

Ceid Cpp Project 2020

Generated by Doxygen 1.9.1

1 Ceid Cpp Project 2020	1
1.1 Introduction	1
1.2 Design	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 Buyer Class Reference	9
5.1.1 Detailed Description	12
5.1.2 Member Enumeration Documentation	13
5.1.2.1 Category	13
5.1.3 Constructor & Destructor Documentation	13
5.1.3.1 Buyer() [1/2]	13
5.1.3.2 Buyer() [2/2]	13
5.1.4 Member Function Documentation	14
5.1.4.1 awardBonus()	14
5.1.4.2 checkout()	15
5.1.4.3 clearCart()	15
5.1.4.4 getBonus()	16
5.1.4.5 getCategory()	17
5.1.4.6 getCategoryName()	17
5.1.4.7 getItemOrder()	18
5.1.4.8 isAdmin()	18
5.1.4.9 operator=()	19
5.1.4.10 operator==()	19
5.1.4.11 placeOrder()	19
5.1.4.12 removeFromOrder()	20
5.1.4.13 setBonus()	21
5.1.4.14 setCategory()	22
5.1.4.15 showCart()	22
5.1.5 Member Data Documentation	23
5.1.5.1 _bonus	23
5.1.5.2 _cart	23
5.1.5.3 _categories	23
5.1.5.4 _category	23
5.1.5.5 _categoryScore	24
5.1.5.6 _categoryString	24

5.2 EShop Class Reference	24
5.2.1 Detailed Description	26
5.2.2 Constructor & Destructor Documentation	26
5.2.2.1 EShop() [1/2]	27
5.2.2.2 EShop() [2/2]	28
5.2.2.3 ~EShop()	28
5.2.3 Member Function Documentation	29
5.2.3.1 addBuyer()	29
5.2.3.2 addItem()	30
5.2.3.3 checkStatus()	30
5.2.3.4 getBuyerByEmail()	31
5.2.3.5 getCategories()	32
5.2.3.6 getItemById()	32
5.2.3.7 getName()	33
5.2.3.8 getOwner()	33
5.2.3.9 getProductsInCategory()	34
5.2.3.10 removeBuyer()	34
5.2.3.11 removeItem()	35
5.2.3.12 setOwner()	35
5.2.3.13 showProduct()	36
5.2.3.14 updateItemStock()	36
5.2.4 Member Data Documentation	37
5.2.4.1 _buyers	37
5.2.4.2 _items	37
5.2.4.3 _name	37
5.2.4.4 _owner	37
5.3 EShopError Class Reference	38
5.3.1 Detailed Description	38
5.3.2 Constructor & Destructor Documentation	39
5.3.2.1 EShopError() [1/2]	39
5.3.2.2 EShopError() [2/2]	39
5.3.3 Member Function Documentation	39
5.3.3.1 error()	39
5.3.4 Member Data Documentation	40
5.3.4.1 _error	40
5.4 Item Class Reference	40
5.4.1 Detailed Description	42
5.4.2 Constructor & Destructor Documentation	43
5.4.2.1 Item()	43
5.4.2.2 ~Item()	43
5.4.3 Member Function Documentation	44
5.4.3.1 getBasicInfo()	44

5.4.3.2 getCategory()	44
5.4.3.3 getDescription()	44
5.4.3.4 getDetails()	45
5.4.3.5 getId()	45
5.4.3.6 getName()	45
5.4.3.7 getPrice()	46
5.4.3.8 getStock()	46
5.4.3.9 operator std::string()	46
5.4.3.10 operator==()	47
5.4.3.11 setCategory()	47
5.4.3.12 setDescription()	47
5.4.3.13 setId() [1/2]	48
5.4.3.14 setId() [2/2]	49
5.4.3.15 setName()	49
5.4.3.16 setPrice()	49
5.4.3.17 setStock()	50
5.4.4 Friends And Related Function Documentation	50
5.4.4.1 operator<<	51
5.4.5 Member Data Documentation	51
5.4.5.1 _category	51
5.4.5.2 _desc	51
5.4.5.3 _id	51
5.4.5.4 _name	52
5.4.5.5 _price	52
5.4.5.6 _stock	52
5.5 Menu Class Reference	52
5.5.1 Detailed Description	54
5.5.2 Constructor & Destructor Documentation	54
5.5.2.1 Menu()	54
5.5.2.2 ~Menu()	55
5.5.3 Member Function Documentation	55
5.5.3.1 askYesNo()	55
5.5.3.2 login()	55
5.5.3.3 showBrowseMenu()	56
5.5.3.4 showBuyerMenu()	57
5.5.3.5 showCartMenu()	58
5.5.3.6 showCategoryMenu()	59
5.5.3.7 showCheckoutMenu()	60
5.5.3.8 showLoginMenu()	61
5.5.3.9 showOwnerMenu()	61
5.5.3.10 showProductMenu()	62
5.5.3.11 showStatusMenu()	63

5.5.3.12 showWelcome()	64
5.5.4 Member Data Documentation	65
5.5.4.1 _buyer	65
5.5.4.2 _eshop	65
5.5.4.3 _owner	65
5.5.4.4 _user	65
5.6 Notebook Class Reference	66
5.6.1 Detailed Description	68
5.6.2 Constructor & Destructor Documentation	68
5.6.2.1 Notebook()	68
5.6.3 Member Function Documentation	69
5.6.3.1 getDetails()	69
5.6.3.2 getSubjects()	69
5.6.3.3 setId()	70
5.6.3.4 setSubjects()	70
5.6.4 Member Data Documentation	71
5.6.4.1 _subjects	71
5.7 Owner Class Reference	71
5.7.1 Detailed Description	74
5.7.2 Constructor & Destructor Documentation	74
5.7.2.1 Owner()	74
5.7.3 Member Function Documentation	74
5.7.3.1 isAdmin()	74
5.7.4 Member Data Documentation	75
5.7.4.1 _isAdmin	75
5.8 Paper Class Reference	75
5.8.1 Detailed Description	78
5.8.2 Constructor & Destructor Documentation	78
5.8.2.1 Paper()	78
5.8.3 Member Function Documentation	79
5.8.3.1 getDetails()	79
5.8.3.2 getPages()	79
5.8.3.3 getWeight()	80
5.8.3.4 setId()	80
5.8.3.5 setPages()	81
5.8.3.6 setWeight()	82
5.8.4 Member Data Documentation	83
5.8.4.1 _pages	83
5.8.4.2 _weight	83
5.9 Pen Class Reference	83
5.9.1 Detailed Description	86
5.9.2 Constructor & Destructor Documentation	86

5.9.2.1 Pen()	86
5.9.3 Member Function Documentation	87
5.9.3.1 getColor()	87
5.9.3.2 getDetails()	87
5.9.3.3 getTipSize()	88
5.9.3.4 setColor()	88
5.9.3.5 setId()	89
5.9.3.6 setTipSize()	89
5.9.4 Member Data Documentation	90
5.9.4.1 _color	90
5.9.4.2 _tipSize	90
5.10 Pencil Class Reference	91
5.10.1 Detailed Description	93
5.10.2 Member Enumeration Documentation	93
5.10.2.1 Type	93
5.10.3 Constructor & Destructor Documentation	93
5.10.3.1 Pencil()	93
5.10.4 Member Function Documentation	94
5.10.4.1 getDetails()	94
5.10.4.2 getTipSize()	95
5.10.4.3 getType()	95
5.10.4.4 setId()	95
5.10.4.5 setTipSize()	96
5.10.4.6 setType()	97
5.10.5 Member Data Documentation	97
5.10.5.1 _tipSize	97
5.10.5.2 _type	97
5.10.5.3 _typeMap	98
5.11 ShoppingCart Class Reference	98
5.11.1 Detailed Description	99
5.11.2 Constructor & Destructor Documentation	99
5.11.2.1 ShoppingCart()	99
5.11.3 Member Function Documentation	100
5.11.3.1 addItemOrder()	100
5.11.3.2 calculateCourier()	101
5.11.3.3 calculateNet()	101
5.11.3.4 changeItemOrderQuantity()	102
5.11.3.5 checkout()	102
5.11.3.6 clearCart()	103
5.11.3.7 getItemOrder()	104
5.11.3.8 removeItemOrder()	105
5.11.3.9 showCart()	106

5.11.4 Member Data Documentation	107
5.11.4.1 _buyer	107
5.11.4.2 _order	107
5.12 User Class Reference	108
5.12.1 Detailed Description	110
5.12.2 Constructor & Destructor Documentation	110
5.12.2.1 User()	110
5.12.3 Member Function Documentation	110
5.12.3.1 getEmail()	111
5.12.3.2 getName()	111
5.12.3.3 isAdmin()	112
5.12.3.4 setEmail()	113
5.12.3.5 setName()	113
5.12.4 Member Data Documentation	113
5.12.4.1 _email	114
5.12.4.2 _name	114
6 File Documentation	115
6.1 src/buyer.cpp File Reference	115
6.2 buyer.cpp	116
6.3 src/buyer.h File Reference	117
6.4 buyer.h	118
6.5 src/eshop.cpp File Reference	119
6.6 eshop.cpp	119
6.7 src/eshop.h File Reference	121
6.8 eshop.h	123
6.9 src/eshoperror.cpp File Reference	123
6.10 eshoperror.cpp	124
6.11 src/eshoperror.h File Reference	124
6.12 eshoperror.h	124
6.13 src/item.cpp File Reference	125
6.13.1 Function Documentation	125
6.13.1.1 operator<<()	125
6.14 item.cpp	126
6.15 src/item.h File Reference	127
6.16 item.h	127
6.17 src/main.cpp File Reference	128
6.17.1 Function Documentation	129
6.17.1.1 main()	129
6.18 main.cpp	129
6.19 src/menu.cpp File Reference	130
6.20 menu.cpp	131

6.21 src/menu.h File Reference	135
6.22 menu.h	136
6.23 src/notebook.cpp File Reference	137
6.24 notebook.cpp	137
6.25 src/notebook.h File Reference	138
6.26 notebook.h	139
6.27 src/owner.cpp File Reference	139
6.28 owner.cpp	139
6.29 src/owner.h File Reference	140
6.30 owner.h	141
6.31 src/paper.cpp File Reference	141
6.32 paper.cpp	141
6.33 src/paper.h File Reference	142
6.34 paper.h	143
6.35 src/pen.cpp File Reference	143
6.36 pen.cpp	144
6.37 src/pen.h File Reference	145
6.38 pen.h	146
6.39 src/pencil.cpp File Reference	146
6.40 pencil.cpp	146
6.41 src/pencil.h File Reference	147
6.42 pencil.h	148
6.43 src/shoppingcart.cpp File Reference	149
6.44 shoppingcart.cpp	149
6.45 src/shoppingcart.h File Reference	151
6.46 shoppingcart.h	152
6.47 src/user.cpp File Reference	152
6.48 user.cpp	152
6.49 src/user.h File Reference	153
6.50 user.h	154
Index	155

Chapter 1

Ceid Cpp Project 2020

Author

Tsampas Stilianos (1039884) (4104) tsampas@ceid.upatras.gr ceid4104@upatras.gr

Siamoglou Charalambos (1041601) (5890) siamoglou@ceid.upatras.gr ceid5890@upatras.gr↵

The source code can also be found [here](#).

1.1 Introduction

This is an implementation of the project for the Objective Programming course. The goal is to create an [EShop](#) in the C++ programming language

1.2 Design

The design was based on the proposed classes and structures with very little deviation. A few functions were refactored from "show" to "get" for clarity. The presentation of the results of those functions is handled by the [Menu](#). We also opted to use a map to represent the cart instead of an extra class mostly because it offered STL-defined amenities. Because of the use of "contains" on containers, it requires C++20 compliant compiler to build.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

EShop	24
EShopError	38
Item	40
Notebook	66
Paper	75
Pen	83
Pencil	91
Menu	52
ShoppingCart	98
User	108
Buyer	9
Owner	71

Chapter 3

Class Index

3.1 Class List

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

Buyer	Specialization of User . Describes a Buyer	9
EShop	Class implementing the e-shop	24
EShopError	Exception class for passing error messages on failures	38
Item	Base class for all other items	40
Menu	Creates a menu for the e-shop's interface	52
Notebook	Class representing a Notebook	66
Owner	Specialization of User . Describes an Owner	71
Paper	Class representing a Paper	75
Pen	Class representing a Pen	83
Pencil	Class representing a Pencil	91
ShoppingCart	Class implementing the shopping cart	98
User	Base class for all users	108

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/buyer.cpp	115
src/buyer.h	117
src/eshop.cpp	119
src/eshop.h	121
src/eshoperror.cpp	123
src/eshoperror.h	124
src/item.cpp	125
src/item.h	127
src/main.cpp	128
src/menu.cpp	130
src/menu.h	135
src/notebook.cpp	137
src/notebook.h	138
src/owner.cpp	139
src/owner.h	140
src/paper.cpp	141
src/paper.h	142
src/pen.cpp	143
src/pen.h	145
src/pencil.cpp	146
src/pencil.h	147
src/shoppingcart.cpp	149
src/shoppingcart.h	151
src/user.cpp	152
src/user.h	153

Chapter 5

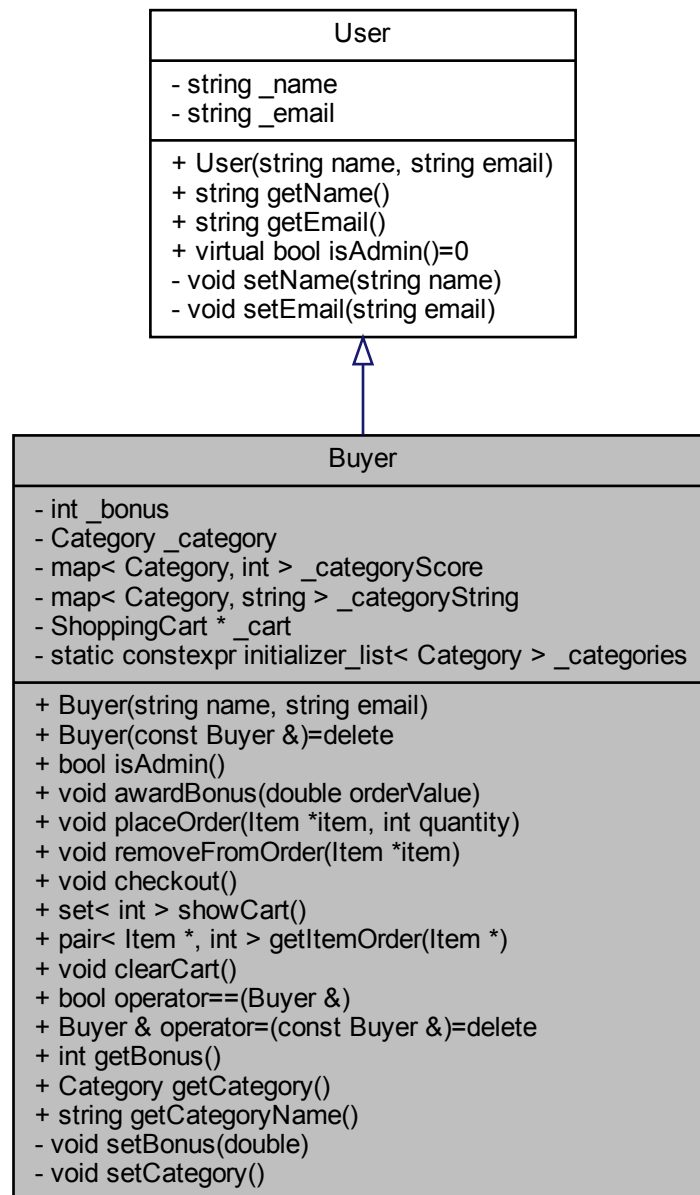
Class Documentation

5.1 Buyer Class Reference

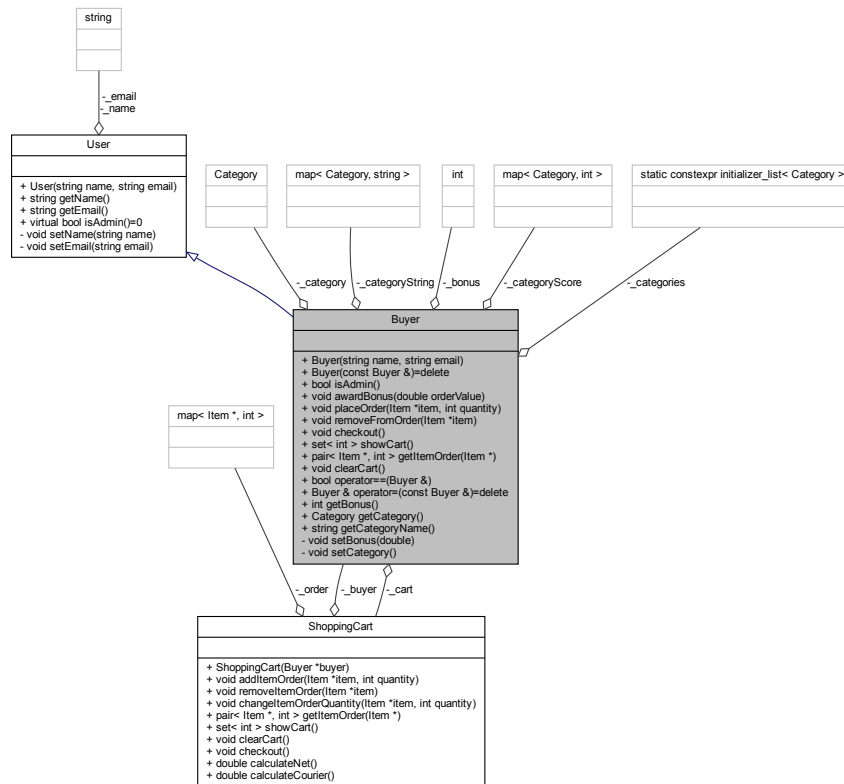
Specialization of [User](#). Describes a [Buyer](#).

```
#include <buyer.h>
```

Inheritance diagram for Buyer:



Collaboration diagram for Buyer:



Public Types

- enum **Category** { **Bronze** , **Silver** , **Gold** }
- Buyer _categories.*

Public Member Functions

- Buyer** (string name, string email)
Constructor of Buyer.
- Buyer** (const **Buyer** &)=delete
Disable the copy constructor for buyer.
- bool **isAdmin** ()
Impementation of isAdmin() of User.
- void **awardBonus** (double orderValue)
Set the buyer's bonus based on the cost of the order.
- void **placeOrder** (**Item** *item, int quantity)
Add an item to the cart with the specified quantity.
- void **removeFromOrder** (**Item** *item)
Removes an item from the cart completely.
- void **checkout** ()
Wrapper to ShoppingCart::checkout()
- set< int > **showCart** ()

- *Wrapper to [ShoppingCart::showCart\(\)](#)*
- `pair< Item *, int > getItemOrder (Item *)`
Wrapper to [ShoppingCart::getItemOrder\(\)](#)
- `void clearCart ()`
Wrapper to [ShoppingCart::clearCart\(\)](#)
- `bool operator== (Buyer &)`
Comparison operator between two Buyers.
- `Buyer & operator= (const Buyer &)=delete`
Disable the copy operator for buyer.
- `int getBonus ()`
Get the [Buyer](#)'s bonus.
- `Category getCategory ()`
Get the [Buyer](#)'s category.
- `string getCategoryName ()`
Get the [Buyer](#)'s category as string.

Private Member Functions

- `void setBonus (double)`
Sets a buyer's bonus based on the cost.
- `void setCategory ()`
Sets a buyer's category based on `_bonus`.

Private Attributes

- `int _bonus`
- `Category _category`
- `map< Category, int > _categoryScore { {Bronze, 0}, {Silver, 100}, {Gold, 200} }`
Map of the categories and their respective lower end scores.
- `map< Category, string > _categoryString { {Bronze, "Bronze"}, {Silver, "Silver"}, {Gold, "Gold"} }`
Map of the categories and their string representations.
- `ShoppingCart * _cart`
Pointer to the user's cart (forward declarations and such)

Static Private Attributes

- `static constexpr initializer_list< Category > _categories = {Bronze, Silver, Gold}`
I just needed an iterator over the categories, so yeah...

5.1.1 Detailed Description

Specialization of [User](#). Describes a [Buyer](#).

Derivative class to specialize a [User](#). Implements [Buyer](#) related functionality.

Definition at line 21 of file [buyer.h](#).

5.1.2 Member Enumeration Documentation

5.1.2.1 Category

```
enum Buyer::Category
```

[Buyer](#) _categories.

Enumeration of the different [Buyer](#) status categories. Used to determine the perks of a [Buyer](#).

Enumerator

Bronze	
Silver	
Gold	

Definition at line 44 of file [buyer.h](#).

5.1.3 Constructor & Destructor Documentation

5.1.3.1 Buyer() [1/2]

```
Buyer::Buyer (  
    string name,  
    string email )
```

Constructor of [Buyer](#).

Parameters

<i>name</i>	User 's name
<i>email</i>	User 's login email

Definition at line 5 of file [buyer.cpp](#).

References [_bonus](#), [_cart](#), [_category](#), and [Bronze](#).

5.1.3.2 Buyer() [2/2]

```
Buyer::Buyer (  
    const Buyer & ) [delete]
```

Disable the copy constructor for buyer.

5.1.4 Member Function Documentation

5.1.4.1 awardBonus()

```
void Buyer::awardBonus (
    double orderValue )
```

Set the buyer's bonus based on the cost of the order.

Parameters

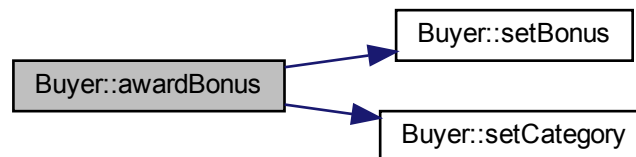
<i>orderValue</i>	double The value of the order
-------------------	-------------------------------

Definition at line 13 of file [buyer.cpp](#).

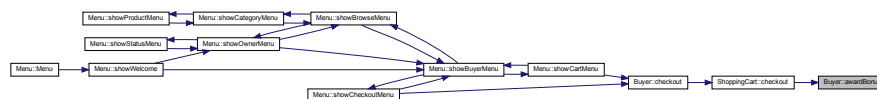
References [setBonus\(\)](#), and [setCategory\(\)](#).

Referenced by [ShoppingCart::checkout\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.4.2 checkout()

```
void Buyer::checkout ( )
```

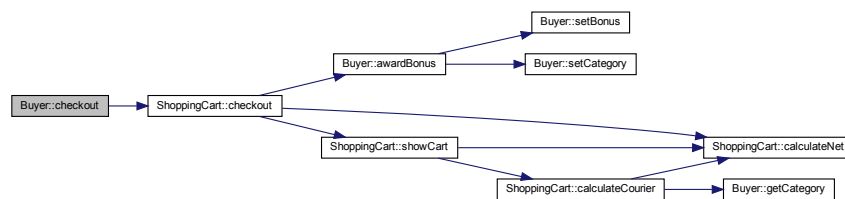
Wrapper to [ShoppingCart::checkout\(\)](#)

Definition at line 46 of file [buyer.cpp](#).

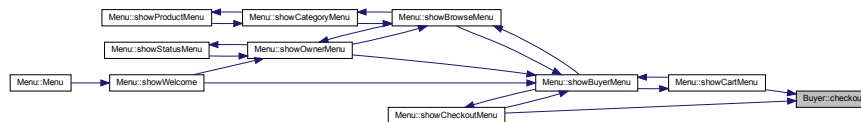
References [_cart](#), and [ShoppingCart::checkout\(\)](#).

Referenced by [Menu::showCartMenu\(\)](#), and [Menu::showCheckoutMenu\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.4.3 clearCart()

```
void Buyer::clearCart ( )
```

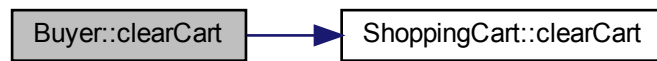
Wrapper to [ShoppingCart::clearCart\(\)](#)

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

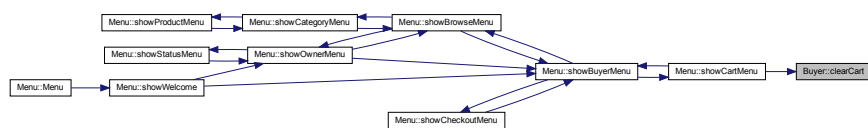
References [_cart](#), and [ShoppingCart::clearCart\(\)](#).

Referenced by [Menu::showCartMenu\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.4.4 getBonus()

```
int Buyer::getBonus ( )
```

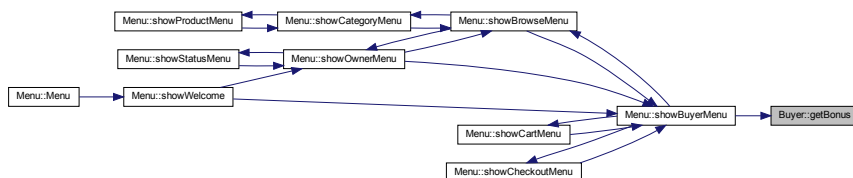
Get the [Buyer](#)'s bonus.

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

References [_bonus](#).

Referenced by [Menu::showBuyerMenu\(\)](#).

Here is the caller graph for this function:



5.1.4.5 getCategory()

```
Buyer::Category Buyer::getCategory ( )
```

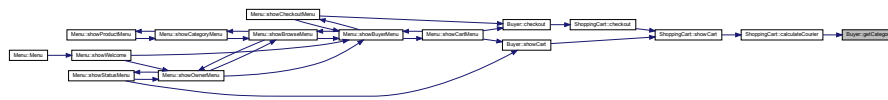
Get the [Buyer](#)'s category.

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

References [_category](#).

Referenced by [ShoppingCart::calculateCourier\(\)](#).

Here is the caller graph for this function:



5.1.4.6 getCategoryName()

```
string Buyer::getCategoryName ( )
```

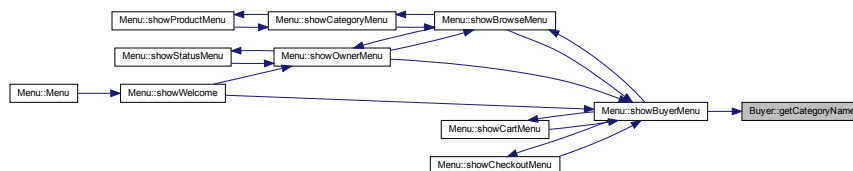
Get the [Buyer](#)'s category as string.

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

References [_category](#), and [_categoryString](#).

Referenced by [Menu::showBuyerMenu\(\)](#).

Here is the caller graph for this function:



5.1.4.7 getItemOrder()

```
pair< Item *, int > Buyer::getItemOrder (
    Item * item )
```

Wrapper to [ShoppingCart::getItemOrder\(\)](#)

Returns

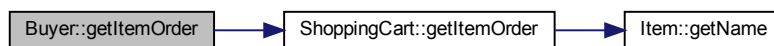
Returns a pair containing an item pointer and the quantity int in the cart

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

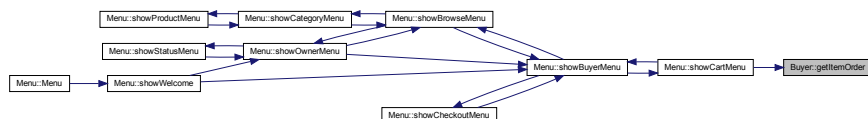
References [_cart](#), and [ShoppingCart::getItemOrder\(\)](#).

Referenced by [Menu::showCartMenu\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.4.8 isAdmin()

```
bool Buyer::isAdmin ( ) [virtual]
```

Implementation of [isAdmin\(\)](#) of [User](#).

Parameters

<i>none</i>

Returns

bool Always false

Implements [User](#).

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

5.1.4.9 operator=()

```
Buyer& Buyer::operator= (
    const Buyer & ) [delete]
```

Disable the copy operator for buyer.

5.1.4.10 operator==()

```
bool Buyer::operator== (
    Buyer & other )
```

Comparison operator between two Buyers.

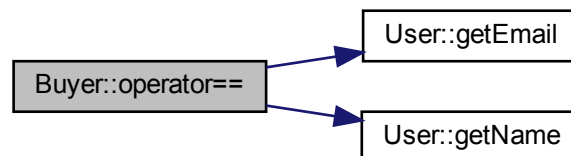
Returns

bool True if they have the same name and email otherwise false

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

References [User::getEmail\(\)](#), and [User::getName\(\)](#).

Here is the call graph for this function:

**5.1.4.11 placeOrder()**

```
void Buyer::placeOrder (
    Item * item,
    int quantity )
```

Add an item to the cart with the specified quantity.

This function checks if the the cart already contains the item. If it does, it updates the quantity in the cart. Also used to remove a quantity using negative values.

Parameters

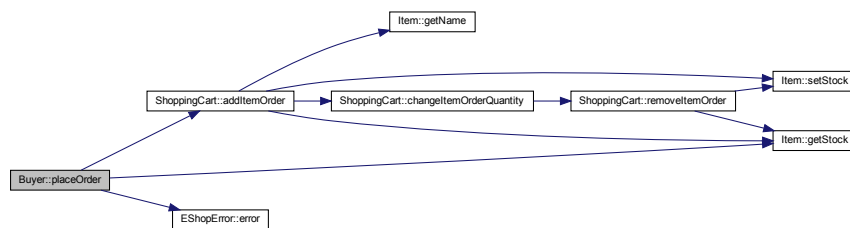
<i>item</i>	Item* The item to add to the cart
<i>quantity</i>	int The selected quantity

Definition at line 20 of file [buyer.cpp](#).

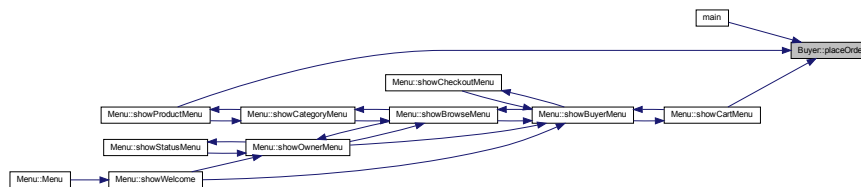
References [_cart](#), [ShoppingCart::addItemOrder\(\)](#), [EShopError::error\(\)](#), and [Item::getStock\(\)](#).

Referenced by [main\(\)](#), [Menu::showCartMenu\(\)](#), and [Menu::showProductMenu\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.4.12 removeFromOrder()

```
void Buyer::removeFromOrder (
    Item * item )
```

Removes an item from the cart completely.

Parameters

<i>item</i>	Item* The item to remove
-------------	--------------------------

Definition at line 40 of file [buyer.cpp](#).

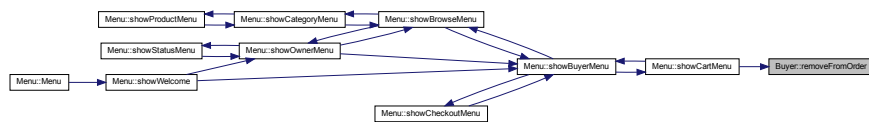
References [_cart](#), and [ShoppingCart::removeItemOrder\(\)](#).

Referenced by [Menu::showCartMenu\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.4.13 setBonus()

```
void Buyer::setBonus (
    double value ) [private]
```

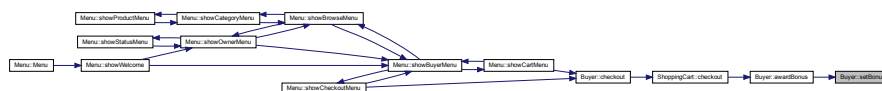
Sets a buyer's bonus based on the cost.

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

References [_bonus](#).

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

Here is the caller graph for this function:



5.1.4.14 setCategory()

```
void Buyer::setCategory ( ) [private]
```

Sets a buyer's category based on `_bonus`.

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

References [_bonus](#), [_categories](#), [_category](#), and [_categoryScore](#).

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

Here is the caller graph for this function:



5.1.4.15 showCart()

```
set< int > Buyer::showCart ( )
```

Wrapper to [ShoppingCart::showCart\(\)](#)

Returns

Returns a set of the IDs of the items in the cart

Definition at line 52 of file [buyer.cpp](#).

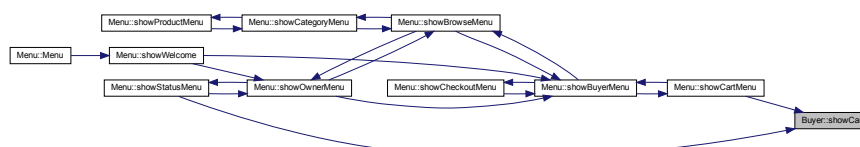
References [_cart](#), and [ShoppingCart::showCart\(\)](#).

Referenced by [Menu::showCartMenu\(\)](#), and [Menu::showStatusMenu\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.5 Member Data Documentation

5.1.5.1 `_bonus`

```
int Buyer::_bonus [private]
```

Definition at line 137 of file [buyer.h](#).

Referenced by [Buyer\(\)](#), [getBonus\(\)](#), [setBonus\(\)](#), and [setCategory\(\)](#).

5.1.5.2 `_cart`

```
ShoppingCart* Buyer::_cart [private]
```

Pointer to the user's cart (forward declarations and such)

Definition at line 146 of file [buyer.h](#).

Referenced by [Buyer\(\)](#), [checkout\(\)](#), [clearCart\(\)](#), [getItemOrder\(\)](#), [placeOrder\(\)](#), [removeFromOrder\(\)](#), and [showCart\(\)](#).

5.1.5.3 `_categories`

```
constexpr initializer_list<Category> Buyer::_categories = {Bronze, Silver, Gold} [static],  
[constexpr], [private]
```

I just needed an iterator over the categories, so yeah...

Definition at line 144 of file [buyer.h](#).

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

5.1.5.4 `_category`

```
Category Buyer::_category [private]
```

Definition at line 138 of file [buyer.h](#).

Referenced by [Buyer\(\)](#), [getCategory\(\)](#), [getCategoryName\(\)](#), and [setCategory\(\)](#).

5.1.5.5 `_categoryScore`

```
map<Category, int> Buyer::_categoryScore { {Bronze, 0}, {Silver, 100}, {Gold, 200} } [private]
```

Map of the categories and their respective lower end scores.

Definition at line 140 of file [buyer.h](#).

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

5.1.5.6 `_categoryString`

```
map<Category, string> Buyer::_categoryString { {Bronze, "Bronze"}, {Silver, "Silver"}, {Gold, "Gold"} } [private]
```

Map of the categories and their string representations.

Definition at line 142 of file [buyer.h](#).

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

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

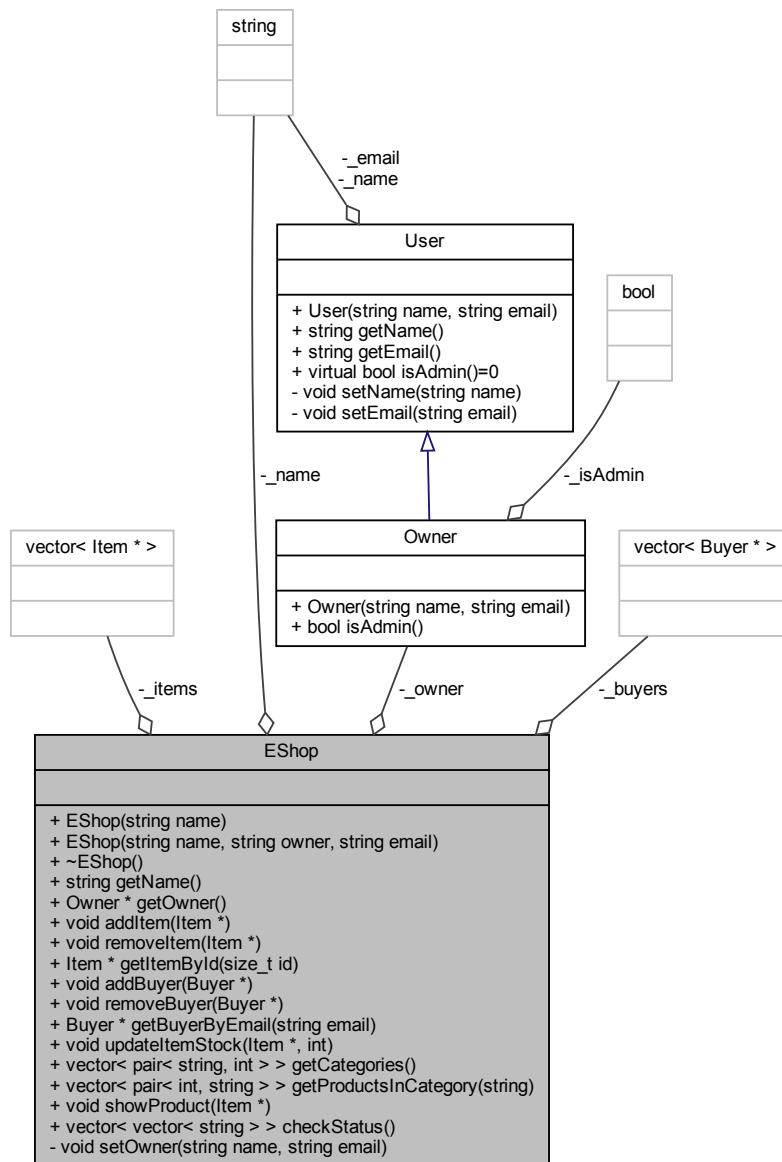
- [src/buyer.h](#)
- [src/buyer.cpp](#)

5.2 EShop Class Reference

Class implementing the e-shop.

```
#include <eshop.h>
```

Collaboration diagram for EShop:



Public Member Functions

- **EShop** (string name)
Constructor of **EShop**.
- **EShop** (string name, string owner, string email)
Constructor of **EShop**.
- **~EShop** ()
The destructor of e-shop.
- string **getName** ()
Get the name of the e-shop.
- **Owner** * **getOwner** ()

- Get the owner.*

 - void [addItem](#) ([Item](#) *)

Add an [Item](#) to the e-shop.
- void [removeItem](#) ([Item](#) *)

Remove an [Item](#) from the eshop if it exists.
- [Item](#) * [getItemById](#) (size_t id)

Return an [Item](#) reference if the [Item](#)'s ID is found in the e-shop.
- void [addBuyer](#) ([Buyer](#) *)

Add a [Buyer](#) to the e-shop.
- void [removeBuyer](#) ([Buyer](#) *)

Remove a [Buyer](#) from the eshop if it exists and clears their cart.
- [Buyer](#) * [getBuyerByEmail](#) (string email)

Return a [Buyer](#) reference if the [Buyer](#)'s email is found in the e-shop.
- void [updateItemStock](#) ([Item](#) *, int)

Update an [Item](#)'s stock.
- vector< pair< string, int > > [getCategories](#) ()

Get the categories of Items that exist in the e-shop.
- vector< pair< int, string > > [getProductsInCategory](#) (string)

Get the Items in a specific category.
- void [showProduct](#) ([Item](#) *)

Shows the details of the specified product.
- vector< vector< string > > [checkStatus](#) ()

Prints the status of the Buyers.

Private Member Functions

- void [setOwner](#) (string name, string email)
- Sets the name of the owner.*

Private Attributes

- string [_name](#)
- [Owner](#) * [_owner](#) = nullptr
- vector< [Buyer](#) * > [_buyers](#)
- vector< [Item](#) * > [_items](#)

5.2.1 Detailed Description

Class implementing the e-shop.

This class implements the e-shop related functionality. There are two ways this class can be instantiated, either by passing the name of the e-shop, in which case we have to specify an owner later on or by passing the name and the email of the owner to the constructor. It also holds the manipulates the Items and the Buyers of the [EShop](#).

Definition at line 22 of file [eshop.h](#).

5.2.2 Constructor & Destructor Documentation

5.2.2.1 EShop() [1/2]

```
EShop::EShop (  
    string name ) [explicit]
```

Constructor of [EShop](#).

Parameters

<i>name</i>	<string> The name of the e-shop
-------------	---------------------------------

Definition at line 7 of file [eshop.cpp](#).

References [_name](#), and [_owner](#).

5.2.2.2 EShop() [2/2]

```
EShop::EShop (
    string name,
    string owner,
    string email )
```

Constructor of [EShop](#).

Parameters

<i>name</i>	<string> The name of the e-shop
<i>owner</i>	<string> The name of the owner
<i>email</i>	<string> The email of the owner

Definition at line 13 of file [eshop.cpp](#).

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

Here is the call graph for this function:

**5.2.2.3 ~EShop()**

```
EShop::~~EShop ( )
```

The destructor of e-shop.

We require this to destroy any [Item](#) or [Buyer](#) objects created during execution

Definition at line 18 of file [eshop.cpp](#).

References [_buyers](#), [_items](#), and [_owner](#).

5.2.3 Member Function Documentation

5.2.3.1 addBuyer()

```
void EShop::addBuyer (
    Buyer * buyer )
```

Add a [Buyer](#) to the e-shop.

Checks if the [Buyer](#) already exists, if it does, throws an exception

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

References [_buyers](#), and [User::getName\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.2 addItem()

```
void EShop::addItem (
    Item * item )
```

Add an [Item](#) to the e-shop.

Checks if the [Item](#) already exists, if it does, throws an exception

Definition at line 41 of file [eshop.cpp](#).

References [_items](#), and [Item::getName\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.3 checkStatus()

```
vector< vector< string > > EShop::checkStatus ( )
```

Prints the status of the Buyers.

Also returns a vector of vectors of strings with the information for each buyer

Returns

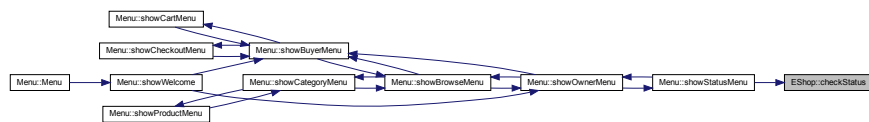
```
<vector<vector<string>>>
```

Definition at line 146 of file [eshop.cpp](#).

References [_buyers](#).

Referenced by [Menu::showStatusMenu\(\)](#).

Here is the caller graph for this function:



5.2.3.4 getBuyerByEmail()

```

Buyer * EShop::getBuyerByEmail (
    string email )

```

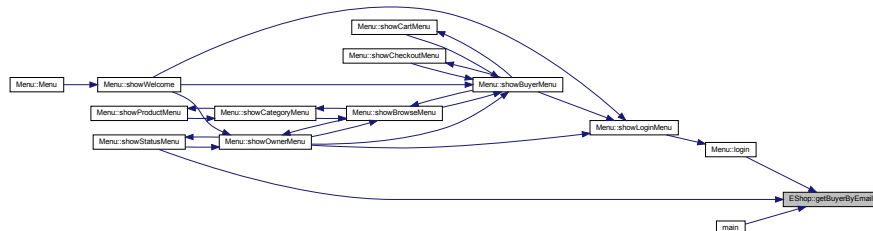
Return a [Buyer](#) reference if the [Buyer](#)'s email is found in the e-shop.

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

References [_buyers](#).

Referenced by [Menu::login\(\)](#), [main\(\)](#), and [Menu::showStatusMenu\(\)](#).

Here is the caller graph for this function:



5.2.3.5 getCategoryes()

```
vector< pair< string, int > > EShop::getCategoryes ( )
```

Get the categories of Items that exist in the e-shop.

Returns a vector of pairs consisting of the categories names and the number of products in each category.

Returns

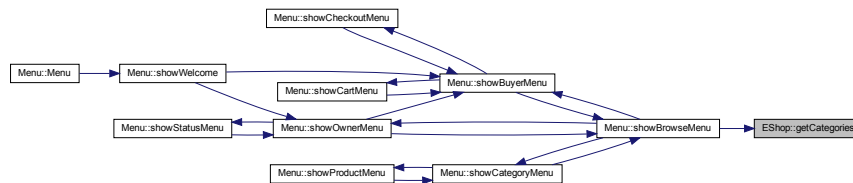
```
<vector<pair<string, int>>>
```

Definition at line 120 of file [eshop.cpp](#).

References [_items](#).

Referenced by [Menu::showBrowseMenu\(\)](#).

Here is the caller graph for this function:



5.2.3.6 getItemById()

```
Item * EShop::getItemById (
    size_t id )
```

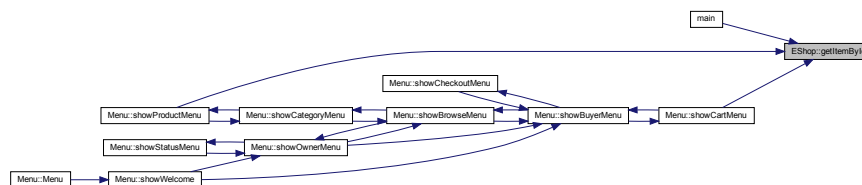
Return an [Item](#) reference if the [Item](#)'s ID is found in the e-shop.

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

References [_items](#).

Referenced by [main\(\)](#), [Menu::showCartMenu\(\)](#), and [Menu::showProductMenu\(\)](#).

Here is the caller graph for this function:



5.2.3.7 getName()

```
string EShop::getName ( )
```

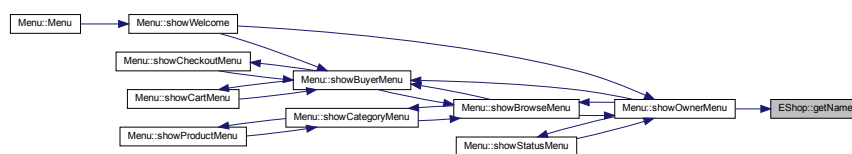
Get the name of the e-shop.

Definition at line 161 of file [eshop.cpp](#).

References [_name](#).

Referenced by [Menu::showOwnerMenu\(\)](#).

Here is the caller graph for this function:



5.2.3.8 getOwner()

```
Owner * EShop::getOwner ( )
```

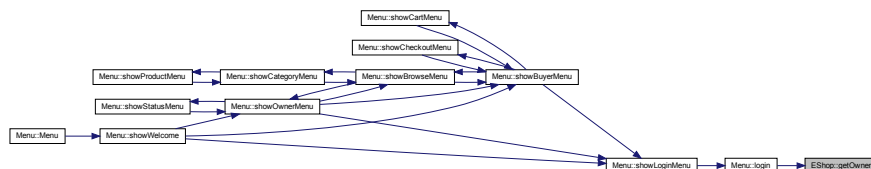
Get the owner.

Definition at line 26 of file [eshop.cpp](#).

References [_owner](#).

Referenced by [Menu::login\(\)](#).

Here is the caller graph for this function:



5.2.3.9 getProductsInCategory()

```
vector< pair< int, string > > EShop::getProductsInCategory (
    string category )
```

Get the Items in a specific category.

Returns a vector of pairs consisting of the [Item](#) IDs and the name of each product

Returns

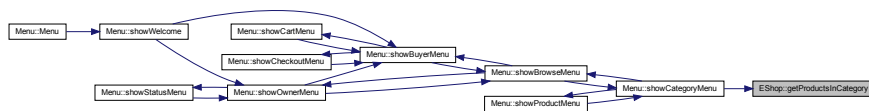
<vector<pair<int, string>>>

Definition at line 136 of file [eshop.cpp](#).

References [_items](#).

Referenced by [Menu::showCategoryMenu\(\)](#).

Here is the caller graph for this function:



5.2.3.10 removeBuyer()

```
void EShop::removeBuyer (
    Buyer * buyer )
```

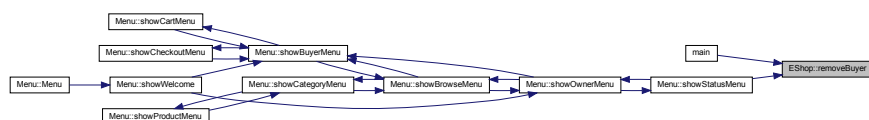
Remove a [Buyer](#) from the eshop if it exists and clears their cart.

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

References [_buyers](#).

Referenced by [main\(\)](#), and [Menu::showStatusMenu\(\)](#).

Here is the caller graph for this function:



5.2.3.11 removeItem()

```
void EShop::removeItem (
    Item * item )
```

Remove an [Item](#) from the eshop if it exists.

Definition at line 55 of file [eshop.cpp](#).

References [_items](#).

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

Here is the caller graph for this function:



5.2.3.12 setOwner()

```
void EShop::setOwner (
    string name,
    string email ) [private]
```

Sets the name of the owner.

Definition at line 33 of file [eshop.cpp](#).

References [_owner](#).

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

Here is the caller graph for this function:



5.2.3.13 showProduct()

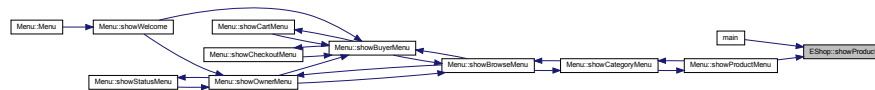
```
void EShop::showProduct (
    Item * item )
```

Shows the details of the specified product.

Definition at line 114 of file [eshop.cpp](#).

Referenced by [main\(\)](#), and [Menu::showProductMenu\(\)](#).

Here is the caller graph for this function:



5.2.3.14 updateItemStock()

```
void EShop::updateItemStock (
    Item * item,
    int delta )
```

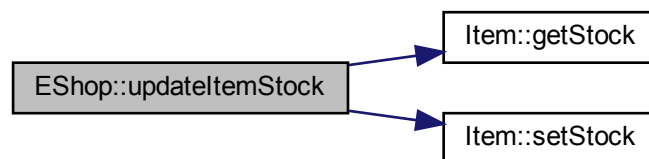
Update an [Item](#)'s stock.

Definition at line 108 of file [eshop.cpp](#).

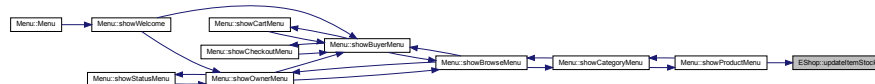
References [Item::getStock\(\)](#), and [Item::setStock\(\)](#).

Referenced by [Menu::showProductMenu\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.4 Member Data Documentation

5.2.4.1 `_buyers`

```
vector<Buyer*> EShop::_buyers [private]
```

Definition at line 128 of file [eshop.h](#).

Referenced by [addBuyer\(\)](#), [checkStatus\(\)](#), [getBuyerByEmail\(\)](#), [removeBuyer\(\)](#), and [~EShop\(\)](#).

5.2.4.2 `_items`

```
vector<Item*> EShop::_items [private]
```

Definition at line 129 of file [eshop.h](#).

Referenced by [addItem\(\)](#), [getCategories\(\)](#), [getItemById\(\)](#), [getProductsInCategory\(\)](#), [removeItem\(\)](#), and [~EShop\(\)](#).

5.2.4.3 `_name`

```
string EShop::_name [private]
```

Definition at line 126 of file [eshop.h](#).

Referenced by [EShop\(\)](#), and [getName\(\)](#).

5.2.4.4 `_owner`

```
Owner* EShop::_owner = nullptr [private]
```

Definition at line 127 of file [eshop.h](#).

Referenced by [EShop\(\)](#), [getOwner\(\)](#), [setOwner\(\)](#), and [~EShop\(\)](#).

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

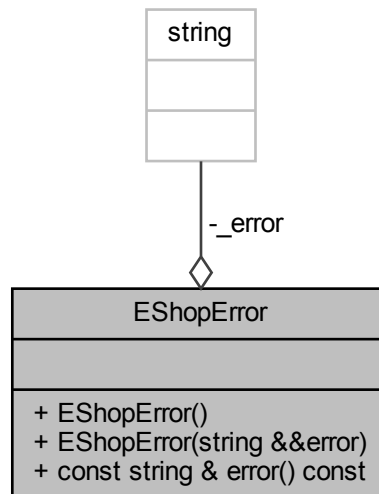
- [src/eshop.h](#)
- [src/eshop.cpp](#)

5.3 EShopError Class Reference

Exception class for passing error messages on failures.

```
#include <eshoperror.h>
```

Collaboration diagram for EShopError:



Public Member Functions

- [EShopError](#) ()
Constructor for empty exception messages.
- [EShopError](#) (string &&[error](#))
Constructor for empty exception messages.
- const string & [error](#) () const
Return the error message.

Private Attributes

- string [_error](#)

5.3.1 Detailed Description

Exception class for passing error messages on failures.

Definition at line 12 of file [eshoperror.h](#).

5.3.2 Constructor & Destructor Documentation

5.3.2.1 EShopError() [1/2]

```
EShopError::EShopError ( ) [inline]
```

Constructor for empty exception messages.

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

5.3.2.2 EShopError() [2/2]

```
EShopError::EShopError (
    string && error ) [inline]
```

Constructor for empty exception messages.

Parameters

<i>error</i>	The error message to pass
--------------	---------------------------

Definition at line 25 of file [eshoperror.h](#).

5.3.3 Member Function Documentation

5.3.3.1 error()

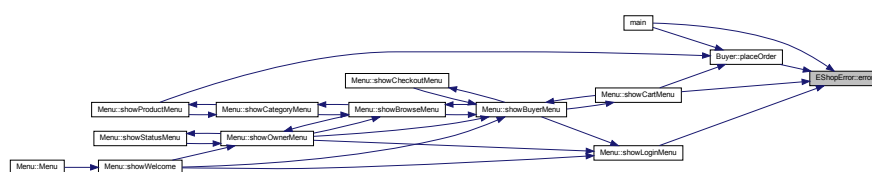
```
const string& EShopError::error ( ) const [inline]
```

Return the error message.

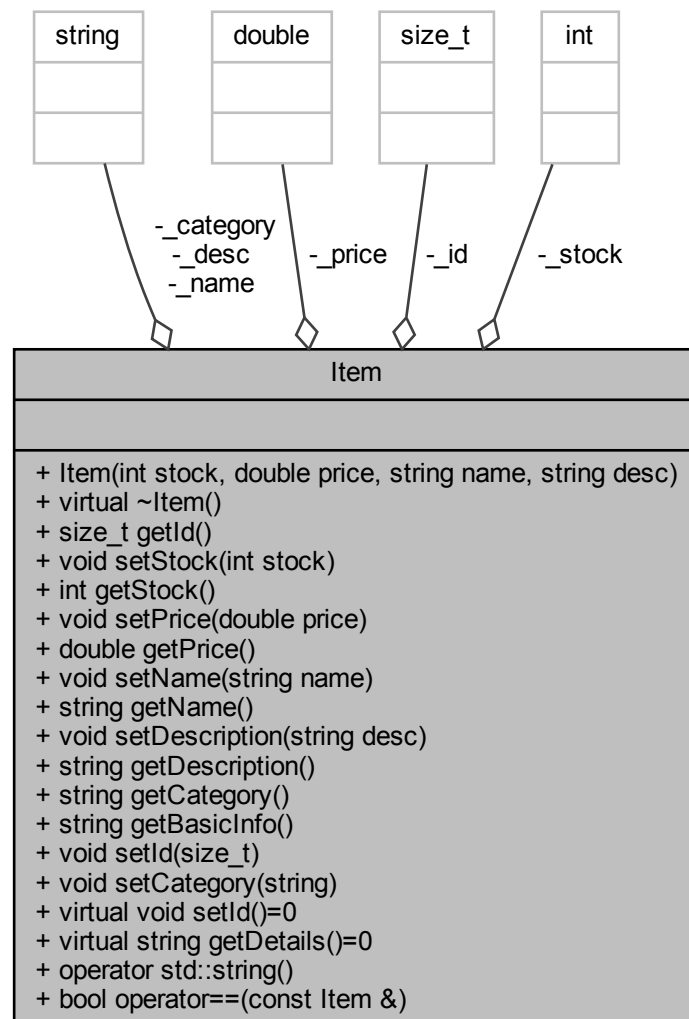
Definition at line 30 of file [eshoperror.h](#).

Referenced by [main\(\)](#), [Buyer::placeOrder\(\)](#), [Menu::showCartMenu\(\)](#), and [Menu::showLoginMenu\(\)](#).

Here is the caller graph for this function:



Collaboration diagram for Item:



Public Member Functions

- **Item** (int stock, double price, string name, string desc)
*Constructor for **Item**, called by the derivative classes.*
- virtual **~Item** ()
Virtual destructor to force the reimplementations in the derivatives.
- size_t **getId** ()
Get the ID of the item.
- void **setStock** (int stock)
Set the stock of an item.
- int **getStock** ()
*Get the **Item**'s stock.*
- void **setPrice** (double price)

- Set the price of an item.*

 - double `getPrice` ()

*Get the *Item*'s price.*
- void `setName` (string name)

Set the stock of an item.
- string `getName` ()

*Get the *Item*'s name.*
- void `setDescription` (string desc)

Set the stock of an item.
- string `getDescription` ()

*Get the *Item*'s description.*
- string `getCategory` ()

*Get the *Item*'s category (*Pen*, *Pencil*, *Paper*, *Notebook*)*
- string `getBasicInfo` ()

*Get the *Item*'s basic `getBasicInfo`.*
- void `setId` (size_t)

*Set the *Item*'s ID based on the hash.*
- void `setCategory` (string)

*Set the *Item*'s category.*
- virtual void `setId` ()=0

*Set the *Item*'s ID.*
- virtual string `getDetails` ()=0

*Get the *Item*'s specialization specific details.*
- `operator std::string` ()

Override the cast to `std::string` operator.
- bool `operator==` (const *Item* &)

Override for the comparison operator.

Private Attributes

- size_t `_id` = 0
- int `_stock`
- double `_price`
- string `_name`
- string `_desc`
- string `_category`

Friends

- ostream & `operator<<` (ostream &, *Item* &)
- Override for the ostream operator.*

5.4.1 Detailed Description

Base class for all other items.

Base abstract class to subclassed by the *Pen*, *Pencil*, *Paper* and *Notebook* specialized classes. Implements the common functions of setting the common characteristics of items, namely the Name, Stock, Description. Implements getters for common characteristics of items as well as virtual functions that should be re-implemented in each item category.

Definition at line 18 of file *item.h*.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Item()

```
Item::Item (
    int stock,
    double price,
    string name,
    string desc )
```

Constructor for [Item](#), called by the derivative classes.

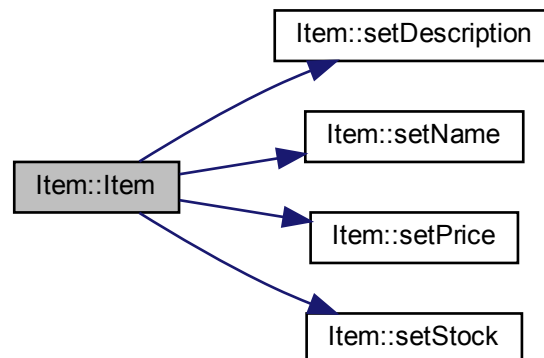
Parameters

<i>stock</i>	<int>.The stock the item should be created with
<i>price</i>	<double> The price of the item
<i>name</i>	<string> The name of the item, doesn't need to be unique
<i>desc</i>	<string> The description of the item

Definition at line 3 of file [item.cpp](#).

References [setDescription\(\)](#), [setName\(\)](#), [setPrice\(\)](#), and [setStock\(\)](#).

Here is the call graph for this function:



5.4.2.2 ~Item()

```
Item::~Item ( ) [virtual]
```

Virtual destructor to force the reimplementaion in the derivatives.

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

5.4.3 Member Function Documentation

5.4.3.1 getBasicInfo()

```
string Item::getBasicInfo ( )
```

Get the [Item](#)'s basic getBasicInfo.

Returns the common information between items as a commaspaced-separated string. The order of the returned information is "id, name, price, stock, description"

Returns

<string>

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

References [_desc](#), [_id](#), [_name](#), [_price](#), and [_stock](#).

5.4.3.2 getCategory()

```
string Item::getCategory ( )
```

Get the [Item](#)'s category ([Pen](#), [Pencil](#), [Paper](#), [Notebook](#))

Returns

<string>

Definition at line 52 of file [item.cpp](#).

References [_category](#).

5.4.3.3 getDescription()

```
string Item::getDescription ( )
```

Get the [Item](#)'s description.

Returns

<string>

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

References [_desc](#).

5.4.3.4 getDetails()

```
virtual string Item::getDetails ( ) [pure virtual]
```

Get the [Item](#)'s specialization specific details.

Pure abstract function that every derivative should implement. Returns a string of specialization specific details of each [Item](#).

Implemented in [Pencil](#), [Pen](#), [Notebook](#), and [Paper](#).

5.4.3.5 getId()

```
size_t Item::getId ( )
```

Get the ID of the item.

The ID of an item is a 4 digit hash generated by the defining characteristics of each item. The hash is computed by each derivative class

Returns

<size_t> The item's hash

Definition at line 57 of file [item.cpp](#).

References [_id](#).

5.4.3.6 getName()

```
string Item::getName ( )
```

Get the [Item](#)'s name.

Returns

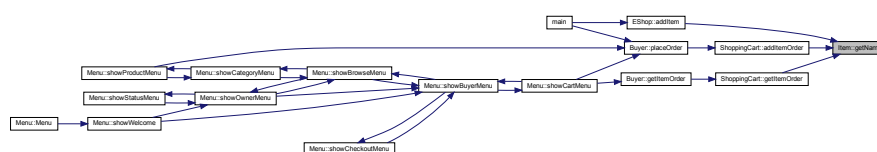
<string>

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

References [_name](#).

Referenced by [EShop::addItem\(\)](#), [ShoppingCart::addItemOrder\(\)](#), and [ShoppingCart::getItemOrder\(\)](#).

Here is the caller graph for this function:



5.4.3.7 getPrice()

```
double Item::getPrice ( )
```

Get the [Item](#)'s price.

Returns

<double>

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

References [_price](#).

5.4.3.8 getStock()

```
int Item::getStock ( )
```

Get the [Item](#)'s stock.

Returns

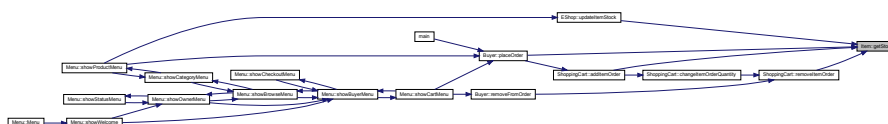
<int>

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

References [_stock](#).

Referenced by [ShoppingCart::addItemOrder\(\)](#), [Buyer::placeOrder\(\)](#), [ShoppingCart::removeItemOrder\(\)](#), and [EShop::updateItemStock\(\)](#).

Here is the caller graph for this function:



5.4.3.9 operator std::string()

```
Item::operator std::string ( )
```

Override the cast to std::string operator.

Override the cast to string operator to use it to return details about each item. Works by calling the [getBasicInfo\(\)](#) and [getDetails\(\)](#) functions

Definition at line 31 of file [item.cpp](#).

5.4.3.10 operator==()

```
bool Item::operator== (
    const Item & other )
```

Override for the comparison operator.

Compares two [Item](#) objects by comparing their IDs

Definition at line 44 of file [item.cpp](#).

References [_id](#).

5.4.3.11 setCategory()

```
void Item::setCategory (
    string category )
```

Set the [Item](#)'s category.

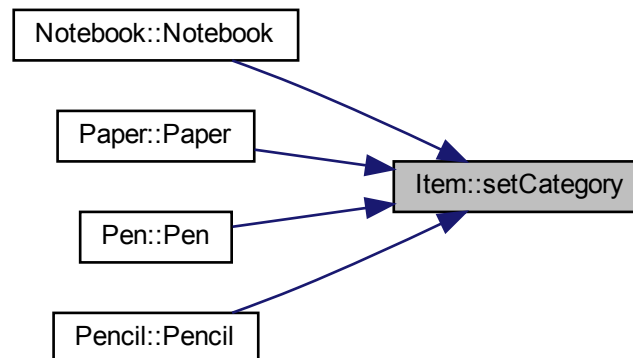
Set the item's category by removing the first character of the string returned by typeid (In the case of gcc 10 on my machine it is the name of the class prefix by the length of the string (according to the reference it is implementation specific))

Definition at line 50 of file [item.cpp](#).

References [_category](#).

Referenced by [Notebook::Notebook\(\)](#), [Paper::Paper\(\)](#), [Pen::Pen\(\)](#), and [Pencil::Pencil\(\)](#).

Here is the caller graph for this function:



5.4.3.12 setDescription()

```
void Item::setDescription (
    string desc )
```

Set the stock of an item.

Parameters

<i>desc</i>	<string> The description to set
-------------	---------------------------------

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

References [_desc](#).

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

Here is the caller graph for this function:



5.4.3.13 setId() [1/2]

```
virtual void Item::setId ( ) [pure virtual]
```

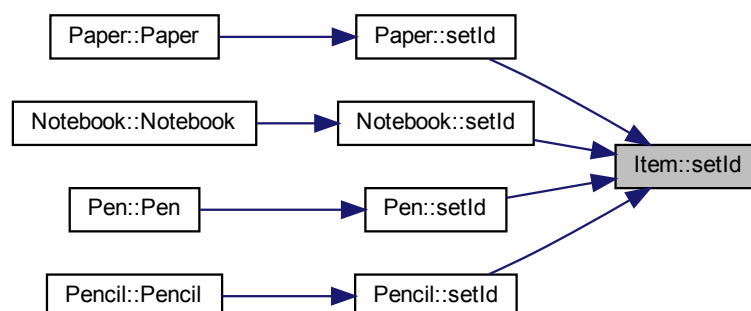
Set the [Item](#)'s ID.

Pure abstract function that every derivative should implement. All implementations of this function work on the same way. They create 2 or 3 hashes based on the unique identifiers of each item with are then XOR'd together. The first is identifier is the name of each class as returned by `typeid`. The second and third (if applicable) are specific to each specialization and documented there.

Implemented in [Pencil](#), [Pen](#), [Notebook](#), and [Paper](#).

Referenced by [Paper::setId\(\)](#), [Notebook::setId\(\)](#), [Pen::setId\(\)](#), and [Pencil::setId\(\)](#).

Here is the caller graph for this function:



5.4.3.14 setId() [2/2]

```
void Item::setId (
    size_t id )
```

Set the [Item](#)'s ID based on the hash.

Provides the common functionality to the derivatives of truncating the hash to a 4 least significant digits. Hopefully they are unique enough to be used as IDs

Definition at line 55 of file [item.cpp](#).

References [_id](#).

5.4.3.15 setName()

```
void Item::setName (
    string name )
```

Set the stock of an item.

Parameters

<i>name</i>	<string> The name to set
-------------	--------------------------

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

References [_name](#).

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

Here is the caller graph for this function:



5.4.3.16 setPrice()

```
void Item::setPrice (
    double price )
```

Set the price of an item.

Parameters

<i>price</i>	<double> The price to set
--------------	---------------------------

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

References [_price](#).

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

Here is the caller graph for this function:



5.4.3.17 setStock()

```
void Item::setStock (
    int stock )
```

Set the stock of an item.

Parameters

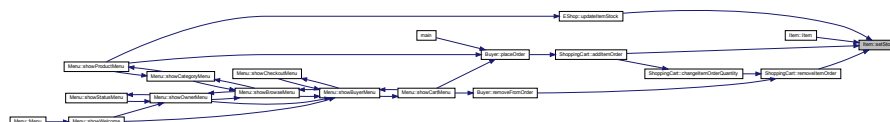
<i>stock</i>	<int> The stock to set
--------------	------------------------

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

References [_stock](#).

Referenced by [ShoppingCart::addItemOrder\(\)](#), [Item\(\)](#), [ShoppingCart::removeItemOrder\(\)](#), and [EShop::updateItemStock\(\)](#).

Here is the caller graph for this function:



5.4.4 Friends And Related Function Documentation

5.4.4.1 operator<<

```
ostream& operator<< (
    ostream & os,
    Item & item ) [friend]
```

Override for the ostream operator.

Used to throw information to the cout garbage can

Definition at line 36 of file [item.cpp](#).

5.4.5 Member Data Documentation

5.4.5.1 _category

```
string Item::_category [private]
```

Definition at line 178 of file [item.h](#).

Referenced by [getCategory\(\)](#), and [setCategory\(\)](#).

5.4.5.2 _desc

```
string Item::_desc [private]
```

Definition at line 177 of file [item.h](#).

Referenced by [getBasicInfo\(\)](#), [getDescription\(\)](#), and [setDescription\(\)](#).

5.4.5.3 _id

```
size_t Item::_id = 0 [private]
```

Definition at line 173 of file [item.h](#).

Referenced by [getBasicInfo\(\)](#), [getId\(\)](#), [operator==\(\)](#), and [setId\(\)](#).

5.4.5.4 `_name`

```
string Item::_name [private]
```

Definition at line 176 of file [item.h](#).

Referenced by [getBasicInfo\(\)](#), [getName\(\)](#), and [setName\(\)](#).

5.4.5.5 `_price`

```
double Item::_price [private]
```

Definition at line 175 of file [item.h](#).

Referenced by [getBasicInfo\(\)](#), [getPrice\(\)](#), and [setPrice\(\)](#).

5.4.5.6 `_stock`

```
int Item::_stock [private]
```

Definition at line 174 of file [item.h](#).

Referenced by [getBasicInfo\(\)](#), [getStock\(\)](#), and [setStock\(\)](#).

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

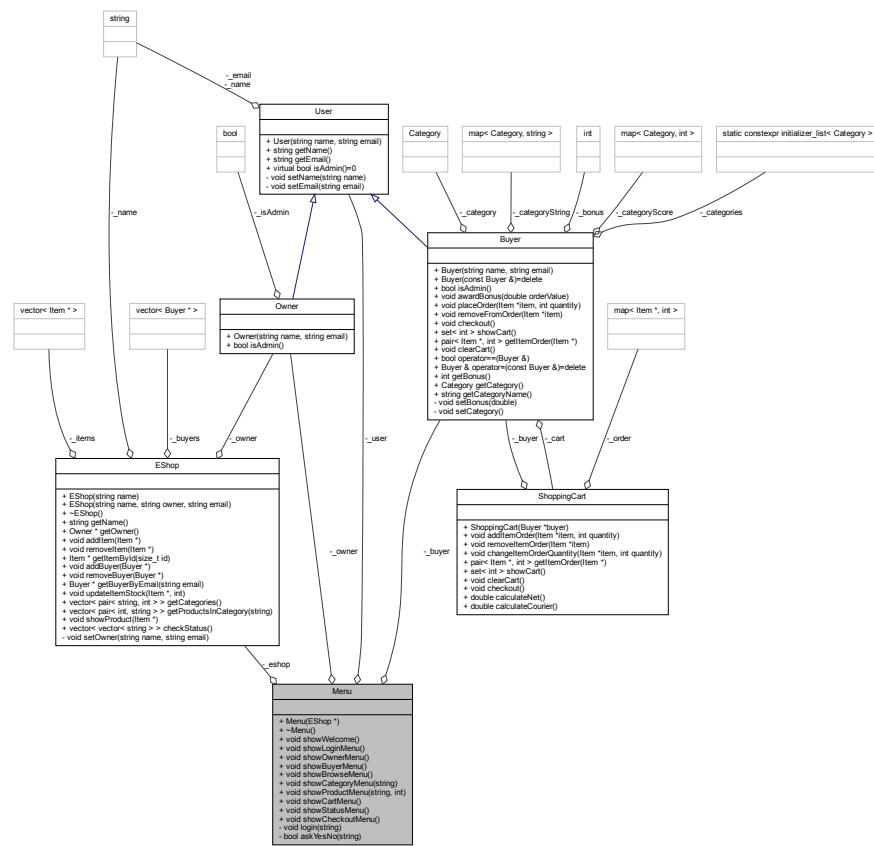
- [src/item.h](#)
- [src/item.cpp](#)

5.5 Menu Class Reference

Creates a menu for the e-shop's interface.

```
#include <menu.h>
```

Collaboration diagram for Menu:



Public Member Functions

- [Menu](#) ([EShop](#) *)
- [~Menu](#) ()
- void [showWelcome](#) ()
- void [showLoginMenu](#) ()
- void [showOwnerMenu](#) ()
- void [showBuyerMenu](#) ()
- void [showBrowseMenu](#) ()
- void [showCategoryMenu](#) (string)
- void [showProductMenu](#) (string, int)
- void [showCartMenu](#) ()
- void [showStatusMenu](#) ()
- void [showCheckoutMenu](#) ()

Private Member Functions

- void [login](#) (string)
User authentication.
- bool [askYesNo](#) (string)
Helper function to encapsulate yes/no questions.

Private Attributes

- `User * _user = nullptr`
- `Owner * _owner = nullptr`
- `Buyer * _buyer = nullptr`
- `EShop * _eshop`

5.5.1 Detailed Description

Creates a menu for the e-shop's interface.

This class provides the majority of user input and output. It takes an instantiated [EShop](#) and provides a navigation [Menu](#).

Definition at line 20 of file [menu.h](#).

5.5.2 Constructor & Destructor Documentation

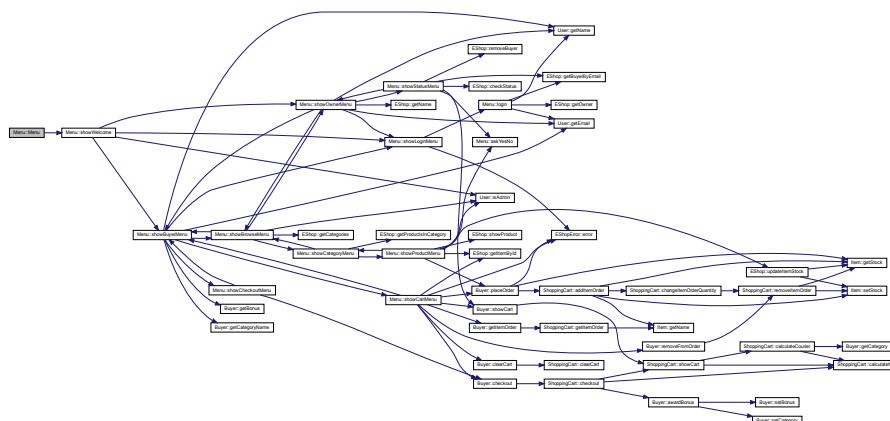
5.5.2.1 Menu()

```
Menu::Menu (
    EShop * eshop )
```

Definition at line 6 of file [menu.cpp](#).

References [_eshop](#), and [showWelcome\(\)](#).

Here is the call graph for this function:



5.5.2.2 ~Menu()

Menu::~~Menu ()

Definition at line 317 of file [menu.cpp](#).

5.5.3 Member Function Documentation

5.5.3.1 askYesNo()

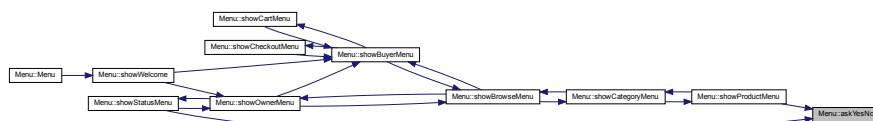
```
bool Menu::askYesNo (
    string message ) [private]
```

Helper function to encapsulate yes/no questions.

Definition at line 341 of file [menu.cpp](#).

Referenced by [showProductMenu\(\)](#), and [showStatusMenu\(\)](#).

Here is the caller graph for this function:



5.5.3.2 login()

```
void Menu::login (
    string email ) [private]
```

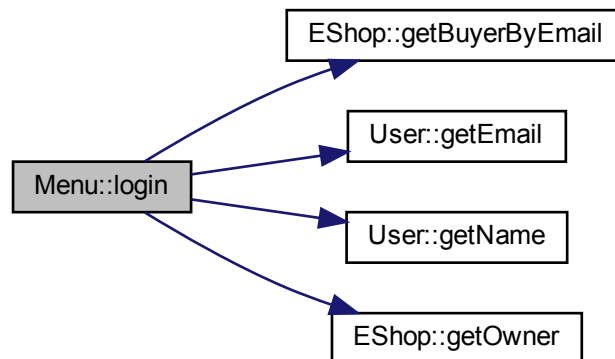
User authentication.

Definition at line 322 of file [menu.cpp](#).

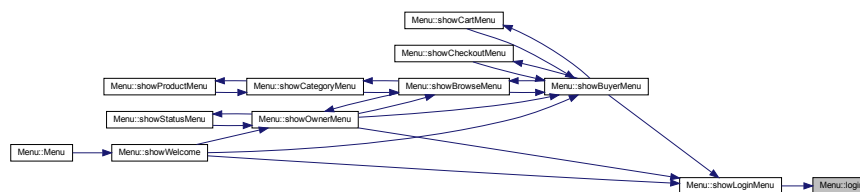
References [_buyer](#), [_eshop](#), [_owner](#), [_user](#), [EShop::getBuyerByEmail\(\)](#), [User::getEmail\(\)](#), [User::getName\(\)](#), and [EShop::getOwner\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.3 showBrowseMenu()

```
void Menu::showBrowseMenu ( )
```

Definition at line 150 of file [menu.cpp](#).

References [_eshop](#), [_user](#), [EShop::getCategories\(\)](#), [User::isAdmin\(\)](#), [showBuyerMenu\(\)](#), [showCategoryMenu\(\)](#), and [showOwnerMenu\(\)](#).

Referenced by [showBuyerMenu\(\)](#), [showCategoryMenu\(\)](#), and [showOwnerMenu\(\)](#).

[illegible]

```

classDiagram
    class Menu {
        +showWelcome()
        +showCheckoutMenu()
        +showBuyerMenu()
        +showCartMenu()
        +showStatusMenu()
        +showOwnerMenu()
        +showBrowseMenu()
        +showProductMenu()
        +showCategoryMenu()
    }

```

```
void Menu::showBuyerMenu ( )
```

References `_buyer`, `Buyer::getBonus()`, `Buyer::getCategoryName()`, `User::getEmail()`, `User::getName()`, `showBrowseMenu()`, `showCartMenu()`, `showCheckoutMenu()`, and `showLoginMenu()`.

Generated by Doxygen

[illegible]

```

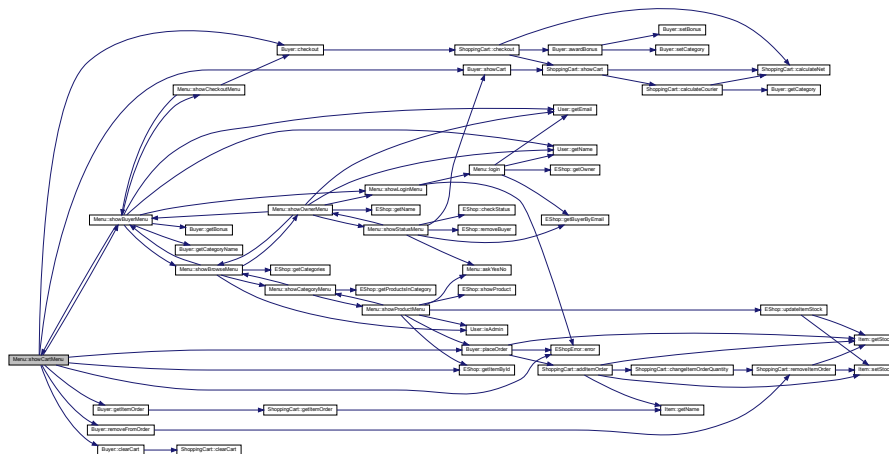
classDiagram
    class Menu {
    }
    class Menu_showWelcome {
    }
    class Menu_showCategoryMenu {
    }
    class Menu_showBrowseMenu {
    }
    class Menu_showStatusMenu {
    }
    class Menu_showOwnerMenu {
    }
    class Menu_showBuyerMenu {
    }
    class Menu_showCartMenu {
    }
    class Menu_showCheckoutMenu {
    }
    Menu --> Menu_showWelcome
    Menu --> Menu_showCategoryMenu
    Menu --> Menu_showBrowseMenu
    Menu --> Menu_showStatusMenu
    Menu --> Menu_showOwnerMenu
    Menu --> Menu_showBuyerMenu
    Menu --> Menu_showCartMenu
    Menu --> Menu_showCheckoutMenu
    Menu_showWelcome --> Menu_showCategoryMenu
    Menu_showWelcome --> Menu_showBrowseMenu
    Menu_showWelcome --> Menu_showBuyerMenu
    Menu_showCategoryMenu --> Menu_showBrowseMenu
    Menu_showCategoryMenu --> Menu_showBuyerMenu
    Menu_showBrowseMenu --> Menu_showBuyerMenu
    Menu_showStatusMenu --> Menu_showOwnerMenu
    Menu_showStatusMenu --> Menu_showBuyerMenu
    Menu_showOwnerMenu --> Menu_showBuyerMenu
    Menu_showBuyerMenu --> Menu_showCartMenu
    Menu_showBuyerMenu --> Menu_showCheckoutMenu
  
```

```
void Menu::showCartMenu ( )
```

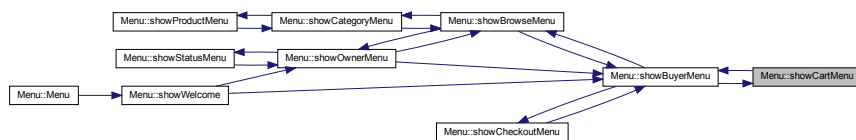
References [_buyer](#), [_eshop](#), [Buyer::checkout\(\)](#), [Buyer::clearCart\(\)](#), [EShopError::error\(\)](#), [EShop::getItemById\(\)](#), [Buyer::getItemOrder\(\)](#), [Buyer::placeOrder\(\)](#), [Buyer::removeFromOrder\(\)](#), [showBuyerMenu\(\)](#), and [Buyer::showCart\(\)](#).

Generated by Doxygen

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.6 showCategoryMenu()

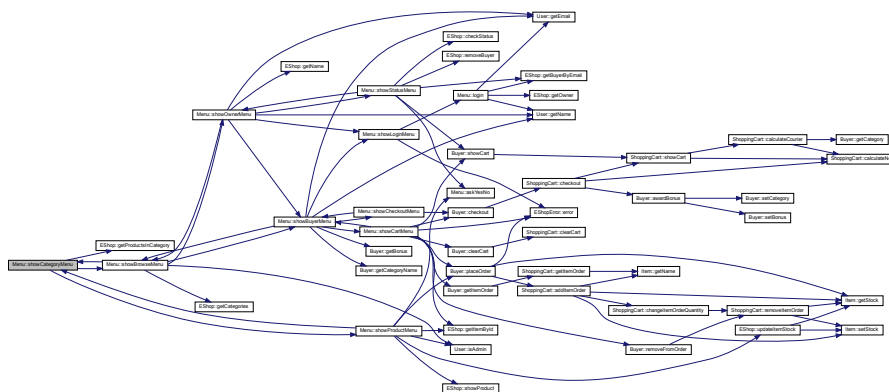
```
void Menu::showCategoryMenu (
    string category )
```

Definition at line 178 of file menu.cpp.

References `_eshop`, `EShop::getProductsInCategory()`, `showBrowseMenu()`, and `showProductMenu()`.

Referenced by `showBrowseMenu()`, and `showProductMenu()`.

Here is the call graph for this function:



5.5.3.8 showLoginMenu()

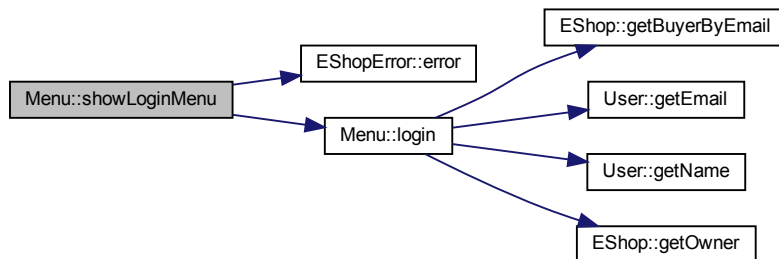
```
void Menu::showLoginMenu ( )
```

Definition at line 25 of file [menu.cpp](#).

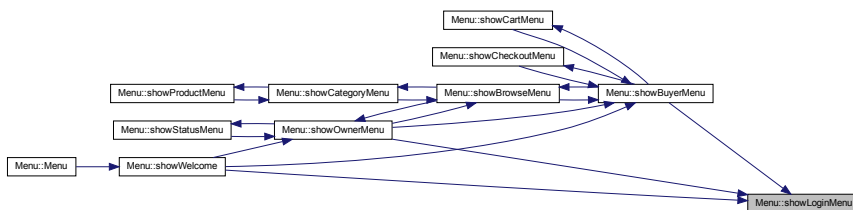
References [EShopError::error\(\)](#), and [login\(\)](#).

Referenced by [showBuyerMenu\(\)](#), [showOwnerMenu\(\)](#), and [showWelcome\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.9 showOwnerMenu()

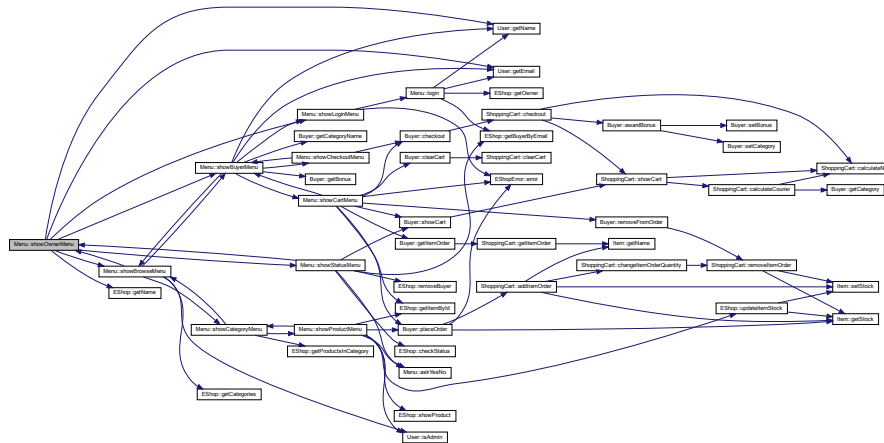
```
void Menu::showOwnerMenu ( )
```

Definition at line 39 of file [menu.cpp](#).

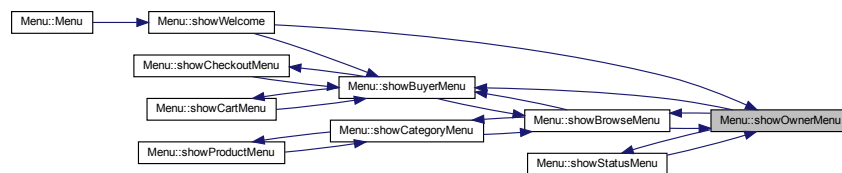
References [_eshop](#), [_owner](#), [User::getEmail\(\)](#), [EShop::getName\(\)](#), [User::getName\(\)](#), [showBrowseMenu\(\)](#), [showBuyerMenu\(\)](#), [showLoginMenu\(\)](#), and [showStatusMenu\(\)](#).

Referenced by [showBrowseMenu\(\)](#), [showStatusMenu\(\)](#), and [showWelcome\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.10 showProductMenu()

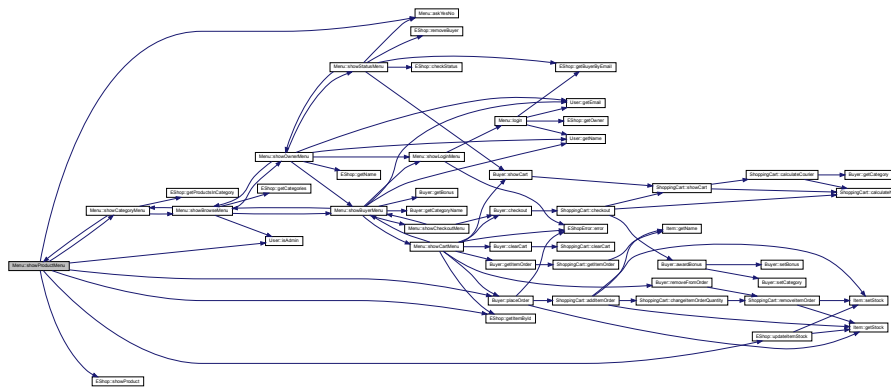
```
void Menu::showProductMenu (
    string back,
    int id )
```

Definition at line 203 of file [menu.cpp](#).

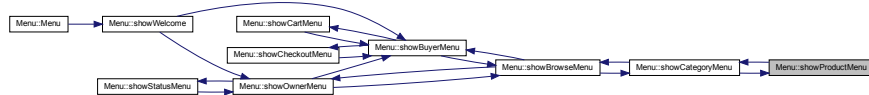
References [_buyer](#), [_eshop](#), [_user](#), [askYesNo\(\)](#), [EShop::getItemById\(\)](#), [User::isAdmin\(\)](#), [Buyer::placeOrder\(\)](#), [showCategoryMenu\(\)](#), [EShop::showProduct\(\)](#), and [EShop::updateItemStock\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.11 showStatusMenu()

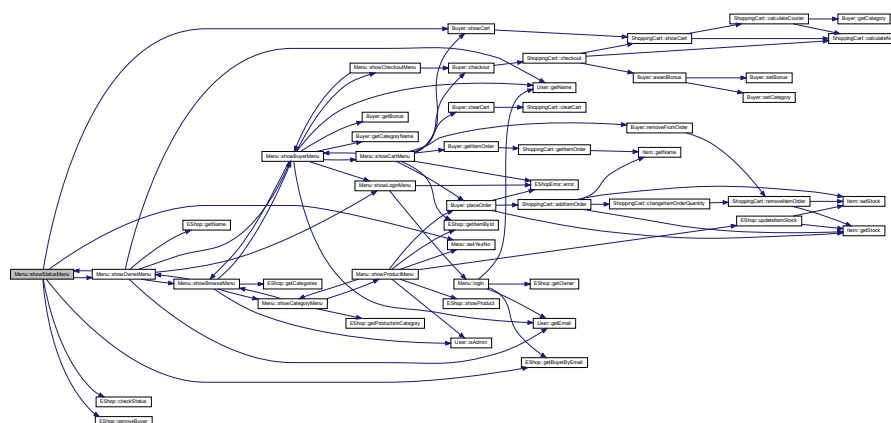
```
void Menu::showStatusMenu ( )
```

Definition at line 77 of file menu.cpp.

References `_eshop`, `askYesNo()`, `EShop::checkStatus()`, `EShop::getBuyerByEmail()`, `EShop::removeBuyer()`, `Buyer::showCart()`, and `showOwnerMenu()`.

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

Here is the call graph for this function:



5.5.4 Member Data Documentation

5.5.4.1 `_buyer`

```
Buyer* Menu::_buyer = nullptr [private]
```

Definition at line 47 of file [menu.h](#).

Referenced by [login\(\)](#), [showBuyerMenu\(\)](#), [showCartMenu\(\)](#), [showCheckoutMenu\(\)](#), and [showProductMenu\(\)](#).

5.5.4.2 `_eshop`

```
EShop* Menu::_eshop [private]
```

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

Referenced by [login\(\)](#), [Menu\(\)](#), [showBrowseMenu\(\)](#), [showCartMenu\(\)](#), [showCategoryMenu\(\)](#), [showOwnerMenu\(\)](#), [showProductMenu\(\)](#), [showStatusMenu\(\)](#), and [showWelcome\(\)](#).

5.5.4.3 `_owner`

```
Owner* Menu::_owner = nullptr [private]
```

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

Referenced by [login\(\)](#), and [showOwnerMenu\(\)](#).

5.5.4.4 `_user`

```
User* Menu::_user = nullptr [private]
```

Definition at line 45 of file [menu.h](#).

Referenced by [login\(\)](#), [showBrowseMenu\(\)](#), [showProductMenu\(\)](#), and [showWelcome\(\)](#).

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

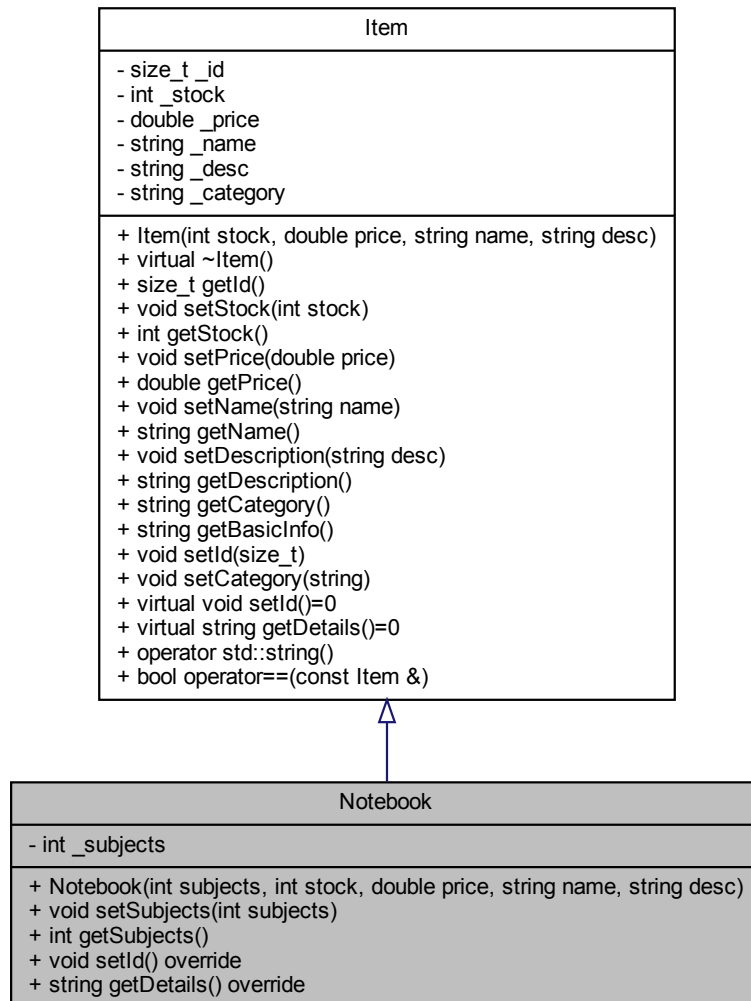
- [src/menu.h](#)
- [src/menu.cpp](#)

5.6 Notebook Class Reference

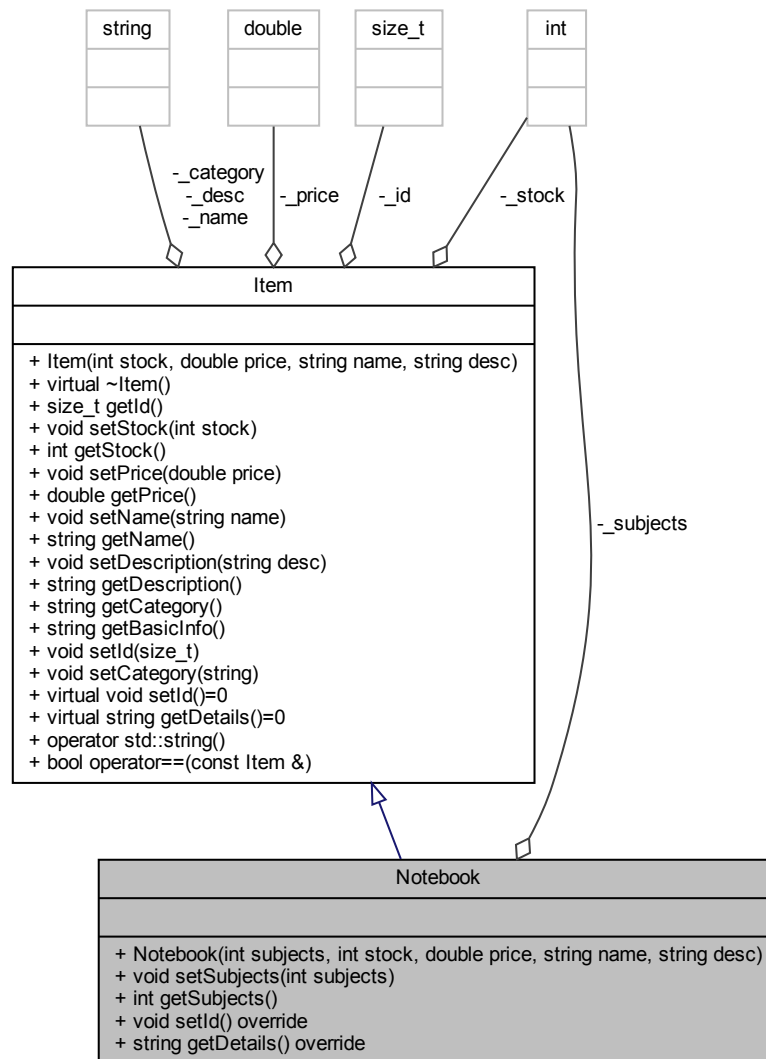
Class representing a [Notebook](#).

```
#include <notebook.h>
```

Inheritance diagram for Notebook:



Collaboration diagram for Notebook:



Public Member Functions

- **Notebook** (int subjects, int stock, double price, string name, string desc)
*Constructor for **Notebook**.*
- void **setSubjects** (int subjects)
Set the number of subjects of the notebook.
- int **getSubjects** ()
*Get the **Notebook**'s number of subjects.*
- void **setId** () override
Override of `Item::setId()`
- string **getDetails** () override
*Implements `Item::getDetails()` for **Notebook**.*

Private Attributes

- `int _subjects`

5.6.1 Detailed Description

Class representing a [Notebook](#).

Definition at line 12 of file [notebook.h](#).

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Notebook()

```

Notebook::Notebook (
    int subjects,
    int stock,
    double price,
    string name,
    string desc )

```

Constructor for [Notebook](#).

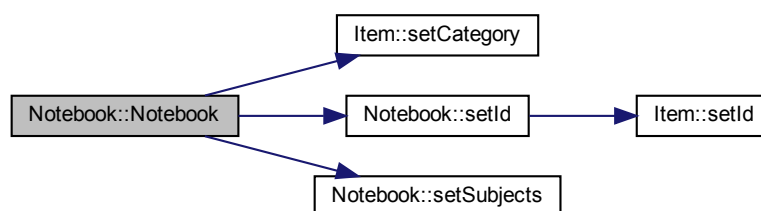
Parameters

<i>subjects</i>	<int>
<i>stock</i>	<int>
<i>price</i>	<double>
<i>name</i>	<string>
<i>desc</i>	<string>

Definition at line 4 of file [notebook.cpp](#).

References [Item::setCategory\(\)](#), [setId\(\)](#), and [setSubjects\(\)](#).

Here is the call graph for this function:



5.6.3 Member Function Documentation

5.6.3.1 getDetails()

```
string Notebook::getDetails ( ) [override], [virtual]
```

Implements [Item::getDetails\(\)](#) for [Notebook](#).

The result is a string of the number of subjects

Implements [Item](#).

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

References [_subjects](#).

5.6.3.2 getSubjects()

```
int Notebook::getSubjects ( )
```

Get the [Notebook](#)'s number of subjects.

Returns

<int>

Definition at line 31 of file [notebook.cpp](#).

References [_subjects](#).

5.6.3.3 setId()

```
void Notebook::setId ( ) [override], [virtual]
```

Override of `Item::setId()`

Computes the `Notebook`'s item id by generating the hashes of the class name and the number of subjects and then XOR'ing the hashes.

Implements `Item`.

Definition at line 21 of file `notebook.cpp`.

References `_subjects`, and `Item::setId()`.

Referenced by `Notebook()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.4 setSubjects()

```
void Notebook::setSubjects (
    int subjects )
```

Set the number of subjects of the notebook.

Parameters

<i>subjects</i>	<int>
-----------------	-------

Definition at line 29 of file [notebook.cpp](#).

References [_subjects](#).

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

Here is the caller graph for this function:



5.6.4 Member Data Documentation

5.6.4.1 `_subjects`

```
int Notebook::_subjects [private]
```

Definition at line 54 of file [notebook.h](#).

Referenced by [getDetails\(\)](#), [getSubjects\(\)](#), [setId\(\)](#), and [setSubjects\(\)](#).

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

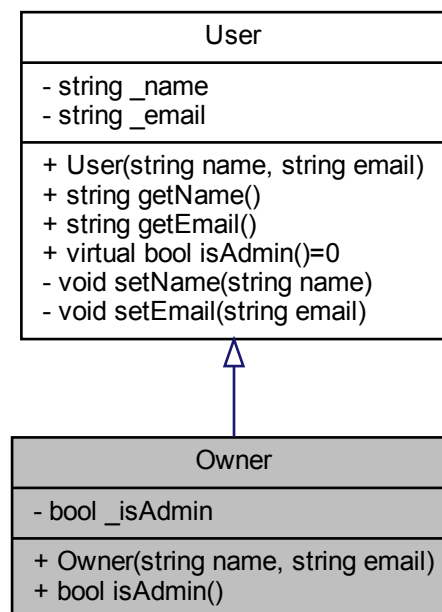
- [src/notebook.h](#)
- [src/notebook.cpp](#)

5.7 Owner Class Reference

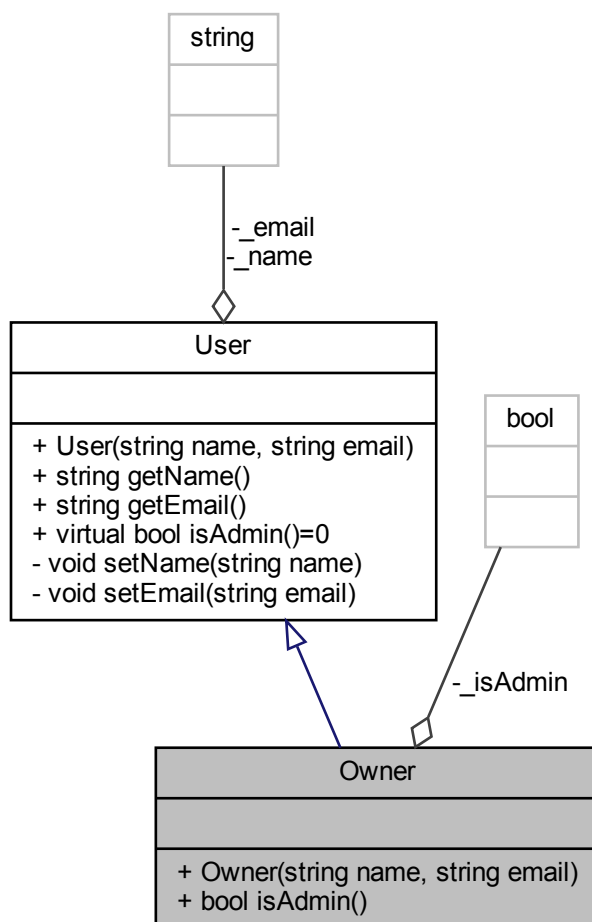
Specialization of [User](#). Describes an [Owner](#).

```
#include <owner.h>
```

Inheritance diagram for Owner:



Collaboration diagram for Owner:



Public Member Functions

- [Owner](#) (string name, string email)
Constructor of [Owner](#).
- bool [isAdmin](#) ()
Impementation of [isAdmin\(\)](#) of [User](#).

Private Attributes

- bool [_isAdmin](#)
This was requested by the project, it is pointless.

5.7.1 Detailed Description

Specialization of [User](#). Describes an [Owner](#).

Derivative class to specialize a [User](#). Implements [Owner](#) related functionality. Basically nothing because I didn't look forward enough Things like managing Buyers could be in here

Definition at line 17 of file [owner.h](#).

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Owner()

```
Owner::Owner (
    string name,
    string email )
```

Constructor of [Owner](#).

Parameters

<i>name</i>	Owner's name
<i>email</i>	Owner's login email

Definition at line 3 of file [owner.cpp](#).

References [_isAdmin](#).

5.7.3 Member Function Documentation

5.7.3.1 isAdmin()

```
bool Owner::isAdmin ( ) [virtual]
```

Implementation of [isAdmin\(\)](#) of [User](#).

Parameters

<i>none</i>	
-------------	--

Returns

bool Always true

Implements [User](#).

Definition at line 9 of file [owner.cpp](#).

References [_isAdmin](#).

5.7.4 Member Data Documentation

5.7.4.1 `_isAdmin`

```
bool Owner::_isAdmin [private]
```

This was requested by the project, it is pointless.

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

Referenced by [isAdmin\(\)](#), and [Owner\(\)](#).

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

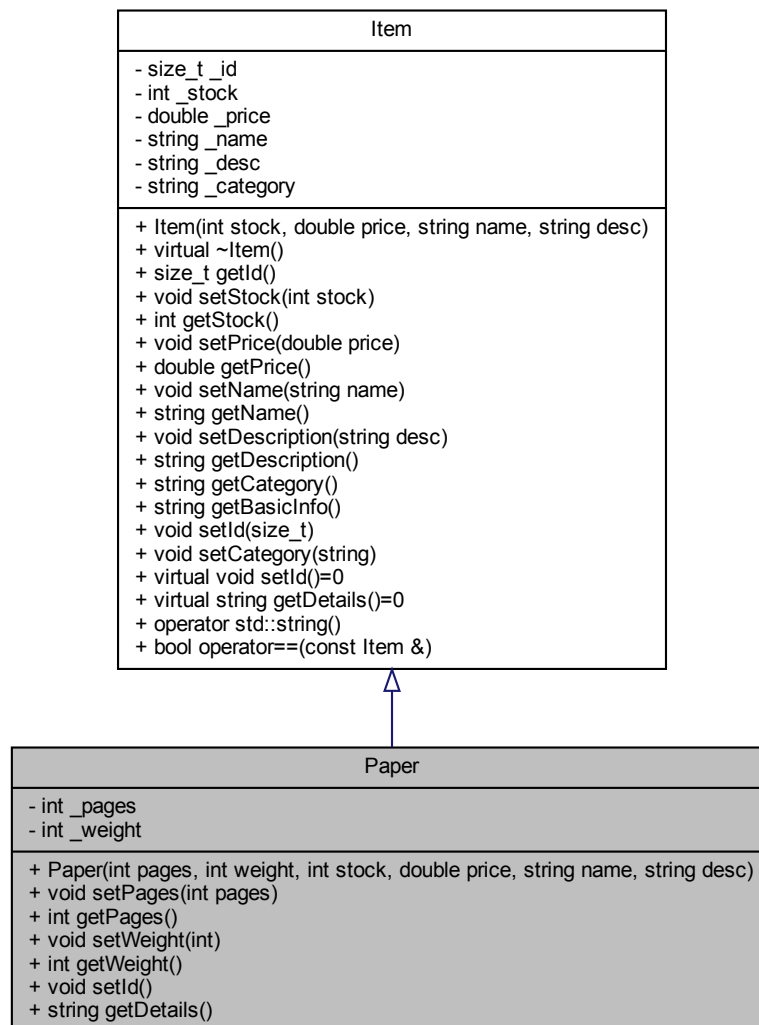
- [src/owner.h](#)
- [src/owner.cpp](#)

5.8 Paper Class Reference

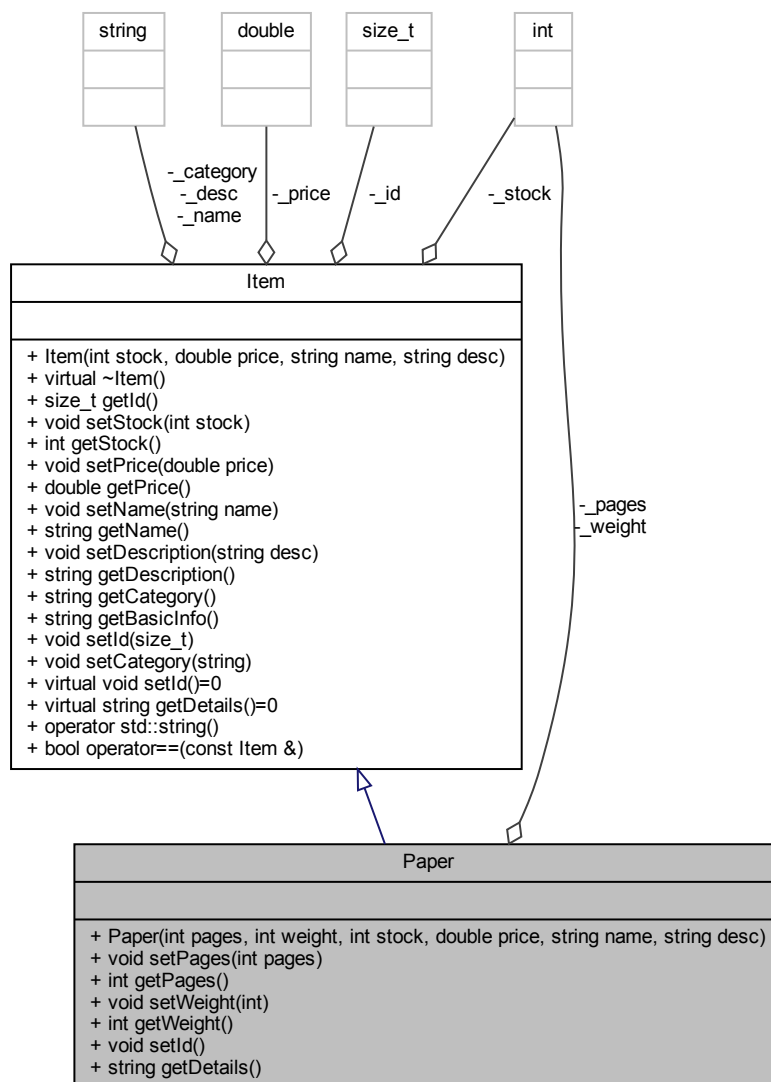
Class representing a [Paper](#).

```
#include <paper.h>
```

Inheritance diagram for Paper:



Collaboration diagram for Paper:



Public Member Functions

- [Paper](#) (int pages, int weight, int stock, double price, string name, string desc)
Constructor for [Paper](#).
- void [setPages](#) (int pages)
Set the pages of [Paper](#).
- int [getPages](#) ()
Get the pages of [Paper](#).
- void [setWeight](#) (int)
Set the weight of [Paper](#).
- int [getWeight](#) ()
Get the weight of [Paper](#).
- void [setId](#) ()

Override of `Item::setID()`

- string `getDetails()`

Implements `Item::getDetails()` for `Paper`.

Private Attributes

- int `_pages`
- int `_weight`

5.8.1 Detailed Description

Class representing a `Paper`.

Definition at line 12 of file `paper.h`.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 `Paper()`

```
Paper::Paper (
    int pages,
    int weight,
    int stock,
    double price,
    string name,
    string desc )
```

Constructor for `Paper`.

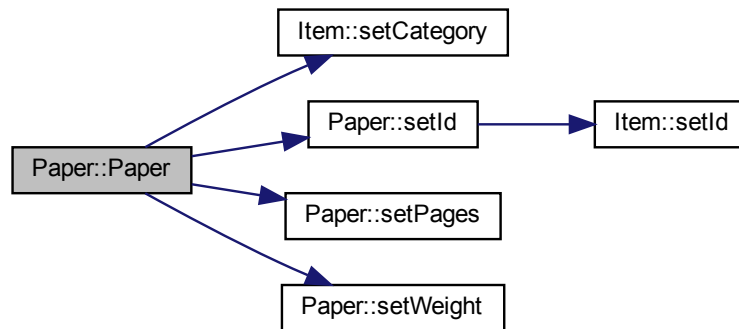
Parameters

<i>int</i>	<pages>
<i>int</i>	<weight>
<i>int</i>	<stock>
<i>double</i>	<price>
<i>string</i>	<name>
<i>string</i>	<desc>

Definition at line 5 of file `paper.cpp`.

References `Item::setCategory()`, `setId()`, `setPages()`, and `setWeight()`.

Here is the call graph for this function:



5.8.3 Member Function Documentation

5.8.3.1 `getDetails()`

```
string Paper::getDetails ( ) [virtual]
```

Implements [Item::getDetails\(\)](#) for [Paper](#).

The result is a comma-space-separated string in the order of "pages, weight"

Implements [Item](#).

Definition at line 13 of file [paper.cpp](#).

References [_pages](#), and [_weight](#).

5.8.3.2 `getPages()`

```
int Paper::getPages ( )
```

Get the pages of [Paper](#).

Returns

<int>

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

References [_pages](#).

5.8.3.3 `getWeight()`

```
int Paper::getWeight ( )
```

Get the weight of [Paper](#).

Returns

<int>

Definition at line 39 of file [paper.cpp](#).

References [_weight](#).

5.8.3.4 `setId()`

```
void Paper::setId ( ) [virtual]
```

Override of [Item::setId\(\)](#)

Computes the [Paper](#)'s item ID by generating the hashes of the class name, the number of pages and the weight and then XOR'ing the hashes.

Implements [Item](#).

Definition at line 23 of file [paper.cpp](#).

References [_pages](#), [_weight](#), and [Item::setId\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.3.5 setPages()

```
void Paper::setPages (
    int pages )
```

Set the pages of [Paper](#).

Parameters

<i>pages</i>	<int>
--------------	-------

Definition at line 32 of file [paper.cpp](#).

References [_pages](#).

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

Here is the caller graph for this function:

**5.8.3.6 setWeight()**

```
void Paper::setWeight (
    int weight )
```

Set the weight of [Paper](#).

Parameters

<i>subjects</i>	<int>
-----------------	-------

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

References [_weight](#).

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

Here is the caller graph for this function:



5.8.4 Member Data Documentation

5.8.4.1 `_pages`

```
int Paper::_pages [private]
```

Definition at line 69 of file [paper.h](#).

Referenced by [getDetails\(\)](#), [getPages\(\)](#), [setId\(\)](#), and [setPages\(\)](#).

5.8.4.2 `_weight`

```
int Paper::_weight [private]
```

Definition at line 70 of file [paper.h](#).

Referenced by [getDetails\(\)](#), [getWeight\(\)](#), [setId\(\)](#), and [setWeight\(\)](#).

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

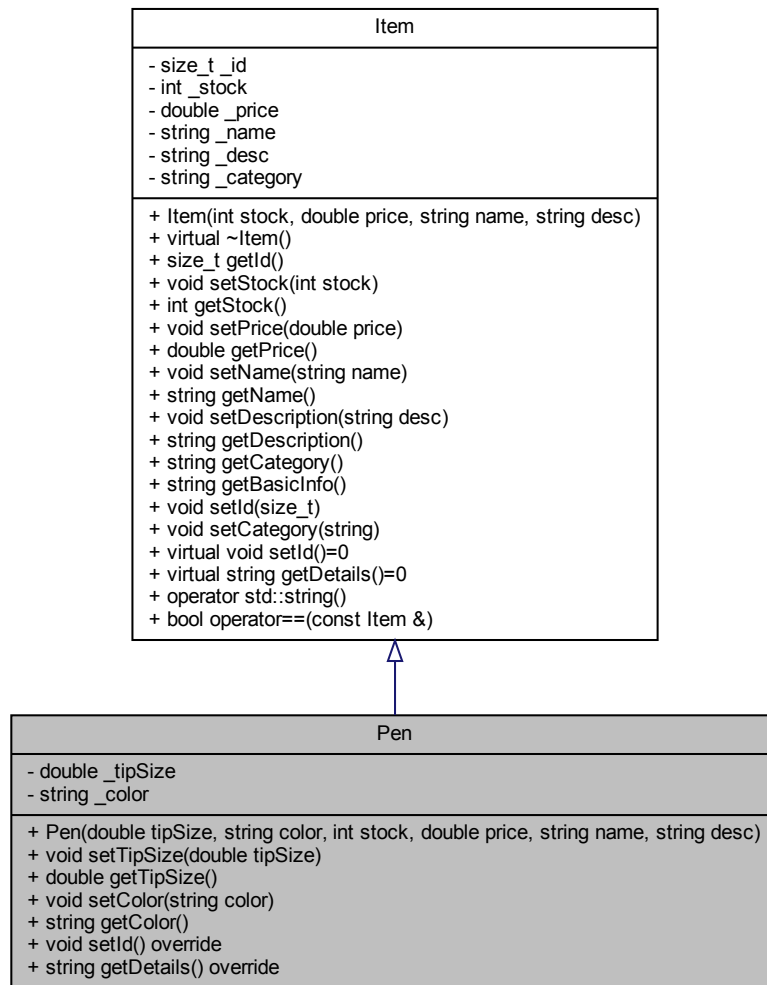
- [src/paper.h](#)
- [src/paper.cpp](#)

5.9 Pen Class Reference

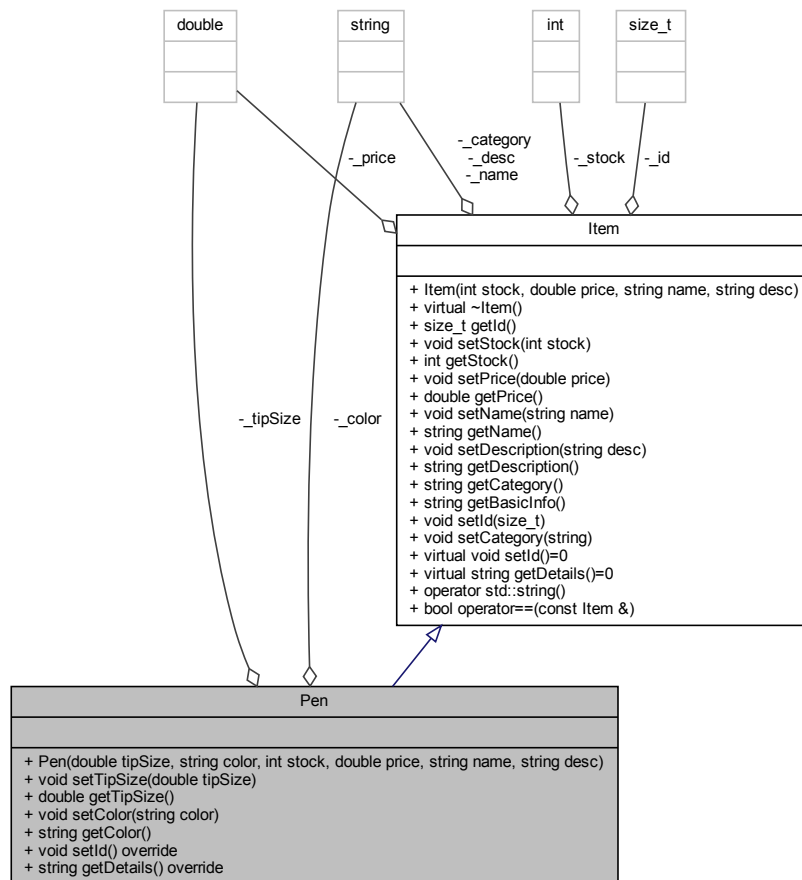
Class representing a [Pen](#).

```
#include <pen.h>
```

Inheritance diagram for Pen:



Collaboration diagram for Pen:



Public Member Functions

- **Pen** (double tipSize, string color, int stock, double price, string name, string desc)
*Constructor for **Pen**.*
- void **setTipSize** (double tipSize)
*Set the tip size of the **Pen**.*
- double **getTipSize** ()
*Get the tip size of the **Pen**.*
- void **setColor** (string color)
*Set the color of the **Pen**.*
- string **getColor** ()
*Get the color of the **Pen**.*
- void **setId** () override
*Override of **Item::setId()***
- string **getDetails** () override
*Implements **Item::getDetails()** for **Pen**.*

Private Attributes

- double **_tipSize**
- string **_color**

5.9.1 Detailed Description

Class representing a [Pen](#).

Definition at line 12 of file [pen.h](#).

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Pen()

```
Pen::Pen (
    double tipSize,
    string color,
    int stock,
    double price,
    string name,
    string desc )
```

Constructor for [Pen](#).

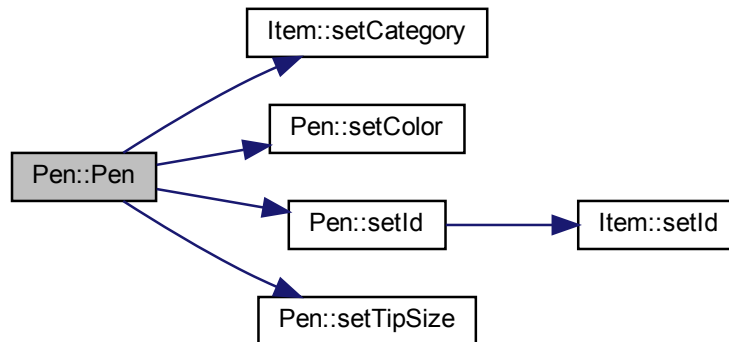
Parameters

<i>tipSize</i>	<double>
<i>color</i>	<string>
<i>stock</i>	<int>
<i>price</i>	<double>
<i>name</i>	<string>
<i>desc</i>	<string>

Definition at line 4 of file [pen.cpp](#).

References [Item::setCategory\(\)](#), [setColor\(\)](#), [setId\(\)](#), and [setTipSize\(\)](#).

Here is the call graph for this function:



5.9.3 Member Function Documentation

5.9.3.1 getColor()

```
string Pen::getColor ( )
```

Get the color of the [Pen](#).

Returns

<string>

Definition at line 40 of file [pen.cpp](#).

References [_color](#).

5.9.3.2 getDetails()

```
string Pen::getDetails ( ) [override], [virtual]
```

Implements [Item::getDetails\(\)](#) for [Pen](#).

The result is a commaspaced string in the order of "tip size, color"

Implements [Item](#).

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

References [_color](#), and [_tipSize](#).

5.9.3.3 `getTipSize()`

```
double Pen::getTipSize ( )
```

Get the tip size of the [Pen](#).

Returns

<double>

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

References [_tipSize](#).

5.9.3.4 `setColor()`

```
void Pen::setColor (
    string color )
```

Set the color of the [Pen](#).

Parameters

<i>color</i>	<string>
--------------	----------

Definition at line 38 of file [pen.cpp](#).

References [_color](#).

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

Here is the caller graph for this function:



5.9.3.5 setId()

```
void Pen::setId ( ) [override], [virtual]
```

Override of [Item::setId\(\)](#)

Computes the [Pen](#)'s item ID by generating the hashes of the class name, the tipSize and the color and then XOR'ing the hashes.

Implements [Item](#).

Definition at line 13 of file [pen.cpp](#).

References [_color](#), [_tipSize](#), and [Item::setId\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.3.6 setTipSize()

```
void Pen::setTipSize (
    double tipSize )
```

Set the tip size of the [Pen](#).

Parameters

<i>tipSize</i>	<double>
----------------	----------

Definition at line 33 of file [pen.cpp](#).

References [_tipSize](#).

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

Here is the caller graph for this function:



5.9.4 Member Data Documentation

5.9.4.1 `_color`

```
string Pen::_color [private]
```

Definition at line 70 of file [pen.h](#).

Referenced by [getColor\(\)](#), [getDetails\(\)](#), [setColor\(\)](#), and [setId\(\)](#).

5.9.4.2 `_tipSize`

```
double Pen::_tipSize [private]
```

Definition at line 69 of file [pen.h](#).

Referenced by [getDetails\(\)](#), [getTipSize\(\)](#), [setId\(\)](#), and [setTipSize\(\)](#).

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

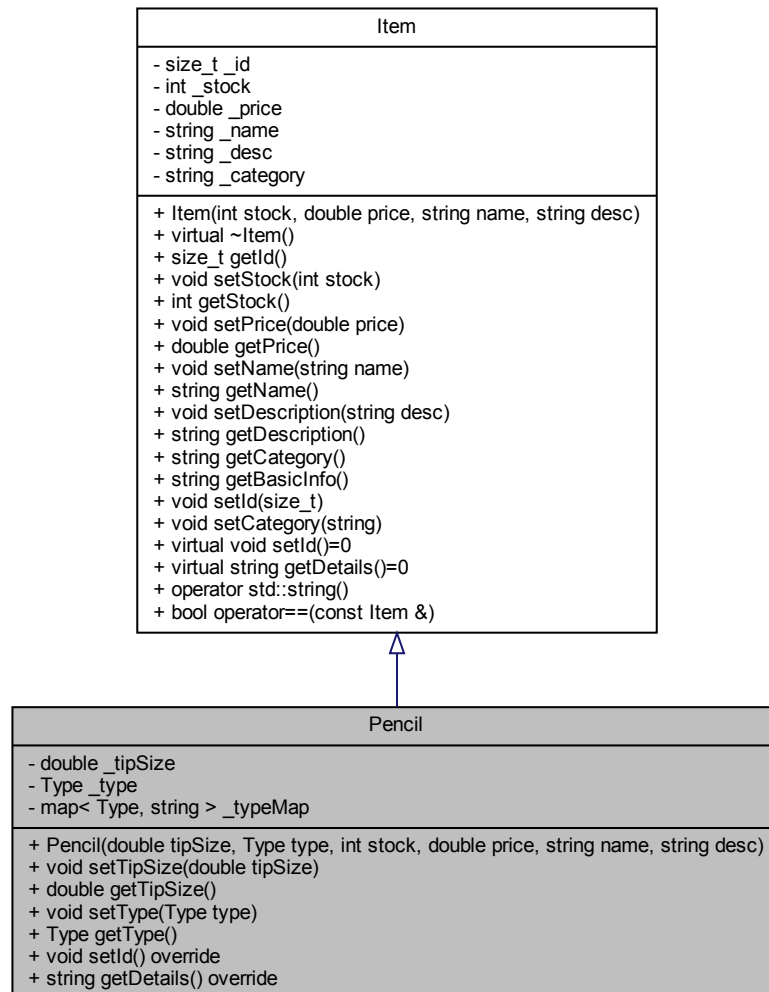
- [src/pen.h](#)
- [src/pen.cpp](#)

5.10 Pencil Class Reference

Class representing a [Pencil](#).

```
#include <pencil.h>
```

Inheritance diagram for Pencil:



Private Attributes

- double [_tipSize](#)
- [Type](#) [_type](#)
- map< [Type](#), string > [_typeMap](#) { {H, "H"}, { B, "B" }, {HB, "HB" } }

Map of [Pencil::Type](#) to string representations.

5.10.1 Detailed Description

Class representing a [Pencil](#).

Definition at line 11 of file [pencil.h](#).

5.10.2 Member Enumeration Documentation

5.10.2.1 Type

```
enum Pencil::Type
```

Enumeration of [Pencil](#) types.

Enumerator

H	
B	
HB	

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

5.10.3 Constructor & Destructor Documentation

5.10.3.1 Pencil()

```
Pencil::Pencil (  
    double tipSize,  
    Pencil::Type type,  
    int stock,  
    double price,  
    string name,  
    string desc )
```

Constructor for [Pencil](#).

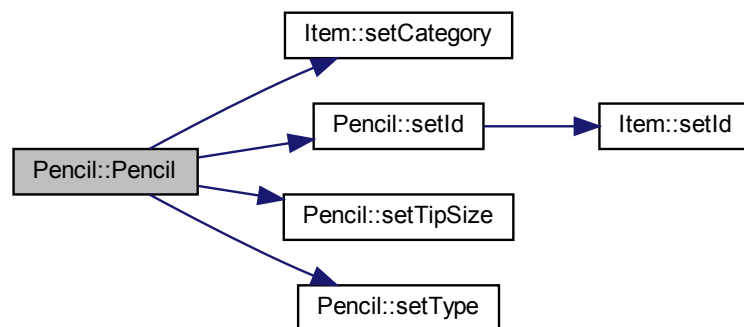
Parameters

<i>tipSize</i>	<double>
<i>type</i>	< Pencil::Type >
<i>stock</i>	<int>
<i>price</i>	<double>
<i>name</i>	<string>
<i>desc</i>	<string>

Definition at line 4 of file [pencil.cpp](#).

References [Item::setCategory\(\)](#), [setId\(\)](#), [setTipSize\(\)](#), and [setType\(\)](#).

Here is the call graph for this function:



5.10.4 Member Function Documentation

5.10.4.1 `getDetails()`

```
string Pencil::getDetails ( ) [override], [virtual]
```

Implements [Item::getDetails\(\)](#) for [Pen](#).

The result is a comma-separated string in the order of "tip size, type"

Implements [Item](#).

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

References [_tipSize](#), [_type](#), and [_typeMap](#).

5.10.4.2 `getTipSize()`

```
double Pencil::getTipSize ( )
```

Get the tip size of the [Pencil](#).

Returns

<double>

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

References [_tipSize](#).

5.10.4.3 `getType()`

```
Pencil::Type Pencil::getType ( )
```

Get the color of the [Pen](#).

Returns

<[Pencil::Type](#)>

Definition at line 40 of file [pencil.cpp](#).

References [_type](#).

5.10.4.4 `setId()`

```
void Pencil::setId ( ) [override], [virtual]
```

Override of [Item::setId\(\)](#)

Computes the [Pencil](#)'s item ID by generating the hashes of the class name, the tip size and the type and then XOR'ing the hashes.

Implements [Item](#).

Definition at line 13 of file [pencil.cpp](#).

References [_tipSize](#), [_type](#), [_typeMap](#), and [Item::setId\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.4.5 setTipSize()

```
void Pencil::setTipSize (  
    double tipSize )
```

Set the tip size of the [Pencil](#).

Parameters

<i>tipSize</i>	<double>
----------------	----------

Definition at line 33 of file [pencil.cpp](#).

References [_tipSize](#).

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

Here is the caller graph for this function:



5.10.4.6 setType()

```
void Pencil::setType (
    Type type )
```

Set the type of the [Pencil](#).

Parameters

<i>color</i>	< Pencil::Type >
--------------	----------------------------------

Definition at line 38 of file [pencil.cpp](#).

References [_type](#).

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

Here is the caller graph for this function:



5.10.5 Member Data Documentation

5.10.5.1 _tipSize

```
double Pencil::_tipSize [private]
```

Definition at line 78 of file [pencil.h](#).

Referenced by [getDetails\(\)](#), [getTipSize\(\)](#), [setId\(\)](#), and [setTipSize\(\)](#).

5.10.5.2 _type

```
Type Pencil::_type [private]
```

Definition at line 79 of file [pencil.h](#).

Referenced by [getDetails\(\)](#), [getType\(\)](#), [setId\(\)](#), and [setType\(\)](#).

5.10.5.3 _typeMap

```
map<Type, string> Pencil::_typeMap { {H, "H"}, {B, "B"}, {HB, "HB"} } [private]
```

Map of [Pencil::Type](#) to string representations.

Definition at line 81 of file [pencil.h](#).

Referenced by [getDetails\(\)](#), and [setId\(\)](#).

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

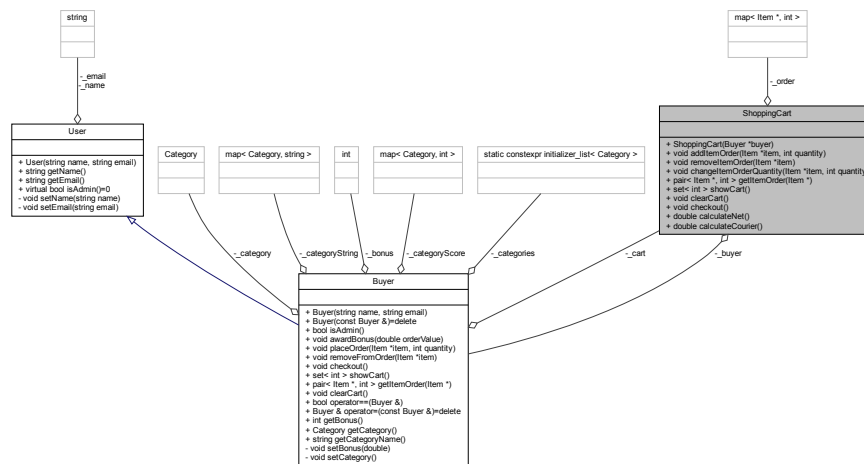
- [src/pencil.h](#)
- [src/pencil.cpp](#)

5.11 ShoppingCart Class Reference

Class implementing the shopping cart.

```
#include <shoppingcart.h>
```

Collaboration diagram for ShoppingCart:



Public Member Functions

- [ShoppingCart](#) ([Buyer](#) *buyer)
Constructor for [ShoppingCart](#).
- void [addItemOrder](#) ([Item](#) *item, int quantity)
Add quantity of an [Item](#) to the cart.
- void [removeItemOrder](#) ([Item](#) *item)
Remove an item from the cart completely.
- void [changeItemOrderQuantity](#) ([Item](#) *item, int quantity)
Changes the quantity of an item in the cart.
- pair< [Item](#) *, int > [getItemOrder](#) ([Item](#) *)

- Get the order information of an item in the cart.*
 - set< int > [showCart](#) ()
- Show the contents of the cart.*
 - void [clearCart](#) ()
- Clears the cart.*
 - void [checkout](#) ()
- Performs the checkout.*
 - double [calculateNet](#) ()
- Calculates the cost of the order.*
 - double [calculateCourier](#) ()
- Calculates the cost of the courier.*

Private Attributes

- map< [Item](#) *, int > [_order](#)
 - The cart represented as a map.*
- [Buyer](#) * [_buyer](#)
 - The [Buyer](#) this cart belongs to.*

5.11.1 Detailed Description

Class implementing the shopping cart.

This class implements the shopping cart related functionality. This class is instantiated for each buyer, and requires to access functionality related to that specific buyer. To satisfy that need we pass a pointer to the constructing buyer during instantiation for later use. The cart is represented as a map between the [Item](#) and an integer representing the quantity of [Item](#) in the cart.

Definition at line 22 of file [shoppingcart.h](#).

5.11.2 Constructor & Destructor Documentation

5.11.2.1 ShoppingCart()

```
ShoppingCart::ShoppingCart (
    Buyer * buyer ) [explicit]
```

Constructor for [ShoppingCart](#).

Parameters

<i>buyer</i>	<Buyer*> Pointer to the buyer owning this cart
--------------	--

Definition at line 5 of file [shoppingcart.cpp](#).

References [_buyer](#).

5.11.3 Member Function Documentation

5.11.3.1 addItemOrder()

```
void ShoppingCart::addItemOrder (
    Item * item,
    int quantity )
```

Add quantity of an [Item](#) to the cart.

This function also check if the [Item](#) is already in the cart, if it is it updates the quantity by adding the requested quantity. It also checks if there is enough stock, if it doesn't, it throws and [EShopError](#) exception to be caught by the caller.

Parameters

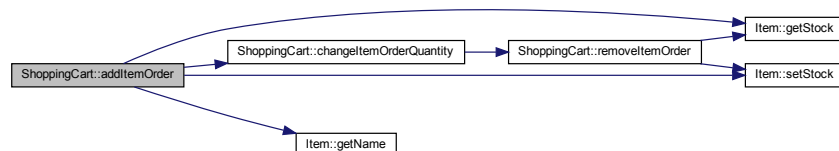
<i>item</i>	<Item*> Reference to an item in the eshop
<i>quantity</i>	<int> The quantity of the item

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

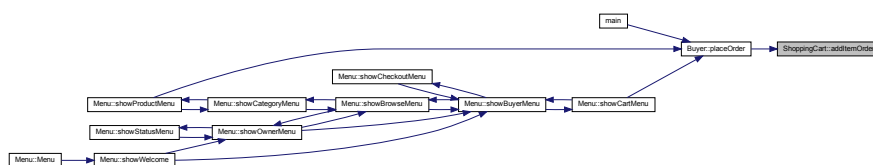
References [_order](#), [changeItemOrderQuantity\(\)](#), [Item::getName\(\)](#), [Item::getStock\(\)](#), and [Item::setStock\(\)](#).

Referenced by [Buyer::placeOrder\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.2 calculateCourier()

```
double ShoppingCart::calculateCourier ( )
```

Calculates the cost of the courier.

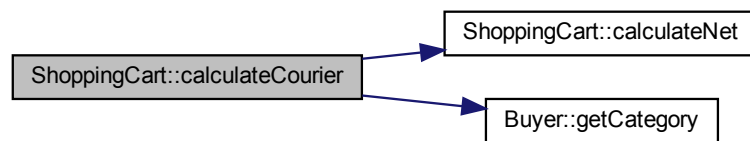
It also takes into account the category of the buyer.

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

References [_buyer](#), [calculateNet\(\)](#), [Buyer::getCategory\(\)](#), [Buyer::Gold](#), and [Buyer::Silver](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.3 calculateNet()

```
double ShoppingCart::calculateNet ( )
```

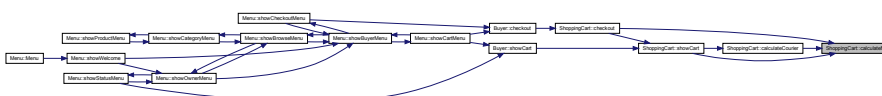
Calculates the cost of the order.

Definition at line 53 of file [shoppingcart.cpp](#).

References [_order](#).

Referenced by [calculateCourier\(\)](#), [checkout\(\)](#), and [showCart\(\)](#).

Here is the caller graph for this function:



5.11.3.4 `changeItemOrderQuantity()`

```
void ShoppingCart::changeItemOrderQuantity (
    Item * item,
    int quantity )
```

Changes the quantity of an item in the cart.

Also checks if the resulting quantity is below zero and if it is it removes the `Item` from the cart.

Parameters

<i>item</i>	<Item*> Reference to the <code>Item</code> to be removed.
<i>quantity</i>	<int> Quantity to remove.

Definition at line 35 of file `shoppingcart.cpp`.

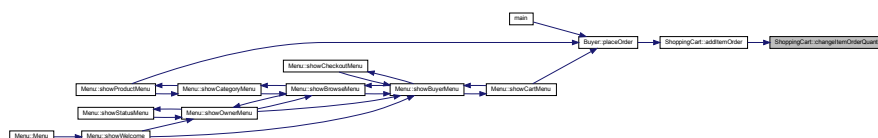
References `_order`, and `removeItemOrder()`.

Referenced by `addItemOrder()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.5 `checkout()`

```
void ShoppingCart::checkout ( )
```

Performs the checkout.

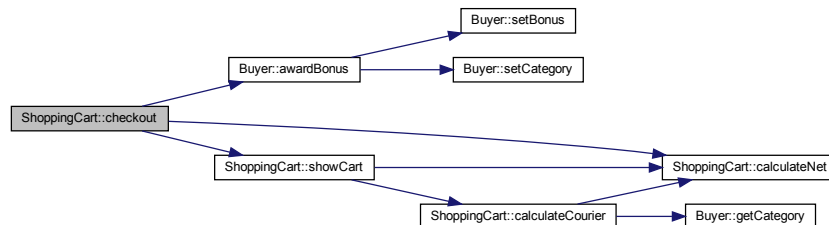
Performs the checkout, asks the user for confirmation. If it is positive it clears the cart and awards the bonus to the user.

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

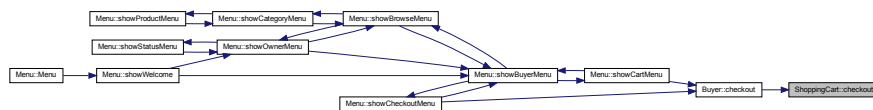
References [_buyer](#), [_order](#), [Buyer::awardBonus\(\)](#), [calculateNet\(\)](#), and [showCart\(\)](#).

Referenced by [Buyer::checkout\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.6 clearCart()

```
void ShoppingCart::clearCart ( )
```

Clears the cart.

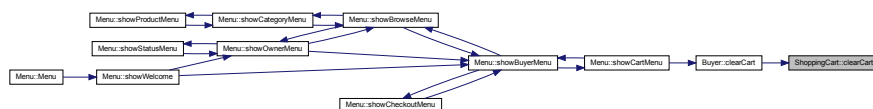
It doesn't call `removeItemOrder` because of the use of an iterator.

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

References [_order](#).

Referenced by [Buyer::clearCart\(\)](#).

Here is the caller graph for this function:



5.11.3.7 getItemOrder()

```
pair< Item *, int > ShoppingCart::getItemOrder (
    Item * item )
```

Get the order information of an item in the cart.

Returns a pair of `Item` reference and quantity of the item specified if the item is in the cart.

Parameters

<i>item</i>	<Item*> Reference to the Item to be returned.
-------------	---

Returns

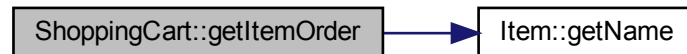
<pair<Item*,int>> A pair from the cart.

Definition at line 42 of file [shoppingcart.cpp](#).

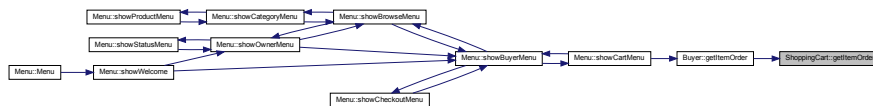
References [_order](#), and [Item::getName\(\)](#).

Referenced by [Buyer::getItemOrder\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.8 removeItemOrder()

```
void ShoppingCart::removeItemOrder (
    Item * item )
```

Remove an item from the cart completely.

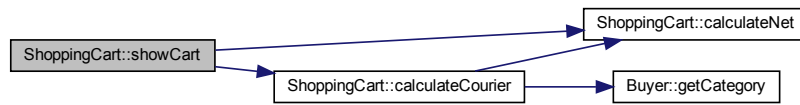
Also updates the cart.

Parameters

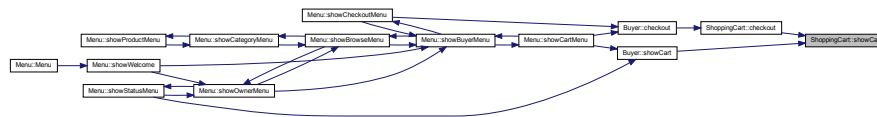
<i>item</i>	<Item*> Reference to the Item to be removed.
-------------	--

Definition at line 26 of file [shoppingcart.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.4 Member Data Documentation

5.11.4.1 `_buyer`

```
Buyer* ShoppingCart::_buyer [private]
```

The [Buyer](#) this cart belongs to.

Definition at line 114 of file [shoppingcart.h](#).

Referenced by [calculateCourier\(\)](#), [checkout\(\)](#), and [ShoppingCart\(\)](#).

5.11.4.2 `_order`

```
map<Item*, int> ShoppingCart::_order [private]
```

The cart represented as a map.

Definition at line 112 of file [shoppingcart.h](#).

Referenced by [addItemOrder\(\)](#), [calculateNet\(\)](#), [changeItemOrderQuantity\(\)](#), [checkout\(\)](#), [clearCart\(\)](#), [getItemOrder\(\)](#), [removeItemOrder\(\)](#), and [showCart\(\)](#).

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

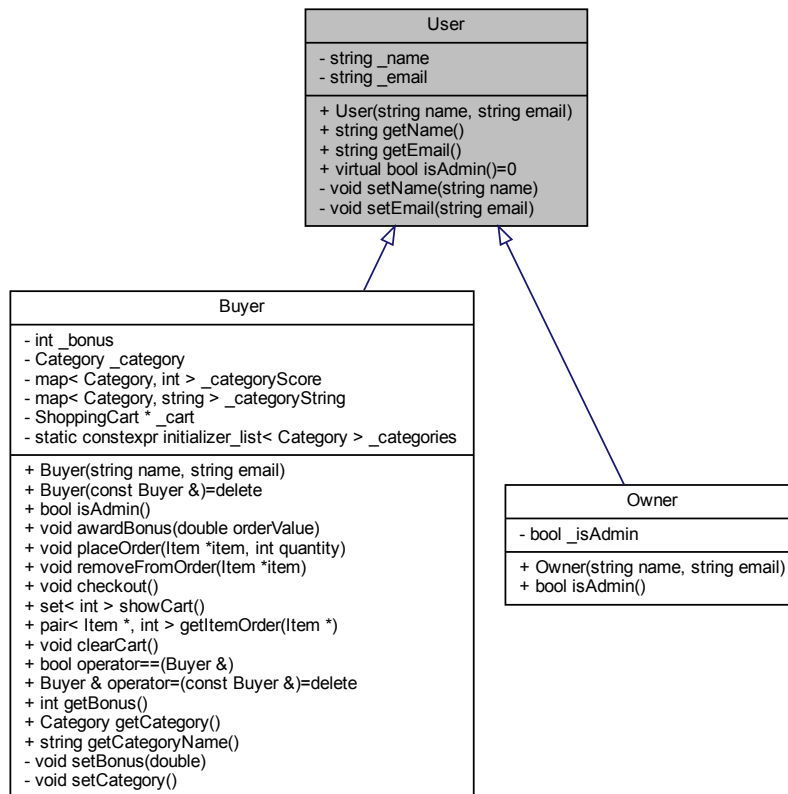
- [src/shoppingcart.h](#)
- [src/shoppingcart.cpp](#)

5.12 User Class Reference

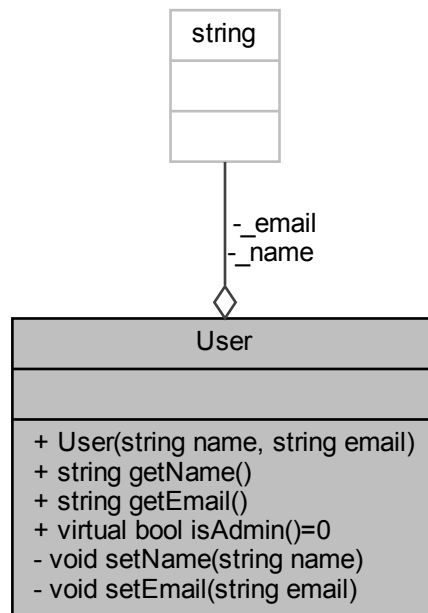
Base class for all users.

```
#include <user.h>
```

Inheritance diagram for User:



Collaboration diagram for User:



Public Member Functions

- `User` (string name, string email)
Constructor for `User`, called by `Owner` and `Buyer`.
- string `getName` ()
Get the Name of the user.
- string `getEmail` ()
Get the Email of the user.
- virtual bool `isAdmin` ()=0
Check if the user is an admin.

Private Member Functions

- void `setName` (string name)
- void `setEmail` (string email)

Private Attributes

- string `_name`
- string `_email`

5.12.1 Detailed Description

Base class for all users.

Base abstract class to subclassed by the [Owner](#) and [Buyer](#) specialized classes. Implements the common functions of setting the Name and the Email of the user.

Definition at line 15 of file [user.h](#).

5.12.2 Constructor & Destructor Documentation

5.12.2.1 User()

```
User::User (
    string name,
    string email )
```

Constructor for [User](#), called by [Owner](#) and [Buyer](#).

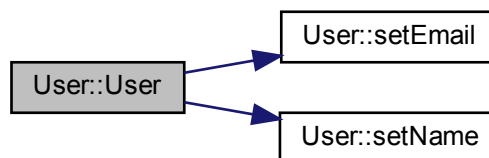
Parameters

<i>name</i>	string. The name of the user
<i>email</i>	string. The email of the user

Definition at line 3 of file [user.cpp](#).

References [setEmail\(\)](#), and [setName\(\)](#).

Here is the call graph for this function:



5.12.3 Member Function Documentation

5.12.3.1 getEmail()

```
string User::getEmail ( )
```

Get the Email of the user.

Parameters

<i>none</i>	
-------------	--

Returns

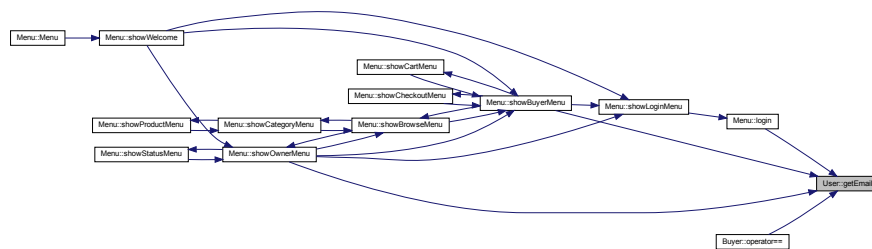
string The user's email

Definition at line 19 of file [user.cpp](#).

References [_email](#).

Referenced by [Menu::login\(\)](#), [Buyer::operator==\(\)](#), [Menu::showBuyerMenu\(\)](#), and [Menu::showOwnerMenu\(\)](#).

Here is the caller graph for this function:



5.12.3.2 getName()

```
string User::getName ( )
```

Get the Name of the user.

Parameters

<i>none</i>	
-------------	--

Returns

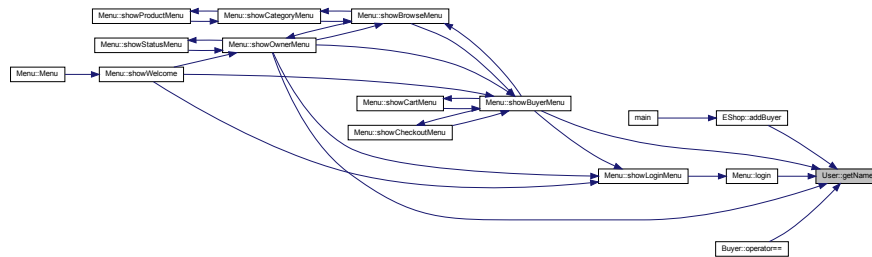
string The user's name

Definition at line 13 of file [user.cpp](#).

References [_name](#).

Referenced by [EShop::addBuyer\(\)](#), [Menu::login\(\)](#), [Buyer::operator==\(\)](#), [Menu::showBuyerMenu\(\)](#), and [Menu::showOwnerMenu\(\)](#).

Here is the caller graph for this function:



5.12.3.3 isAdmin()

```
virtual bool User::isAdmin ( ) [pure virtual]
```

Check if the user is an admin.

Virtual function implemented in the derivative classes

Parameters

<i>none</i>

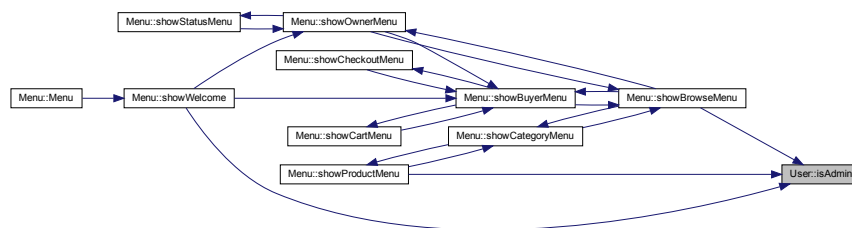
Returns

bool True if the user is an owner, false otherwise.

Implemented in [Owner](#), and [Buyer](#).

Referenced by [Menu::showBrowseMenu\(\)](#), [Menu::showProductMenu\(\)](#), and [Menu::showWelcome\(\)](#).

Here is the caller graph for this function:



5.12.3.4 setEmail()

```
void User::setEmail (
    string email ) [private]
```

Definition at line 16 of file [user.cpp](#).

References [_email](#).

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

Here is the caller graph for this function:



5.12.3.5 setName()

```
void User::setName (
    string name ) [private]
```

Definition at line 10 of file [user.cpp](#).

References [_name](#).

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

Here is the caller graph for this function:



5.12.4 Member Data Documentation

5.12.4.1 `_email`

```
string User::_email [private]
```

Definition at line 57 of file [user.h](#).

Referenced by [getEmail\(\)](#), and [setEmail\(\)](#).

5.12.4.2 `_name`

```
string User::_name [private]
```

Definition at line 56 of file [user.h](#).

Referenced by [getName\(\)](#), and [setName\(\)](#).

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

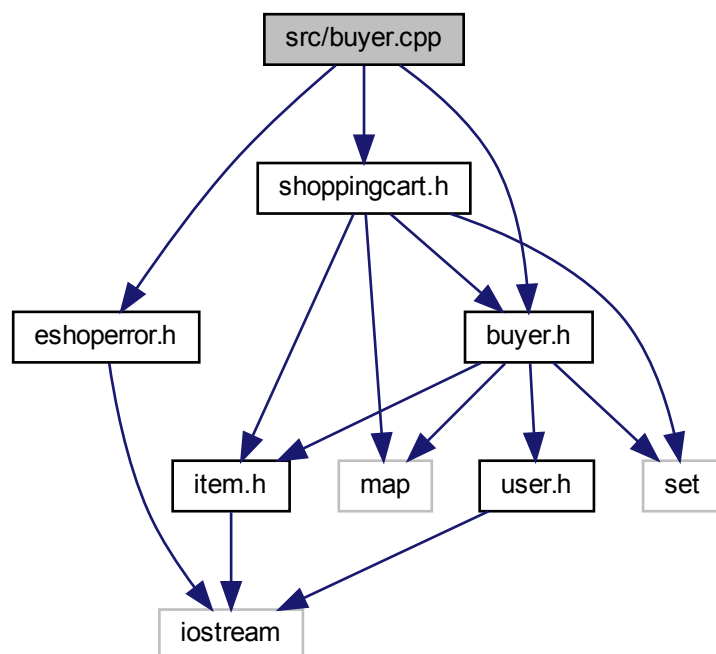
- [src/user.h](#)
- [src/user.cpp](#)

Chapter 6

File Documentation

6.1 src/buyer.cpp File Reference

```
#include "buyer.h"  
#include "shoppingcart.h"  
#include "eshoperror.h"  
Include dependency graph for buyer.cpp:
```



6.2 buyer.cpp

```

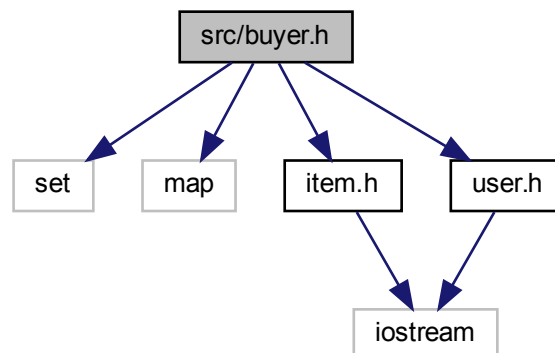
00001 #include "buyer.h"
00002 #include "shoppingcart.h"
00003 #include "eshoperror.h"
00004
00005 Buyer::Buyer(string name, string email) : User(name, email)
00006 {
00007     _bonus = 0;
00008     _category = Bronze;
00009     _cart = new ShoppingCart(this);
00010 }
00011
00012 void
00013 Buyer::awardBonus(double orderValue)
00014 {
00015     setBonus(orderValue);
00016     setCategory();
00017 }
00018
00019 void
00020 Buyer::placeOrder(Item* item, int quantity)
00021 {
00022     try {
00023         _cart->addItemOrder(item, quantity);
00024     } catch (const EShopError& e) {
00025         cout << e.error() << endl;
00026         cout << "Available quantity is " + to_string(item->getStock()) << endl;
00027         string ans;
00028         do {
00029             cout << "Do you want to add the available quantity to the cart? [y/n]: ";
00030             cin >> ans;
00031         }
00032         while( !cin.fail() && ans!="y" && ans!="n" );
00033
00034         if (ans == "y")
00035             _cart->addItemOrder(item, item->getStock());
00036     }
00037 }
00038
00039 void
00040 Buyer::removeFromOrder(Item* item)
00041 {
00042     _cart->removeItemOrder(item);
00043 }
00044
00045 void
00046 Buyer::checkout()
00047 {
00048     _cart->checkout();
00049 }
00050
00051 set<int>
00052 Buyer::showCart()
00053 {
00054     try {
00055         return _cart->showCart();
00056     } catch (const EShopError& e) {
00057         throw e;
00058     }
00059 }
00060
00061 pair<Item*, int>
00062 Buyer::getItemOrder(Item* item)
00063 {
00064     try {
00065         return _cart->getItemOrder(item);
00066     } catch (const EShopError& e) {
00067         throw e;
00068     }
00069 }
00070
00071 void
00072 Buyer::clearCart()
00073 {
00074     _cart->clearCart();
00075 }
00076
00077
00078 bool
00079 Buyer::operator==(Buyer& other)
00080 {
00081     return (this->getName() == other.getName()) && (this->getEmail() == other.getEmail());
00082 }
00083
00084 void
00085 Buyer::setBonus(double value) {

```

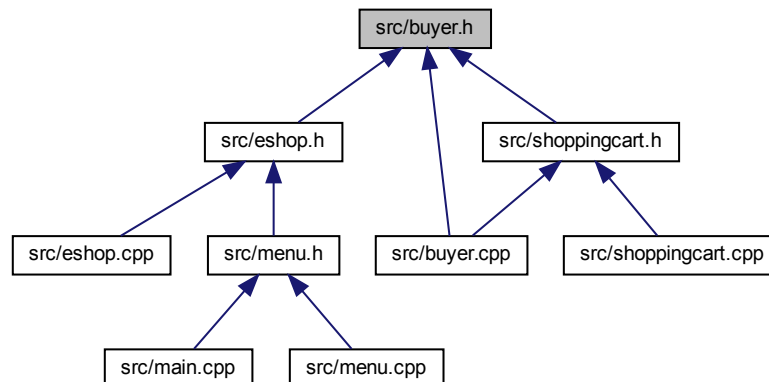
```
00086     _bonus = static_cast<int>(value*0.1);
00087 }
00088
00089 int
00090 Buyer::getBonus() { return _bonus; }
00091
00092 void
00093 Buyer::setCategory() {
00094     for(auto c: _categories)
00095         if (_bonus > _categoryScore[c]) _category = c;
00096 }
00097
00098 Buyer::Category
00099 Buyer::getCategory() { return _category; }
00100
00101 string
00102 Buyer::getCategoryName() { return _categoryString[_category]; }
00103
00104 bool
00105 Buyer::isAdmin() { return false; }
```

6.3 src/buyer.h File Reference

```
#include <set>
#include <map>
#include "item.h"
#include "user.h"
Include dependency graph for buyer.h:
```



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



Classes

- class [Buyer](#)

Specialization of [User](#). Describes a [Buyer](#).

6.4 buyer.h

```

00001 #ifndef BUYER_H
00002 #define BUYER_H
00003
00004 #include <set>
00005 #include <map>
00006 #include "item.h"
00007 #include "user.h"
00008
00009 // Forward declaration of ShoppingCart
00010 class ShoppingCart;
00011
00012 using namespace std;
00013
00021 class Buyer final : public User
00022 {
00023 public:
00030     Buyer(string name, string email);
00031
00035     Buyer(const Buyer&) = delete;
00036
00044     enum Category {
00045         Bronze,
00046         Silver,
00047         Gold,
00048     };
00049
00056     bool isAdmin();
00062     void awardBonus(double orderValue);
00073     void placeOrder(Item* item, int quantity);
00079     void removeFromOrder(Item* item);
00083     void checkout();
00089     set<int> showCart();
00096     pair<Item*, int> getItemOrder(Item*);
00100     void clearCart();
00101
00108     bool operator==(Buyer&);
00112     Buyer& operator=(const Buyer&) = delete;
00113
00117     int getBonus();
00121     Category getCategory();
00125     string getCategoryName();
  
```



```

00126
00127 private:
00131     void setBonus(double);
00135     void setCategory();
00136
00137     int _bonus;
00138     Category _category;
00140     map<Category, int> _categoryScore { {Bronze, 0}, {Silver, 100}, {Gold, 200} };
00142     map<Category, string> _categoryString { {Bronze, "Bronze"}, {Silver, "Silver"}, {Gold, "Gold"} };
00144     static constexpr initializer_list<Category> _categories = {Bronze, Silver, Gold};
00146     ShoppingCart* _cart;
00147 };
00148
00149 #endif // BUYER_H

```

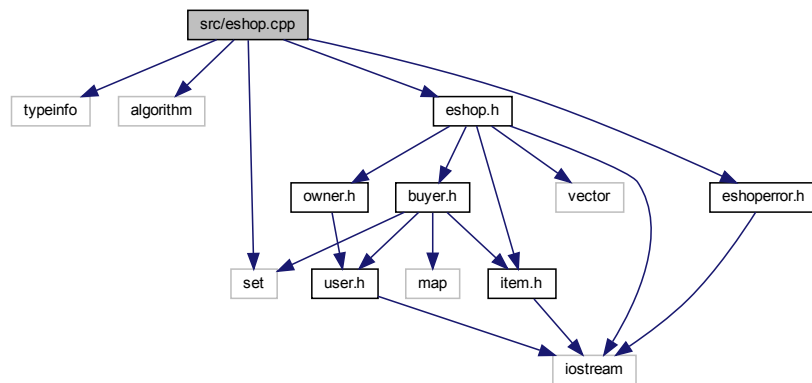
6.5 src/eshop.cpp File Reference

```

#include <typeinfo>
#include <algorithm>
#include <set>
#include "eshop.h"
#include "eshoperror.h"

```

Include dependency graph for eshop.cpp:



6.6 eshop.cpp

```

00001 #include <typeinfo>
00002 #include <algorithm>
00003 #include <set>
00004 #include "eshop.h"
00005 #include "eshoperror.h"
00006
00007 EShop::EShop(string name)
00008 {
00009     _name = name;
00010     _owner = nullptr;
00011 }
00012
00013 EShop::EShop(string name, string owner, string email) : EShop(name)
00014 {
00015     setOwner(owner, email);
00016 }
00017
00018 EShop::~EShop()
00019 {
00020     for(auto i: _items) delete i;
00021     for(auto b: _buyers) delete b;
00022     delete _owner;

```

```

00023 }
00024
00025 Owner*
00026 EShop::getOwner()
00027 {
00028     return _owner;
00029 }
00030
00031
00032 void
00033 EShop::setOwner(string name, string email)
00034 {
00035     if (_owner != nullptr)
00036         throw EShopError("Shop already has an owner");
00037     _owner = new Owner(name, email);
00038 }
00039
00040 void
00041 EShop::addItem(Item* item)
00042 {
00043     //TODO: exception
00044     if (!_items.empty()) {
00045         for(auto i: _items) {
00046             if (*i == *item) {
00047                 throw EShopError("Item " + item->getName() + " already exists.");
00048             }
00049         }
00050     }
00051     _items.push_back(item);
00052 }
00053
00054 void
00055 EShop::removeItem(Item* item)
00056 {
00057     for(auto i = _items.begin(); i <= _items.end(); ++i) {
00058         if (*i == *item) {
00059             delete *i;
00060             _items.erase(i);
00061             break;
00062         }
00063     }
00064 }
00065
00066 Item*
00067 EShop::getItemById(size_t id)
00068 {
00069     for(auto i: _items) if (i->getId() == id) return i;
00070     throw EShopError("Item with ID " + to_string(id) + " does not exist.");
00071     return nullptr;
00072 }
00073
00074 void
00075 EShop::addBuyer(Buyer* buyer)
00076 {
00077     //TODO: exception
00078     for(auto b: _buyers) {
00079         if (*b == *buyer) {
00080             throw EShopError("Buyer " + buyer->getName() + " already exists.");
00081         }
00082     }
00083     _buyers.push_back(buyer);
00084 }
00085
00086 void
00087 EShop::removeBuyer(Buyer* buyer)
00088 {
00089     for(auto b = _buyers.begin(); b <= _buyers.end(); ++b) {
00090         if (*b == *buyer) {
00091             (*b)->clearCart();
00092             delete *b;
00093             _buyers.erase(b);
00094             break;
00095         }
00096     }
00097 }
00098
00099 Buyer*
00100 EShop::getBuyerByEmail(string email)
00101 {
00102     for(auto b: _buyers) if (b->getEmail() == email) return b;
00103     throw EShopError("Buyer with email \' + email + \' does not exist.");
00104     return nullptr;
00105 }
00106
00107 void
00108 EShop::updateItemStock(Item* item, int delta)
00109 {

```

```

00110     item->setStock(item->getStock() + delta);
00111 }
00112
00113 void
00114 EShop::showProduct(Item* item)
00115 {
00116     cout << *item;
00117 }
00118
00119 vector<pair<string, int>>
00120 EShop::getCategories()
00121 {
00122     map<string, int> categories;
00123     for (auto i: _items) {
00124         string category = i->getCategory();
00125         if (categories.find(category) != categories.end())
00126             categories[category]++;
00127         else categories.emplace(category, 1);
00128     }
00129     vector<pair<string, int>> vectored;
00130     for (auto c: categories)
00131         vectored.push_back(make_pair(c.first, c.second));
00132     return vectored;
00133 }
00134
00135 vector<pair<int, string>>
00136 EShop::getProductsInCategory(string category)
00137 {
00138     vector<pair<int, string>> products;
00139     for(auto i: _items)
00140         if (i->getCategory() == category)
00141             products.push_back(make_pair(i->getId(), i->getName()));
00142     return products;
00143 }
00144
00145 vector<vector<string>>
00146 EShop::checkStatus()
00147 {
00148     vector<vector<string>> buyers;
00149     for (auto b: _buyers) {
00150         vector<string> info;
00151         info.push_back(b->getEmail());
00152         info.push_back(b->getName());
00153         info.push_back(b->getCategoryName());
00154         info.push_back(to_string(b->getBonus()));
00155         buyers.push_back(info);
00156     }
00157     return buyers;
00158 }
00159
00160 string
00161 EShop::getName()
00162 {
00163     return _name;
00164 }

```

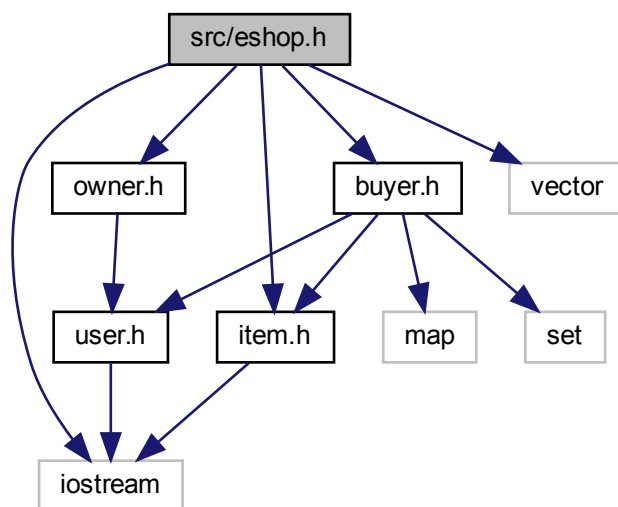
6.7 src/eshop.h File Reference

```

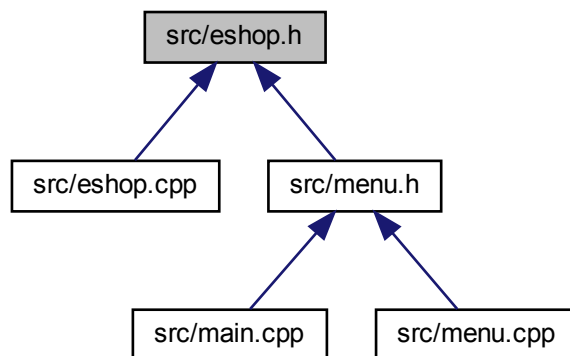
#include <iostream>
#include <vector>
#include "item.h"
#include "owner.h"
#include "buyer.h"

```

Include dependency graph for eshop.h:



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



Classes

- class [EShop](#)

Class implementing the e-shop.

6.8 eshop.h

```

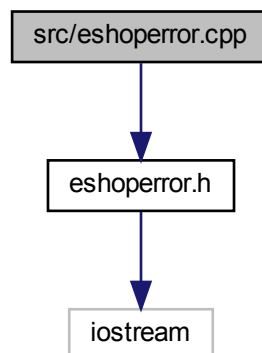
00001 #ifndef ESHOP_H
00002 #define ESHOP_H
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include "item.h"
00007 #include "owner.h"
00008 #include "buyer.h"
00009
00010 using namespace std;
00011
00022 class EShop
00023 {
00024 public:
00030     explicit EShop(string name);
00038     EShop(string name, string owner, string email);
00045     ~EShop();
00046
00050     string getName();
00054     Owner* getOwner();
00055
00061     void addItem(Item*);
00065     void removeItem(Item*);
00069     Item* getItemById(size_t id);
00070
00076     void addBuyer(Buyer*);
00080     void removeBuyer(Buyer*);
00084     Buyer* getBuyerByEmail(string email);
00085
00089     void updateItemStock(Item*, int);
00098     vector<pair<string, int>> getCategories();
00107     vector<pair<int, string>> getProductsInCategory(string);
00111     void showProduct(Item*);
00120     vector<vector<string>> checkStatus();
00121
00122 private:
00124     void setOwner(string name, string email);
00125
00126     string _name;
00127     Owner* _owner = nullptr;
00128     vector<Buyer*> _buyers;
00129     vector<Item*> _items;
00130 };
00131
00132 #endif // ESHOP_H

```

6.9 src/eshoperror.cpp File Reference

#include "eshoperror.h"

Include dependency graph for eshoperror.cpp:



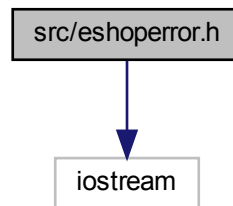
6.10 eshoperror.cpp

```
00001 #include "eshoperror.h"
```

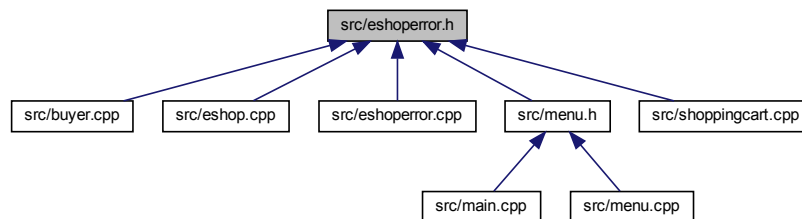
6.11 src/eshoperror.h File Reference

```
#include <iostream>
```

Include dependency graph for eshoperror.h:



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



Classes

- class [EShopError](#)

Exception class for passing error messages on failures.

6.12 eshoperror.h

```
00001 #ifndef ESHOPERROR_H
00002 #define ESHOPERROR_H
00003
00004 #include <iostream>
00005
00006 using namespace std;
00007
00012 class EShopError
00013 {
```

```

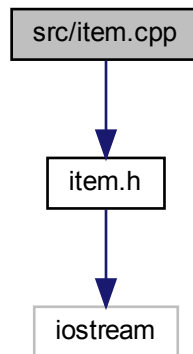
00014 public:
00015
00019     EShopError() { }
00025     EShopError(string&& error) : _error(move(error)) { }
00026
00030     const string& error() const {
00031         return _error;
00032     }
00033
00034 private:
00035     string _error;
00036 };
00037
00038 #endif // ESHOPERROR_H
00039

```

6.13 src/item.cpp File Reference

```
#include "item.h"
```

Include dependency graph for item.cpp:



Functions

- ostream & [operator<<](#) (ostream &os, [Item](#) &item)

6.13.1 Function Documentation

6.13.1.1 operator<<()

```

ostream& operator<< (
    ostream & os,
    Item & item )

```

Used to throw information to the cout garbage can

Definition at line 36 of file [item.cpp](#).

6.14 item.cpp

```

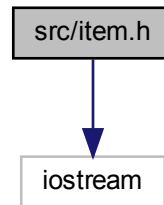
00001 #include "item.h"
00002
00003 Item::Item(int stock, double price, string name, string desc)
00004 {
00005     setStock(stock);
00006     setPrice(price);
00007     setName(name);
00008     setDescription(desc);
00009 }
00010
00011 Item::~Item()
00012 {
00013 }
00014
00015 string Item::getBasicInfo()
00016 {
00017     string ret;
00018     ret += to_string(_id);
00019     ret += ", ";
00020     ret += _name;
00021     ret += ", ";
00022     ret += to_string(_price);
00023     ret += ", ";
00024     ret += to_string(_stock);
00025     ret += ", ";
00026     ret += _desc;
00027     ret += ", ";
00028     return ret;
00029 }
00030
00031 Item::operator std::string ()
00032 {
00033     return getBasicInfo() + getDetails();
00034 }
00035
00036 ostream&
00037 operator<<(ostream& os, Item& item)
00038 {
00039     os << static_cast<std::string>(item);
00040     return os;
00041 }
00042
00043 bool
00044 Item::operator==(const Item& other)
00045 {
00046     return this->_id == other._id;
00047 }
00048
00049 void
00050 Item::setCategory(string category) { _category = category.substr(1); }
00051 string
00052 Item::getCategory() { return _category; }
00053
00054 void
00055 Item::setId(size_t id) { _id = id % 10000; }
00056 size_t
00057 Item::getId() { return _id; }
00058
00059 void
00060 Item::setStock(int stock) { _stock = stock; }
00061 int
00062 Item::getStock() { return _stock; }
00063
00064 void
00065 Item::setPrice(double price) { _price = price; }
00066 double
00067 Item::getPrice() { return _price; }
00068
00069 void
00070 Item::setName(string name) { _name = name; }
00071 string
00072 Item::getName() { return _name; }
00073
00074 void
00075 Item::setDescription(string desc) { _desc = desc; }
00076 string
00077 Item::getDescription() { return _desc; }
00078

```

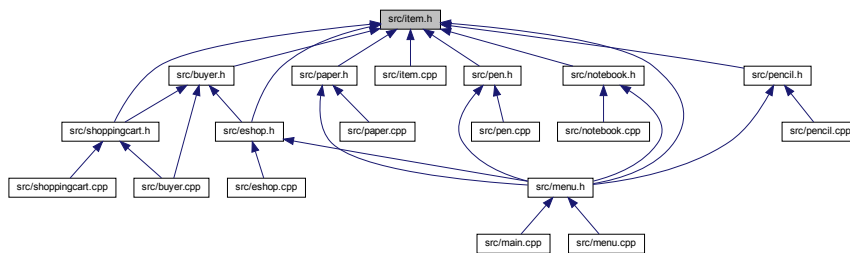

6.15 src/item.h File Reference

```
#include <iostream>
```

Include dependency graph for item.h:



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



Classes

- class [Item](#)

Base class for all other items.

6.16 item.h

```

00001 #ifndef ITEM_H
00002 #define ITEM_H
00003
00004 #include <iostream>
00005
00006 using namespace std;
00007
00018 class Item
00019 {
00020 public:
00029     Item(int stock, double price, string name, string desc);
00033     virtual ~Item();
00034
00043     size_t getId();
00044
00050     void setStock(int stock);
00056     int  getStock();
00057
  
```

```

00063     void    setPrice(double price);
00069     double  getPrice();
00070
00076     void    setName(string name);
00082     string  getName();
00083
00089     void    setDescription(string desc);
00095     string  getDescription();
00096
00102     string  getCategory();
00112     string  getBasicInfo();
00113
00121     void    setId(size_t);
00130     void    setCategory(string);
00131
00142     virtual void setId() = 0;
00149     virtual string getDetails() = 0;
00150
00157     operator std::string ();
00163     friend ostream& operator<<(ostream&, Item&);
00164
00170     bool operator==(const Item&);
00171
00172 private:
00173     size_t _id = 0;
00174     int     _stock;
00175     double  _price;
00176     string  _name;
00177     string  _desc;
00178     string  _category;
00179 };
00180
00181 #endif // ITEM_H

```

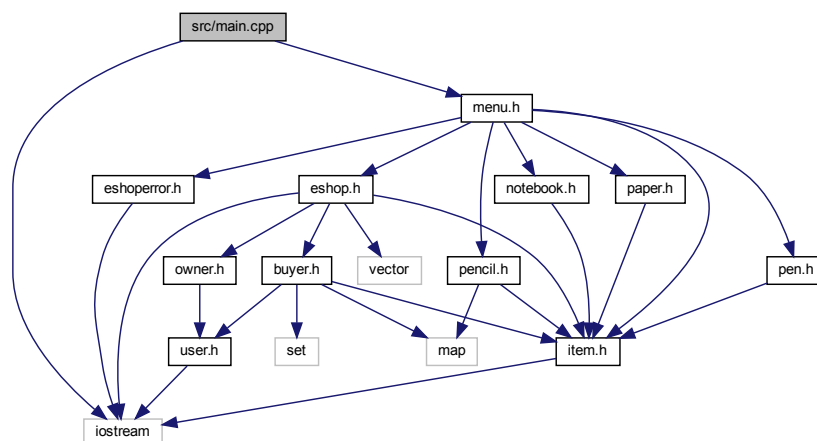
6.17 src/main.cpp File Reference

```

#include <iostream>
#include "menu.h"

```

Include dependency graph for main.cpp:



Functions

- int `main` (int argc, char **argv)

6.17.1 Function Documentation

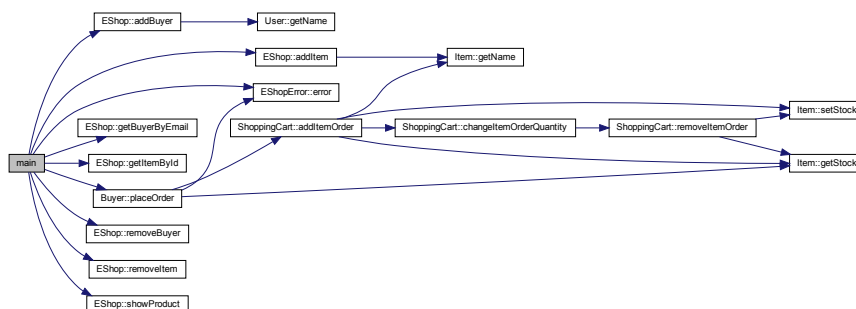
6.17.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

Definition at line 27 of file [main.cpp](#).

References [EShop::addBuyer\(\)](#), [EShop::addItem\(\)](#), [Pencil::B](#), [EShopError::error\(\)](#), [EShop::getBuyerByEmail\(\)](#), [EShop::getItemById\(\)](#), [Pencil::H](#), [Pencil::HB](#), [Buyer::placeOrder\(\)](#), [EShop::removeBuyer\(\)](#), [EShop::removeItem\(\)](#), and [EShop::showProduct\(\)](#).

Here is the call graph for this function:



6.18 main.cpp

```
00001
00024 #include <iostream>
00025 #include "menu.h"
00026
00027 int main(int argc, char **argv) {
00028     EShop eshop = EShop("EShop", "Owner", "owner@eshop.com");
00029
00030     try {
00031         eshop.addItem(new Pencil(0.2, Pencil::B, 10, 3.1, "Yellow Pencil Name", "Yellow Pencil
Desc"));
00032         eshop.addItem(new Pencil(0.3, Pencil::H, 15, 2.1, "Orange Pencil Name", "Orange Pencil
Desc"));
00033         eshop.addItem(new Pencil(0.4, Pencil::HB, 20, 2.6, "Purple Pencil Name", "Purple Pencil
Desc"));
00034         eshop.addItem(new Pencil(0.4, Pencil::HB, 10, 3.1, "Purple Pencil Name", "Purple Pencil
Desc"));
00035     } catch(const EShopError& e) {
00036         cout << e.error() << endl;
00037     }
00038
00039     try {
00040         eshop.addItem(new Pen(0.2, "yellow", 10, 3.1, "Yellow Pen Name", "Yellow Pen Desc"));
00041         eshop.addItem(new Pen(0.3, "orange", 15, 2.1, "Orange Pen Name", "Orange Pen Desc"));
00042         eshop.addItem(new Pen(0.4, "purple", 20, 2.6, "Purple Pen Name", "Purple Pen Desc"));
00043         eshop.addItem(new Pen(0.4, "purple", 10, 3.1, "Purple Pen Name", "Purple Pen Desc"));
00044     } catch(const EShopError& e) {
00045         cout << e.error() << endl;
00046     }
00047
00048     try {
00049         eshop.addItem(new Notebook(2, 15, 10.5, "2 Subject Notebook", "Fancy 2 Subject Notebook"));
00049     }
```

```

00050         eshop.addItem(new Notebook(3, 15, 11.5, "3 Subject Notebook", "Fancy 3 Subject Notebook"));
00051         eshop.addItem(new Notebook(4, 15, 12.5, "4 Subject Notebook", "Fancy 4 Subject Notebook"));
00052         eshop.addItem(new Notebook(4, 15, 12.5, "4 Subject Notebook", "Fancy 4 Subject Notebook"));
00053     } catch(const EShopError& e) {
00054         cout << e.error() << endl;
00055     }
00056
00057     try {
00058         eshop.addItem(new Paper(100, 2, 15, 10.5, "100 Pages", "Fancy 100"));
00059         eshop.addItem(new Paper(200, 3, 15, 11.5, "200 Pages", "Fancy 200"));
00060         eshop.addItem(new Paper(300, 4, 15, 12.5, "300 Pages", "Fancy 300"));
00061         eshop.addItem(new Paper(300, 4, 15, 12.5, "300 Pages", "Fancy 300"));
00062     } catch(const EShopError& e) {
00063         cout << e.error() << endl;
00064     }
00065
00066     try {
00067         eshop.showProduct(eshop.getItemById(6091));
00068         eshop.showProduct(eshop.getItemById(6092));
00069         eshop.removeItem(eshop.getItemById(6091));
00070         eshop.removeItem(eshop.getItemById(6092));
00071     } catch (const EShopError& e) {
00072         cout << e.error() << endl;
00073     }
00074
00075     try {
00076         eshop.removeItem(eshop.getItemById(6091));
00077         eshop.removeItem(eshop.getItemById(6092));
00078     } catch (const EShopError& e) {
00079         cout << e.error() << endl;
00080     }
00081
00082     try {
00083         eshop.addBuyer(new Buyer("buyer_a", "buyer_a@isp.org"));
00084         eshop.addBuyer(new Buyer("buyer_b", "buyer_b@isp.org"));
00085         eshop.addBuyer(new Buyer("buyer_c", "buyer_c@isp.org"));
00086         eshop.addBuyer(new Buyer("buyer_b", "buyer_b@isp.org"));
00087     } catch(const EShopError& e) {
00088         cout << e.error() << endl;
00089     }
00090
00091     try {
00092         eshop.removeBuyer(eshop.getBuyerByEmail("buyer_c@isp.org"));
00093     } catch(const EShopError& e) {
00094         cout << e.error() << endl;
00095     }
00096
00097     Buyer* buyer = eshop.getBuyerByEmail("buyer_a@isp.org");
00098     try {
00099         buyer->placeOrder(eshop.getItemById(7093), 3);
00100         buyer->placeOrder(eshop.getItemById(7093), 3);
00101         buyer->placeOrder(eshop.getItemById(7093), 3);
00102         buyer->placeOrder(eshop.getItemById(7093), 13);
00103     } catch(const EShopError& e) {
00104         cout << e.error() << endl;
00105     }
00106
00107     Menu menu = Menu(&eshop);
00108     return 0;
00109 }

```

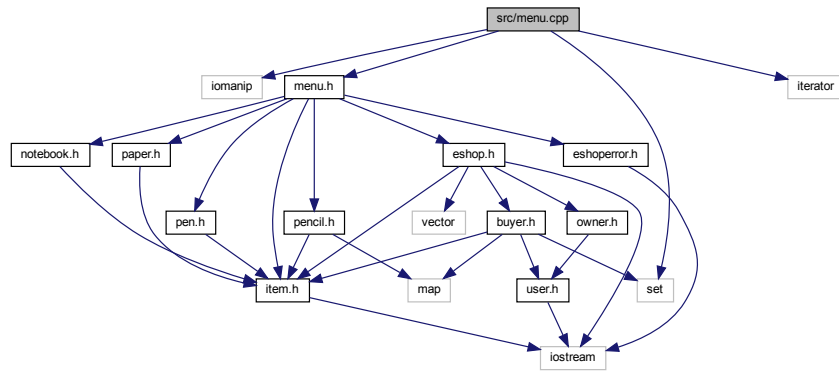
6.19 src/menu.cpp File Reference

```

#include <iomanip>
#include <set>
#include <iterator>
#include "menu.h"

```

Include dependency graph for menu.cpp:



6.20 menu.cpp

```

00001 #include <iomanip>
00002 #include <set>
00003 #include <iterator>
00004 #include "menu.h"
00005
00006 Menu::Menu(EShop* eshop)
00007 {
00008     _eshop = eshop;
00009     showWelcome();
00010 }
00011
00012 void
00013 Menu::showWelcome()
00014 {
00015     cout << "Welcome to " << quoted(_eshop->getName()) << endl;
00016     showLoginMenu();
00017     if(_user->isAdmin()) {
00018         showOwnerMenu();
00019     } else {
00020         showBuyerMenu();
00021     }
00022 }
00023
00024 void
00025 Menu::showLoginMenu()
00026 {
00027     string email;
00028     cout << "Enter the email address of a user to log in: ";
00029     cin >> email;
00030     try {
00031         login(email);
00032     } catch (const EShopError& e) {
00033         cout << e.error() << endl;
00034         exit(-1);
00035     }
00036 }
00037
00038 void
00039 Menu::showOwnerMenu()
00040 {
00041     cout << "Hello " << quoted(_owner->getName()) << " " << quoted(_owner->getEmail()) << endl;
00042     cout << "You are the owner of " << quoted(_eshop->getName()) << endl;
00043     cout << endl;
00044     cout << "Please choose an action." << endl;
00045     cout << "1. Browse Store" << endl;
00046     cout << "2. Check Status" << endl;
00047     cout << "3. Back" << endl;
00048     cout << "4. Logout" << endl;
00049     cout << "5. Exit" << endl;
00050
00051     string ans; int chc;
00052     do {
00053         cout << "Action: ";
00054         cin >> ans;
00055         chc = stoi(ans);

```

```

00056     } while( !cin.fail() && chc < 1 && chc > 5);
00057
00058     switch (stoi(ans)) {
00059         case 1:
00060             showBrowseMenu();
00061             break;
00062         case 2:
00063             showStatusMenu();
00064             break;
00065         case 3:
00066             showBuyerMenu();
00067             break;
00068         case 4:
00069             showLoginMenu();
00070             break;
00071         default:
00072             exit(0);
00073     }
00074 }
00075
00076 void
00077 Menu::showStatusMenu()
00078 {
00079     cout << "Please choose a buyer." << endl;
00080     vector<vector<string>> buyers = _eshop->checkStatus();
00081     int idx = 0;
00082     for(auto b: buyers) {
00083         cout << to_string(++idx) << ". " << b[0] << " " << b[1] << " " << b[2] << " " << b[3] << " " << endl;
00084     }
00085     cout << to_string(idx+1) << ". Back" << endl;
00086
00087     string ans; int chc;
00088     do {
00089         cout << "Buyer: ";
00090         cin >> ans;
00091         chc = stoi(ans);
00092     } while( !cin.fail() && chc < 1 && chc > idx+1);
00093
00094     if (chc <= idx) {
00095         Buyer* buyer = _eshop->getBuyerByEmail (buyers[chc][0]);
00096         buyer->showCart();
00097
00098         if (askYesNo("Do you want to delete this buyer?")) {
00099             _eshop->removeBuyer(buyer);
00100         }
00101         showStatusMenu();
00102     } else {
00103         showOwnerMenu();
00104     }
00105 }
00106
00107 void
00108 Menu::showBuyerMenu()
00109 {
00110     cout << "Hello " << quoted(_buyer->getName()) << " " << quoted(_buyer->getEmail()) << endl;
00111     cout << "You are a " << quoted(_buyer->getCategoryName()) << " customer with " << _buyer->getBonus() <<
00112     " points" << endl;
00113     cout << endl;
00114     cout << "Please choose an action." << endl;
00115     cout << "1. Browse Store" << endl;
00116     cout << "2. View Cart" << endl;
00117     cout << "3. Checkout" << endl;
00118     cout << "4. Back" << endl;
00119     cout << "5. Logout" << endl;
00120     cout << "6. Exit" << endl;
00121
00122     string ans;
00123     do {
00124         cout << "Action: ";
00125         cin >> ans;
00126     } while( !cin.fail() && stoi(ans) < 1 && stoi(ans) > 6);
00127
00128     switch (stoi(ans)) {
00129         case 1:
00130             showBrowseMenu();
00131             break;
00132         case 2:
00133             showCartMenu();
00134             break;
00135         case 3:
00136             showCheckoutMenu();
00137             break;
00138         case 4:
00139             showBuyerMenu();
00140             break;
00141         case 5:

```

```

00142         showLoginMenu();
00143         break;
00144     default:
00145         exit(0);
00146     }
00147 }
00148
00149 void
00150 Menu::showBrowseMenu()
00151 {
00152     cout << "Please choose a category." << endl;
00153     vector<pair<string, int>> categories = _eshop->getCategories();
00154     int idx = 0;
00155     for(auto c: categories) {
00156         cout << to_string(++idx) << ". " << c.first << " " << "(" << c.second << ")" << endl;
00157     }
00158     cout << to_string(idx+1) << ". Back" << endl;
00159
00160     string ans; int chc;
00161     do {
00162         cout << "Category: ";
00163         cin >> ans;
00164         chc = stoi(ans);
00165     } while( !cin.fail() && chc < 1 && chc > idx+1);
00166
00167     if (chc <= idx) {
00168         showCategoryMenu(categories[chc-1].first);
00169     } else {
00170         if (_user->isAdmin())
00171             showOwnerMenu();
00172         else
00173             showBuyerMenu();
00174     }
00175 }
00176
00177 void
00178 Menu::showCategoryMenu(string category)
00179 {
00180     cout << "Please select a product." << endl;
00181     vector<pair<int, string>> products = _eshop->getProductsInCategory(category);
00182     int idx = 0;
00183     for(auto p: products) {
00184         cout << to_string(++idx) << ". " << p.first << " " << "(" << p.second << ")" << endl;
00185     }
00186     cout << to_string(idx+1) << ". Back" << endl;
00187
00188     string ans; int chc;
00189     do {
00190         cout << "Product: ";
00191         cin >> ans;
00192         chc = stoi(ans);
00193     } while( !cin.fail() && chc < 1 && chc > idx+1);
00194
00195     if (chc <= idx) {
00196         showProductMenu(category, products[chc-1].first);
00197     } else {
00198         showBrowseMenu();
00199     }
00200 }
00201
00202 void
00203 Menu::showProductMenu(string back, int id){
00204     Item* item = _eshop->getItemById(id);
00205     _eshop->showProduct(item);
00206     if (_user->isAdmin()) {
00207         if (askYesNo("Do you want update the stock of this product?")) {
00208             int quantity;
00209             cout << "Quantity: ";
00210             cin >> quantity;
00211             _eshop->updateItemStock(item, quantity);
00212         }
00213     } else {
00214         if (askYesNo("Do you want to buy this product?")) {
00215             int quantity;
00216             cout << "Quantity: ";
00217             cin >> quantity;
00218             _buyer->placeOrder(item, quantity);
00219         }
00220     }
00221     showCategoryMenu(back);
00222 }
00223
00224 void
00225 Menu::showCartMenu()
00226 {
00227     try {
00228         set<int> ids = _buyer->showCart();

```

```

00229     int chc;
00230     string ans;
00231     Item* item;
00232
00233     cout << "Actions" << endl;
00234     cout << "1. Clear Cart" << endl;
00235     cout << "2. Checkout" << endl;
00236     cout << "3. Back" << endl;
00237
00238     do {
00239         cout << "Please select an action or a product ID to edit your order: ";
00240         cin >> ans;
00241         chc = stoi(ans);
00242         if (ids.contains(chc)) {
00243             item = _eshop->getItemById(chc);
00244             break;
00245         }
00246     } while (!cin.fail() && chc < 1 && chc > 3);
00247
00248     switch (chc) {
00249         case 1:
00250             _buyer->clearCart();
00251             break;
00252         case 2:
00253             _buyer->checkout();
00254             break;
00255         case 3:
00256             break;
00257         default:
00258             pair<Item*, int> item_order;
00259             try {
00260                 item_order = _buyer->getItemOrder(item);
00261             } catch (const EShopError& e) {
00262                 cout << e.error() << endl;
00263                 showCartMenu();
00264             }
00265             cout << "Editing Item Order: ";
00266             cout << item_order.first->getName() << " (" << item_order.second << ")" << endl;
00267             cout << "1. Delete Item Order" << endl;
00268             cout << "2. Change Item Order" << endl;
00269
00270             do {
00271                 cout << "Action: ";
00272                 cin >> ans;
00273                 chc = stoi(ans);
00274             }
00275             while( (!cin.fail() && chc < 1 && chc > 2) );
00276
00277             switch (chc) {
00278                 case 1:
00279                     _buyer->removeFromOrder(item);
00280                     break;
00281                 case 2:
00282                     do {
00283                         cout << "Do you want to (1)add or (2)delete: ";
00284                         cin >> ans;
00285                         chc = stoi(ans);
00286                     } while( (!cin.fail() && chc < 1 && chc > 2) );
00287
00288                     int qnt;
00289                     do {
00290                         cout << "Please enter the quantity: ";
00291                         cin >> ans;
00292                         qnt = stoi(ans);
00293                     }
00294                     while( (!cin.fail() && qnt < 0 && qnt > item_order.second) );
00295
00296                     if (chc == 1)
00297                         _buyer->placeOrder(item, qnt);
00298                     else
00299                         _buyer->placeOrder(item, -qnt);
00300                     break;
00301             }
00302             showCartMenu();
00303         }
00304     } catch (const EShopError& e) {
00305         cout << e.error() << endl;
00306     }
00307     showBuyerMenu();
00308 }
00309
00310 void
00311 Menu::showCheckoutMenu()
00312 {
00313     _buyer->checkout();
00314     showBuyerMenu();
00315 }

```



```

00316
00317 Menu::~Menu()
00318 {
00319 }
00320
00321 void
00322 Menu::login(string email)
00323 {
00324     try {
00325         _buyer = nullptr;
00326         _owner = nullptr;
00327         if (email == _eshop->getOwner()->getEmail()) {
00328             _owner = _eshop->getOwner();
00329             _user = _owner;
00330         } else {
00331             _buyer = _eshop->getBuyerByEmail(email);
00332             _user = _buyer;
00333         }
00334     } catch (const EShopError& e) {
00335         throw e;
00336     }
00337     cout << "Authenticated as: " << _user->getName() << " " << _user->getEmail() << endl;
00338 }
00339
00340 bool
00341 Menu::askYesNo(string message) {
00342     string ans;
00343     do {
00344         cout << message << " [y/n]: ";
00345         cin >> ans;
00346     }
00347     while( !cin.fail() && ans!="y" && ans!="n" );
00348
00349     if (ans == "y") return true;
00350     else return false;
00351 }

```

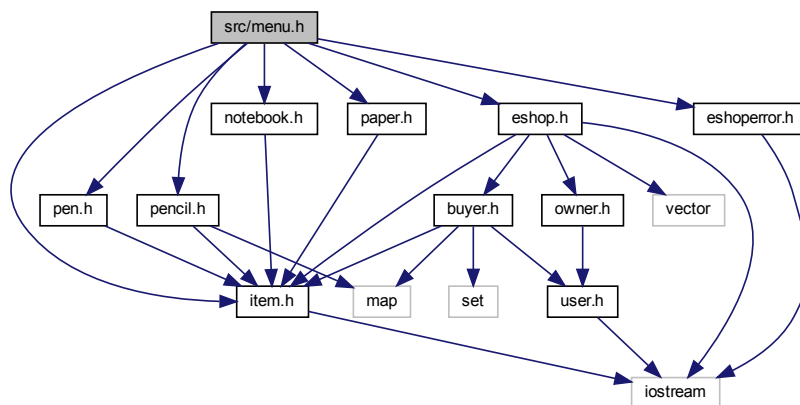
6.21 src/menu.h File Reference

```

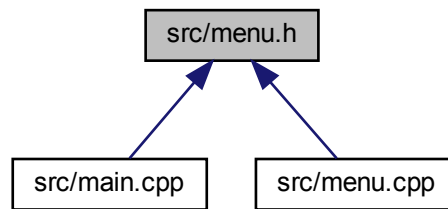
#include "eshop.h"
#include "item.h"
#include "pen.h"
#include "pencil.h"
#include "notebook.h"
#include "paper.h"
#include "eshoperror.h"

```

Include dependency graph for menu.h:



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



Classes

- class [Menu](#)

Creates a menu for the e-shop's interface.

6.22 menu.h

```

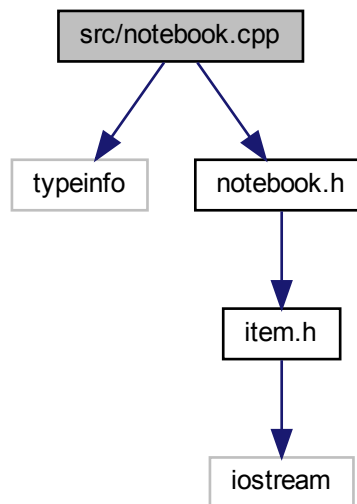
00001 #ifndef MENU_H
00002 #define MENU_H
00003
00004 #include "eshop.h"
00005 #include "item.h"
00006 #include "pen.h"
00007 #include "pencil.h"
00008 #include "notebook.h"
00009 #include "paper.h"
00010 #include "eshoperror.h"
00011
00012
00020 class Menu
00021 {
00022 public:
00023     Menu(EShop*);
00024     ~Menu();
00025
00026     void showWelcome();
00027     void showLoginMenu();
00028
00029     void showOwnerMenu();
00030     void showBuyerMenu();
00031
00032     void showBrowseMenu();
00033     void showCategoryMenu(string);
00034     void showProductMenu(string, int);
00035     void showCartMenu();
00036     void showStatusMenu();
00037     void showCheckoutMenu();
00038
00039 private:
00041     void login(string);
00043     bool askYesNo(string);
00044
00045     User* _user = nullptr;
00046     Owner* _owner = nullptr;
00047     Buyer* _buyer = nullptr;
00048     EShop* _eshop;
00049 };
00050
00051 #endif // MENU_H
  
```

6.23 src/notebook.cpp File Reference

```
#include <typeinfo>
```

```
#include "notebook.h"
```

Include dependency graph for notebook.cpp:



6.24 notebook.cpp

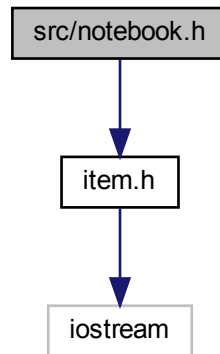
```

00001 #include <typeinfo>
00002 #include "notebook.h"
00003
00004 Notebook::Notebook(int subjects, int stock, double price, string name, string desc) : Item(stock,
00005 price, name, desc)
00006 {
00007     setSubjects(subjects);
00008     setCategory(typeid(*this).name());
00009     setId();
00010 }
00011 string
00012 Notebook::getDetails()
00013 {
00014     string ret;
00015     ret += to_string(_subjects);
00016     ret += "\n";
00017     return ret;
00018 }
00019
00020 void
00021 Notebook::setId()
00022 {
00023     size_t h_obj = hash<string>{}(typeid(*this).name());
00024     size_t h_sub = hash<int>{}(_subjects);
00025     Item::setId(h_obj ^ (h_sub << 1));
00026 }
00027
00028 void
00029 Notebook::setSubjects(int subjects) { _subjects = subjects; }
00030 int
00031 Notebook::getSubjects() { return _subjects; }
  
```

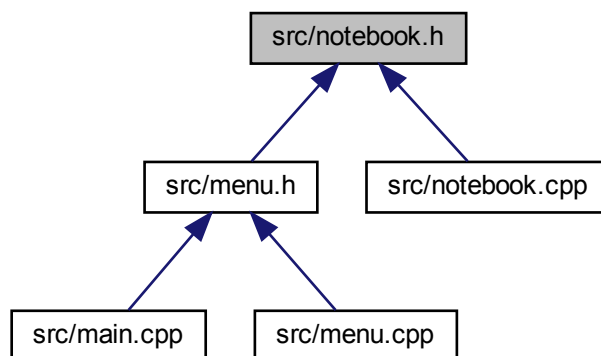
6.25 src/notebook.h File Reference

```
#include "item.h"
```

Include dependency graph for notebook.h:



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



Classes

- class [Notebook](#)

Class representing a [Notebook](#).

6.26 notebook.h

```

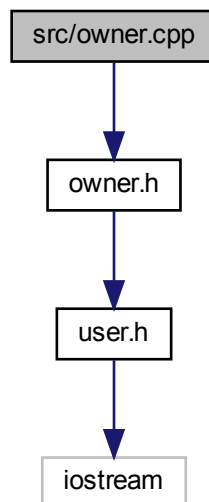
00001 #ifndef NOTEBOOK_H
00002 #define NOTEBOOK_H
00003
00004 #include "item.h"
00005
00006 using namespace std;
00007
00012 class Notebook final : public Item
00013 {
00014 public:
00024     Notebook(int subjects, int stock, double price, string name, string desc);
00025
00031     void setSubjects(int subjects);
00037     int  getSubjects();
00038
00045     void setId() override;
00051     string getDetails() override;
00052
00053 private:
00054     int _subjects;
00055 };
00056
00057 #endif // NOTEBOOK_H

```

6.27 src/owner.cpp File Reference

```
#include "owner.h"
```

Include dependency graph for owner.cpp:



6.28 owner.cpp

```

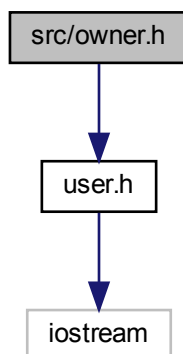
00001 #include "owner.h"
00002
00003 Owner::Owner(string name, string email) : User(name, email)
00004 {
00005     _isAdmin = true;
00006 }
00007
00008 bool
00009 Owner::isAdmin() { return _isAdmin; }

```

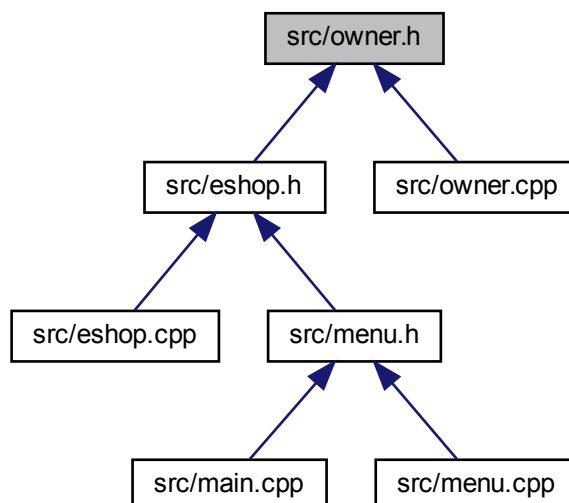
6.29 src/owner.h File Reference

```
#include "user.h"
```

Include dependency graph for owner.h:



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



Classes

- class [Owner](#)

Specialization of [User](#). Describes an [Owner](#).

6.30 owner.h

```

00001 #ifndef OWNER_H
00002 #define OWNER_H
00003
00004 #include "user.h"
00005
00006 using namespace std;
00007
00017 class Owner final: public User {
00018 public:
00025     Owner(string name, string email);
00026
00033     bool isAdmin();
00034
00035 private:
00037     bool _isAdmin;
00038 };
00039
00040 #endif // OWNER_H

```

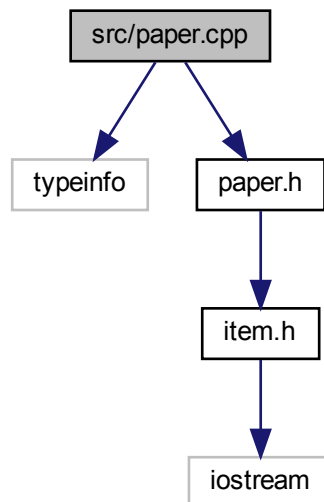
6.31 src/paper.cpp File Reference

```

#include <typeinfo>
#include "paper.h"

```

Include dependency graph for paper.cpp:



6.32 paper.cpp

```

00001 #include <typeinfo>
00002 #include "paper.h"
00003
00004
00005 Paper::Paper(int pages, int weight, int stock, double price, string name, string desc) : Item(stock,
    price, name, desc)
00006 {
00007     setPages(pages);
00008     setWeight(weight);
00009     setCategory(typeid(*this).name());

```

```

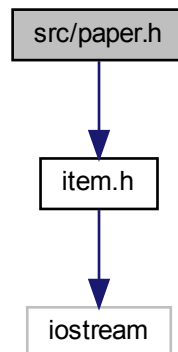
00010     setId();
00011 }
00012
00013 string Paper::getDetails()
00014 {
00015     string ret;
00016     ret += to_string(_pages);
00017     ret += ", ";
00018     ret += to_string(_weight);
00019     ret += "\n";
00020     return ret;
00021 }
00022
00023 void Paper::setId()
00024 {
00025     size_t h_obj = hash<string>{}(typeid(*this).name());
00026     size_t h_pgs = hash<int>{}(_pages);
00027     size_t h_wgt = hash<int>{}(_weight);
00028     Item::setId(h_obj ^ ((h_pgs ^ (h_wgt << 1)) << 1));
00029 }
00030
00031 void
00032 Paper::setPages(int pages) { _pages = pages; }
00033 int
00034 Paper::getPages() { return _pages; }
00035
00036 void
00037 Paper::setWeight(int weight) { _weight = weight; }
00038 int
00039 Paper::getWeight() { return _weight; }

```

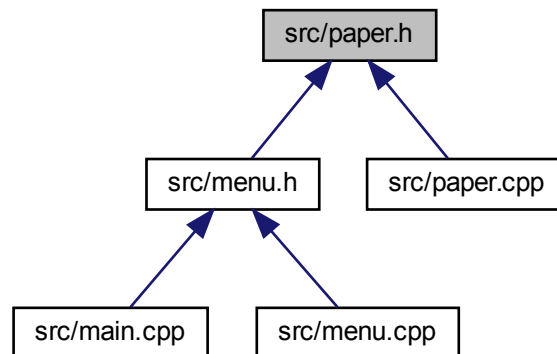
6.33 src/paper.h File Reference

```
#include "item.h"
```

Include dependency graph for paper.h:



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



Classes

- class [Paper](#)
Class representing a [Paper](#).

6.34 paper.h

```

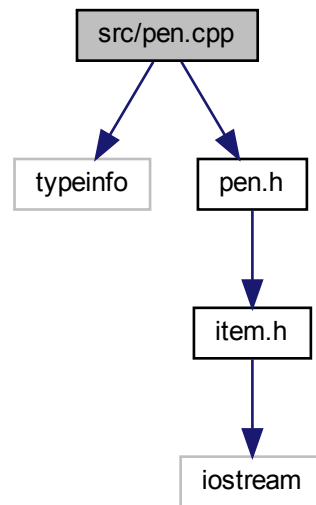
00001 #ifndef PAPER_H
00002 #define PAPER_H
00003
00004 #include "item.h"
00005
00006 using namespace std;
00007
00012 class Paper final : public Item
00013 {
00014 public:
00025     Paper(int pages, int weight, int stock, double price, string name, string desc);
00026
00032     void setPages(int pages);
00038     int  getPages();
00039
00045     void setWeight(int);
00051     int  getWeight();
00052
00059     void  setId();
00066     string getDetails();
00067
00068 private:
00069     int  _pages;
00070     int  _weight;
00071 };
00072
00073 #endif // PAPER_H
  
```

6.35 src/pen.cpp File Reference

```

#include <typeinfo>
#include "pen.h"
  
```

Include dependency graph for pen.cpp:



6.36 pen.cpp

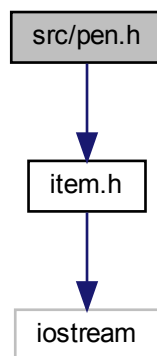
```

00001 #include <typeinfo>
00002 #include "pen.h"
00003
00004 Pen::Pen(double tipSize, string color, int stock, double price, string name, string desc) :
00005     Item(stock, price, name, desc)
00006 {
00007     setTipSize(tipSize);
00008     setColor(color);
00009     setCategory(typeid(*this).name());
00010     setId();
00011 }
00012 void
00013 Pen::setId()
00014 {
00015     size_t h_obj = hash<string>{}(typeid(*this).name());
00016     size_t h_clr = hash<string>{}(_color);
00017     size_t h_tip = hash<double>{}(_tipSize);
00018     Item::setId(h_obj ^ ((h_clr ^ (h_tip << 1)) << 1));
00019 }
00020
00021 string
00022 Pen::getDetails()
00023 {
00024     string ret;
00025     ret += to_string(_tipSize);
00026     ret += ", ";
00027     ret += _color;
00028     ret += "\n";
00029     return ret;
00030 }
00031
00032 void
00033 Pen::setTipSize(double size) { _tipSize = size; }
00034 double
00035 Pen::getTipSize() { return _tipSize; }
00036
00037 void
00038 Pen::setColor(string color) { _color = color; }
00039 string
00040 Pen::getColor() { return _color; }
  
```

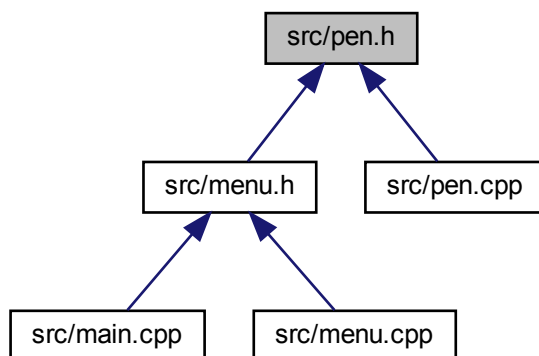
6.37 src/pen.h File Reference

```
#include "item.h"
```

Include dependency graph for pen.h:



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



Classes

- class [Pen](#)
Class representing a [Pen](#).

6.38 pen.h

```

00001 #ifndef PEN_H
00002 #define PEN_H
00003
00004 #include "item.h"
00005
00006 using namespace std;
00007
00012 class Pen final : public Item
00013 {
00014 public:
00025     Pen(double tipSize, string color, int stock, double price, string name, string desc);
00026
00032     void    setTipSize(double tipSize);
00038     double getTipSize();
00039
00045     void    setColor(string color);
00051     string getColor();
00052
00059     void    setId() override;
00066     string getDetails() override;
00067
00068 private:
00069     double _tipSize;
00070     string _color;
00071 };
00072
00073 #endif // PEN_H

```

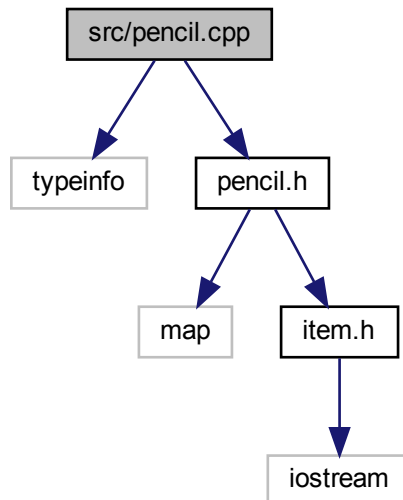
6.39 src/pencil.cpp File Reference

```

#include <typeinfo>
#include "pencil.h"

```

Include dependency graph for pencil.cpp:



6.40 pencil.cpp

```

00001 #include <typeinfo>

```

```

00002 #include "pencil.h"
00003
00004 Pencil::Pencil(double tipSize, Pencil::Type type, int stock, double price, string name, string desc) :
    Item(stock, price, name, desc)
00005 {
00006     setTipSize(tipSize);
00007     setType(type);
00008     setCategory(typeid(*this).name());
00009     setId();
00010 }
00011
00012 void
00013 Pencil::setId()
00014 {
00015     size_t h_obj = hash<string>{}(typeid(*this).name());
00016     size_t h_typ = hash<string>{}(_typeMap[_type]);
00017     size_t h_tip = hash<double>{}(_tipSize);
00018     Item::setId(h_obj ^ ((h_typ ^ (h_tip << 1)) << 1));
00019 }
00020
00021 string
00022 Pencil::getDetails()
00023 {
00024     string ret;
00025     ret += to_string(_tipSize);
00026     ret += ", ";
00027     ret += _typeMap[_type];
00028     ret += "\n";
00029     return ret;
00030 }
00031
00032 void
00033 Pencil::setTipSize(double size) { _tipSize = size; }
00034 double
00035 Pencil::getTipSize() { return _tipSize; }
00036
00037 void
00038 Pencil::setType(Type type) { _type = type; }
00039 Pencil::Type
00040 Pencil::getType() { return _type; }

```

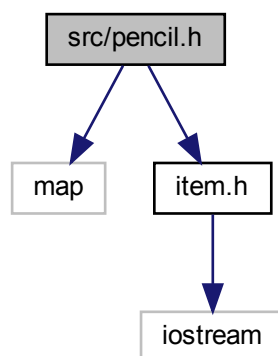
6.41 src/pencil.h File Reference

```

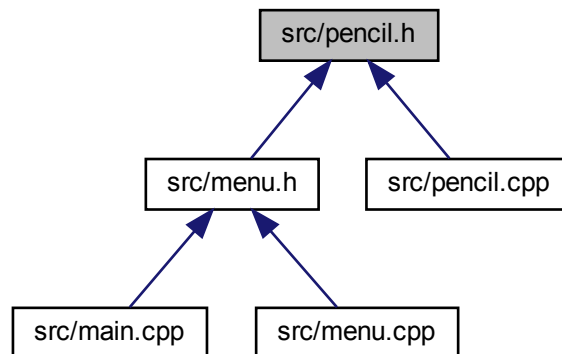
#include <map>
#include "item.h"

```

Include dependency graph for pencil.h:



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



Classes

- class [Pencil](#)

Class representing a [Pencil](#).

6.42 pencil.h

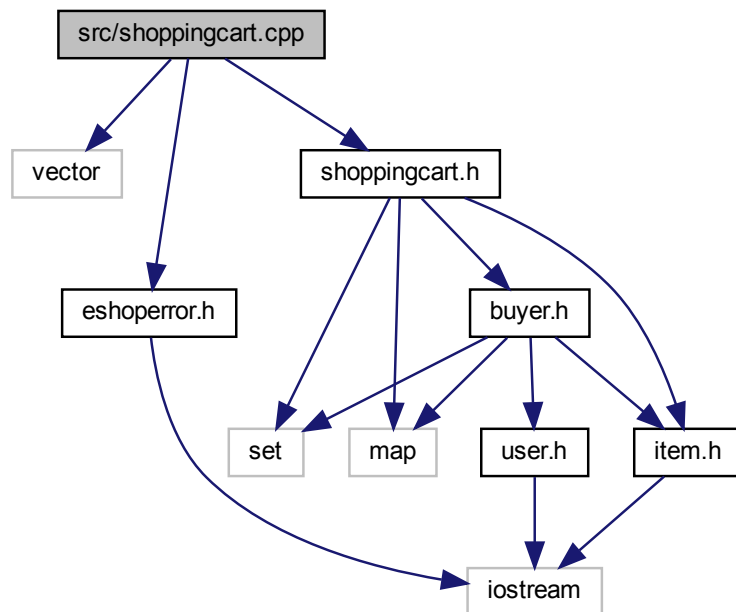
```

00001 #ifndef PENCIL_H
00002 #define PENCIL_H
00003
00004 #include <map>
00005 #include "item.h"
00006
00011 class Pencil final : public Item
00012 {
00013 public:
00014
00018     enum Type {
00019         H,
00020         B,
00021         HB,
00022     };
00023
00034     Pencil(double tipSize, Type type, int stock, double price, string name, string desc);
00035
00041     void setTipSize(double tipSize);
00047     double getTipSize();
00048
00054     void setType(Type type);
00060     Type getType();
00061
00068     void setId() override;
00075     string getDetails() override;
00076
00077 private:
00078     double _tipSize;
00079     Type _type;
00081     map<Type, string> _typeMap { {H, "H"}, { B, "B" }, {HB, "HB" } };
00082 };
00083
00084 #endif // PENCIL_H
  
```

6.43 src/shoppingcart.cpp File Reference

```
#include <vector>
#include "shoppingcart.h"
#include "eshoperror.h"
```

Include dependency graph for shoppingcart.cpp:



6.44 shoppingcart.cpp

```

00001 #include <vector>
00002 #include "shoppingcart.h"
00003 #include "eshoperror.h"
00004
00005 ShoppingCart::ShoppingCart(Buyer* buyer)
00006 {
00007     _buyer = buyer;
00008 }
00009
00010 void
00011 ShoppingCart::addItemOrder(Item* item, int quantity)
00012 {
00013     //TODO: exception
00014     if (item->getStock() >= quantity) {
00015         if (_order.find(item) != _order.end())
00016             changeItemOrderQuantity(item, quantity);
00017         else
00018             _order.emplace(item, quantity);
00019         item->setStock(item->getStock() - quantity);
00020     } else {
00021         throw EShopError("Item " + item->getName() + " does not have enough stock.");
00022     }
00023 }
00024
00025 void
00026 ShoppingCart::removeItemOrder(Item* item)
00027 {
00028     if (_order.find(item) != _order.end()) {
00029         item->setStock(item->getStock() + _order[item]);
00030     }
00031 }
```

```

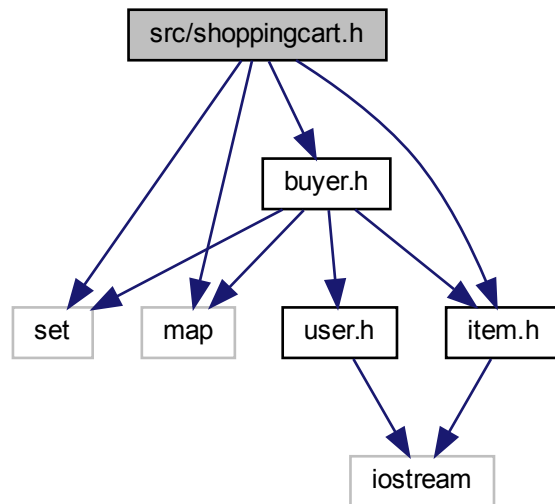
00030         _order.erase(item);
00031     }
00032 }
00033
00034 void
00035 ShoppingCart::changeItemOrderQuantity(Item* item, int quantity)
00036 {
00037     _order[item] += quantity;
00038     if (_order[item] <= 0) removeItemOrder(item);
00039 }
00040
00041 pair<Item*, int>
00042 ShoppingCart::getItemOrder(Item* item)
00043 {
00044     if (_order.find(item) != _order.end())
00045         return make_pair(item, _order[item]);
00046     else {
00047         throw EShopError("Item " + item->getName() + " not found in the cart");
00048         return make_pair(nullptr, 0);
00049     }
00050 }
00051
00052 double
00053 ShoppingCart::calculateNet()
00054 {
00055     double value = 0;
00056     for(auto o: _order) value += o.first->getPrice()*o.second;
00057     return value;
00058 }
00059
00060 double
00061 ShoppingCart::calculateCourier()
00062 {
00063     if (_buyer->getCategory() == Buyer::Gold) return 0;
00064     else {
00065         double value = calculateNet();
00066         if (_buyer->getCategory() == Buyer::Silver) return value*0.01;
00067         else return (value*0.02 >= 3 ? value*0.02 : 3);
00068     }
00069 }
00070
00071 set<int>
00072 ShoppingCart::showCart()
00073 {
00074     set<int> ids;
00075     if (_order.empty()) throw EShopError("Shopping cart is empty.");
00076     for (auto o: _order) {
00077         ids.insert(o.first->getId());
00078         cout << o.first->getId() << ", " << o.first->getName() << ", " << o.second << ", " <<
o.first->getPrice()*o.second << endl;
00079     }
00080     cout << "Order value: " << calculateNet() << endl;
00081     cout << "Courier value: " << calculateCourier() << endl;
00082     return ids;
00083 }
00084
00085 void
00086 ShoppingCart::clearCart()
00087 {
00088     for (auto o: _order)
00089         o.first->setStock(o.first->getStock() + o.second);
00090     _order.clear();
00091 }
00092
00093 void
00094 ShoppingCart::checkout()
00095 {
00096     showCart();
00097
00098     string ans;
00099     do {
00100         cout << "Do you want to continue the checkout? [y/n]: ";
00101         cin >> ans;
00102     }
00103     while( (!cin.fail() && ans!="y" && ans!="n") );
00104
00105     if (ans == "y") {
00106         _buyer->awardBonus(calculateNet());
00107         _order.clear();
00108     }
00109 }
00110

```

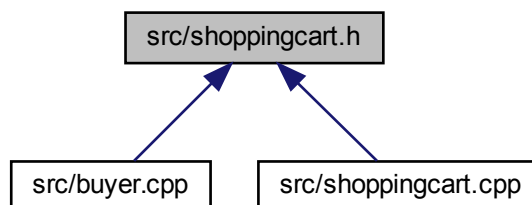

6.45 src/shoppingcart.h File Reference

```
#include <set>
#include <map>
#include "item.h"
#include "buyer.h"
```

Include dependency graph for shoppingcart.h:



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



Classes

- class [ShoppingCart](#)
Class implementing the shopping cart.

6.46 shoppingcart.h

```

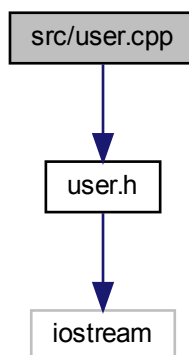
00001 #ifndef SHOPPINGCART_H
00002 #define SHOPPINGCART_H
00003
00004 #include <set>
00005 #include <map>
00006 #include "item.h"
00007 #include "buyer.h"
00008
00009 using namespace std;
00010
00022 class ShoppingCart
00023 {
00024 public:
00030     explicit ShoppingCart(Buyer* buyer);
00031
00043     void addItemOrder(Item* item, int quantity);
00051     void removeItemOrder(Item* item);
00061     void changeItemOrderQuantity(Item* item, int quantity);
00071     pair<Item*, int> getItemOrder(Item*);
00072
00083     set<int> showCart();
00089     void clearCart();
00090
00097     void checkout();
00098
00102     double calculateNet();
00108     double calculateCourier();
00109
00110 private:
00112     map<Item*, int> _order;
00114     Buyer* _buyer;
00115 };
00116
00117 #endif // SHOPPINGCART_H

```

6.47 src/user.cpp File Reference

```
#include "user.h"
```

Include dependency graph for user.cpp:



6.48 user.cpp

```

00001 #include "user.h"
00002
00003 User::User(string name, string email)

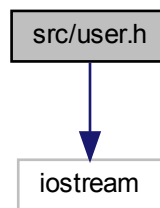
```

```
00004 {  
00005     setName(name);  
00006     setEmail(email);  
00007 }  
00008  
00009 void  
00010 User::setName(string name) { _name = name; }  
00011  
00012 string  
00013 User::getName() { return _name; }  
00014  
00015 void  
00016 User::setEmail(string email) { _email = email; }  
00017  
00018 string  
00019 User::getEmail() { return _email; }
```

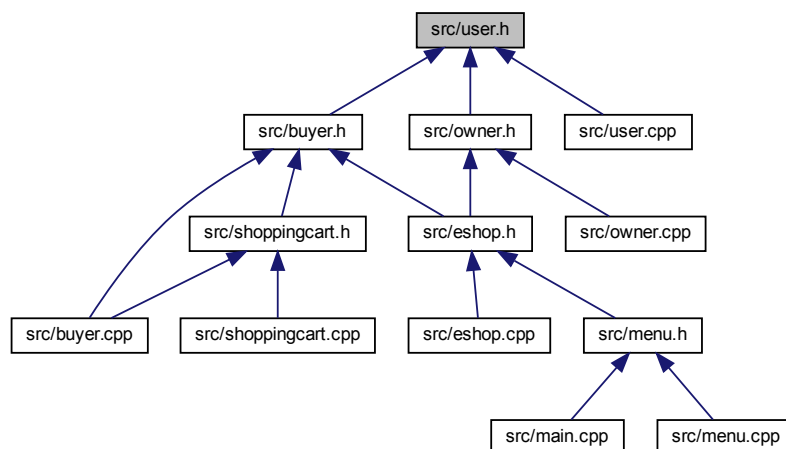
6.49 src/user.h File Reference

```
#include <iostream>
```

Include dependency graph for user.h:



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



Classes

- class [User](#)

Base class for all users.

6.50 user.h

```
00001 #ifndef USER_H
00002 #define USER_H
00003
00004 #include <iostream>
00005
00006 using namespace std;
00007
00015 class User {
00016 public:
00023     User(string name, string email);
00024
00032     string getName();
00040     string getEmail();
00041
00050     virtual bool isAdmin() = 0;
00051
00052 private:
00053     void setName(string name);
00054     void setEmail(string email);
00055
00056     string _name;
00057     string _email;
00058 };
00059
00060 #endif // USER_H
```

Index

- [_bonus](#)
 - [Buyer, 23](#)
 - [_buyer](#)
 - [Menu, 65](#)
 - [ShoppingCart, 107](#)
 - [_buyers](#)
 - [EShop, 37](#)
 - [_cart](#)
 - [Buyer, 23](#)
 - [_categories](#)
 - [Buyer, 23](#)
 - [_category](#)
 - [Buyer, 23](#)
 - [Item, 51](#)
 - [_categoryScore](#)
 - [Buyer, 23](#)
 - [_categoryString](#)
 - [Buyer, 24](#)
 - [_color](#)
 - [Pen, 90](#)
 - [_desc](#)
 - [Item, 51](#)
 - [_email](#)
 - [User, 113](#)
 - [_error](#)
 - [EShopError, 40](#)
 - [_eshop](#)
 - [Menu, 65](#)
 - [_id](#)
 - [Item, 51](#)
 - [_isAdmin](#)
 - [Owner, 75](#)
 - [_items](#)
 - [EShop, 37](#)
 - [_name](#)
 - [EShop, 37](#)
 - [Item, 51](#)
 - [User, 114](#)
 - [_order](#)
 - [ShoppingCart, 107](#)
 - [_owner](#)
 - [EShop, 37](#)
 - [Menu, 65](#)
 - [_pages](#)
 - [Paper, 83](#)
 - [_price](#)
 - [Item, 52](#)
 - [_stock](#)
 - [Item, 52](#)
- [_subjects](#)
 - [Notebook, 71](#)
 - [_tipSize](#)
 - [Pen, 90](#)
 - [Pencil, 97](#)
 - [_type](#)
 - [Pencil, 97](#)
 - [_typeMap](#)
 - [Pencil, 97](#)
 - [_user](#)
 - [Menu, 65](#)
 - [_weight](#)
 - [Paper, 83](#)
 - [~EShop](#)
 - [EShop, 28](#)
 - [~Item](#)
 - [Item, 43](#)
 - [~Menu](#)
 - [Menu, 54](#)
- [addBuyer](#)
 - [EShop, 29](#)
- [addItem](#)
 - [EShop, 29](#)
- [addItemOrder](#)
 - [ShoppingCart, 100](#)
- [askYesNo](#)
 - [Menu, 55](#)
- [awardBonus](#)
 - [Buyer, 14](#)
- [B](#)
 - [Pencil, 93](#)
- [Bronze](#)
 - [Buyer, 13](#)
- [Buyer, 9](#)
 - [_bonus, 23](#)
 - [_cart, 23](#)
 - [_categories, 23](#)
 - [_category, 23](#)
 - [_categoryScore, 23](#)
 - [_categoryString, 24](#)
 - [awardBonus, 14](#)
 - [Bronze, 13](#)
 - [Buyer, 13](#)
 - [Category, 13](#)
 - [checkout, 14](#)
 - [clearCart, 15](#)
 - [getBonus, 16](#)
 - [getCategory, 16](#)

- getCategoryName, [17](#)
- getItemOrder, [17](#)
- Gold, [13](#)
- isAdmin, [18](#)
- operator=, [19](#)
- operator==, [19](#)
- placeOrder, [19](#)
- removeFromOrder, [20](#)
- setBonus, [21](#)
- setCategory, [21](#)
- showCart, [22](#)
- Silver, [13](#)
- calculateCourier
 - ShoppingCart, [100](#)
- calculateNet
 - ShoppingCart, [101](#)
- Category
 - Buyer, [13](#)
- changeItemOrderQuantity
 - ShoppingCart, [101](#)
- checkout
 - Buyer, [14](#)
 - ShoppingCart, [102](#)
- checkStatus
 - EShop, [30](#)
- clearCart
 - Buyer, [15](#)
 - ShoppingCart, [103](#)
- error
 - EShopError, [39](#)
- EShop, [24](#)
 - _buyers, [37](#)
 - _items, [37](#)
 - _name, [37](#)
 - _owner, [37](#)
 - ~EShop, [28](#)
 - addBuyer, [29](#)
 - addItem, [29](#)
 - checkStatus, [30](#)
 - EShop, [26, 28](#)
 - getBuyerByEmail, [31](#)
 - getCategories, [31](#)
 - getItemById, [32](#)
 - getName, [32](#)
 - getOwner, [33](#)
 - getProductsInCategory, [33](#)
 - removeBuyer, [34](#)
 - removeItem, [34](#)
 - setOwner, [35](#)
 - showProduct, [35](#)
 - updateItemStock, [36](#)
- EShopError, [38](#)
 - _error, [40](#)
 - error, [39](#)
 - EShopError, [39](#)
- getBasicInfo
 - Item, [44](#)
- getBonus
 - Buyer, [16](#)
- getBuyerByEmail
 - EShop, [31](#)
- getCategories
 - EShop, [31](#)
- getCategory
 - Buyer, [16](#)
 - Item, [44](#)
- getCategoryName
 - Buyer, [17](#)
- getColor
 - Pen, [87](#)
- getDescription
 - Item, [44](#)
- getDetails
 - Item, [44](#)
 - Notebook, [69](#)
 - Paper, [79](#)
 - Pen, [87](#)
 - Pencil, [94](#)
- getEmail
 - User, [110](#)
- getId
 - Item, [45](#)
- getItemById
 - EShop, [32](#)
- getItemOrder
 - Buyer, [17](#)
 - ShoppingCart, [103](#)
- getName
 - EShop, [32](#)
 - Item, [45](#)
 - User, [111](#)
- getOwner
 - EShop, [33](#)
- getPages
 - Paper, [79](#)
- getPrice
 - Item, [45](#)
- getProductsInCategory
 - EShop, [33](#)
- getStock
 - Item, [46](#)
- getSubjects
 - Notebook, [69](#)
- getTipSize
 - Pen, [87](#)
 - Pencil, [94](#)
- getType
 - Pencil, [95](#)
- getWeight
 - Paper, [79](#)
- Gold
 - Buyer, [13](#)
- H
 - Pencil, [93](#)

- HB
 - Pencil, [93](#)
- isAdmin
 - Buyer, [18](#)
 - Owner, [74](#)
 - User, [112](#)
- Item, [40](#)
 - _category, [51](#)
 - _desc, [51](#)
 - _id, [51](#)
 - _name, [51](#)
 - _price, [52](#)
 - _stock, [52](#)
 - ~Item, [43](#)
 - getBasicInfo, [44](#)
 - getCategory, [44](#)
 - getDescription, [44](#)
 - getDetails, [44](#)
 - getId, [45](#)
 - getName, [45](#)
 - getPrice, [45](#)
 - getStock, [46](#)
 - Item, [43](#)
 - operator std::string, [46](#)
 - operator<<, [50](#)
 - operator==, [46](#)
 - setCategory, [47](#)
 - setDescription, [47](#)
 - setId, [48](#)
 - setName, [49](#)
 - setPrice, [49](#)
 - setStock, [50](#)
- item.cpp
 - operator<<, [125](#)
- login
 - Menu, [55](#)
- main
 - main.cpp, [129](#)
- main.cpp
 - main, [129](#)
- Menu, [52](#)
 - _buyer, [65](#)
 - _eshop, [65](#)
 - _owner, [65](#)
 - _user, [65](#)
 - ~Menu, [54](#)
 - askYesNo, [55](#)
 - login, [55](#)
 - Menu, [54](#)
 - showBrowseMenu, [56](#)
 - showBuyerMenu, [57](#)
 - showCartMenu, [58](#)
 - showCategoryMenu, [59](#)
 - showCheckoutMenu, [60](#)
 - showLoginMenu, [60](#)
 - showOwnerMenu, [61](#)
 - showProductMenu, [62](#)
 - showStatusMenu, [63](#)
 - showWelcome, [64](#)
- Notebook, [66](#)
 - _subjects, [71](#)
 - getDetails, [69](#)
 - getSubjects, [69](#)
 - Notebook, [68](#)
 - setId, [69](#)
 - setSubjects, [70](#)
- operator std::string
 - Item, [46](#)
- operator<<
 - Item, [50](#)
 - item.cpp, [125](#)
- operator=
 - Buyer, [19](#)
- operator==
 - Buyer, [19](#)
 - Item, [46](#)
- Owner, [71](#)
 - _isAdmin, [75](#)
 - isAdmin, [74](#)
 - Owner, [74](#)
- Paper, [75](#)
 - _pages, [83](#)
 - _weight, [83](#)
 - getDetails, [79](#)
 - getPages, [79](#)
 - getWeight, [79](#)
 - Paper, [78](#)
 - setId, [80](#)
 - setPages, [80](#)
 - setWeight, [82](#)
- Pen, [83](#)
 - _color, [90](#)
 - _tipSize, [90](#)
 - getColor, [87](#)
 - getDetails, [87](#)
 - getTipSize, [87](#)
 - Pen, [86](#)
 - setColor, [88](#)
 - setId, [88](#)
 - setTipSize, [89](#)
- Pencil, [91](#)
 - _tipSize, [97](#)
 - _type, [97](#)
 - _typeMap, [97](#)
 - B, [93](#)
 - getDetails, [94](#)
 - getTipSize, [94](#)
 - getType, [95](#)
 - H, [93](#)
 - HB, [93](#)
 - Pencil, [93](#)
 - setId, [95](#)

- setTipSize, 96
 - setType, 97
 - Type, 93
- placeOrder
 - Buyer, 19
- removeBuyer
 - EShop, 34
- removeFromOrder
 - Buyer, 20
- removeItem
 - EShop, 34
- removeItemOrder
 - ShoppingCart, 105
- setBonus
 - Buyer, 21
- setCategory
 - Buyer, 21
 - Item, 47
- setColor
 - Pen, 88
- setDescription
 - Item, 47
- setEmail
 - User, 112
- setId
 - Item, 48
 - Notebook, 69
 - Paper, 80
 - Pen, 88
 - Pencil, 95
- setName
 - Item, 49
 - User, 113
- setOwner
 - EShop, 35
- setPages
 - Paper, 80
- setPrice
 - Item, 49
- setStock
 - Item, 50
- setSubjects
 - Notebook, 70
- setTipSize
 - Pen, 89
 - Pencil, 96
- setType
 - Pencil, 97
- setWeight
 - Paper, 82
- ShoppingCart, 98
 - _buyer, 107
 - _order, 107
 - addItemOrder, 100
 - calculateCourier, 100
 - calculateNet, 101
 - changeItemOrderQuantity, 101
 - checkout, 102
 - clearCart, 103
 - getItemOrder, 103
 - removeItemOrder, 105
 - ShoppingCart, 99
 - showCart, 106
 - showBrowseMenu
 - Menu, 56
 - showBuyerMenu
 - Menu, 57
 - showCart
 - Buyer, 22
 - ShoppingCart, 106
 - showCartMenu
 - Menu, 58
 - showCategoryMenu
 - Menu, 59
 - showCheckoutMenu
 - Menu, 60
 - showLoginMenu
 - Menu, 60
 - showOwnerMenu
 - Menu, 61
 - showProduct
 - EShop, 35
 - showProductMenu
 - Menu, 62
 - showStatusMenu
 - Menu, 63
 - showWelcome
 - Menu, 64
 - Silver
 - Buyer, 13
 - src/buyer.cpp, 115, 116
 - src/buyer.h, 117, 118
 - src/eshop.cpp, 119
 - src/eshop.h, 121, 123
 - src/eshoperror.cpp, 123, 124
 - src/eshoperror.h, 124
 - src/item.cpp, 125, 126
 - src/item.h, 127
 - src/main.cpp, 128, 129
 - src/menu.cpp, 130, 131
 - src/menu.h, 135, 136
 - src/notebook.cpp, 137
 - src/notebook.h, 138, 139
 - src/owner.cpp, 139
 - src/owner.h, 140, 141
 - src/paper.cpp, 141
 - src/paper.h, 142, 143
 - src/pen.cpp, 143, 144
 - src/pen.h, 145, 146
 - src/pencil.cpp, 146
 - src/pencil.h, 147, 148
 - src/shoppingcart.cpp, 149
 - src/shoppingcart.h, 151, 152
 - src/user.cpp, 152
 - src/user.h, 153, 154

Type

Pencil, [93](#)

updateItemStock

EShop, [36](#)

User, [108](#)

_email, [113](#)

_name, [114](#)

getEmail, [110](#)

getName, [111](#)

isAdmin, [112](#)

setEmail, [112](#)

setName, [113](#)

User, [110](#)