t nif )
```

A member function that signs up the user.

Parameters

<i>name</i>	Name of the client/user
<i>nif</i>	NIF othe client/user

Returns

A boolean, true if signing up was done successfully.

3.9.4 Friends And Related Function Documentation

3.9.4.1 `operator<<`

```
ostream& operator<< (
    ostream & out,
    const Market & m ) [friend]
```

Overload of Operator << for class [Market](#). Prints the [Market](#) statistics.

Parameters

<i>out</i>	The outstream to write to.
<i>m</i>	The Market .

Returns

Returns the output stream to allow chaining

3.9.5 Member Data Documentation

3.9.5.1 clients

```
map<nif_t, Client *> Market::clients [private]
```

Map clients. A map where the key's are clients NIF's and the values are client pointers. Corresponds a NIF and a client.

3.9.5.2 clientsChanged

```
bool Market::clientsChanged [private]
```

bool clientsChanged. Boolean set to true if any changes done to the clients during execution.

3.9.5.3 clientsFile

```
string Market::clientsFile [private]
```

string clientsFile. String with the client's file name.

3.9.5.4 companys

```
set<Company> Market::companys [private]
```

Set companys. Implemented as a Binary Search Tree, of [Company](#) objects.

3.9.5.5 companysChanged

```
bool Market::companysChanged [private]
```

bool companysChanged. Boolean set to true if any changes done to the companys during execution.

3.9.5.6 companysFile

```
string Market::companysFile [private]
```

string companysFile. String with the company's file name.

3.9.5.7 currentNIF

```
nif_t Market::currentNIF [private]
```

nif_t NIF. Saves the NIF of the current user.

3.9.5.8 inactive_investors

```
unordered_set<Investor*, investorPtrHash, investorPtrHash> Market::inactive_investors [private]
```

Hash table inactive_investors. Implemented as an unordered_set of investors with hash function

3.9.5.9 investors

```
priority_queue<Investor> Market::investors [private]
```

Priority Queue investors. Implemented as a priority queue of [Investor](#) objects.

3.9.5.10 investorsChanged

```
bool Market::investorsChanged [private]
```

bool investorsChanged. Boolean set to true if any changes done to the investors during execution.

3.9.5.11 investorsFile

```
string Market::investorsFile [private]
```

string investorsFile. String with the investor's file name.

3.9.5.12 ordersChanged

```
bool Market::ordersChanged [private]
```

bool ordersChanged. Boolean set to true if any changes done to the orders during execution.

3.9.5.13 ordersFile

```
string Market::ordersFile [private]
```

string ordersFile. String with the order's file name.

3.9.5.14 singleton_instance

```
Market * Market::singleton_instance = NULL [static], [private]
```

[Market](#) pointer. Contains the pointer to the singleton instance of [Market](#).

3.9.5.15 transactions

```
vector<Transaction *> Market::transactions [private]
```

Vector transactions. A vector saving pointers of all [Market](#)'s transactions.

3.9.5.16 transactionsChanged

```
bool Market::transactionsChanged [private]
```

bool transactionsChanged. Boolean set to true if any changes done to the transactions during execution.

3.9.5.17 transactionsFile

```
string Market::transactionsFile [private]
```

string transactionsFile. String with the transaction's file name.

3.9.5.18 unfulfilled_orders

```
vector<Order *> Market::unfulfilled_orders [private]
```

Vector unfulfilled_orders. A vector saving pointers of all [Market](#)'s unfulfilled orders.

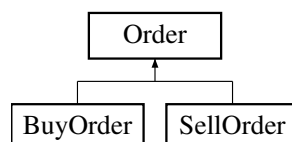
The documentation for this class was generated from the following files:

- D:/ProjetosC++/AEDAv2/StockMarket/Market.h
- D:/ProjetosC++/AEDAv2/StockMarket/Market.cpp

3.10 Order Class Reference

```
#include <Order.h>
```

Inheritance diagram for Order:



Classes

- class [InvalidValue](#)

Public Member Functions

- [Order](#) (ifstream &in)
- [Order](#) (string s, double value, unsigned quant)
- virtual [~Order](#) ()=default
- [Date](#) [getDatePlaced](#) () const
- string [getStock](#) () const
- double [getValue](#) () const
- unsigned [getQuantity](#) () const
- void [printInfo](#) () const
- virtual nif_t [getClientNIF](#) () const =0
- virtual [Transaction](#) * [operator\(\)](#) ([Order](#) *o)=0
- virtual void [saveChanges](#) (ofstream &out) const

Protected Attributes

- string [stock](#)
- double [valuePerStock](#)
- unsigned [quantity](#)
- [Date](#) [datePlaced](#)

3.10.1 Detailed Description

Abstract Base class used to represent an order.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 [Order\(\)](#) [1/2]

```
Order::Order (
    ifstream & in )
```

A constructor. The construtor creates an order object, reading the data from the input stream passed as argument.

Parameters

<i>in</i>	The input stream to read from in order to build the order object.
-----------	---

3.10.2.2 [Order\(\)](#) [2/2]

```
Order::Order (
    string s,
```

```
double value,  
unsigned quant )
```

A constructor. The constructor creates an order object using the data passed as arguments.

Parameters

<i>s</i>	A string with the stock name.
<i>value</i>	A double with the value per stock.
<i>quant</i>	An unsigned with the stock quantity.

3.10.2.3 ~Order()

```
virtual Order::~~Order ( ) [virtual], [default]
```

A virtual destructor.

3.10.3 Member Function Documentation

3.10.3.1 getClientNIF()

```
virtual nif_t Order::getClientNIF ( ) const [pure virtual]
```

A const abstract member function that returns the NIF of the client associated with the [Order](#).

Returns

The NIF of the Buyer/Seller associated with the [Order](#).

Implemented in [SellOrder](#), and [BuyOrder](#).

3.10.3.2 getDatePlaced()

```
Date Order::getDatePlaced ( ) const
```

A const member function with no arguments to get the orders's place date.

Returns

A [Date](#) object , the date when the order was placed.

3.10.3.3 getQuantity()

```
unsigned Order::getQuantity ( ) const
```

A const member function with no arguments to get the order stock quantity.

Returns

An unsigned, the quantity of stock.

3.10.3.4 getStock()

```
string Order::getStock ( ) const
```

A const member function with no arguments to get the orders's stock name.

Returns

A string that is the name of the stock.

3.10.3.5 getValue()

```
double Order::getValue ( ) const
```

A const member function with no arguments to get the value per stock.

Returns

A double, the value per stock.

3.10.3.6 operator()()

```
virtual Transaction* Order::operator() (
    Order * o ) [pure virtual]
```

Abstract overload of function operator. Cheks whether this instance of an [Order](#) Object can be fulfilled by the provided [Order](#).

Parameters

<i>an</i>	Order pointer
-----------	-------------------------------

Returns

A pointer to the transaction generated if successful, NULL otherwise.

Implemented in [SellOrder](#), and [BuyOrder](#).

3.10.3.7 printInfo()

```
void Order::printInfo ( ) const
```

A const member function that prints the order information.

3.10.3.8 saveChanges()

```
void Order::saveChanges (
    ostream & out ) const [virtual]
```

A virtual function to save changes in the derived classes.

Reimplemented in [SellOrder](#), and [BuyOrder](#).

3.10.4 Member Data Documentation**3.10.4.1 datePlaced**

```
Date Order::datePlaced [protected]
```

[Date](#) datePlaced. The date when the order was placed.

3.10.4.2 quantity

```
unsigned Order::quantity [protected]
```

unsigned quantity. An unsigned with the stock quantity.

3.10.4.3 stock

```
string Order::stock [protected]
```

string stock. A string with stock name.

3.10.4.4 valuePerStock

```
double Order::valuePerStock [protected]
```

valuePerStock. A double with the stock value.

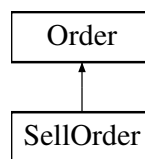
The documentation for this class was generated from the following files:

- D:/ProjetosC++/AEDAv2/StockMarket/Order.h
- D:/ProjetosC++/AEDAv2/StockMarket/Order.cpp

3.11 SellOrder Class Reference

```
#include <Order.h>
```

Inheritance diagram for SellOrder:



Public Member Functions

- [SellOrder](#) (ifstream &in)
///
- [SellOrder](#) (string [stock](#), double val, unsigned [quantity](#), nif_t [sellerNIF](#))
- nif_t [getClientNIF](#) () const
- [Transaction](#) * [operator\(\)](#) ([Order](#) *o)
- void [saveChanges](#) (ofstream &out) const

Private Attributes

- friend [BuyOrder](#)
- nif_t [sellerNIF](#)

Additional Inherited Members

3.11.1 Detailed Description

Class used to represent a sell order. Derives from [Order](#).

3.11.2 Constructor & Destructor Documentation

3.11.2.1 SellOrder() [1/2]

```
SellOrder::SellOrder (
    ifstream & in )
```

```
///
```

A constructor. The constructor creates a [SellOrder](#) object, reading the data from the input stream passed as argument.

Parameters

<i>in</i>	The input stream to read from in order to build the SellOrder object.
-----------	---

3.11.2.2 [SellOrder\(\)](#) [2/2]

```
SellOrder::SellOrder (
    string stock,
    double val,
    unsigned quantity,
    nif_t sellerNIF )
```

A constructor. The constructor creates a [SellOrder](#) object using the data passed as arguments.

Parameters

<i>stock</i>	A string with the stock name.
<i>val</i>	A double with the value per stock.
<i>quantity</i>	An unsigned with the stock quantity.
<i>sellerNIF</i>	The seller's NIF.

3.11.3 Member Function Documentation

3.11.3.1 [getClientNIF\(\)](#)

```
nif_t SellOrder::getClientNIF ( ) const [virtual]
```

A const member function that returns the NIF of the client associated with this [SellOrder](#).

Returns

The NIF of the Seller associated with this [Order](#).

Implements [Order](#).

3.11.3.2 [operator\(\)\(\)](#)

```
Transaction * SellOrder::operator() (
    Order * o ) [virtual]
```

Overload of [operator\(\)](#) for class [Order](#).

Parameters

<i>o</i>	Pointer of an object Order .
----------	--

Returns

A transaction type object.

Implements [Order](#).

3.11.3.3 saveChanges()

```
void SellOrder::saveChanges (
    ofstream & out ) const [virtual]
```

A const member function to save changes in an output stream.

Parameters

<i>out</i>	The outstream file to write to.
------------	---------------------------------

Reimplemented from [Order](#).

3.11.4 Member Data Documentation

3.11.4.1 sellerNIF

```
nif_t SellOrder::sellerNIF [private]
```

nif_t sellerNIF. The NIF of the seller associated with this [Order](#).

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

- D:/ProjetosC++/AEDAv2/StockMarket/Order.h
- D:/ProjetosC++/AEDAv2/StockMarket/Order.cpp

3.12 Transaction Class Reference

```
#include <Transaction.h>
```

Public Member Functions

- [Transaction](#) ()=default
- [Transaction](#) (ifstream &in)
- [Transaction](#) (nif_t [buyerNIF](#), nif_t [sellerNIF](#), string [stock](#), double [value](#), unsigned [quantity](#))
- string [getStock](#) () const
- double [getValue](#) () const
- unsigned [getQuantity](#) () const
- nif_t [getSellerNIF](#) () const
- nif_t [getBuyerNIF](#) () const
- [Date](#) [getDate](#) () const
- void [saveChanges](#) (ofstream &out) const

Private Attributes

- nif_t [sellerNIF](#)
- nif_t [buyerNIF](#)
- string [stock](#)
- double [value](#)
- unsigned [quantity](#)
- [Date](#) [time_stamp](#)

Friends

- ostream & [operator<<](#) (ostream &, const [Transaction](#) &)

3.12.1 Detailed Description

A class used to represent a transaction.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 [Transaction\(\)](#) [1/3]

```
Transaction::Transaction ( ) [default]
```

A default constructor.

3.12.2.2 [Transaction\(\)](#) [2/3]

```
Transaction::Transaction (
    ifstream & in )
```

A constructor. The constructor creates a transaction object, reading the data from the input stream passed as argument.

Parameters

<i>in</i>	The input stream to read from in order to build the transaction object.
-----------	---

3.12.2.3 Transaction() [3/3]

```
Transaction::Transaction (
    nif_t buyerNIF,
    nif_t sellerNIF,
    string stock,
    double value,
    unsigned quantity )
```

A constructor. The constructor creates a transaction object using the data passed as arguments.

Parameters

<i>buyer</i> ↔ <i>NIF</i>	The NIF from the buyer.
<i>seller</i> ↔ <i>NIF</i>	The NIF from the seller.
<i>stock</i>	The stock name.
<i>value</i>	The value of the stock.
<i>quantity</i>	The amount of stock.

3.12.3 Member Function Documentation

3.12.3.1 getBuyerNIF()

```
nif_t Transaction::getBuyerNIF ( ) const
```

A const member function with no arguments to get the transaction's buyer NIF.

Returns

A `nif_t`, the buyer NIF.

3.12.3.2 getDate()

```
Date Transaction::getDate ( ) const
```

A const member function with no arguments to get the transaction's date.

Returns

A `Date` object, the date of the transaction.

3.12.3.3 getQuantity()

```
unsigned Transaction::getQuantity ( ) const
```

A const member function with no arguments to get the transaction's quantity of stock.

Returns

An unsigned, the quantity.

3.12.3.4 getSellerNIF()

```
nif_t Transaction::getSellerNIF ( ) const
```

A const member function with no arguments to get the transaction's seller NIF.

Returns

A nif_t , the seller NIF.

3.12.3.5 getStock()

```
string Transaction::getStock ( ) const
```

A const member function with no arguments to get the transaction's Stock name.

Returns

A string, the Stock's name.

3.12.3.6 getValue()

```
double Transaction::getValue ( ) const
```

A const member function with no arguments to get the transaction's Value Per Stock.

Returns

A double, the value per stock transacted.

3.12.3.7 saveChanges()

```
void Transaction::saveChanges (
    ofstream & out ) const
```

A const member function to write the transaction to a save file.

Parameters

<i>out</i>	The outputstream file to write to.
------------	------------------------------------

3.12.4 Friends And Related Function Documentation**3.12.4.1 operator<<**

```
ostream& operator<< (
    ostream & ,
    const Transaction & ) [friend]
```

Overload of Operator << for class [Transaction](#). Prints the transaction in a human friendly way.

Parameters

<i>out</i>	The outstream to write to.
<i>t</i>	The transaction to be written.

Returns

Returns the output stream to allow chaining

3.12.5 Member Data Documentation**3.12.5.1 buyerNIF**

```
nif_t Transaction::buyerNIF [private]
```

nif_t buyerNIF. The NIF from the client that bought the stock.

3.12.5.2 quantity

```
unsigned Transaction::quantity [private]
```

unsigned quantity. The quantity of stock.

3.12.5.3 sellerNIF

```
nif_t Transaction::sellerNIF [private]
```

nif_t sellerNIF. The NIF from the client that sold the stock.

3.12.5.4 stock

```
string Transaction::stock [private]
```

string stock. The name of the stock product.

3.12.5.5 time_stamp

```
Date Transaction::time_stamp [private]
```

Date time_stamp. The time where the transaction occurred.

3.12.5.6 value

```
double Transaction::value [private]
```

double value. The value of the stock.

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

- D:/ProjetosC++/AEDAv2/StockMarket/Transaction.h
- D:/ProjetosC++/AEDAv2/StockMarket/Transaction.cpp

Index

- ~Market
 - Market, [29](#)
- ~Order
 - Order, [40](#)
- addBudget
 - Investor, [22](#)
- availableBudget
 - Investor, [25](#)
- business_area
 - Company, [13](#)
- BuyOrder, [5](#)
 - BuyOrder, [6](#)
 - buyerNIF, [7](#)
 - getClientNIF, [6](#)
 - operator(), [7](#)
 - saveChanges, [7](#)
- buyerNIF
 - BuyOrder, [7](#)
 - Transaction, [49](#)
- changeCompany
 - Market, [29](#)
- changeInvestorContact
 - Market, [29](#)
- Client, [8](#)
 - Client, [8](#), [9](#)
 - getNIF, [9](#)
 - getName, [9](#)
 - name, [10](#)
 - nif, [10](#)
 - saveChanges, [9](#)
- Client::InvalidNIF, [19](#)
 - getNIF, [20](#)
 - InvalidNIF, [19](#)
- clientHistory
 - Market, [29](#)
- clients
 - Market, [36](#)
- clientsChanged
 - Market, [36](#)
- clientsFile
 - Market, [36](#)
- Company, [10](#)
 - business_area, [13](#)
 - Company, [11](#)
 - getArea, [12](#)
 - getName, [12](#)
 - getValue, [12](#)
 - max_transaction_value, [13](#)
 - NIF, [14](#)
 - name, [14](#)
 - operator<, [12](#)
 - operator<<, [13](#)
 - saveChanges, [12](#)
 - setValue, [12](#)
- companys
 - Market, [36](#)
- companysChanged
 - Market, [36](#)
- companysFile
 - Market, [36](#)
- currentNIF
 - Market, [36](#)
- Date, [14](#)
 - Date, [15](#)
 - day, [18](#)
 - get_day, [15](#)
 - get_month, [16](#)
 - get_year, [16](#)
 - month, [18](#)
 - operator<, [16](#)
 - operator<<, [17](#)
 - operator<=, [17](#)
 - operator>>, [18](#)
 - operator==, [17](#)
 - year, [19](#)
- datePlaced
 - Order, [42](#)
- day
 - Date, [18](#)
- debitInvest
 - Investor, [23](#)
- deleteCompany
 - Market, [30](#)
- eraseClientOrder
 - Market, [30](#)
- get_day
 - Date, [15](#)
- get_month
 - Date, [16](#)
- get_year
 - Date, [16](#)
- getArea
 - Company, [12](#)
- getBudget

- Investor, 23
- getBuyerNIF
 - Transaction, 47
- getClientNIF
 - BuyOrder, 6
 - Order, 40
 - SellOrder, 44
- getCurrentNIF
 - Market, 30
- getDate
 - Transaction, 47
- getDatePlaced
 - Order, 40
- getMaxInv
 - Investor, 23
- getNIF
 - Client, 9
 - Client::InvalidNIF, 20
- getName
 - Client, 9
 - Company, 12
- getPhoneNumber
 - Investor, 23
- getQuantity
 - Order, 40
 - Transaction, 47
- getSellerNIF
 - Transaction, 48
- getStock
 - Order, 41
 - Transaction, 48
- getValue
 - Company, 12
 - Order, 41
 - Transaction, 48
- inactive_investors
 - Market, 37
- insertCompany
 - Market, 30
- instance
 - Market, 31
- InvalidNIF
 - Client::InvalidNIF, 19
- InvalidValue
 - Order::InvalidValue, 20
- Investor, 21
 - addBudget, 22
 - availableBudget, 25
 - debitInvest, 23
 - getBudget, 23
 - getMaxInv, 23
 - getPhoneNumber, 23
 - Investor, 22
 - maxInvestment, 25
 - name, 26
 - operator<, 24
 - operator<<, 24
 - operator==, 25
 - phone, 26
 - updatePhoneN, 24
- investorPtrHash, 26
 - operator(), 26, 27
- investors
 - Market, 37
- investorsChanged
 - Market, 37
- investorsFile
 - Market, 37
- listBuyOrders
 - Market, 31
- listCompanyys
 - Market, 31
- listInactiveInvestors
 - Market, 31
- listInvestors
 - Market, 31
- listInvestorsB
 - Market, 32
- listInvestorsI
 - Market, 32
- listSellOrders
 - Market, 32
- Market, 27
 - ~Market, 29
 - changeCompany, 29
 - changeInvestorContact, 29
 - clientHistory, 29
 - clients, 36
 - clientsChanged, 36
 - clientsFile, 36
 - companyys, 36
 - companyysChanged, 36
 - companyysFile, 36
 - currentNIF, 36
 - deleteCompany, 30
 - eraseClientOrder, 30
 - getCurrentNIF, 30
 - inactive_investors, 37
 - insertCompany, 30
 - instance, 31
 - investors, 37
 - investorsChanged, 37
 - investorsFile, 37
 - listBuyOrders, 31
 - listCompanyys, 31
 - listInactiveInvestors, 31
 - listInvestors, 31
 - listInvestorsB, 32
 - listInvestorsI, 32
 - listSellOrders, 32
 - Market, 29
 - operator<<, 35
 - ordersChanged, 37
 - ordersFile, 37
 - placeOrder, 32

- printTransactions, [32](#), [33](#)
- recreditInvestor, [33](#)
- requestInvestement, [33](#)
- saveChanges, [34](#)
- showClientHistory, [34](#)
- showClientInfo, [34](#)
- showClientOrders, [34](#)
- signIn, [34](#)
- signOut, [35](#)
- signUp, [35](#)
- singleton_instance, [37](#)
- transactions, [37](#)
- transactionsChanged, [38](#)
- transactionsFile, [38](#)
- unfulfilled_orders, [38](#)
- max_transaction_value
 - Company, [13](#)
- maxInvestment
 - Investor, [25](#)
- month
 - Date, [18](#)
- NIF
 - Company, [14](#)
- name
 - Client, [10](#)
 - Company, [14](#)
 - Investor, [26](#)
- nif
 - Client, [10](#)
- operator<
 - Company, [12](#)
 - Date, [16](#)
 - Investor, [24](#)
- operator<<
 - Company, [13](#)
 - Date, [17](#)
 - Investor, [24](#)
 - Market, [35](#)
 - Transaction, [49](#)
- operator<=
 - Date, [17](#)
- operator>>
 - Date, [18](#)
- operator()
 - BuyOrder, [7](#)
 - investorPtrHash, [26](#), [27](#)
 - Order, [41](#)
 - SellOrder, [44](#)
- operator==
 - Date, [17](#)
 - Investor, [25](#)
- Order, [38](#)
 - ~Order, [40](#)
 - datePlaced, [42](#)
 - getClientNIF, [40](#)
 - getDatePlaced, [40](#)
 - getQuantity, [40](#)
 - getStock, [41](#)
 - getValue, [41](#)
 - operator(), [41](#)
 - Order, [39](#)
 - printInfo, [42](#)
 - quantity, [42](#)
 - saveChanges, [42](#)
 - stock, [42](#)
 - valuePerStock, [42](#)
- Order::InvalidValue, [20](#)
 - InvalidValue, [20](#)
 - value, [21](#)
- ordersChanged
 - Market, [37](#)
- ordersFile
 - Market, [37](#)
- phone
 - Investor, [26](#)
- placeOrder
 - Market, [32](#)
- printInfo
 - Order, [42](#)
- printTransactions
 - Market, [32](#), [33](#)
- quantity
 - Order, [42](#)
 - Transaction, [49](#)
- recreditInvestor
 - Market, [33](#)
- requestInvestement
 - Market, [33](#)
- saveChanges
 - BuyOrder, [7](#)
 - Client, [9](#)
 - Company, [12](#)
 - Market, [34](#)
 - Order, [42](#)
 - SellOrder, [45](#)
 - Transaction, [48](#)
- SellOrder, [43](#)
 - getClientNIF, [44](#)
 - operator(), [44](#)
 - saveChanges, [45](#)
 - SellOrder, [43](#), [44](#)
 - sellerNIF, [45](#)
- sellerNIF
 - SellOrder, [45](#)
 - Transaction, [49](#)
- setValue
 - Company, [12](#)
- showClientHistory
 - Market, [34](#)
- showClientInfo
 - Market, [34](#)
- showClientOrders

- Market, [34](#)
- signIn
 - Market, [34](#)
- signOut
 - Market, [35](#)
- signUp
 - Market, [35](#)
- singleton_instance
 - Market, [37](#)
- stock
 - Order, [42](#)
 - Transaction, [49](#)
- time_stamp
 - Transaction, [50](#)
- Transaction, [45](#)
 - buyerNIF, [49](#)
 - getBuyerNIF, [47](#)
 - getDate, [47](#)
 - getQuantity, [47](#)
 - getSellerNIF, [48](#)
 - getStock, [48](#)
 - getValue, [48](#)
 - operator<<, [49](#)
 - quantity, [49](#)
 - saveChanges, [48](#)
 - sellerNIF, [49](#)
 - stock, [49](#)
 - time_stamp, [50](#)
 - Transaction, [46](#), [47](#)
 - value, [50](#)
- transactions
 - Market, [37](#)
- transactionsChanged
 - Market, [38](#)
- transactionsFile
 - Market, [38](#)
- unfulfilled_orders
 - Market, [38](#)
- updatePhoneN
 - Investor, [24](#)
- value
 - Order::InvalidValue, [21](#)
 - Transaction, [50](#)
- valuePerStock
 - Order, [42](#)
- year
 - Date, [19](#)