Ceid Cpp Project 2020

Generated by Doxygen 1.9.1

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 where 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 compile to build.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

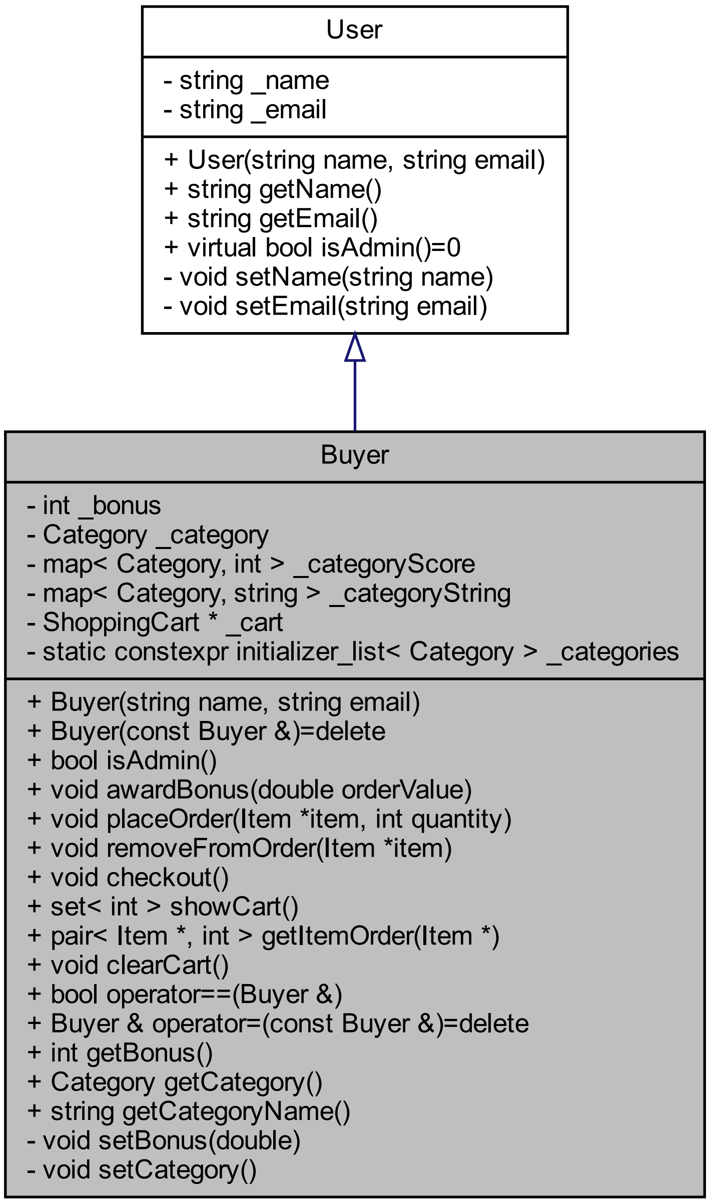
Class Documentation

5.1 Buyer Class Reference

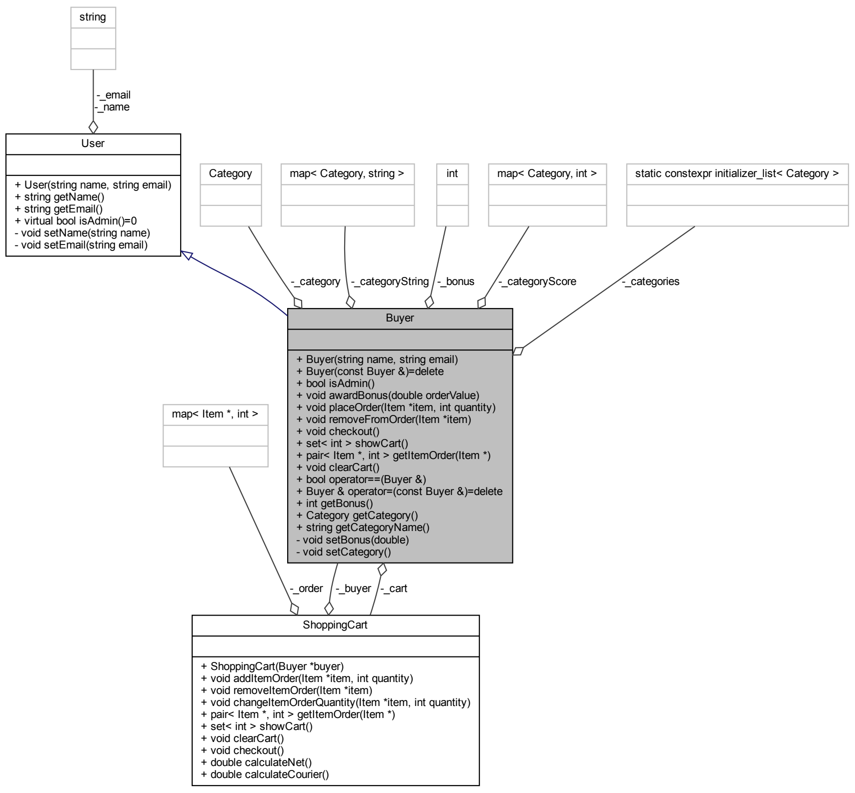
Specialization of User. Describes a Buyer.

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

Inheritance diagram for Buyer:


User
- string _name
- string _email
+ User(string name, string email)
+ string getName()
+ string getEmail()
+ virtual bool isAdmin()=0
- void setName(string name)
- void setEmail(string email)
Buyer
- int _bonus
- Category _category
- map< Category, int > _categoryScore
- map< Category, string > _categoryString
- ShoppingCart * _cart
- static constexpr initializer_list< Category > _categories
+ Buyer(string name, string email)
+ Buyer(const Buyer &)=delete
+ bool isAdmin()
+ void awardBonus(double orderValue)
+ void placeOrder(Item *item, int quantity)
+ void removeFromOrder(Item *item)
+ void checkout()
+ set< int > showCart()
+ pair< Item *, int > getItemOrder(Item *)
+ void clearCart()
+ bool operator==(Buyer &)
+ Buyer & operator=(const Buyer &)=delete
+ int getBonus()
+ Category getCategory()
+ string getCategoryName()
- void setBonus(double)
- void setCategory()

Collaboration diagram for Buyer:


string
-_email
-_name
User
+ User(string name, string email)
+ string getName()
+ string getEmail()
+ virtual bool isAdmin()=0
- void setName(string name)
- void setEmail(string email)
Category
map< Category, string >
int
map< Category, int >
static constexpr initializer_list< Category >
-_category
-_categoryString
-_bonus
-_categoryScore
-_categories
Buyer
+ Buyer(string name, string email)
+ Buyer(const Buyer &)=delete
+ bool isAdmin()
+ void awardBonus(double orderValue)
+ void placeOrder(Item *item, int quantity)
+ void removeFromOrder(Item *item)
+ void checkout()
+ set< int > showCart()
+ pair< Item *, int > getItemOrder(Item *)
+ void clearCart()
+ bool operator==(Buyer &)
+ Buyer & operator=(const Buyer &)=delete
+ int getBonus()
+ Category getCategory()
+ string getCategoryName()
- void setBonus(double)
- void setCategory()
map< Item *, int >
-_order
-_buyer
-_cart
ShoppingCart
+ ShoppingCart(Buyer *buyer)
+ void addItemOrder(Item *item, int quantity)
+ void removeItemOrder(Item *item)
+ void changeItemOrderQuantity(Item *item, int quantity)
+ pair< Item *, int > getItemOrder(Item *)
+ set< int > showCart()
+ void clearCart()
+ void checkout()
+ double calculateNet()
+ double calculateCourier()

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 &)

  Comparission 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

name	User's name
email	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

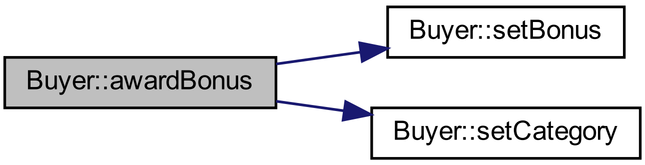
orderValue	double The value of the order

Definition at line 13 of file buyer.cpp.

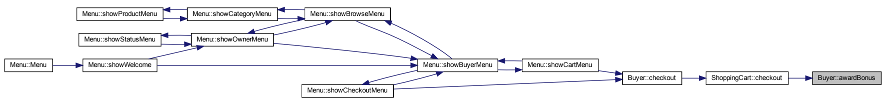
References setBonus(), and setCategory().

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 ( )
```

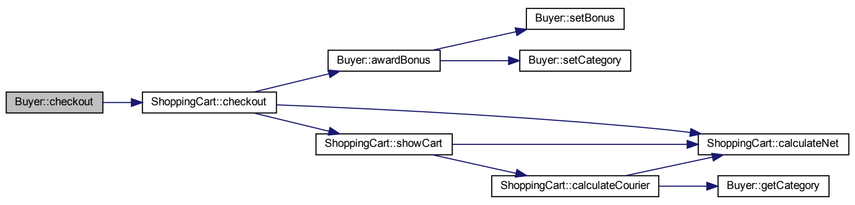
Wrapper to ShoppingCart::checkout()

Definition at line 46 of file buyer.cpp.

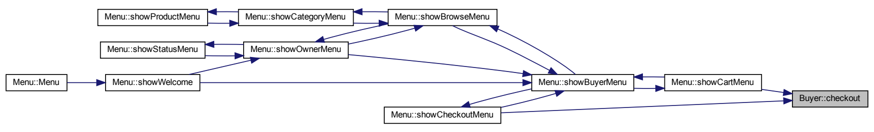
References _cart, and ShoppingCart::checkout().

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 ( )
```

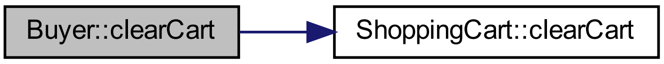
Wrapper to ShoppingCart::clearCart()

Definition at line 72 of file buyer.cpp.

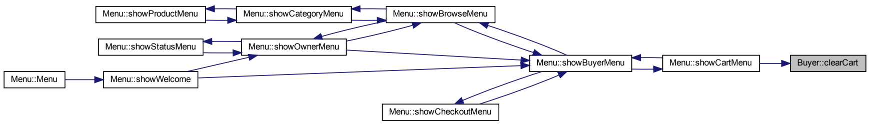
References _cart, and ShoppingCart::clearCart().

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 ( )
```

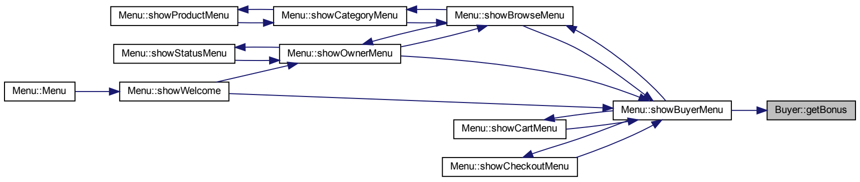
Get the Buyer's bonus.

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 ( )`

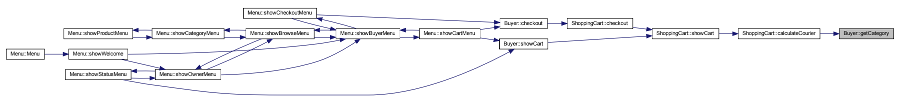
Get the Buyer's category.

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 ( )`

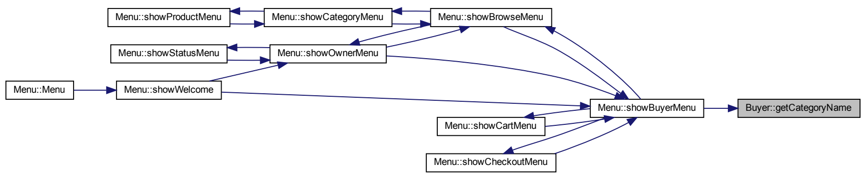
Get the Buyer's category as string.

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

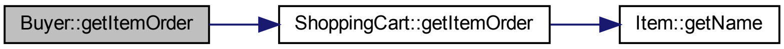
Returns a pair containing an item pointer and the quantity int in the cart

Definition at line 62 of file buyer.cpp.

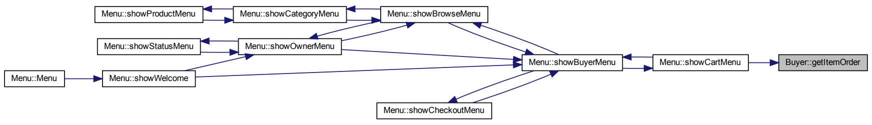
References _cart, and ShoppingCart::getItemOrder().

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]
```

Impementation of isAdmin() of User.

Parameters

none	

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 )
```

Comparission operator between two Buyers.

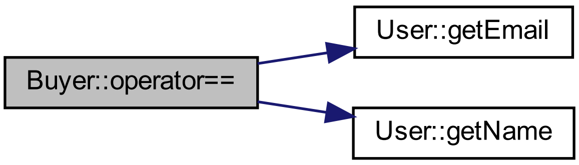
Returns

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

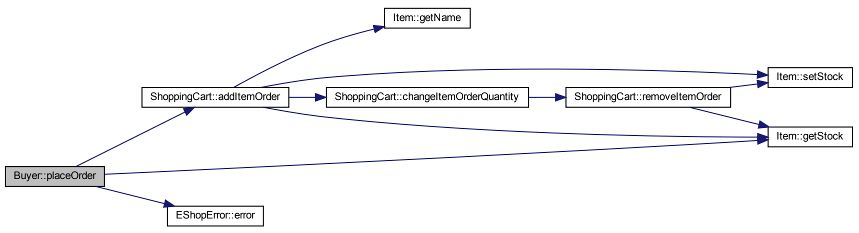
item	Item∗ The item to add to the cart
quantity	int The selected quantity

Definition at line 20 of file buyer.cpp.

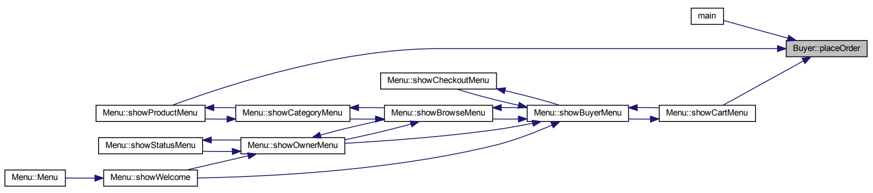
References _cart, ShoppingCart::addItemOrder(), EShopError::error(), and Item::getStock().

Referenced by main(), Menu::showCartMenu(), and Menu::showProductMenu().

Here is the call graph for this function:


Item::getName
Item::setStock
ShoppingCart::addItemOrder
ShoppingCart::changeItemOrderQuantity
ShoppingCart::removeItemOrder
Item::getStock
Buyer::placeOrder
EShopError::error

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

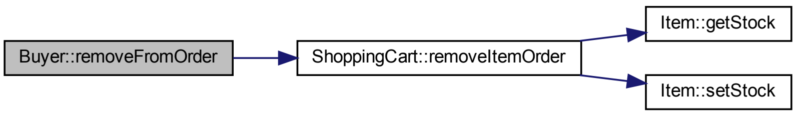
item	Item∗ The item to remove

Definition at line 40 of file buyer.cpp.

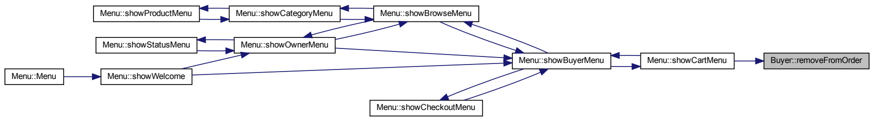
References _cart, and ShoppingCart::removeItemOrder().

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]
```

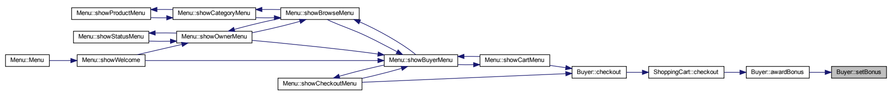
Sets a buyer's bonus based on the cost.

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]
```

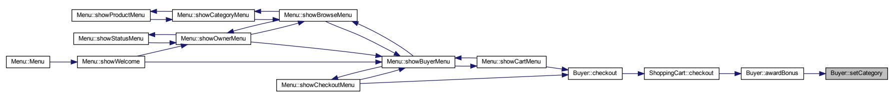
Sets a buyer's category based on _bonus.

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

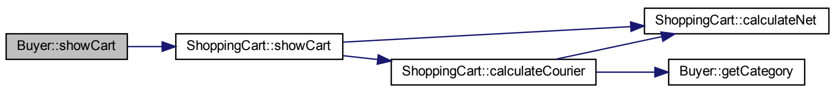
Returns a set of the IDs of the items in the cart

Definition at line 52 of file buyer.cpp.

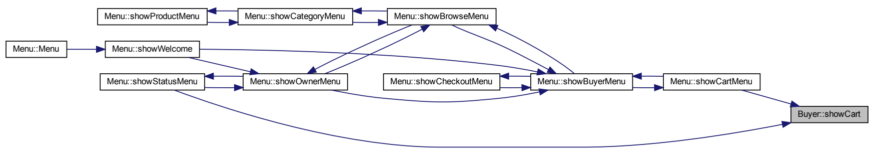
References _cart, and ShoppingCart::showCart().

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:

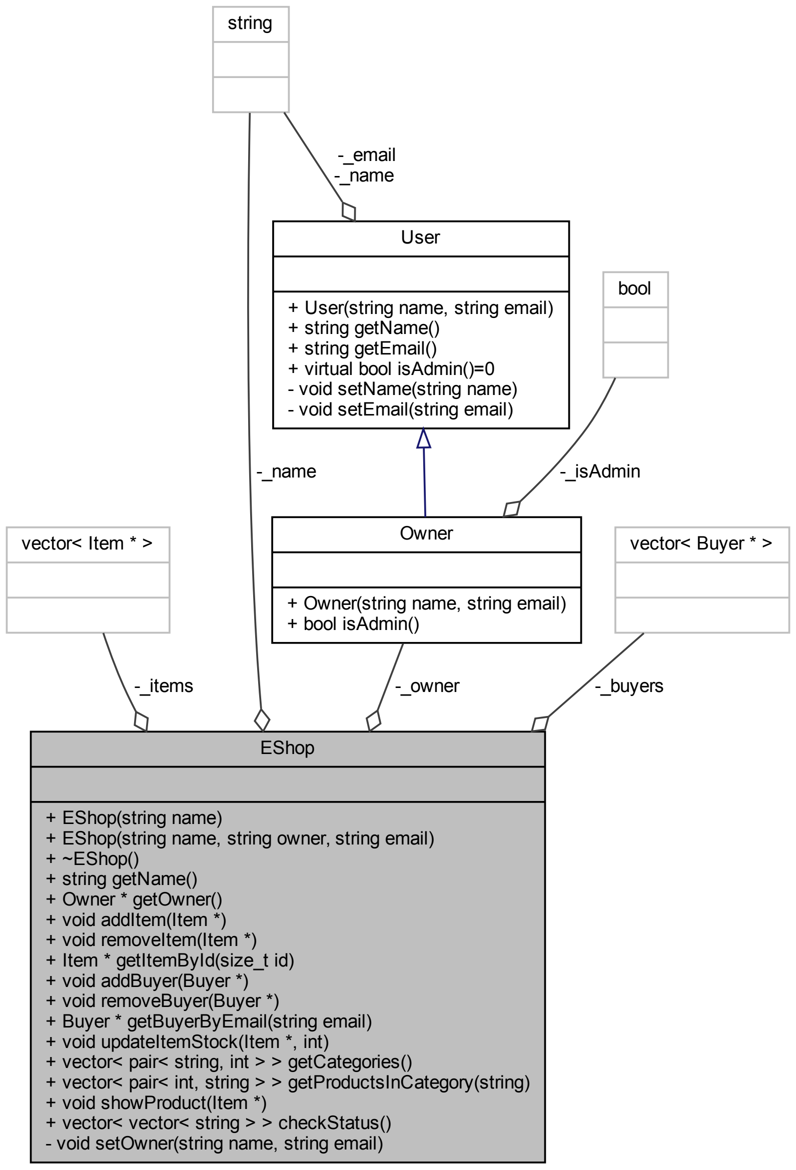
- src/buyer.h
- src/buyer.cpp

5.2 EShop Class Reference

Class implementing the e-shop.

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

Collaboration diagram for EShop:


string
-_email
-_name
User
+ User(string name, string email)
+ string getName()
+ string getEmail()
+ virtual bool isAdmin()=0
- void setName(string name)
- void setEmail(string email)
bool
-_name
-_isAdmin
Owner
+ Owner(string name, string email)
+ bool isAdmin()
vector< Item * >
vector< Buyer * >
-_items
-_owner
-_buyers
EShop
+ EShop(string name)
+ EShop(string name, string owner, string email)
+ ~EShop()
+ string getName()
+ Owner * getOwner()
+ void addItem(Item *)
+ void removeItem(Item *)
+ Item * getItemById(size_t id)
+ void addBuyer(Buyer *)
+ void removeBuyer(Buyer *)
+ Buyer * getBuyerByEmail(string email)
+ void updateItemStock(Item *, int)
+ vector< pair< string, int > > getCategories()
+ vector< pair< int, string > > getProductsInCategory(string)
+ void showProduct(Item *)
+ vector< vector< string > > checkStatus()
- void setOwner(string name, string email)

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

name	<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.

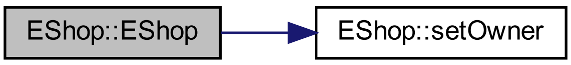
Parameters

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

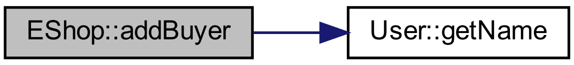
Checks if the Buyer already exists, if it does, throws and exception

Definition at line 75 of file eshop.cpp.

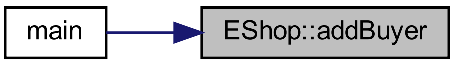
References _buyers, and User::getName().

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.

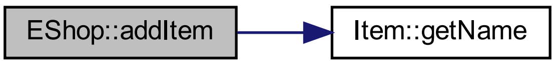
Checks if the Item already exists, if it does, throws and exception

Definition at line 41 of file eshop.cpp.

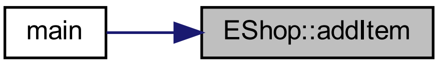
References _items, and Item::getName().

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

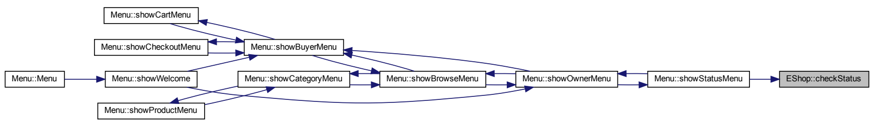
<vector<vector<string>>>

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 )
```

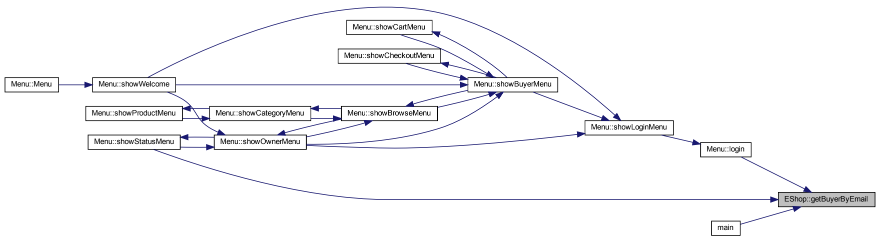
Return a Buyer reference if the Buyer's email is found in the e-shop.

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 getCategories()

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

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

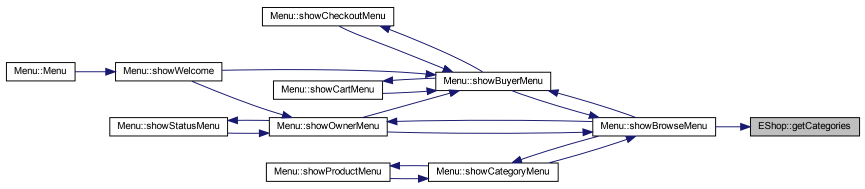
<vector<pair<string, int>>>

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 )
```

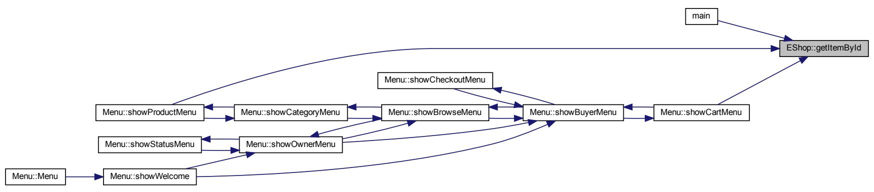
Return an Item reference if the Item's ID is found in the e-shop.

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 ( )
```

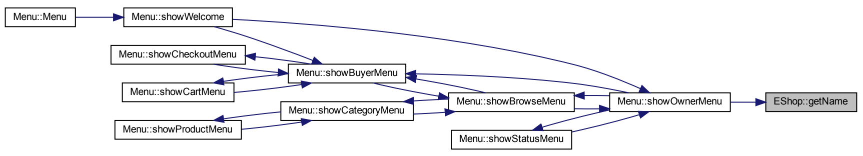
Get the name of the e-shop.

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 ( )
```

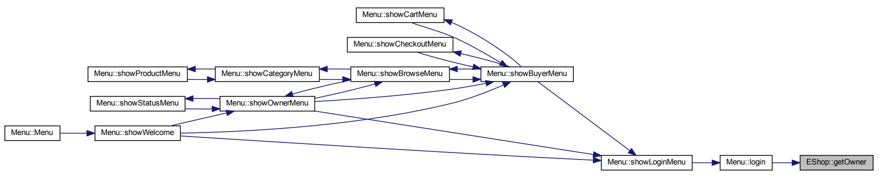
Get the owner.

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

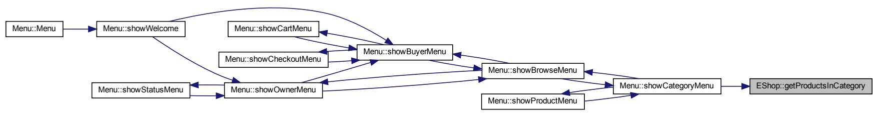
<vector<pair<int, string>>>

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 )
```

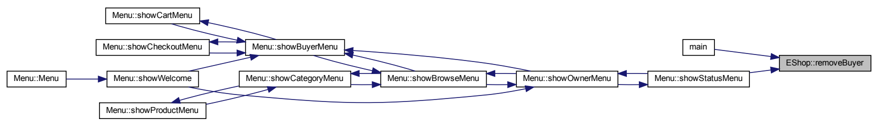
Remove a Buyer from the eshop if it exists and clears their cart.

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 )
```

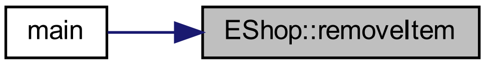
Remove an Item from the eshop if it exists.

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]
```

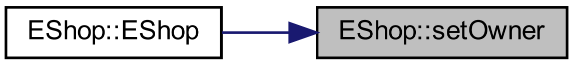
Sets the name of the owner.

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()

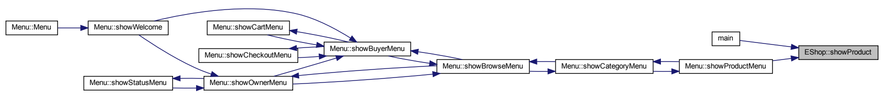
```
void EShop::showProduct (
            Item * item )
```

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 )
```

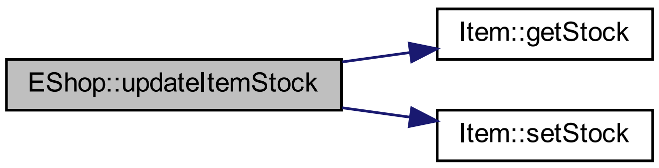
Update an Item's stock.

Definition at line 108 of file eshop.cpp.

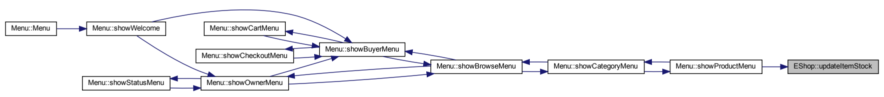
References Item::getStock(), and Item::setStock().

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:

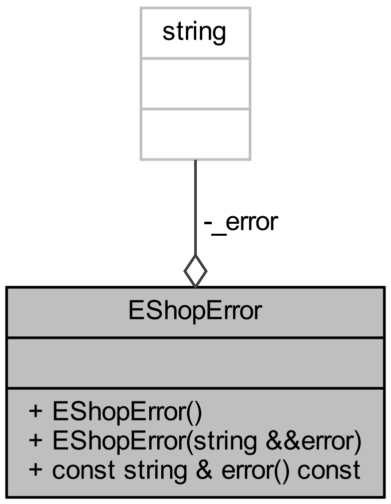
- src/eshop.h
- src/eshop.cpp

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

error	The error message to pass

Definition at line 25 of file eshoperror.h.

5.3.3 Member Function Documentation

5.3.3.1 error()

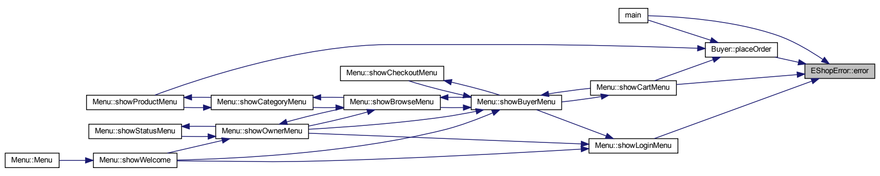
```
const string& EShopError::error ( ) const  [inline]
```

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:



5.3.4 Member Data Documentation

5.3.4.1 _error

`string EShopError::_error [private]`

Definition at line 35 of file eshoperror.h.

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

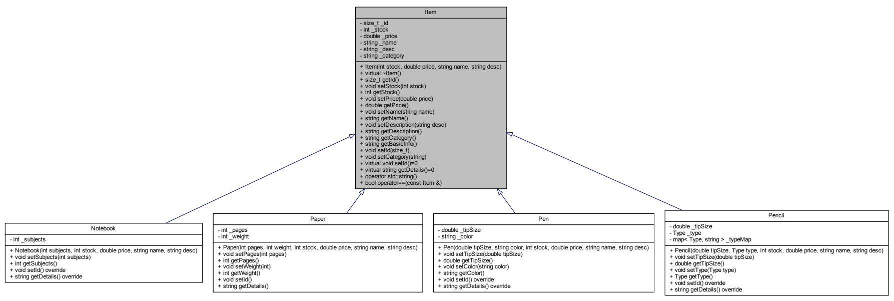
- src/eshoperror.h

5.4 Item Class Reference

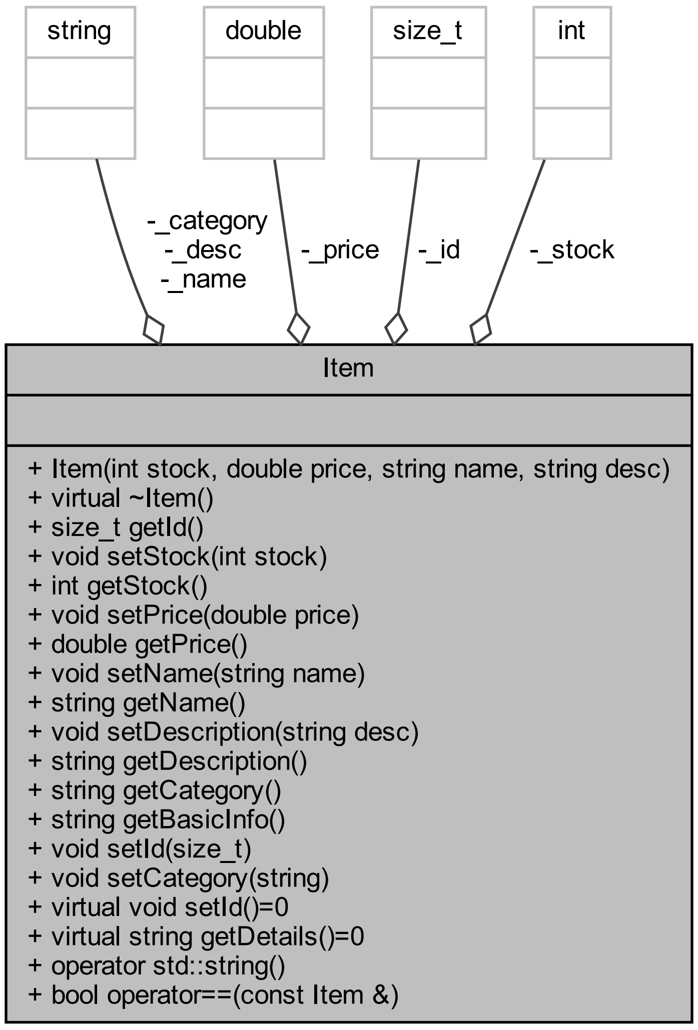
Base class for all other items.

`#include <item.h>`

Inheritance diagram for Item:


Item
- size_t _id
- int _stock
- double _price
- string _name
- string _desc
- string _category
+ Item(int stock, double price, string name, string desc)
+ virtual ~Item()
+ size_t getId()
+ void setStock(int stock)
+ int getStock()
+ void setPrice(double price)
+ double getPrice()
+ void setName(string name)
+ string getName()
+ void setDescription(string desc)
+ string getDescription()
+ string getCategory()
+ string getBasicInfo()
+ void setId(size_t)
+ void setCategory(string)
+ virtual void setId()=0
+ virtual string getDetails()=0
+ operator std::string()
+ bool operator==(const Item &)
Notebook
- int _subjects
+ Notebook(int subjects, int stock, double price, string name, string desc)
+ void setSubjects(int subjects)
+ int getSubjects()
+ void setId() override
+ string getDetails() override
Paper
- int _pages
- int _weight
+ Paper(int pages, int weight, int stock, double price, string name, string desc)
+ void setPages(int pages)
+ int getPages()
+ void setWeight(int)
+ int getWeight()
+ void setId()
+ string getDetails()
Pen
- double _tipSize
- string _color
+ Pen(double tipSize, string color, int stock, double price, string name, string desc)
+ void setTipSize(double tipSize)
+ double getTipSize()
+ void setColor(string color)
+ string getColor()
+ void setId() override
+ string getDetails() override
Pencil
- double _tipSize
- Type _type
- map< Type, string > _typeMap
+ Pencil(double tipSize, Type type, int stock, double price, string name, string desc)
+ void setTipSize(double tipSize)
+ double getTipSize()
+ void setType(Type type)
+ Type getType()
+ void setId() override
+ string getDetails() override

Collaboration diagram for Item:


string
double
size_t
int
-_category
-_desc
-_name
-_price
-_id
-_stock
Item
+ Item(int stock, double price, string name, string desc)
+ virtual ~Item()
+ size_t getId()
+ void setStock(int stock)
+ int getStock()
+ void setPrice(double price)
+ double getPrice()
+ void setName(string name)
+ string getName()
+ void setDescription(string desc)
+ string getDescription()
+ string getCategory()
+ string getBasicInfo()
+ void setId(size_t)
+ void setCategory(string)
+ virtual void setId()=0
+ virtual string getDetails()=0
+ operator std::string()
+ bool operator==(const 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 reimplementation 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 outstream 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, Descritpion. 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.

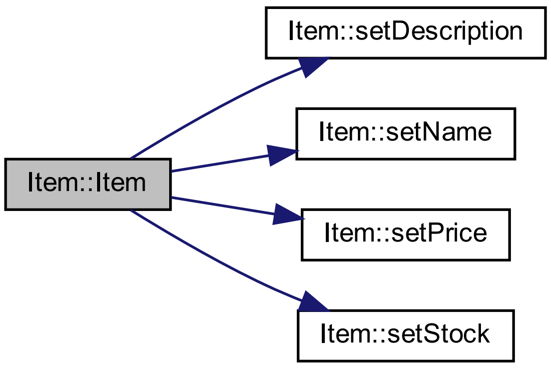
Parameters

stock	<int>.The stock the item should be created with
price	<double> The price of the item
name	<string> The name of the item, doesn't need to be unique
desc	<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 reimplementation 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 commaspace-separated string. The order of the returned inforation 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

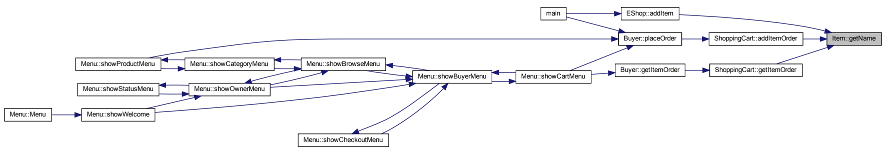
<string>

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

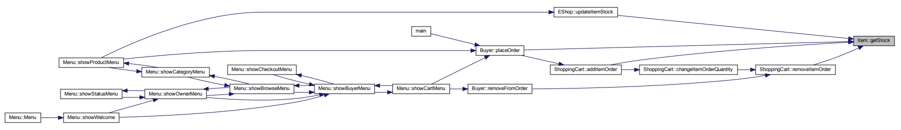
<int>

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.

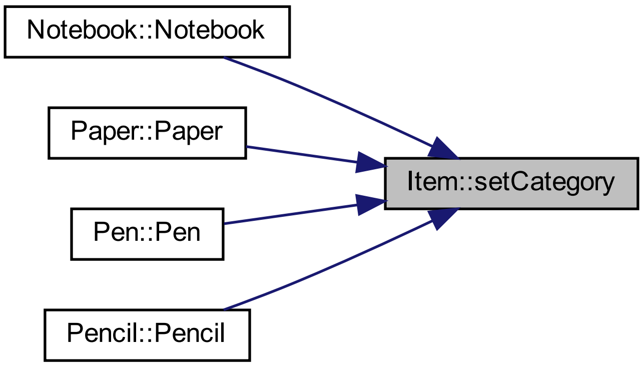
Set the item's category by removing the first character of the string returned by typeid (In the case of gcc 10 on my machine it is the name of the class prefix by the length of the string (according to the reference it is implementation specific))

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

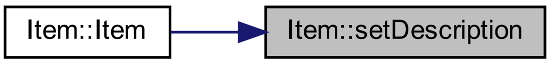
desc	<string> The description to set

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]
```

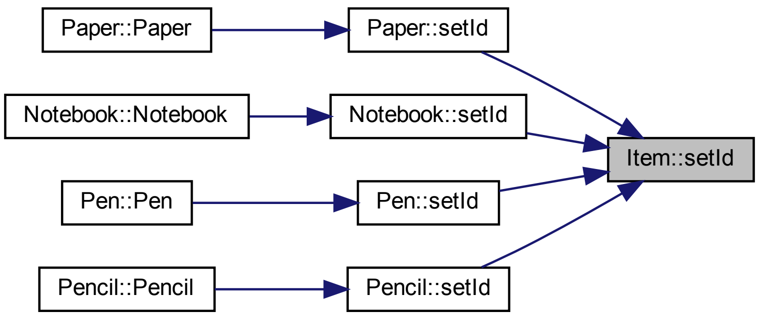
Set the Item's ID.

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

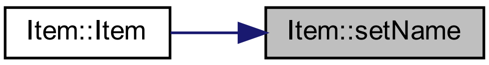
name	<string> The name to set

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

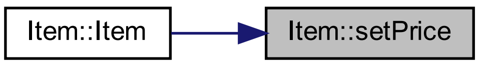
price	<double> The price to set

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

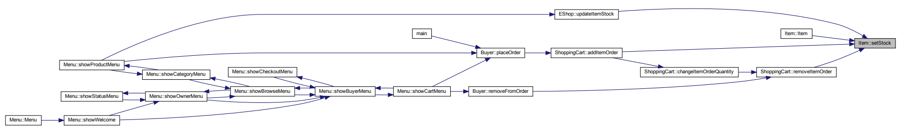
stock	<int> The stock to set

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 outstream 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:

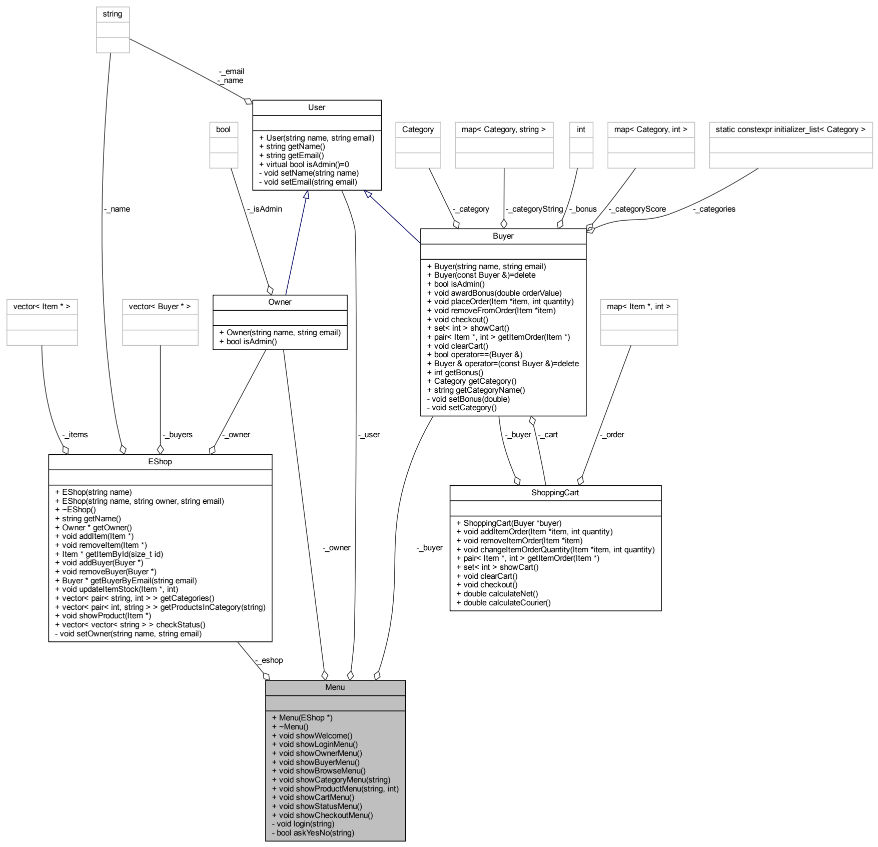
- src/item.h
- src/item.cpp

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

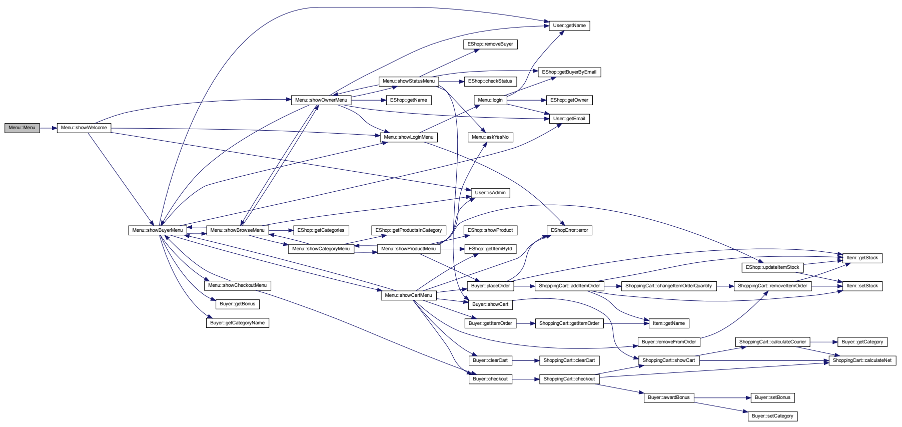
5.5.2.1 Menu()

```
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()

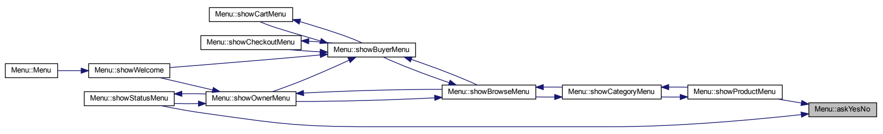
```
bool Menu::askYesNo (
             string message )  [private]
```

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]
```

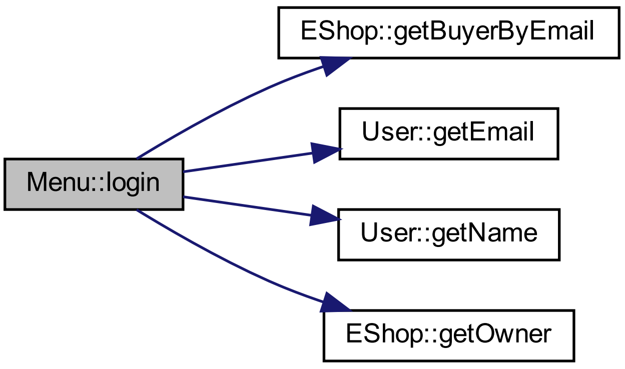
User authentication.

Definition at line 322 of file menu.cpp.

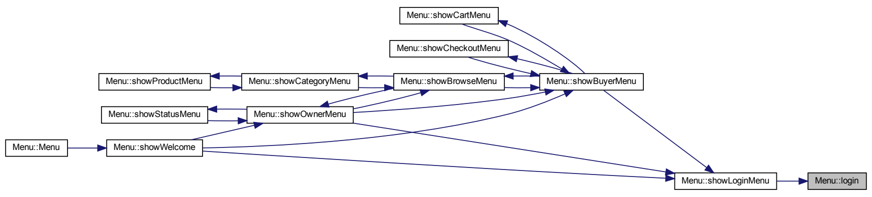
References _buyer, _eshop, _owner, _user, EShop::getBuyerByEmail(), User::getEmail(), User::getName(), and EShop::getOwner().

Referenced by showLoginMenu().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.3 showBrowseMenu()

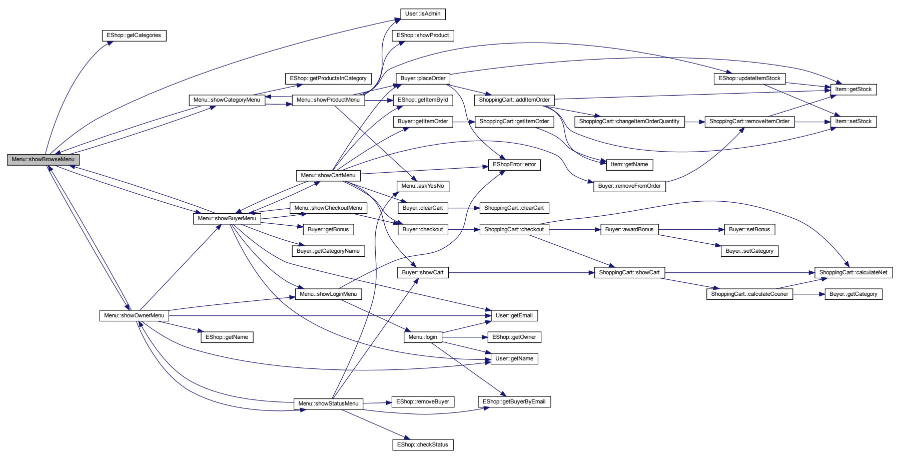
```
void Menu::showBrowseMenu ( )
```

Definition at line 150 of file menu.cpp.

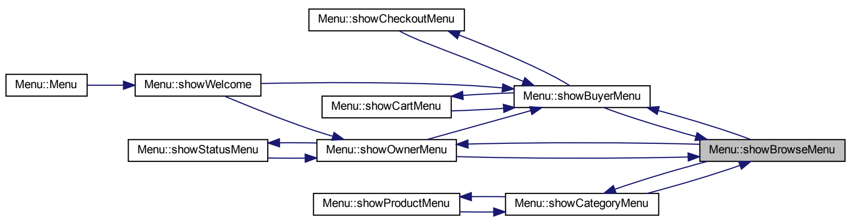
References _eshop, _user, EShop::getCategories(), User::isAdmin(), showBuyerMenu(), showCategoryMenu(), and showOwnerMenu().

Referenced by showBuyerMenu(), showCategoryMenu(), and showOwnerMenu().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.4 showBuyerMenu()

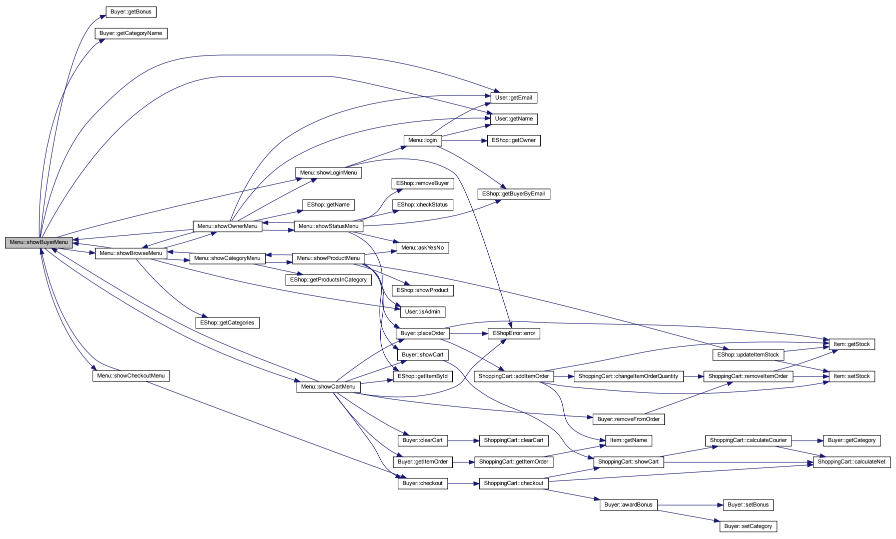
```
void Menu::showBuyerMenu ( )
```

Definition at line 109 of file menu.cpp.

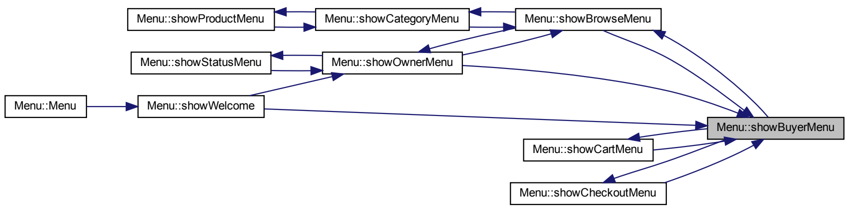
References _buyer, Buyer::getBonus(), Buyer::getCategoryName(), User::getEmail(), User::getName(), showBrowseMenu(), showCartMenu(), showCheckoutMenu(), and showLoginMenu().

Referenced by showBrowseMenu(), showCartMenu(), showCheckoutMenu(), showOwnerMenu(), and showWelcome().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.5 showCartMenu()

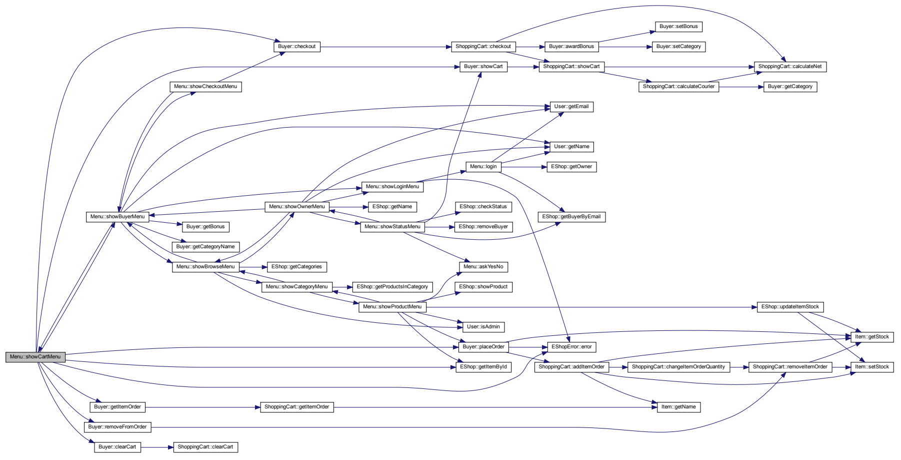
```
void Menu::showCartMenu ( )
```

Definition at line 225 of file menu.cpp.

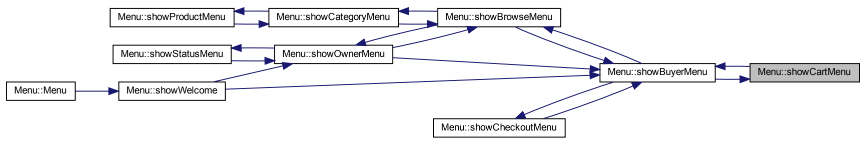
References _buyer, _eshop, Buyer::checkout(), Buyer::clearCart(), EShopError::error(), EShop::getItemById(), Buyer::getItemOrder(), Buyer::placeOrder(), Buyer::removeFromOrder(), showBuyerMenu(), and Buyer::showCart().

Referenced by showBuyerMenu().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.6 showCategoryMenu()

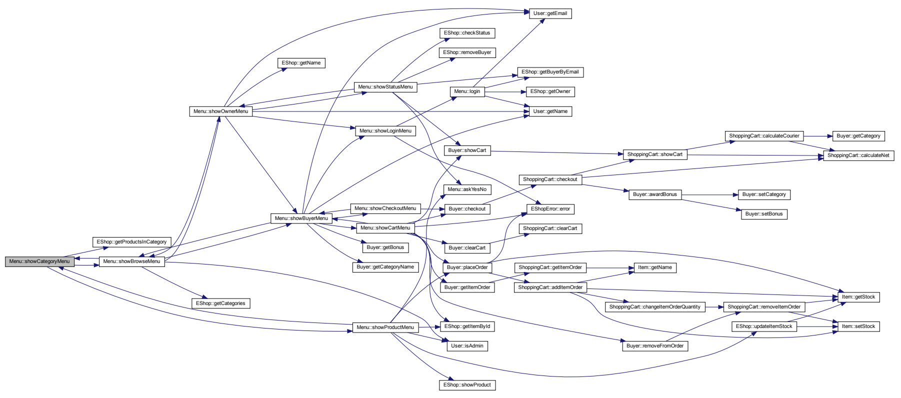
```
void Menu::showCategoryMenu (
            string category )
```

Definition at line 178 of file menu.cpp.

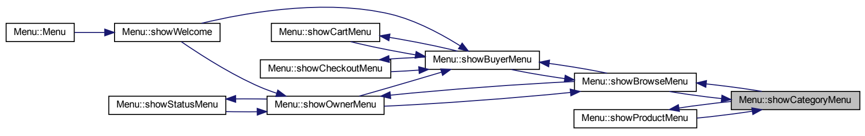
References _eshop, EShop::getProductsInCategory(), showBrowseMenu(), and showProductMenu().

Referenced by showBrowseMenu(), and showProductMenu().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.7 showCheckoutMenu()

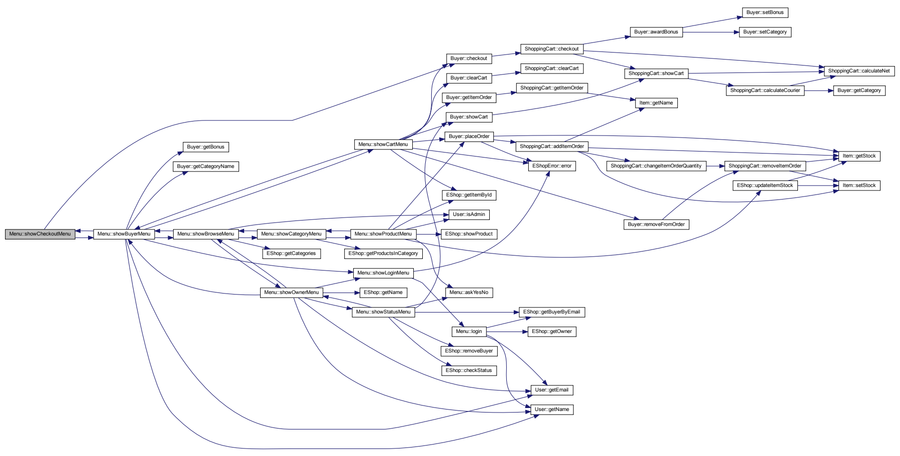
```
void Menu::showCheckoutMenu ( )
```

Definition at line 311 of file menu.cpp.

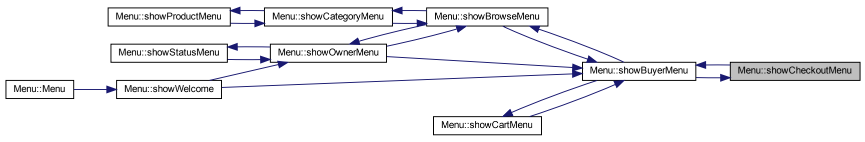
References _buyer, Buyer::checkout(), and showBuyerMenu().

Referenced by showBuyerMenu().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.8 showLoginMenu()

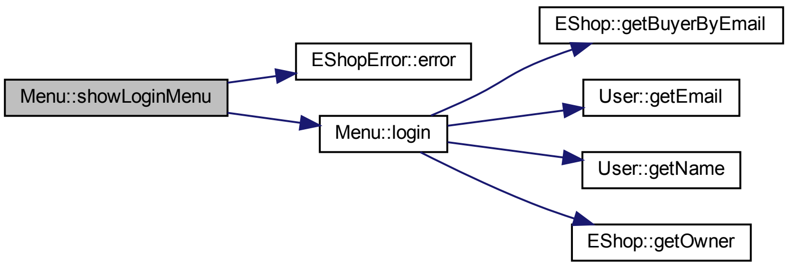
```
void Menu::showLoginMenu ( )
```

Definition at line 25 of file menu.cpp.

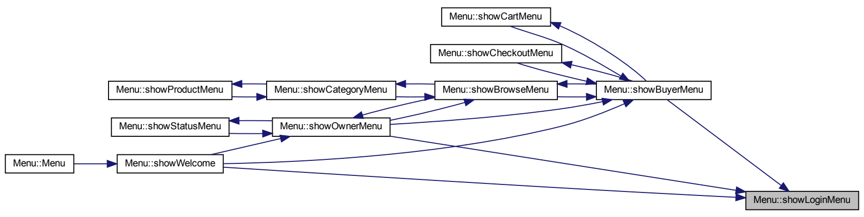
References EShopError::error(), and login().

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()

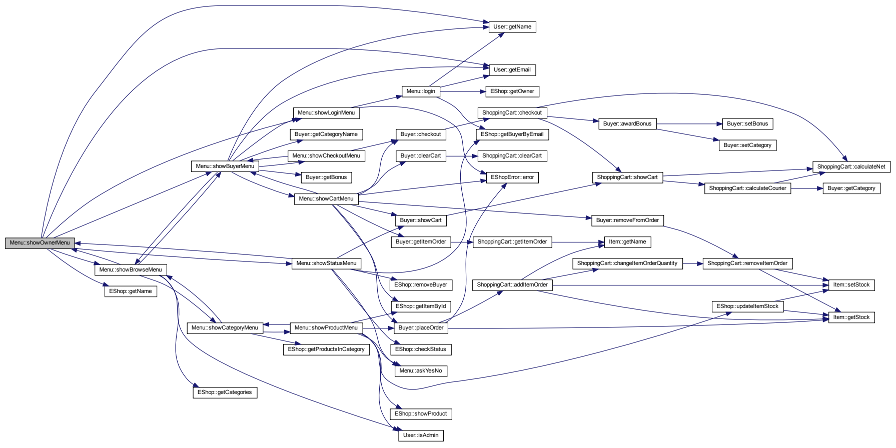
```
void Menu::showOwnerMenu ( )
```

Definition at line 39 of file menu.cpp.

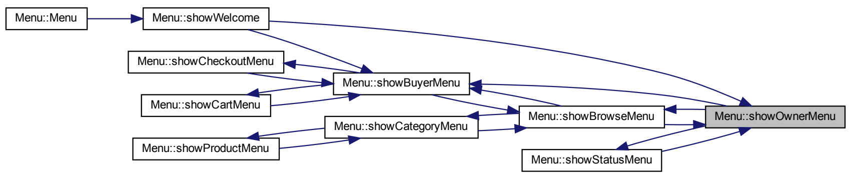
References _eshop, _owner, User::getEmail(), EShop::getName(), User::getName(), showBrowseMenu(), showBuyerMenu(), showLoginMenu(), and showStatusMenu().

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()

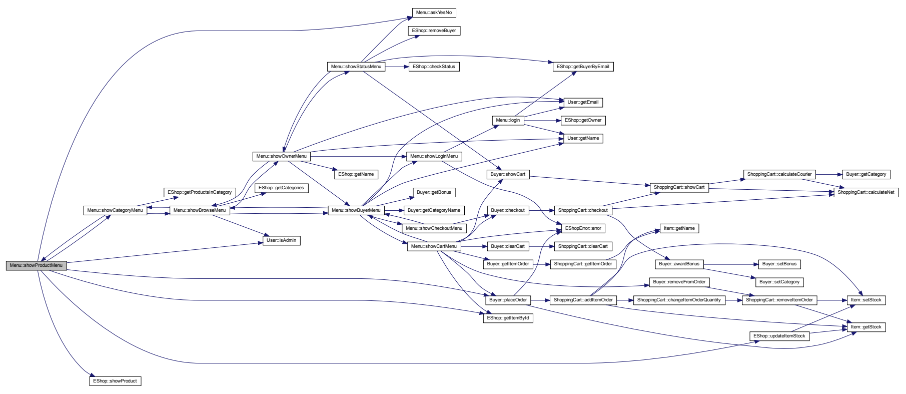
```
void Menu::showProductMenu (
            string back,
            int id )
```

Definition at line 203 of file menu.cpp.

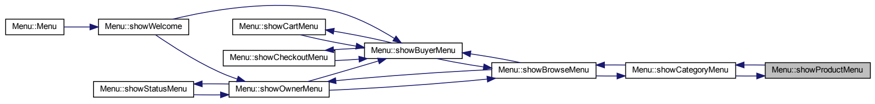
References _buyer, _eshop, _user, askYesNo(), EShop::getItemById(), User::isAdmin(), Buyer::placeOrder(), showCategoryMenu(), EShop::showProduct(), and EShop::updateItemStock().

Referenced by showCategoryMenu().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.11 showStatusMenu()

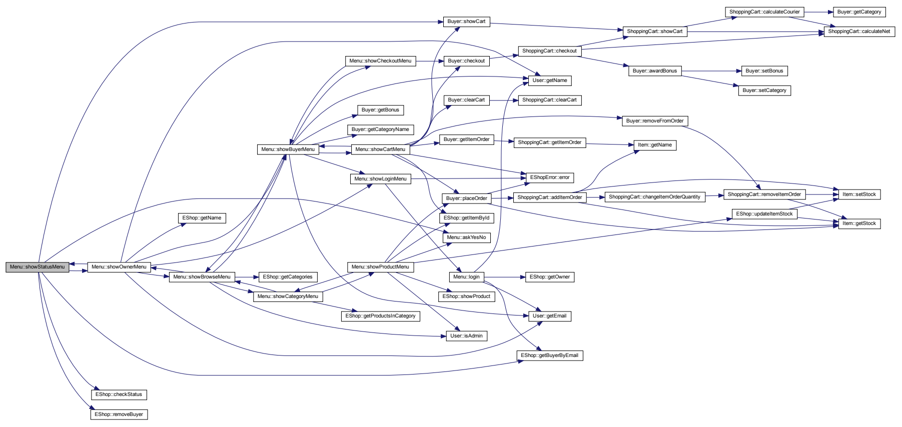
```
void Menu::showStatusMenu ( )
```

Definition at line 77 of file menu.cpp.

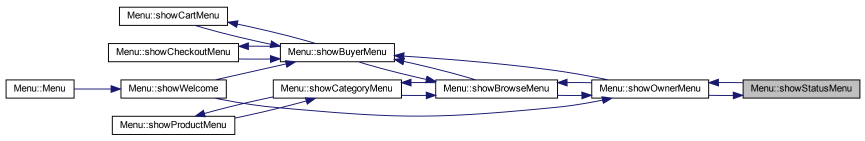
References _eshop, askYesNo(), EShop::checkStatus(), EShop::getBuyerByEmail(), EShop::removeBuyer(), Buyer::showCart(), and showOwnerMenu().

Referenced by showOwnerMenu().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.12 showWelcome()

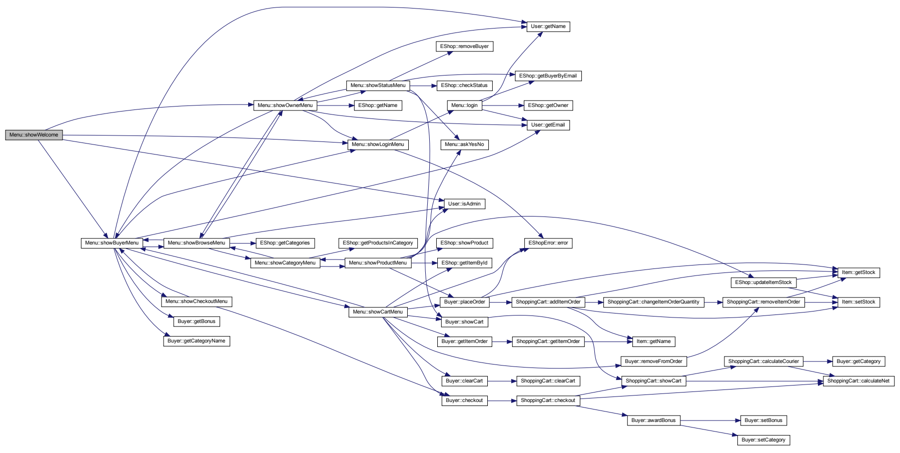
```
void Menu::showWelcome ( )
```

Definition at line 13 of file menu.cpp.

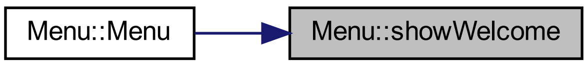
References _eshop, _user, User::isAdmin(), showBuyerMenu(), showLoginMenu(), and showOwnerMenu().

Referenced by Menu().

Here is the call graph for this function:



Here is the caller 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:

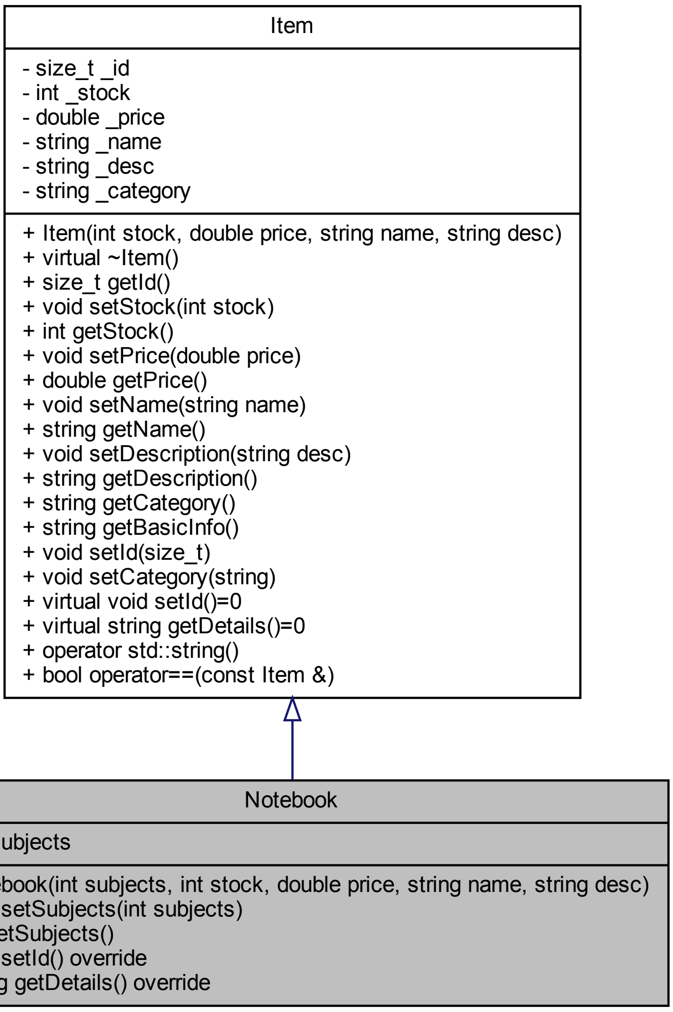
- src/menu.h
- src/menu.cpp

5.6 Notebook Class Reference

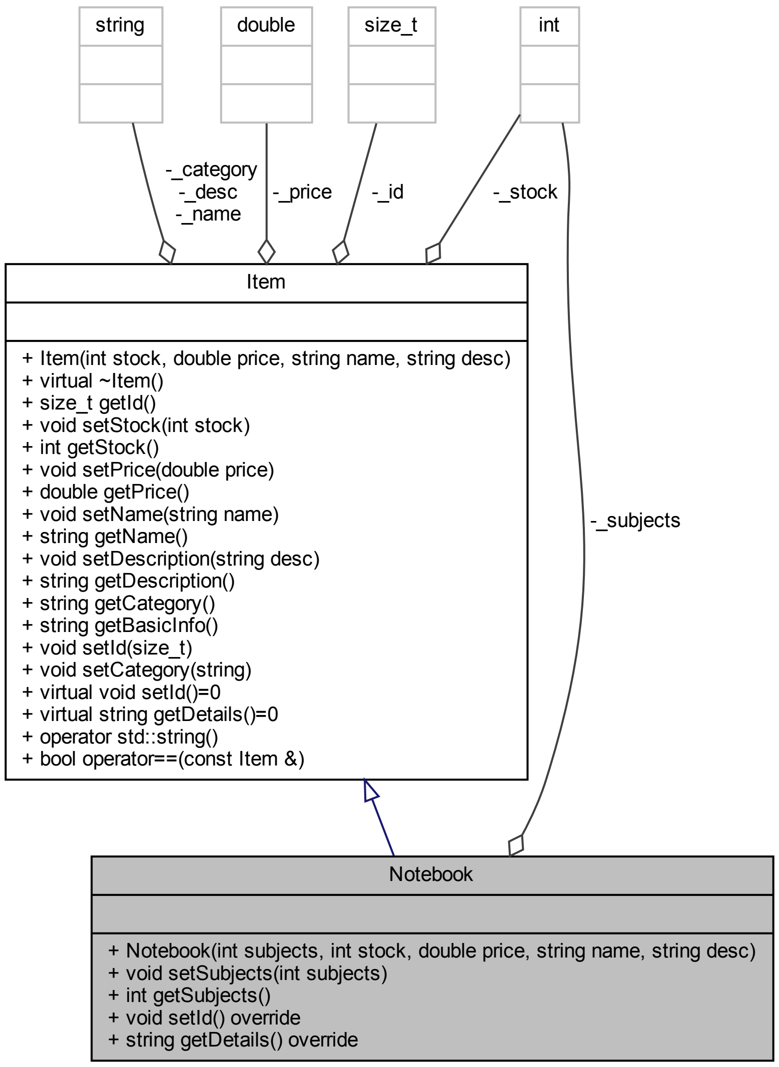
Class representing a Notebook.

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

Inheritance diagram for Notebook:


Item
- size_t _id
- int _stock
- double _price
- string _name
- string _desc
- string _category
+ Item(int stock, double price, string name, string desc)
+ virtual ~Item()
+ size_t getId()
+ void setStock(int stock)
+ int getStock()
+ void setPrice(double price)
+ double getPrice()
+ void setName(string name)
+ string getName()
+ void setDescription(string desc)
+ string getDescription()
+ string getCategory()
+ string getBasicInfo()
+ void setId(size_t)
+ void setCategory(string)
+ virtual void setId()=0
+ virtual string getDetails()=0
+ operator std::string()
+ bool operator==(const Item &)
Notebook
- int _subjects
+ Notebook(int subjects, int stock, double price, string name, string desc)
+ void setSubjects(int subjects)
+ int getSubjects()
+ void setId() override
+ string getDetails() override

Collaboration diagram for Notebook:


string
double
size_t
int
-_category
-_desc
-_name
-_price
-_id
-_stock
-_subjects
Item
+ Item(int stock, double price, string name, string desc)
+ virtual ~Item()
+ size_t getId()
+ void setStock(int stock)
+ int getStock()
+ void setPrice(double price)
+ double getPrice()
+ void setName(string name)
+ string getName()
+ void setDescription(string desc)
+ string getDescription()
+ string getCategory()
+ string getBasicInfo()
+ void setId(size_t)
+ void setCategory(string)
+ virtual void setId()=0
+ virtual string getDetails()=0
+ operator std::string()
+ bool operator==(const Item &)
Notebook
+ Notebook(int subjects, int stock, double price, string name, string desc)
+ void setSubjects(int subjects)
+ int getSubjects()
+ void setId() override
+ string getDetails() override

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.

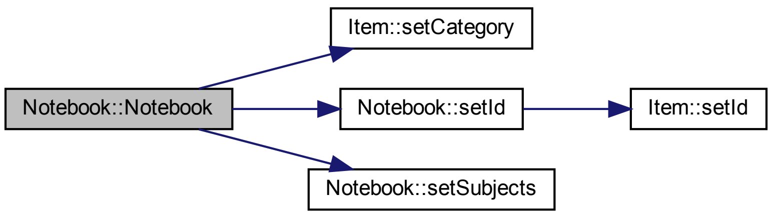
Parameters

subjects	<int>
stock	<int>
price	<double>
name	<string>
desc	<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.

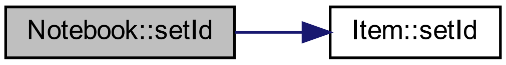
Implements Item.

Definition at line 21 of file notebook.cpp.

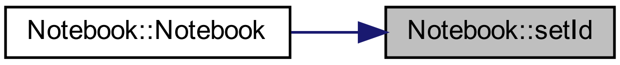
References _subjects, and Item::setId().

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

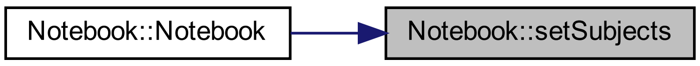
subjects	<int>

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:

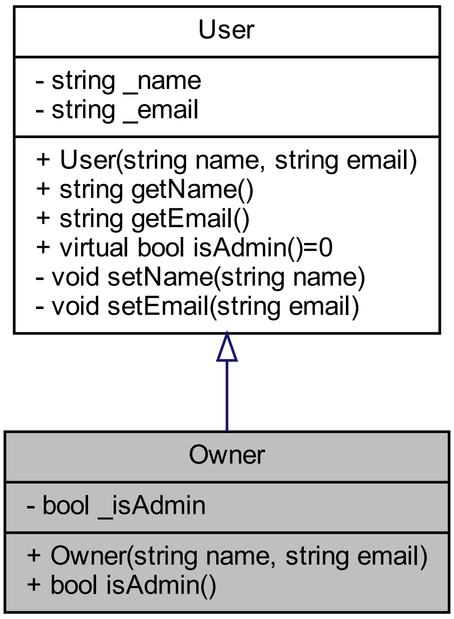
- src/notebook.h
- src/notebook.cpp

5.7 Owner Class Reference

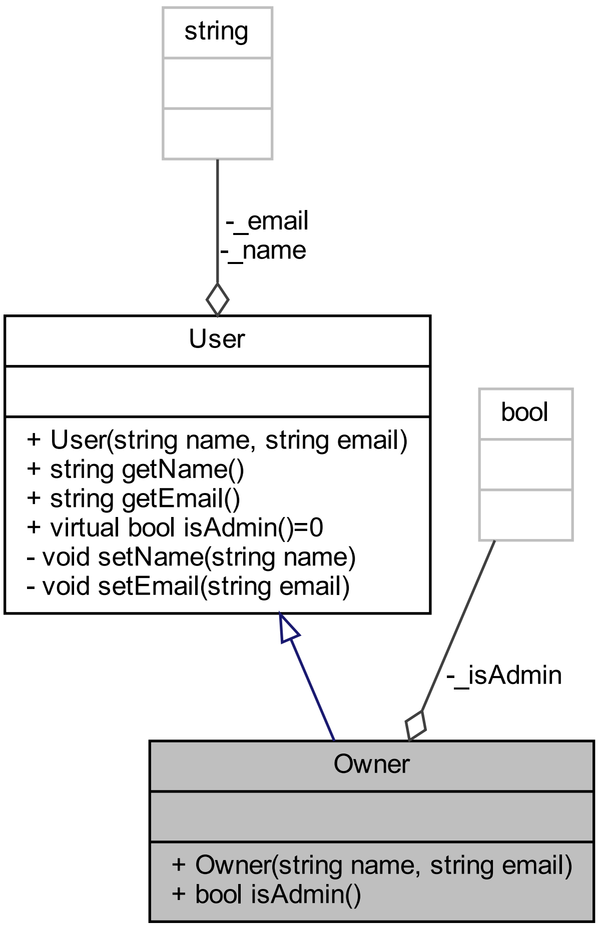
Specialization of User. Describes an Owner.

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

Inheritance diagram for Owner:


User
- string _name
- string _email
+ User(string name, string email)
+ string getName()
+ string getEmail()
+ virtual bool isAdmin()=0
- void setName(string name)
- void setEmail(string email)
Owner
- bool _isAdmin
+ Owner(string name, string email)
+ bool isAdmin()

Collaboration diagram for Owner:


string
-_email
-_name
User
+ User(string name, string email)
+ string getName()
+ string getEmail()
+ virtual bool isAdmin()=0
- void setName(string name)
- void setEmail(string email)
bool
-_isAdmin
Owner
+ Owner(string name, string email)
+ bool isAdmin()

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

name	Owner's name
email	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]
```

Impementation of isAdmin() of User.

Parameters

none	

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:

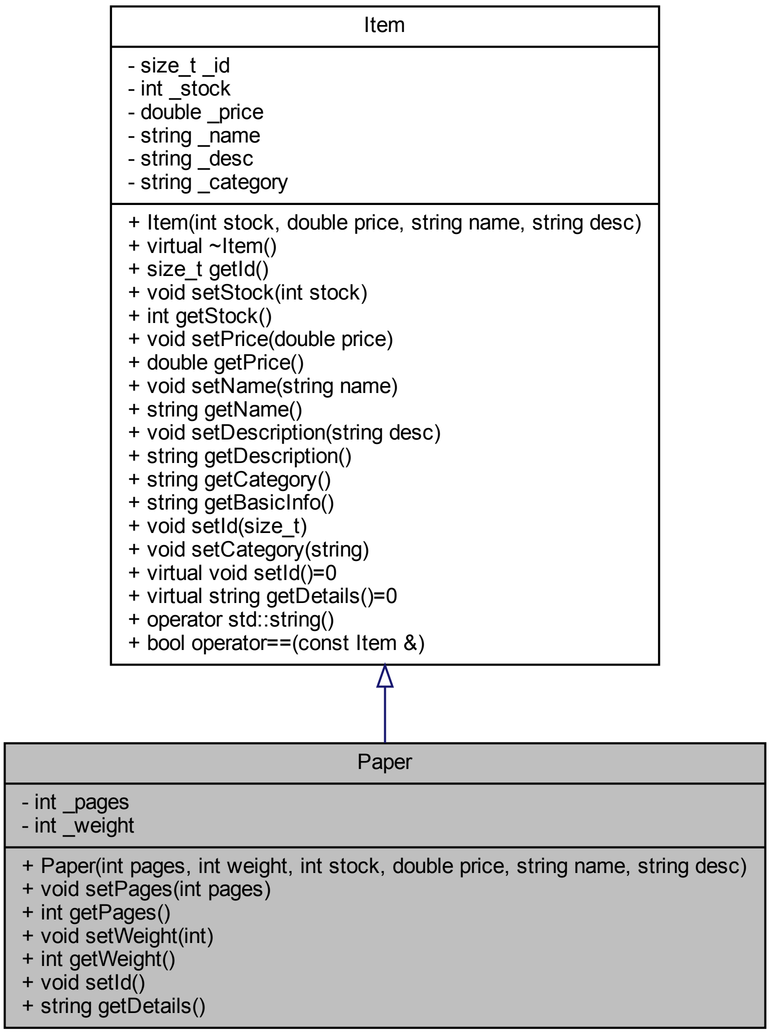
- src/owner.h
- src/owner.cpp

5.8 Paper Class Reference

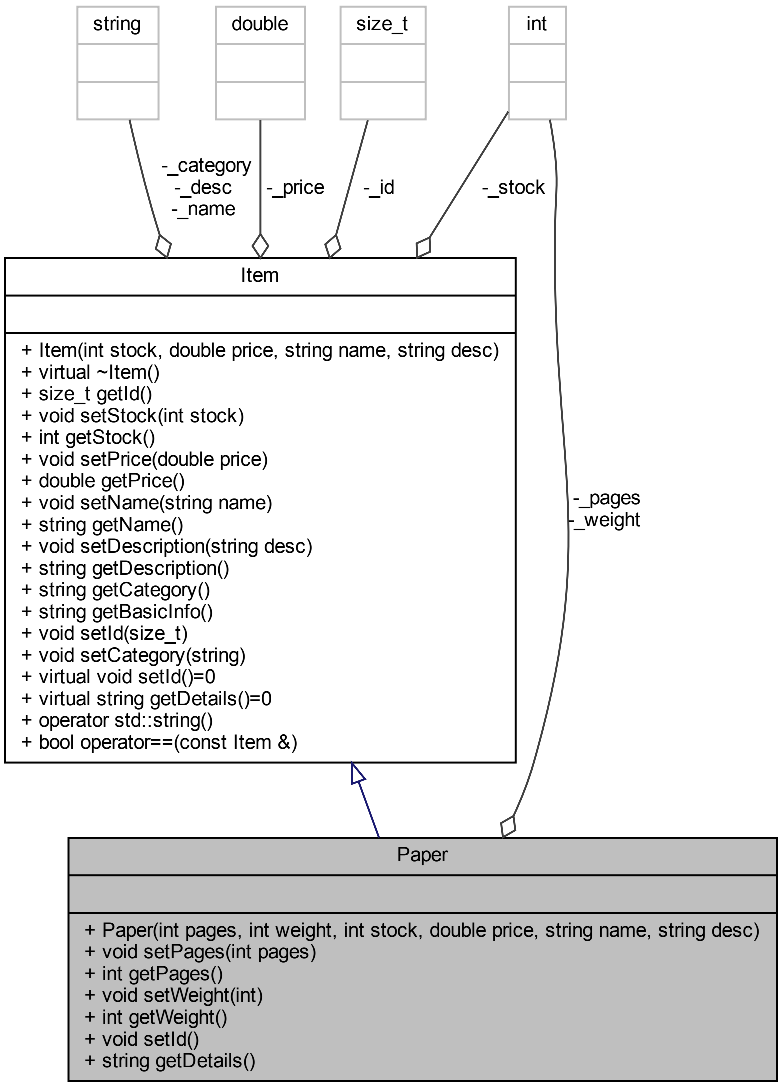
Class representing a Paper.

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

Inheritance diagram for Paper:



Collaboration diagram for Paper:


string
double
size_t
int
-_category
-_desc
-_name
-_price
-_id
-_stock
-_pages
-_weight
Item
+ Item(int stock, double price, string name, string desc)
+ virtual ~Item()
+ size_t getId()
+ void setStock(int stock)
+ int getStock()
+ void setPrice(double price)
+ double getPrice()
+ void setName(string name)
+ string getName()
+ void setDescription(string desc)
+ string getDescription()
+ string getCategory()
+ string getBasicInfo()
+ void setId(size_t)
+ void setCategory(string)
+ virtual void setId()=0
+ virtual string getDetails()=0
+ operator std::string()
+ bool operator==(const Item &)
Paper
+ Paper(int pages, int weight, int stock, double price, string name, string desc)
+ void setPages(int pages)
+ int getPages()
+ void setWeight(int)
+ int getWeight()
+ void setId()
+ string getDetails()

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.

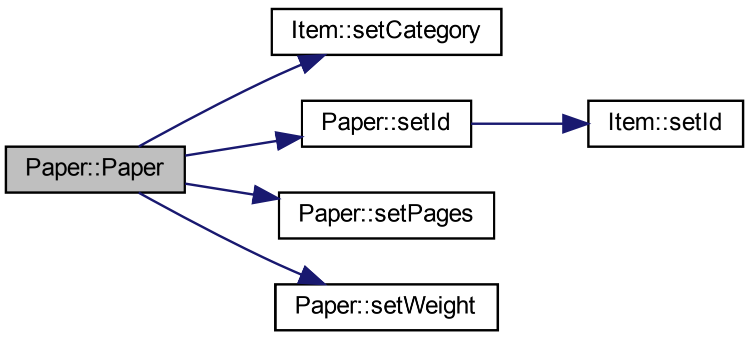
Parameters

int	<pages>
int	<weight>
int	<stock>
double	<price>
string	<name>
string	<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 commaspace-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.

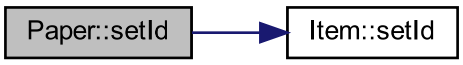
Implements Item.

Definition at line 23 of file paper.cpp.

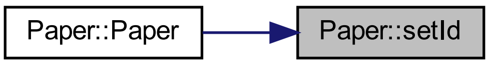
References _pages, _weight, and Item::setId().

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

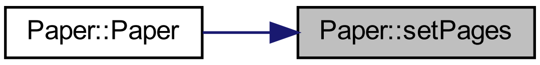
pages	<int>

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

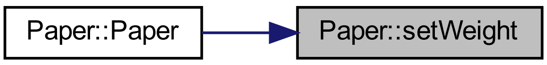
subjects	<int>

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:

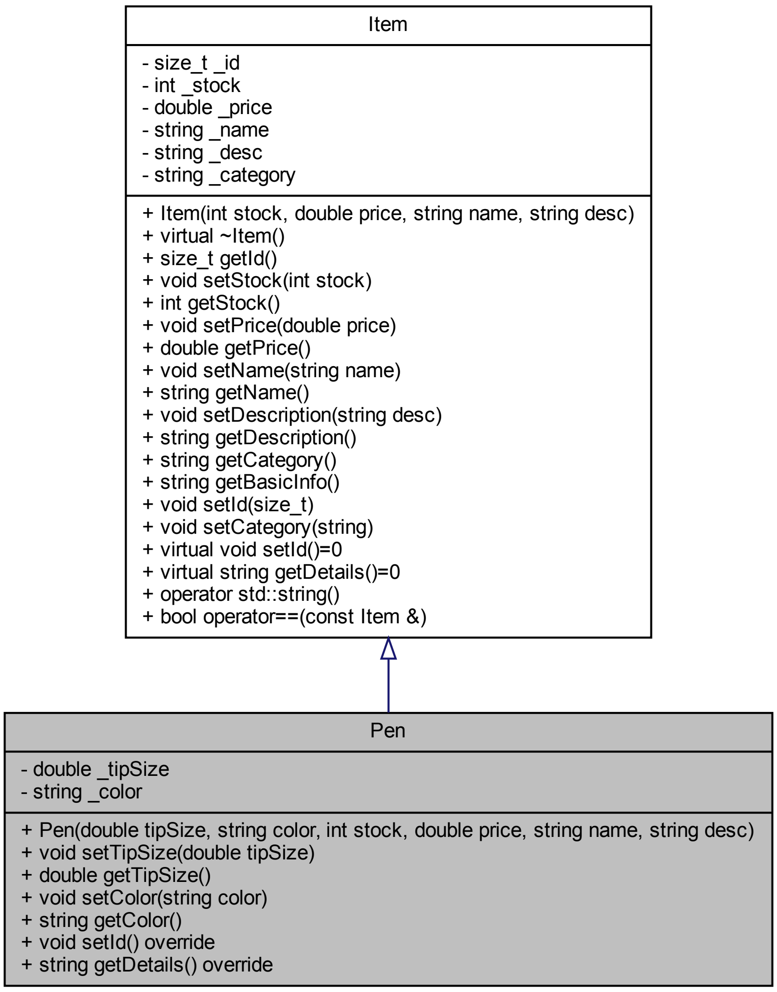
- src/paper.h
- src/paper.cpp

5.9 Pen Class Reference

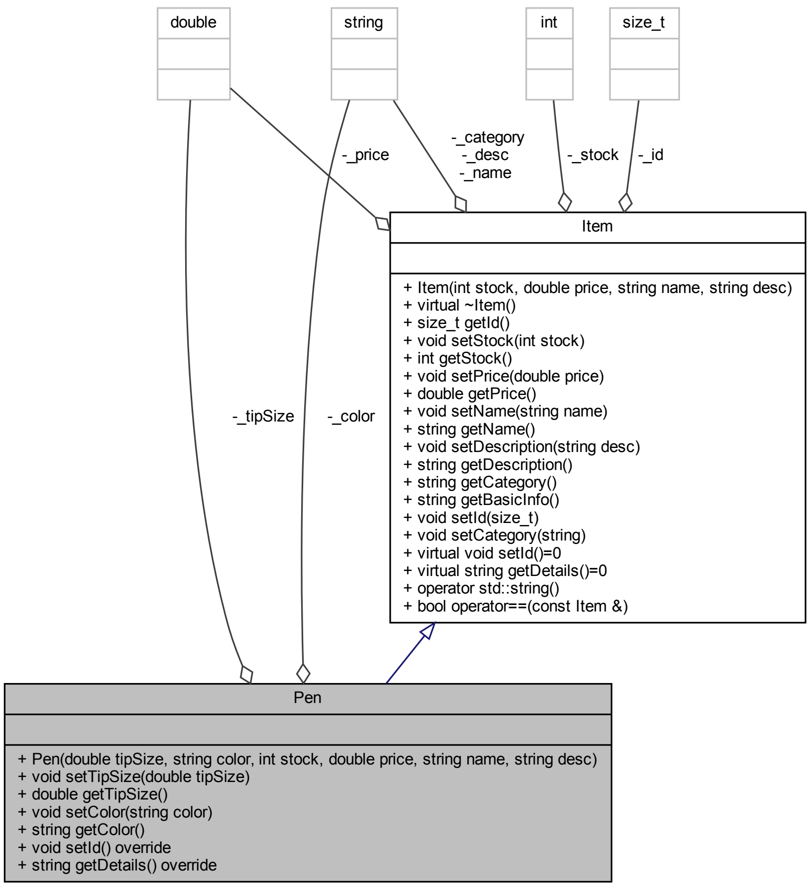
Class representing a Pen.

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

Inheritance diagram for Pen:


Item
- size_t _id
- int _stock
- double _price
- string _name
- string _desc
- string _category
+ Item(int stock, double price, string name, string desc)
+ virtual ~Item()
+ size_t getId()
+ void setStock(int stock)
+ int getStock()
+ void setPrice(double price)
+ double getPrice()
+ void setName(string name)
+ string getName()
+ void setDescription(string desc)
+ string getDescription()
+ string getCategory()
+ string getBasicInfo()
+ void setId(size_t)
+ void setCategory(string)
+ virtual void setId()=0
+ virtual string getDetails()=0
+ operator std::string()
+ bool operator==(const Item &)
Pen
- double _tipSize
- string _color
+ Pen(double tipSize, string color, int stock, double price, string name, string desc)
+ void setTipSize(double tipSize)
+ double getTipSize()
+ void setColor(string color)
+ string getColor()
+ void setId() override
+ string getDetails() override

Collaboration diagram for Pen:


double
string
int
size_t
-_price
-_category
-_desc
-_name
-_stock
-_id
-_tipSize
-_color
Item
+ Item(int stock, double price, string name, string desc)
+ virtual ~Item()
+ size_t getId()
+ void setStock(int stock)
+ int getStock()
+ void setPrice(double price)
+ double getPrice()
+ void setName(string name)
+ string getName()
+ void setDescription(string desc)
+ string getDescription()
+ string getCategory()
+ string getBasicInfo()
+ void setId(size_t)
+ void setCategory(string)
+ virtual void setId()=0
+ virtual string getDetails()=0
+ operator std::string()
+ bool operator==(const Item &)
Pen
+ Pen(double tipSize, string color, int stock, double price, string name, string desc)
+ void setTipSize(double tipSize)
+ double getTipSize()
+ void setColor(string color)
+ string getColor()
+ void setId() override
+ string getDetails() override

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.

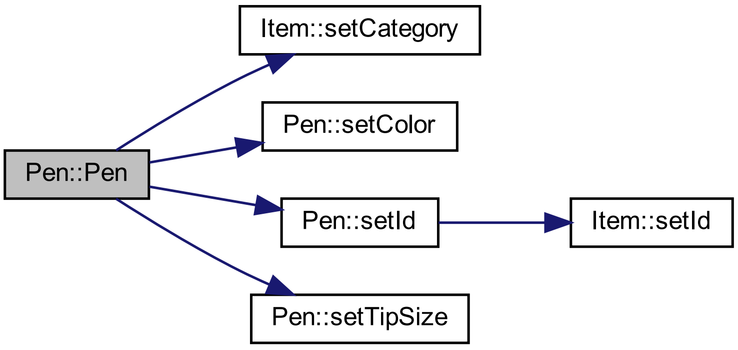
Parameters

tipSize	<double>
color	<string>
stock	<int>
price	<double>
name	<string>
desc	<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 commaspace-separated 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

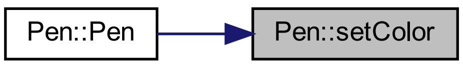
color	<string>

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.

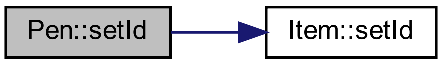
Implements Item.

Definition at line 13 of file pen.cpp.

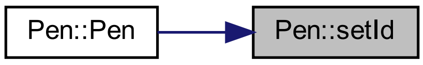
References _color, _tipSize, and Item::setId().

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

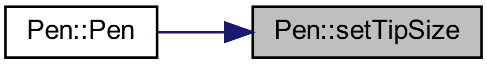
tipSize	<double>

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:

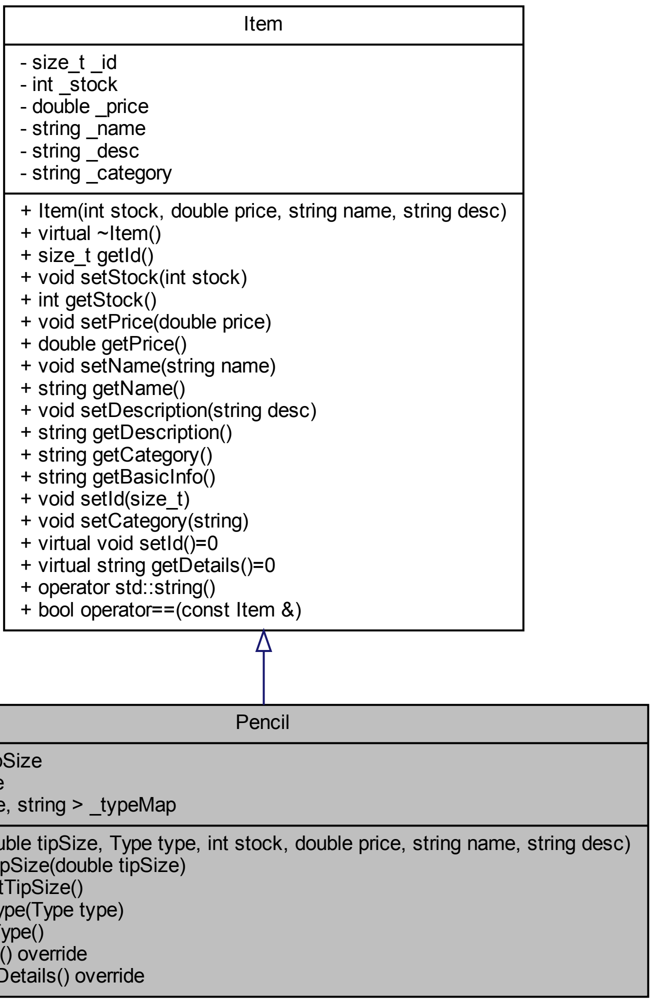
- src/pen.h
- src/pen.cpp

5.10 Pencil Class Reference

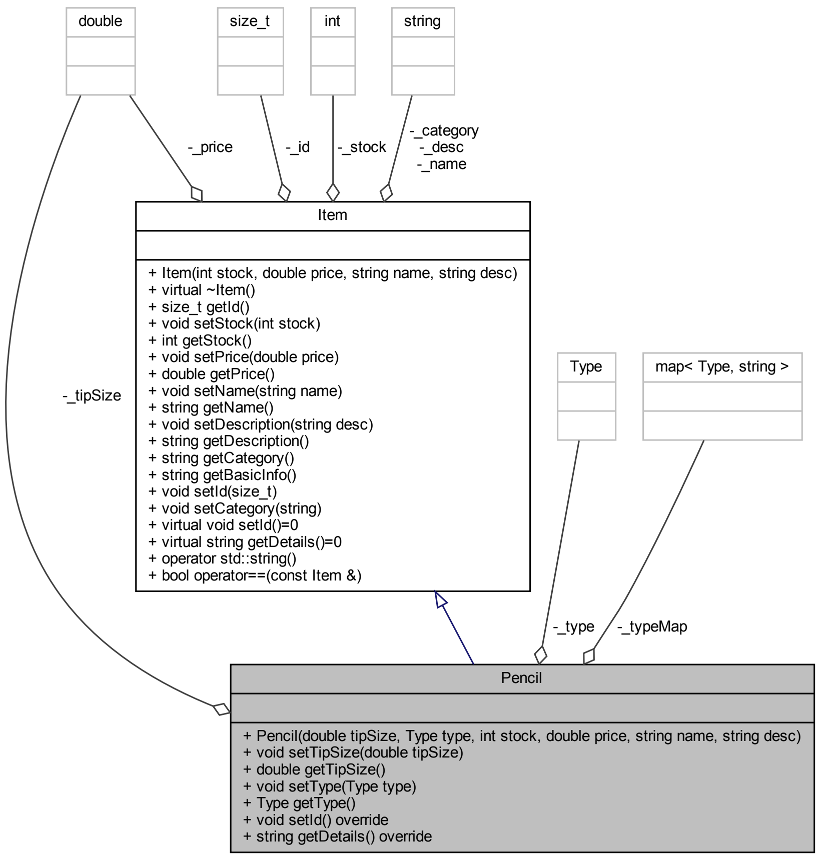
Class representing a Pencil.

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

Inheritance diagram for Pencil:


Item
- size_t _id
- int _stock
- double _price
- string _name
- string _desc
- string _category
+ Item(int stock, double price, string name, string desc)
+ virtual ~Item()
+ size_t getId()
+ void setStock(int stock)
+ int getStock()
+ void setPrice(double price)
+ double getPrice()
+ void setName(string name)
+ string getName()
+ void setDescription(string desc)
+ string getDescription()
+ string getCategory()
+ string getBasicInfo()
+ void setId(size_t)
+ void setCategory(string)
+ virtual void setId()=0
+ virtual string getDetails()=0
+ operator std::string()
+ bool operator==(const Item &)
Pencil
- double _tipSize
- Type _type
- map< Type, string > _typeMap
+ Pencil(double tipSize, Type type, int stock, double price, string name, string desc)
+ void setTipSize(double tipSize)
+ double getTipSize()
+ void setType(Type type)
+ Type getType()
+ void setId() override
+ string getDetails() override

Collaboration diagram for Pencil:


double
size_t
int
string
-_price
-_id
-_stock
-_category
-_desc
-_name
Item
+ Item(int stock, double price, string name, string desc)
+ virtual ~Item()
+ size_t getId()
+ void setStock(int stock)
+ int getStock()
+ void setPrice(double price)
+ double getPrice()
+ void setName(string name)
+ string getName()
+ void setDescription(string desc)
+ string getDescription()
+ string getCategory()
+ string getBasicInfo()
+ void setId(size_t)
+ void setCategory(string)
+ virtual void setId()=0
+ virtual string getDetails()=0
+ operator std::string()
+ bool operator==(const Item &)
-_tipSize
Type
map< Type, string >
-_type
-_typeMap
Pencil
+ Pencil(double tipSize, Type type, int stock, double price, string name, string desc)
+ void setTipSize(double tipSize)
+ double getTipSize()
+ void setType(Type type)
+ Type getType()
+ void setId() override
+ string getDetails() override

Public Types

- enum Type { H , B , HB }

  Enumaration of Pencil types.

Public Member Functions

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

  Constructor for Pencil.
- void setTipSize (double tipSize)

  Set the tip size of the Pencil.
- double getTipSize ()

  Get the tip size of the Pencil.
- void setType (Type type)

  Set the type of the Pencil.
- Type getType ()

  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
- 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
```

Enumaration 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.

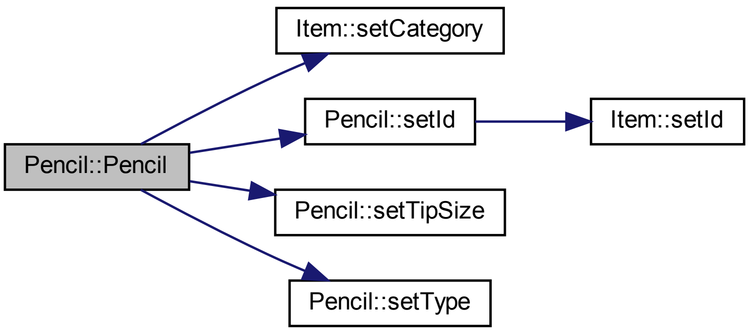
Parameters

tipSize	<double>
type	<Pencil::Type>
stock	<int>
price	<double>
name	<string>
desc	<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 commaspace-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.

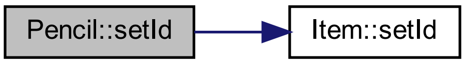
Implements Item.

Definition at line 13 of file pencil.cpp.

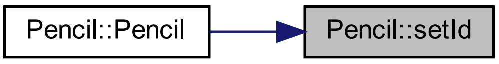
References _tipSize, _type, _typeMap, and Item::setId().

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

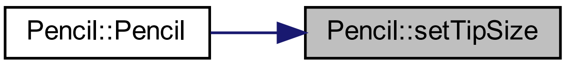
tipSize	<double>

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

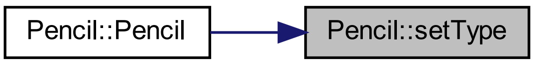
color	<Pencil::Type>

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:

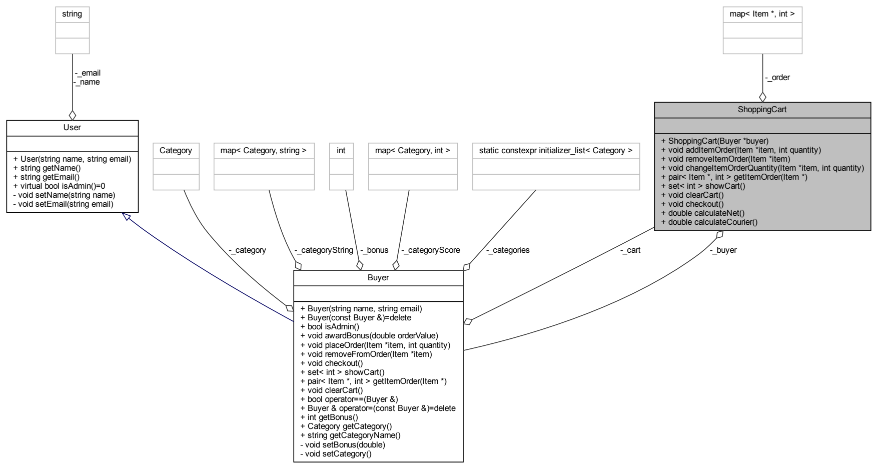
- src/pencil.h
- src/pencil.cpp

5.11 ShoppingCart Class Reference

Class implementing the shopping cart.

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

Collaboration diagram for ShoppingCart:


string
-_email
-_name
User
+ User(string name, string email)
+ string getName()
+ string getEmail()
+ virtual bool isAdmin()=0
- void setName(string name)
- void setEmail(string email)
Category
map< Category, string >
int
map< Category, int >
static constexpr initializer_list< Category >
map< Item *, int >
-_order
ShoppingCart
+ ShoppingCart(Buyer *buyer)
+ void addItemOrder(Item *item, int quantity)
+ void removeItemOrder(Item *item)
+ void changeItemOrderQuantity(Item *item, int quantity)
+ pair< Item *, int > getItemOrder(Item *)
+ set< int > showCart()
+ void clearCart()
+ void checkout()
+ double calculateNet()
+ double calculateCourier()
-_category
-_categoryString
-_bonus
-_categoryScore
-_categories
-_cart
-_buyer
Buyer
+ Buyer(string name, string email)
+ Buyer(const Buyer &)=delete
+ bool isAdmin()
+ void awardBonus(double orderValue)
+ void placeOrder(Item *item, int quantity)
+ void removeFromOrder(Item *item)
+ void checkout()
+ set< int > showCart()
+ pair< Item *, int > getItemOrder(Item *)
+ void clearCart()
+ bool operator==(Buyer &)
+ Buyer & operator=(const Buyer &)=delete
+ int getBonus()
+ Category getCategory()
+ string getCategoryName()
- void setBonus(double)
- void setCategory()

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

buyer	<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

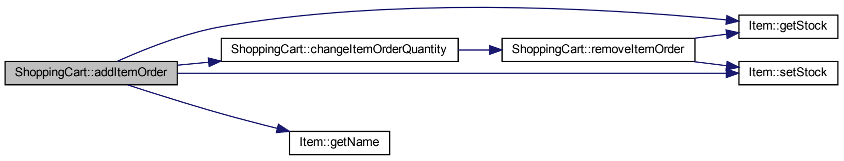
item	<Item∗> Reference to an item in the eshop
quantity	<int> The quantity of the item

Definition at line 11 of file shoppingcart.cpp.

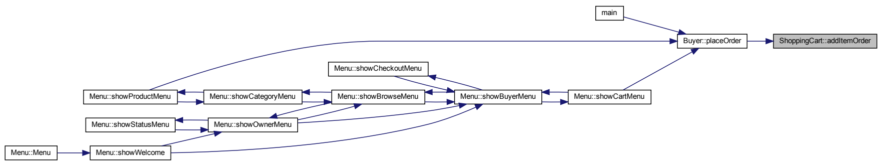
References _order, changeItemOrderQuantity(), Item::getName(), Item::getStock(), and Item::setStock().

Referenced by Buyer::placeOrder().

Here is the call graph for this function:


ShoppingCart::addItemOrder
ShoppingCart::changeItemOrderQuantity
ShoppingCart::removeItemOrder
Item::getStock
Item::setStock
Item::getName

Here is the caller graph for this function:



5.11.3.2 calculateCourier()

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

Calculates the cost of the courier.

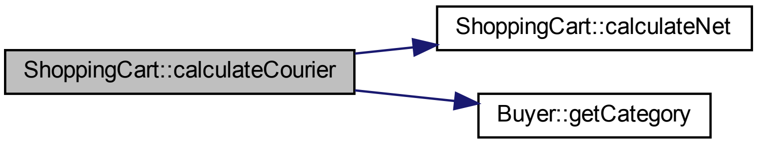
It also takes into account the category of the buyer.

Definition at line 61 of file shoppingcart.cpp.

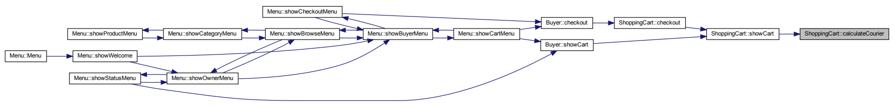
References _buyer, calculateNet(), Buyer::getCategory(), Buyer::Gold, and Buyer::Silver.

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 ( )
```

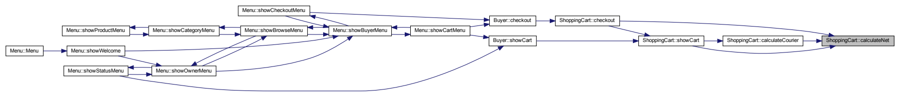
Calculates the cost of the order.

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 of the resulting quantity is below zero and if it is it removes the Item from the cart.

Parameters

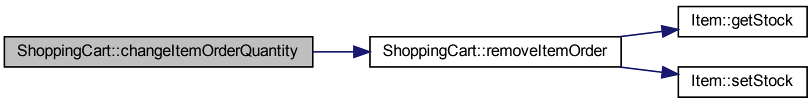
item	<Item∗> Reference to the Item to be removed.
quantity	<int> Quantity to remove.

Definition at line 35 of file shoppingcart.cpp.

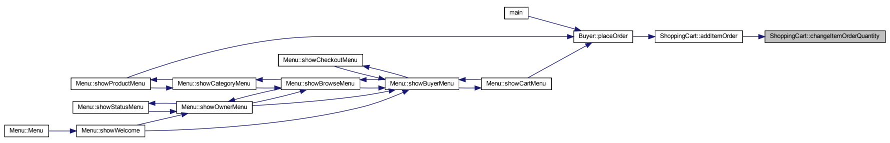
References _order, and removeItemOrder().

Referenced by addItemOrder().

Here is the call graph for this function:


ShoppingCart::changeItemOrderQuantity
ShoppingCart::removeItemOrder
Item::getStock
Item::setStock

Here is the caller graph for this function:



5.11.3.5 checkout()

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

Performs the checkout.

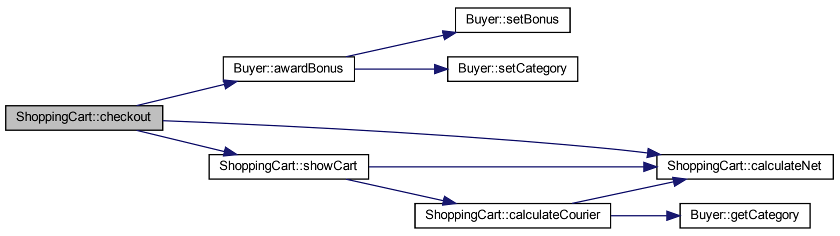
Performs the checkout, asks the user for confirmation. If it is positive it clears the cart and awards the bonus to the user.

Definition at line 94 of file shoppingcart.cpp.

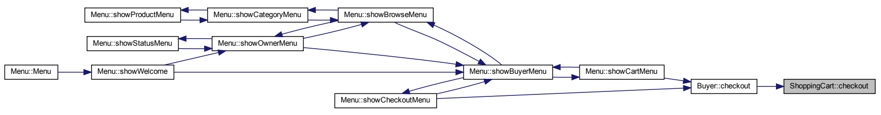
References _buyer, _order, Buyer::awardBonus(), calculateNet(), and showCart().

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.

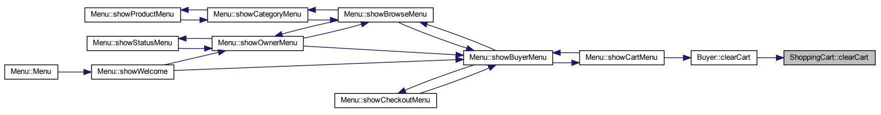
It doesn't call removeItemOrder because of the use of an iterator.

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

item	<Item∗> Reference to the Item to be returned.

Returns

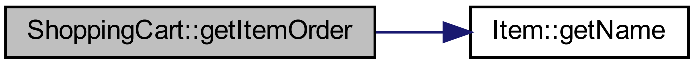
<pair<Item∗,int>> A pair from the cart.

Definition at line 42 of file shoppingcart.cpp.

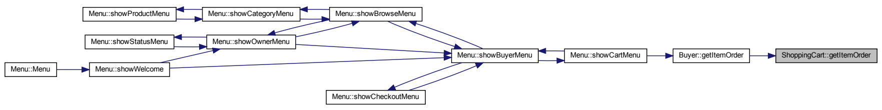
References _order, and Item::getName().

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

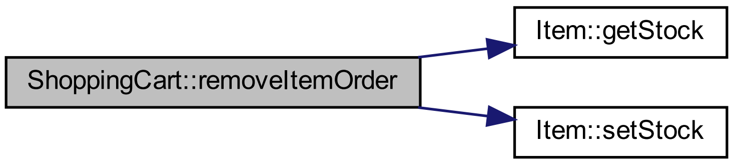
item	<Item∗> Reference to the Item to be removed.

Definition at line 26 of file shoppingcart.cpp.

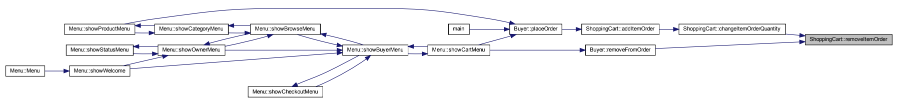
References _order, Item::getStock(), and Item::setStock().

Referenced by changeItemOrderQuantity(), and Buyer::removeFromOrder().

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.9 showCart()

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

Show the contents of the cart.

This function prints the contents of the cart and returns a set of Item IDs of the items in the cart for erroneous input in the menu. It also prints the value of the order and the cost of the courier.

Returns

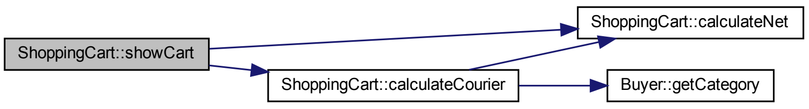
<set<int>> A set of Item IDs in the cart.

Definition at line 72 of file shoppingcart.cpp.

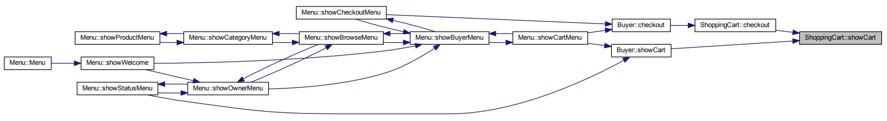
References _order, calculateCourier(), and calculateNet().

Referenced by checkout(), and Buyer::showCart().

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:

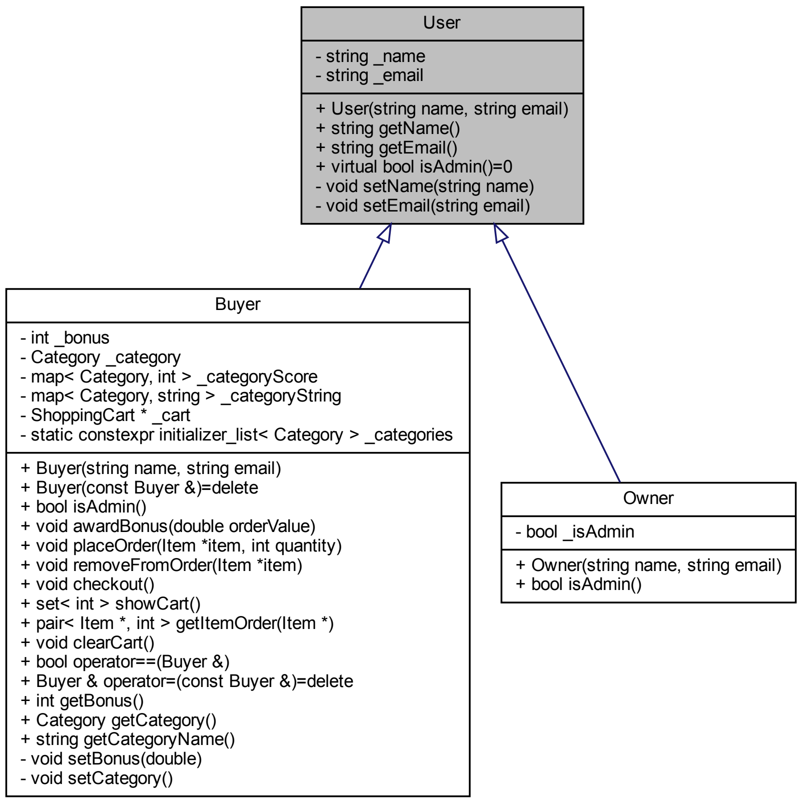
- src/shoppingcart.h
- src/shoppingcart.cpp

5.12 User Class Reference

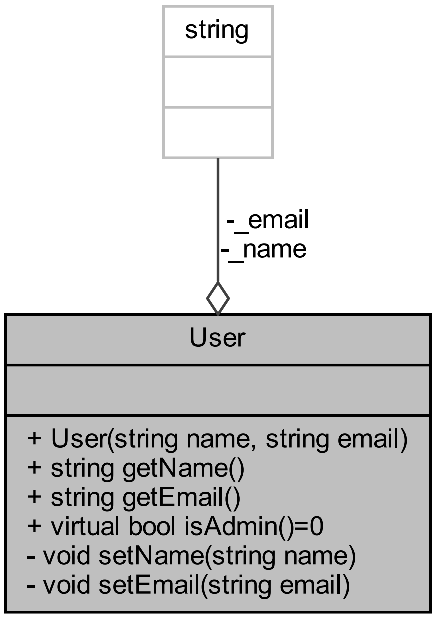
Base class for all users.

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

Inheritance diagram for User:


User
- string _name
- string _email
+ User(string name, string email)
+ string getName()
+ string getEmail()
+ virtual bool isAdmin()=0
- void setName(string name)
- void setEmail(string email)
Buyer
- int _bonus
- Category _category
- map< Category, int > _categoryScore
- map< Category, string > _categoryString
- ShoppingCart * _cart
- static constexpr initializer_list< Category > _categories
+ Buyer(string name, string email)
+ Buyer(const Buyer &)=delete
+ bool isAdmin()
+ void awardBonus(double orderValue)
+ void placeOrder(Item *item, int quantity)
+ void removeFromOrder(Item *item)
+ void checkout()
+ set< int > showCart()
+ pair< Item *, int > getItemOrder(Item *)
+ void clearCart()
+ bool operator==(Buyer &)
+ Buyer & operator=(const Buyer &)=delete
+ int getBonus()
+ Category getCategory()
+ string getCategoryName()
- void setBonus(double)
- void setCategory()
Owner
- bool _isAdmin
+ Owner(string name, string email)
+ bool isAdmin()

Collaboration diagram for User:


string
-_email
-_name
User
+ User(string name, string email)
+ string getName()
+ string getEmail()
+ virtual bool isAdmin()=0
- void setName(string name)
- void setEmail(string email)

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.

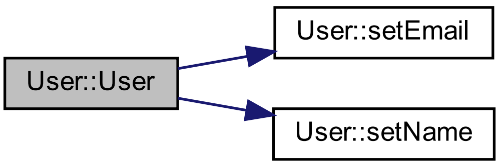
Parameters

name	string. The name of the user
email	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

none	

Returns

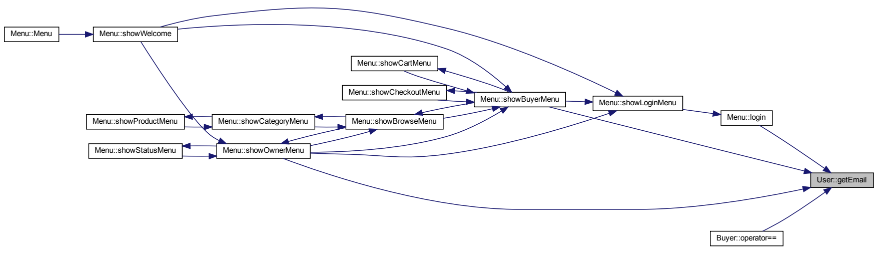
string The user's email

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

none	

Returns

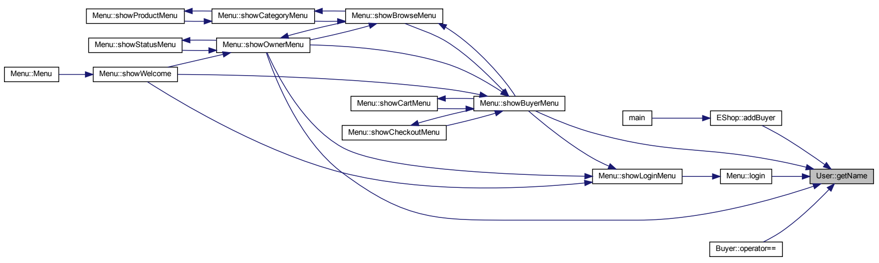
string The user's name

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

none	

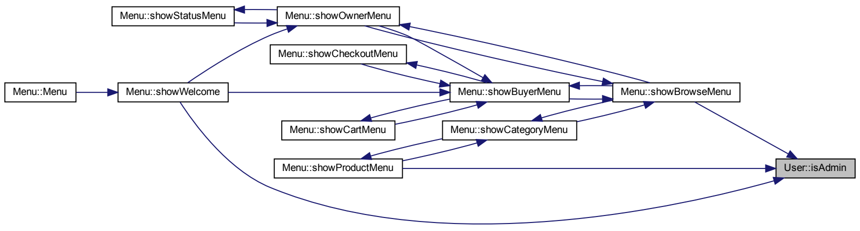
Returns

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()

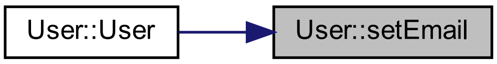
```
void User::setEmail (
             string email )  [private]
```

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()

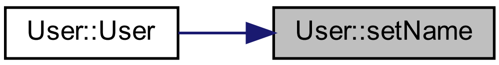
```
void User::setName (
             string name )  [private]
```

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

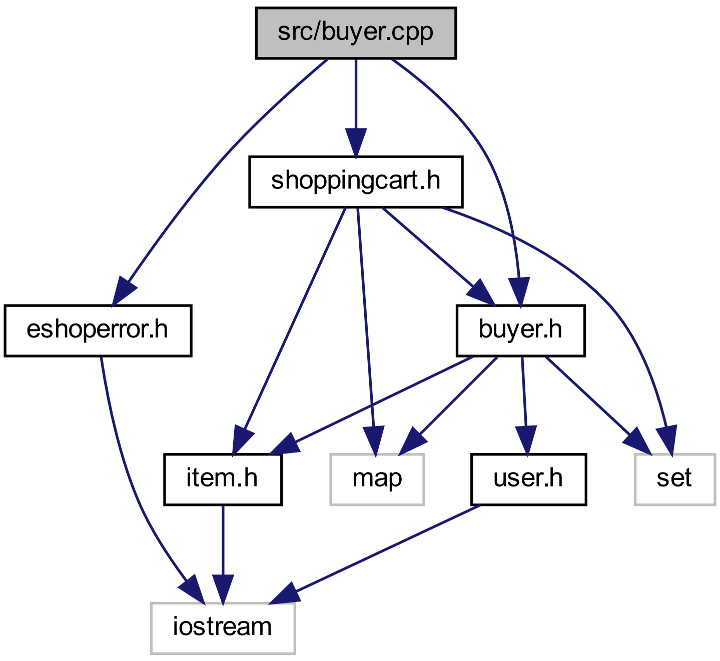
Chapter 6

File Documentation

6.1 src/buyer.cpp File Reference

```
#include "buyer.h"
#include "shoppingcart.h"
#include "eshoperror.h"
```

Include dependency graph for buyer.cpp:



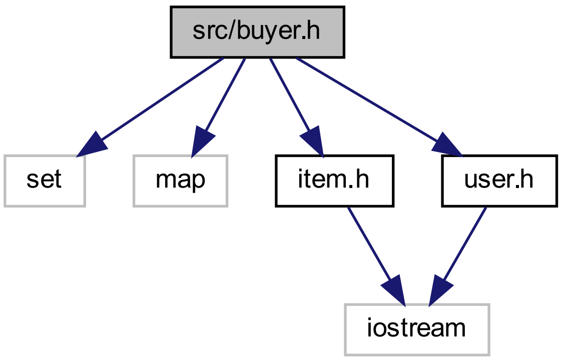
6.2 buyer.cpp

```
#include "buyer.h"
#include "shoppingcart.h"
#include "eshoperror.h"

Buyer::Buyer(string name, string email) : User(name, email)
{
    _bonus = 0;
    _category = Bronze;
    _cart = new ShoppingCart(this);
}

void
Buyer::awardBonus(double orderValue)
{
    setBonus(orderValue);
    setCategory();
}

void
Buyer::placeOrder(Item* item, int quantity)
{
    try {
        _cart->addItemOrder(item, quantity);
    } catch (const EShopError& e){
        cout « e.error() « endl;
        cout « "Available quantity is " + to_string(item->getStock()) « endl;
        string ans;
        do {
            cout « "Do you want to add the available quantity to the cart? [y/n]: ";
            cin » ans;
        }
        while( !cin.fail() && ans!="y" && ans!="n" );

        if (ans == "y")
            _cart->addItemOrder(item, item->getStock());
    }
}

void
Buyer::removeFromOrder(Item* item)
{
    _cart->removeItemOrder(item);
}

void
Buyer::checkout()
{
    _cart->checkout();
}

set<int>
Buyer::showCart()
{
    try {
        return _cart->showCart();
    } catch (const EShopError& e) {
        throw e;
    }
}

pair<Item*, int>
Buyer::getItemOrder(Item* item)
{
    try {
        return _cart->getItemOrder(item);
    } catch (const EShopError& e) {
        throw e;
    }
}

void
Buyer::clearCart()
{
    _cart->clearCart();
}


bool
Buyer::operator==(Buyer& other)
{
    return (this->getName() == other.getName()) && (this->getEmail() == other.getEmail());
}

void
Buyer::setBonus(double value) {
    _bonus = static_cast<int>(value*0.1);
}

int
Buyer::getBonus() { return _bonus; }

void
Buyer::setCategory() {
    for(auto c: _categories)
        if (_bonus > _categoryScore[c]) _category = c;
}

Buyer::Category
Buyer::getCategory() { return _category; }

string
Buyer::getCategoryName() { return _categoryString[_category]; }

bool
Buyer::isAdmin() { return false; }
```

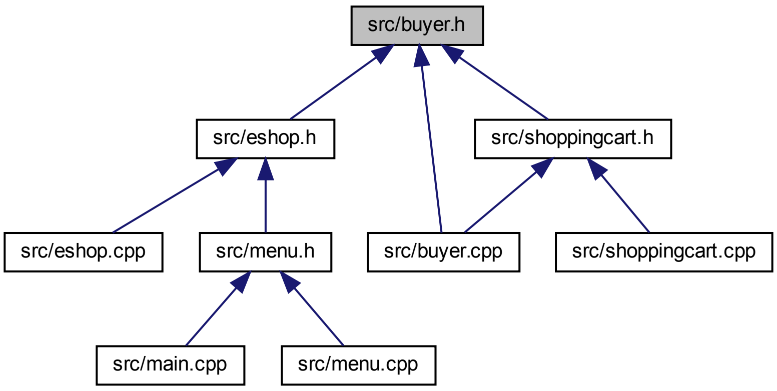
6.3 src/buyer.h File Reference

```
#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.

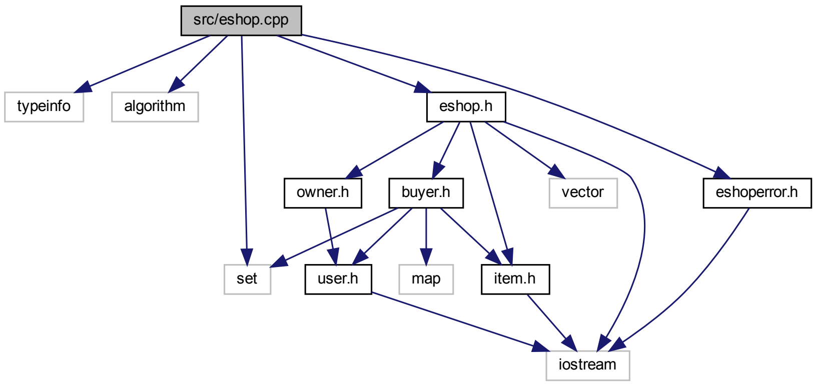
6.4 buyer.h

```
#ifndef BUYER_H
#define BUYER_H

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

// Forward declaration of ShoppingCart
class ShoppingCart;

using namespace std;

class Buyer final : public  User
{
public:
    Buyer(string name, string email);

    Buyer(const Buyer&) = delete;

    enum Category {
        Bronze,
        Silver,
        Gold,
    };

    bool isAdmin();
    void awardBonus(double orderValue);
    void placeOrder(Item* item, int quantity);
    void removeFromOrder(Item* item);
    void checkout();
    set<int> showCart();
    pair<Item*, int> getItemOrder(Item*);
    void clearCart();

    bool operator==(Buyer&);
    Buyer& operator=(const Buyer&) = delete;

    int      getBonus();
    Category getCategory();
    string   getCategoryName();
private:
    void setBonus(double);
    void setCategory();

    int _bonus;
    Category _category;
    map<Category, int> _categoryScore { {Bronze, 0}, {Silver, 100}, {Gold, 200} };
    map<Category, string> _categoryString { {Bronze, "Bronze"}, {Silver, "Silver"}, {Gold, "Gold"} };
    static constexpr initializer_list<Category> _categories = {Bronze, Silver, Gold};
    ShoppingCart* _cart;
};

#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

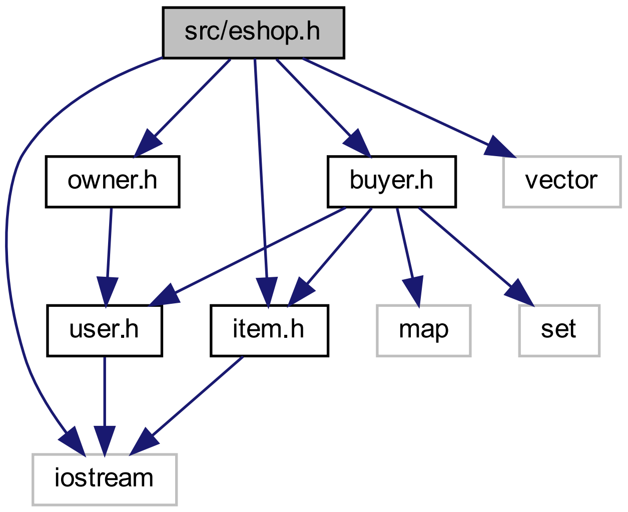
```
#include <typeinfo>
#include <algorithm>
#include <set>
#include "eshop.h"
#include "eshoperror.h"

EShop::EShop(string name)
{
    _name = name;
    _owner = nullptr;
}

EShop::EShop(string name, string owner, string email) : EShop(name)
{
    setOwner(owner, email);
}

EShop::~EShop()
{
    for(auto i: _items) delete i;
    for(auto b: _buyers) delete b;
    delete _owner;
```

```
}

Owner*
EShop::getOwner()
{
    return _owner;
}


void
EShop::setOwner(string name, string email)
{
    if (_owner != nullptr)
        throw EShopError("Shop already has an owner");
    _owner = new Owner(name, email);
}

void
EShop::addItem(Item* item)
{
    //TODO: exception
    if (!_items.empty()) {
        for(auto i: _items) {
            if (*i == *item) {
                throw EShopError("Item " + item->getName() + " already exists.");
            }
        }
    }
    _items.push_back(item);
}

void
EShop::removeItem(Item* item)
{
    for(auto i = _items.begin(); i <= _items.end(); ++i) {
        if (**i == *item) {
            delete *i;
            _items.erase(i);
            break;
        }
    }
}

Item*
EShop::getItemById(size_t id)
{
    for(auto i: _items) if (i->getId() == id) return i;
    throw EShopError("Item with ID " + to_string(id) + " does not exist.");
    return nullptr;
}

void
EShop::addBuyer(Buyer* buyer)
{
    //TODO: exception
    for(auto b: _buyers) {
        if (*b == *buyer) {
            throw EShopError("Buyer " + buyer->getName() + " already exists.");
        }
    }
    _buyers.push_back(buyer);
}

void
EShop::removeBuyer(Buyer* buyer)
{
    for(auto b = _buyers.begin(); b <= _buyers.end(); ++b ){
        if (**b == *buyer) {
            (*b)->clearCart();
            delete *b;
            _buyers.erase(b);
            break;
        }
    }
}

Buyer*
EShop::getBuyerByEmail(string email)
{
    for(auto b: _buyers) if (b->getEmail() == email) return b;
    throw EShopError("Buyer with email \'" + email + "\' does not exist.");
    return nullptr;
}

void
EShop::updateItemStock(Item* item, int delta)
{
    item->setStock(item->getStock() + delta);
}

void
EShop::showProduct(Item* item)
{
    cout « *item;
}

vector<pair<string, int»
EShop::getCategories()
{
    map<string, int> categories;
    for (auto i: _items) {
        string category = i->getCategory();
        if (categories.find(category) != categories.end())
            categories[category]++;
        else categories.emplace(category, 1);
    }
    vector<pair<string, int» vectored;
    for (auto c: categories)
        vectored.push_back(make_pair(c.first, c.second));
    return vectored;
}

vector<pair<int, string»
EShop::getProductsInCategory(string category)
{
    vector<pair<int, string» products;
    for(auto i: _items)
        if (i->getCategory() == category)
            products.push_back(make_pair(i->getId(), i->getName()));
    return products;
}

vector<vector<string»
EShop::checkStatus()
{
    vector<vector<string» buyers;
    for (auto b: _buyers) {
        vector<string> info;
        info.push_back(b->getEmail());
        info.push_back(b->getName());
        info.push_back(b->getCategoryName());
        info.push_back(to_string(b->getBonus()));
        buyers.push_back(info);
    }
    return buyers;
}

string
EShop::getName()
{
    return _name;
}
```

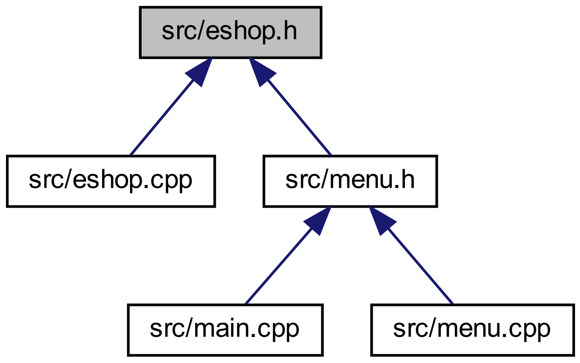
6.7 src/eshop.h File Reference

```
#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.

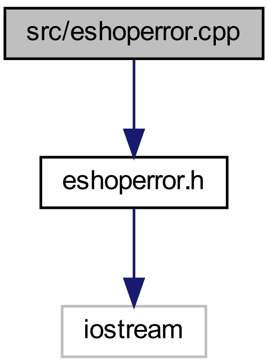
6.8 eshop.h

```
#ifndef ESHOP_H
#define ESHOP_H

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

using namespace std;

class EShop
{
public:
    explicit EShop(string name);
    EShop(string name, string owner, string email);
    ~EShop();

    string getName();
    Owner* getOwner();

    void addItem(Item*);
    void removeItem(Item*);
    Item* getItemById(size_t id);

    void addBuyer(Buyer*);
    void removeBuyer(Buyer*);
    Buyer* getBuyerByEmail(string email);

    void updateItemStock(Item*, int);
    vector<pair<string, int» getCategories();
    vector<pair<int, string» getProductsInCategory(string);
    void showProduct(Item*);
    vector<vector<string» checkStatus();

private:
    void setOwner(string name, string email);

    string _name;
    Owner* _owner = nullptr;
    vector<Buyer*> _buyers;
    vector<Item*>  _items;
};

#endif // ESHOP_H
```

6.9 src/eshoperror.cpp File Reference

`#include "eshoperror.h"`

Include dependency graph for eshoperror.cpp:



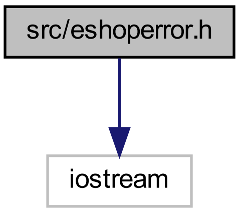
6.10 eshoperror.cpp

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

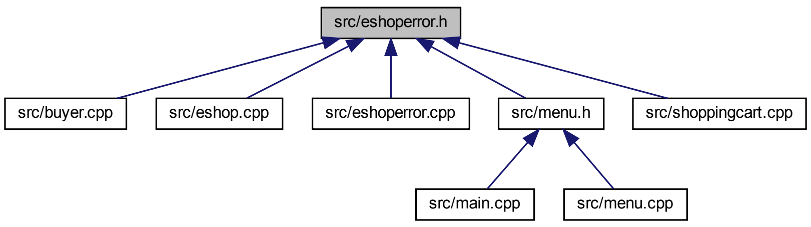
6.11 src/eshoperror.h File Reference

```
#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

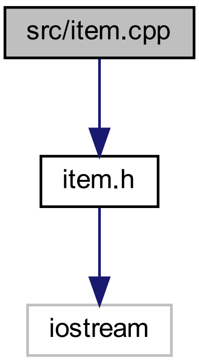
```
#ifndef ESHOPERROR_H
#define ESHOPERROR_H

#include <iostream>

using namespace std;

class EShopError
{
```

```
public:

    EShopError() { }
    EShopError(string&& error) : _error(move(error)) { }

    const string& error() const {
        return _error;
    }

private:
    string _error;
};

#endif // ESHOPERROR_H

```

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

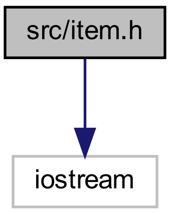
Definition at line 36 of file item.cpp.

6.14 item.cpp

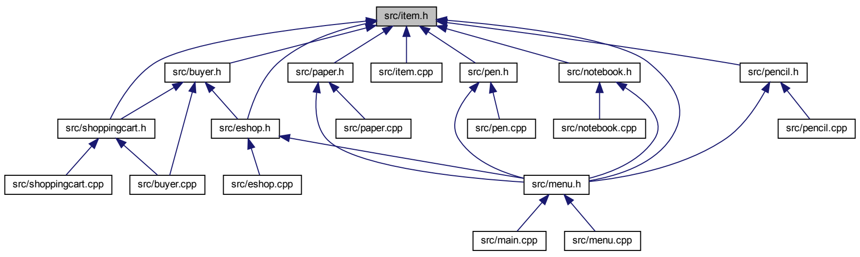
```
#include "item.h"

Item::Item(int stock, double price, string name, string desc)
{
    setStock(stock);
    setPrice(price);
    setName(name);
    setDescription(desc);
}

Item::~Item()
{
}

string Item::getBasicInfo()
{
    string ret;
    ret += to_string(_id);
    ret += ", ";
    ret += _name;
    ret += ", ";
    ret += to_string(_price);
    ret += ", ";
    ret += to_string(_stock);
    ret += ", ";
    ret += _desc;
    ret += ", ";
    return ret;
}

Item::operator std::string ()
{
    return getBasicInfo() + getDetails();
}

ostream&
operator«(ostream& os, Item& item)
{
    os « static_cast<std::string>(item);
    return os;
}

bool
Item::operator==(const Item& other)
{
    return this->_id == other._id;
}

void
Item::setCategory(string category) { _category = category.substr(1); }
string
Item::getCategory() { return _category; }

void
Item::setId(size_t id) { _id = id % 10000; }
size_t
Item::getId() { return _id; }

void
Item::setStock(int stock) { _stock = stock; }
int
Item::getStock() { return _stock; }

void
Item::setPrice(double price) { _price = price; }
double
Item::getPrice() { return _price; }

void
Item::setName(string name) { _name = name; }
string
Item::getName() { return _name; }

void
Item::setDescription(string desc) { _desc = desc; }
string
Item::getDescription() { return _desc; }
```

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.

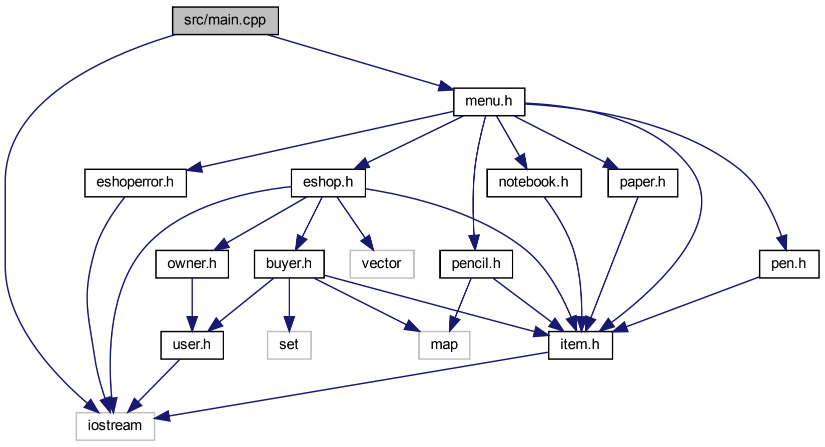
6.16 item.h

```
#ifndef ITEM_H
#define ITEM_H

#include <iostream>

using namespace std;

class Item
{
public:
    Item(int stock, double price, string name, string desc);
    virtual ~Item();

    size_t getId();

    void setStock(int stock);
    int  getStock();

    void   setPrice(double price);
    double getPrice();

    void   setName(string name);
    string getName();

    void   setDescription(string desc);
    string getDescription();

    string getCategory();
    string getBasicInfo();

    void setId(size_t);
    void setCategory(string);

    virtual void setId() = 0;
    virtual string getDetails() = 0;

    operator std::string ();
    friend ostream& operator«(ostream&, Item&);

    bool operator==(const Item&);

private:
    size_t _id = 0;
    int    _stock;
    double _price;
    string _name;
    string _desc;
    string _category;
};

#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

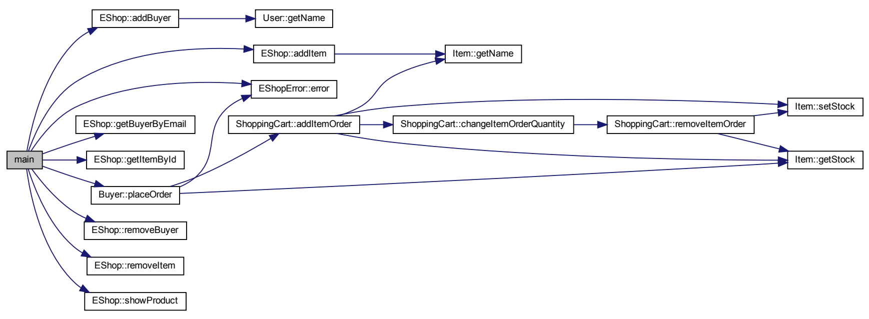
6.17.1.1 main()

```
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

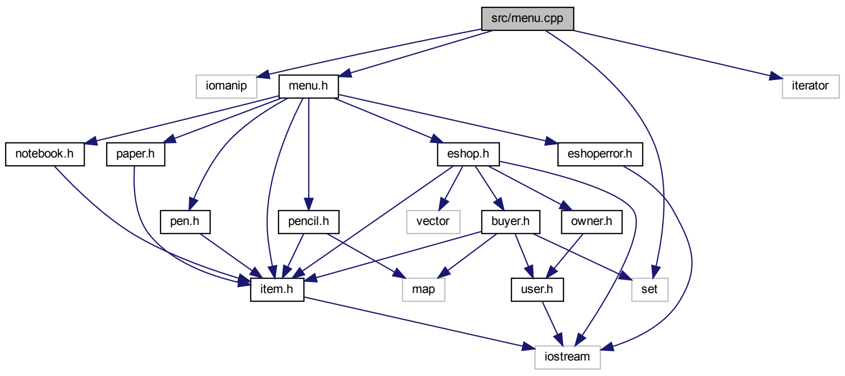
```
00001
00024 #include <iostream>
00025 #include "menu.h"
00026
00027 int main(int argc, char **argv) {
00028     EShop eshop = EShop("EShop", "Owner", "owner@eshop.com");
00029
00030     try {
00031         eshop.addItem(new Pencil(0.2, Pencil::B, 10, 3.1, "Yellow Pencil Name", "Yellow Pencil
      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"));
```

```
        eshop.addItem(new Notebook(3, 15, 11.5, "3 Subject Notebook", "Fancy 3 Subject Notebook"));
        eshop.addItem(new Notebook(4, 15, 12.5, "4 Subject Notebook", "Fancy 4 Subject Notebook"));
        eshop.addItem(new Notebook(4, 15, 12.5, "4 Subject Notebook", "Fancy 4 Subject Notebook"));
    } catch(const EShopError& e) {
        cout « e.error() « endl;
    }

    try {
        eshop.addItem(new Paper(100, 2, 15, 10.5, "100 Pages", "Fancy 100"));
        eshop.addItem(new Paper(200, 3, 15, 11.5, "200 Pages", "Fancy 200"));
        eshop.addItem(new Paper(300, 4, 15, 12.5, "300 Pages", "Fancy 300"));
        eshop.addItem(new Paper(300, 4, 15, 12.5, "300 Pages", "Fancy 300"));
    } catch(const EShopError& e) {
        cout « e.error() « endl;
    }

    try {
        eshop.showProduct(eshop.getItemById(6091));
        eshop.showProduct(eshop.getItemById(6092));
        eshop.removeItem(eshop.getItemById(6091));
        eshop.removeItem(eshop.getItemById(6092));
    } catch (const EShopError& e) {
        cout « e.error() « endl;
    }

    try {
        eshop.removeItem(eshop.getItemById(6091));
        eshop.removeItem(eshop.getItemById(6092));
    } catch (const EShopError& e) {
        cout « e.error() « endl;
    }

    try {
        eshop.addBuyer(new Buyer("buyer_a", "buyer_a@isp.org"));
        eshop.addBuyer(new Buyer("buyer_b", "buyer_b@isp.org"));
        eshop.addBuyer(new Buyer("buyer_c", "buyer_c@isp.org"));
        eshop.addBuyer(new Buyer("buyer_b", "buyer_b@isp.org"));
    } catch(const EShopError& e) {
        cout « e.error() « endl;
    }

    try {
        eshop.removeBuyer(eshop.getBuyerByEmail("buyer_c@isp.org"));
    } catch(const EShopError& e) {
        cout « e.error() « endl;
    }

    Buyer* buyer = eshop.getBuyerByEmail("buyer_a@isp.org");
    try {
        buyer->placeOrder(eshop.getItemById(7093), 3);
        buyer->placeOrder(eshop.getItemById(7093), 3);
        buyer->placeOrder(eshop.getItemById(7093), 3);
        buyer->placeOrder(eshop.getItemById(7093), 13);
    } catch(const EShopError& e) {
        cout « e.error() « endl;
    }

    Menu menu = Menu(&eshop);
    return 0;
}
```

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

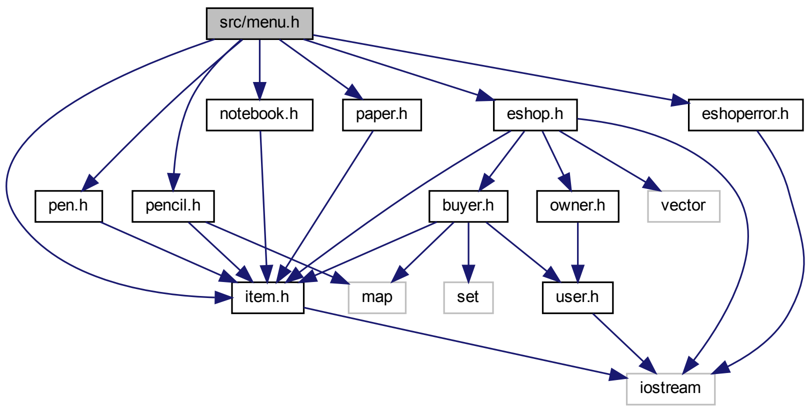
```
#include <iomanip>
#include <set>
#include <iterator>
#include "menu.h"

Menu::Menu(EShop* eshop)
{
    _eshop = eshop;
    showWelcome();
}

void
Menu::showWelcome()
{
    cout « "Welcome to " « quoted(_eshop ->getName()) « endl;
    showLoginMenu();
    if(_user->isAdmin()) {
        showOwnerMenu();
    } else {
        showBuyerMenu();
    }
}

void
Menu::showLoginMenu()
{
    string email;
    cout « "Enter the email address of a user to log in: ";
    cin » email;
    try {
        login(email);
    } catch (const EShopError& e) {
        cout « e.error() « endl;
        exit(-1);
    }
}

void
Menu::showOwnerMenu()
{
    cout « "Hello " « quoted(_owner->getName()) « " " « quoted(_owner->getEmail()) « endl;
    cout « "You are the owner of " « quoted(_eshop->getName()) « endl;
    cout « endl;
    cout « "Please choose an action." « endl;
    cout « "1. Browse Store" « endl;
    cout « "2. Check Status" « endl;
    cout « "3. Back" « endl;
    cout « "4. Logout" « endl;
    cout « "5. Exit" « endl;

    string ans; int chc;
    do {
        cout « "Action: ";
        cin » ans;
        chc = stoi(ans);
    } while( !cin.fail() && chc < 1 && chc > 5);

    switch (stoi(ans)) {
        case 1:
            showBrowseMenu();
            break;
        case 2:
            showStatusMenu();
            break;
        case 3:
            showBuyerMenu();
            break;
        case 4:
            showLoginMenu();
            break;
        default:
            exit(0);
    }
}

void
Menu::showStatusMenu()
{
    cout « "Please choose a buyer." « endl;
    vector<vector<string» buyers = _eshop->checkStatus();
    int idx = 0;
    for(auto b: buyers) {
        cout « to_string(++idx) « ". " « b[0] « " " « b[1] « " " « b[2] « " " « b[3] « " " « endl;
    }
    cout « to_string(idx+1) « ". Back" « endl;

    string ans; int chc;
    do {
        cout « "Buyer: ";
        cin » ans;
        chc = stoi(ans);
    } while( !cin.fail() && chc < 1 && chc > idx+1);

    if (chc <= idx) {
        Buyer* buyer = _eshop->getBuyerByEmail(buyers[chc][0]);
        buyer->showCart();

        if (askYesNo("Do you want to delete this buyer?")) {
            _eshop->removeBuyer(buyer);
        }
        showStatusMenu();
    } else {
        showOwnerMenu();
    }
}


void
Menu::showBuyerMenu()
{
    cout « "Hello " « quoted(_buyer->getName()) « " " « quoted(_buyer->getEmail()) « endl;
    cout « "You are a " « quoted(_buyer->getCategoryName()) « " customer with " « _buyer->getBonus() «
 " points" « endl;
    cout « endl;
    cout « "Please choose an action." « endl;
    cout « "1. Browse Store" « endl;
    cout « "2. View Cart" « endl;
    cout « "3. Checkout" « endl;
    cout « "4. Back" « endl;
    cout « "5. Logout" « endl;
    cout « "6. Exit" « endl;

    string ans;
    do {
        cout « "Action: ";
        cin » ans;
    } while( !cin.fail() && stoi(ans) < 1 && stoi(ans) > 6);

    switch (stoi(ans)) {
        case 1:
            showBrowseMenu();
            break;
        case 2:
            showCartMenu();
            break;
        case 3:
            showCheckoutMenu();
            break;
        case 4:
            showBuyerMenu();
            break;
        case 5:
            showLoginMenu();
            break;
        default:
            exit(0);
    }
}

void
Menu::showBrowseMenu()
{
    cout « "Please choose a category." « endl;
    vector<pair<string, int» categories = _eshop->getCategories();
    int idx = 0;
    for(auto c: categories) {
        cout « to_string(++idx) « ". " « c.first « " " « "(" « c.second « ")" « endl;
    }
    cout « to_string(idx+1) « ". Back" « endl;

    string ans; int chc;
    do {
        cout « "Category: ";
        cin » ans;
        chc = stoi(ans);
    } while( !cin.fail() && chc < 1 && chc > idx+1);

    if (chc <= idx) {
        showCategoryMenu(categories[chc-1].first);
    } else {
        if (_user->isAdmin())
            showOwnerMenu();
        else
            showBuyerMenu();
    }
}

void
Menu::showCategoryMenu(string category)
{
    cout « "Please select a product." « endl;
    vector<pair<int, string» products = _eshop->getProductsInCategory(category);
    int idx = 0;
    for(auto p: products) {
        cout « to_string(++idx) « ". " « p.first « " " « "(" « p.second « ")" « endl;
    }
    cout « to_string(idx+1) « ". Back" « endl;

    string ans; int chc;
    do {
        cout « "Product: ";
        cin » ans;
        chc = stoi(ans);
    } while( !cin.fail() && chc < 1 && chc > idx+1);

    if (chc <= idx) {
        showProductMenu(category, products[chc-1].first);
    } else {
        showBrowseMenu();
    }
}

void
Menu::showProductMenu(string back, int id){
    Item* item = _eshop->getItemById(id);
    _eshop->showProduct(item);
    if (_user->isAdmin()) {
        if (askYesNo("Do you want update the stock of this product?")) {
            int quantity;
            cout « "Quantity: ";
            cin » quantity;
            _eshop->updateItemStock(item, quantity);
        }
    } else {
        if (askYesNo("Do you want to buy this product?")) {
            int quantity;
            cout « "Quantity: ";
            cin » quantity;
            _buyer->placeOrder(item, quantity);
        }
    }
    showCategoryMenu(back);
}

void
Menu::showCartMenu()
{
    try {
        set<int> ids = _buyer->showCart();
        int chc;
        string ans;
        Item* item;

        cout « "Actions" « endl;
        cout « "1. Clear Cart" « endl;
        cout « "2. Checkout" « endl;
        cout « "3. Back" « endl;

        do {
            cout « "Please select an action or a product ID to edit your order: ";
            cin » ans;
            chc = stoi(ans);
            if (ids.contains(chc)) {
                item = _eshop->getItemById(chc);
                break;
            }
        } while (!cin.fail() && chc < 1 && chc > 3);

        switch (chc) {
            case 1:
                _buyer->clearCart();
                break;
            case 2:
                _buyer->checkout();
                break;
            case 3:
                break;
            default:
                pair<Item*, int> item_order;
                try {
                    item_order = _buyer->getItemOrder(item);
                } catch (const EShopError& e) {
                    cout « e.error() « endl;
                    showCartMenu();
                }
                cout « "Editing Item Order: ";
                cout « item_order.first->getName() « " (" « item_order.second « ")" « endl;
                cout « "1. Delete Item Order" « endl;
                cout « "2. Change Item Order" « endl;

                do {
                    cout « "Action: ";
                    cin » ans;
                    chc = stoi(ans);
                }
                while( (!cin.fail() && chc < 1 && chc > 2) );

                switch (chc) {
                    case 1:
                        _buyer->removeFromOrder(item);
                        break;
                    case 2:
                        do {
                            cout « "Do you want to (1)add or (2)delete: ";
                            cin » ans;
                            chc = stoi(ans);
                        } while( (!cin.fail() && chc < 1 && chc > 2) );

                        int qnt;
                        do {
                            cout « "Please enter the quantity: ";
                            cin » ans;
                            qnt = stoi(ans);
                        }
                        while( (!cin.fail() && qnt < 0 && qnt > item_order.second) );

                        if (chc == 1)
                            _buyer->placeOrder(item, qnt);
                        else
                            _buyer->placeOrder(item, -qnt);
                        break;
                }
                showCartMenu();
        }
    } catch (const EShopError& e) {
        cout « e.error() « endl;
    }
    showBuyerMenu();
}

void
Menu::showCheckoutMenu()
{
    _buyer->checkout();
    showBuyerMenu();
}
```

```
Menu::~Menu()
{
}

void
Menu::login(string email)
{
    try {
        _buyer = nullptr;
        _owner = nullptr;
        if (email == _eshop->getOwner()->getEmail()) {
            _owner = _eshop->getOwner();
            _user = _owner;
        } else {
            _buyer = _eshop->getBuyerByEmail(email);
            _user = _buyer;
        }
    } catch (const EShopError& e) {
        throw e;
    }
    cout « "Authenticated as: " « _user->getName() « " " « _user->getEmail() « endl;
}

bool
Menu::askYesNo(string message) {
    string ans;
    do {
        cout « message « " [y/n]: ";
        cin » ans;
    }
    while( !cin.fail() && ans!="y" && ans!="n" );

    if (ans == "y") return true;
    else return false;
}
```

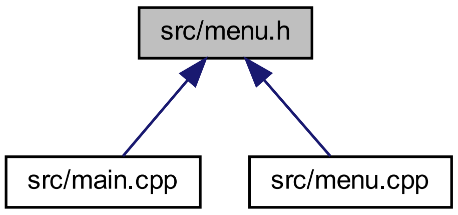
6.21 src/menu.h File Reference

```
#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.

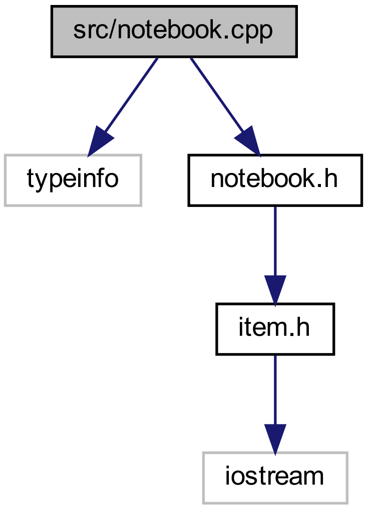
6.22 menu.h

```
#ifndef MENU_H
#define MENU_H

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


class Menu
{
public:
    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:
    void login(string);
    bool askYesNo(string);

    User* _user = nullptr;
    Owner* _owner = nullptr;
    Buyer* _buyer = nullptr;
    EShop* _eshop;
};

#endif // MENU_H
```

6.23 src/notebook.cpp File Reference

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

Include dependency graph for notebook.cpp:



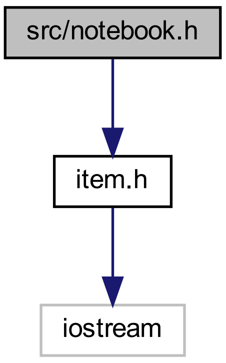
6.24 notebook.cpp

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

Notebook::Notebook(int subjects, int stock, double price, string name, string desc) : Item(stock,
 price, name, desc)
{
    setSubjects(subjects);
    setCategory(typeid(*this).name());
    setId();
}

string
Notebook::getDetails()
{
    string ret;
    ret += to_string(_subjects);
    ret += "\n";
    return ret;
}

void
Notebook::setId()
{
    size_t h_obj = hash<string>{}(typeid(*this).name());
    size_t h_sub = hash<int>{}(_subjects);
    Item::setId(h_obj ^ (h_sub « 1));
}

void
Notebook::setSubjects(int subjects) { _subjects = subjects; }
int
Notebook::getSubjects() { return _subjects; }
```

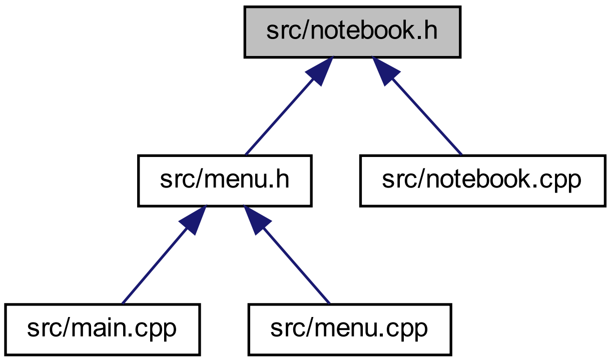
6.25 src/notebook.h File Reference

```
#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.

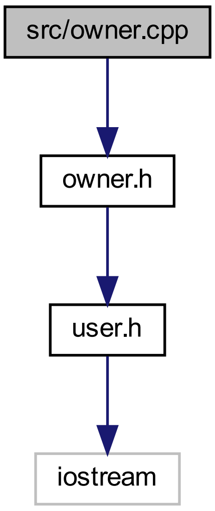
6.26 notebook.h

```
#ifndef NOTEBOOK_H
#define NOTEBOOK_H

#include "item.h"

using namespace std;

class Notebook final : public Item
{
public:
    Notebook(int subjects, int stock, double price, string name, string desc);

    void setSubjects(int subjects);
    int  getSubjects();

    void   setId() override;
    string getDetails() override;

private:
    int _subjects;
};

#endif // NOTEBOOK_H
```

6.27 src/owner.cpp File Reference

`#include "owner.h"`

Include dependency graph for owner.cpp:



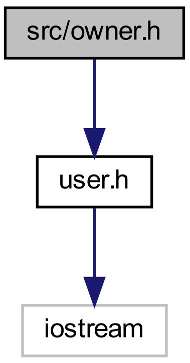
6.28 owner.cpp

```
#include "owner.h"

Owner::Owner(string name, string email) : User(name, email)
{
    _isAdmin = true;
}

bool
Owner::isAdmin() { return _isAdmin; }
```

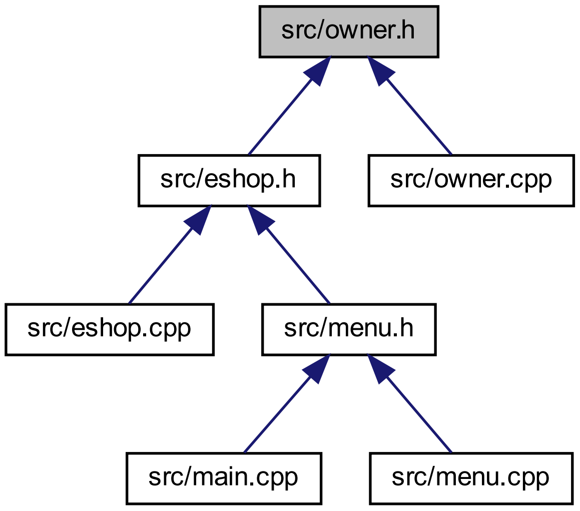
6.29 src/owner.h File Reference

```
#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.

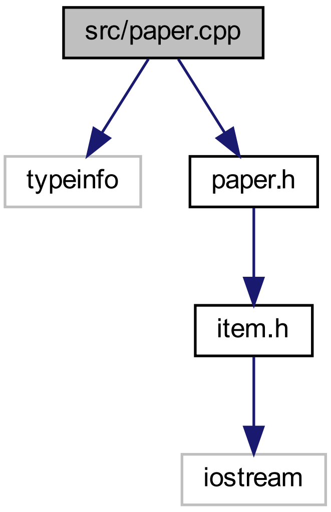
6.30 owner.h

```
#ifndef OWNER_H
#define OWNER_H

#include "user.h"

using namespace std;

class Owner final: public User {
public:
    Owner(string name, string email);

    bool isAdmin();

private:
    bool _isAdmin;
};

#endif // OWNER_H
```

6.31 src/paper.cpp File Reference

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

Include dependency graph for paper.cpp:



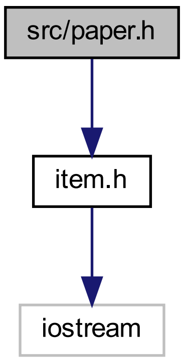
6.32 paper.cpp

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


Paper::Paper(int pages, int weight, int stock, double price, string name, string desc) : Item(stock,
      price, name, desc)
{
    setPages(pages);
    setWeight(weight);
    setCategory(typeid(*this).name());
    setId();
}

string Paper::getDetails()
{
    string ret;
    ret += to_string(_pages);
    ret += ", ";
    ret += to_string(_weight);
    ret += "\n";
    return ret;
}

void Paper::setId()
{
    size_t h_obj = hash<string>{}(typeid(*this).name());
    size_t h_pgs = hash<int>{}(_pages);
    size_t h_wgt = hash<int>{}(_weight);
    Item::setId(h_obj ^ ((h_pgs ^ (h_wgt « 1)) « 1));
}

void
Paper::setPages(int pages) { _pages = pages; }
int
Paper::getPages() { return _pages; }

void
Paper::setWeight(int weight) { _weight = weight; }
int
Paper::getWeight() { return _weight; }
```

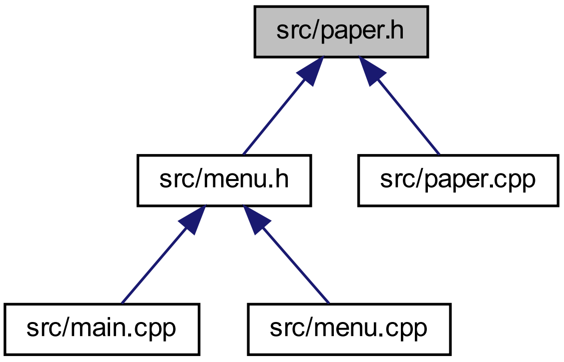
6.33 src/paper.h File Reference

`#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.

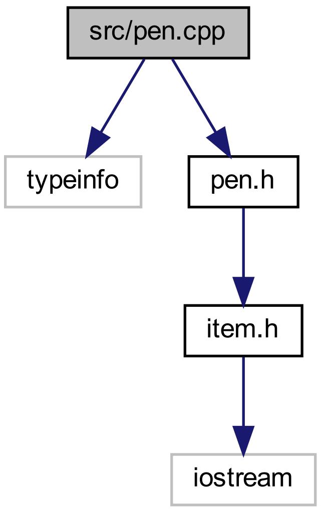
6.34 paper.h

```
#ifndef PAPER_H
#define PAPER_H

#include "item.h"

using namespace std;

class Paper final : public Item
{
public:
    Paper(int pages, int weight, int stock, double price, string name, string desc);

    void setPages(int pages);
    int  getPages();

    void setWeight(int);
    int  getWeight();

    void   setId();
    string getDetails();

private:
    int _pages;
    int _weight;
};

#endif // PAPER_H
```

6.35 src/pen.cpp File Reference

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

Include dependency graph for pen.cpp:



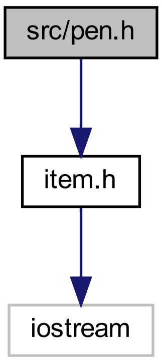
6.36 pen.cpp

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

Pen::Pen(double tipSize, string color, int stock, double price, string name, string desc) :
 Item(stock, price, name, desc)
{
    setTipSize(tipSize);
    setColor(color);
    setCategory(typeid(*this).name());
    setId();
}

void
Pen::setId()
{
    size_t h_obj = hash<string>{}(typeid(*this).name());
    size_t h_clr = hash<string>{}(_color);
    size_t h_tip = hash<double>{}(_tipSize);
    Item::setId(h_obj ^ ((h_clr ^ (h_tip « 1)) « 1));
}

string
Pen::getDetails()
{
    string ret;
    ret += to_string(_tipSize);
    ret += ", ";
    ret += _color;
    ret += "\n";
    return ret;
}

void
Pen::setTipSize(double size) { _tipSize = size; }
double
Pen::getTipSize() { return _tipSize; }

void
Pen::setColor(string color) { _color = color; }
string
Pen::getColor() { return _color; }
```

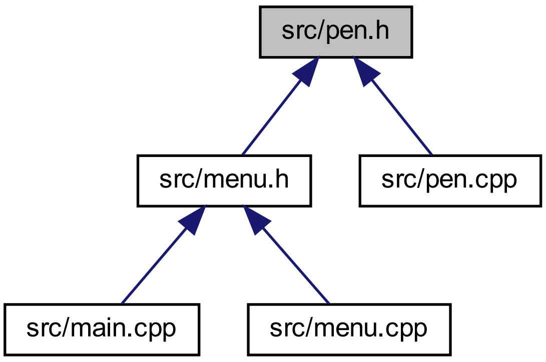
6.37 src/pen.h File Reference

```
#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.

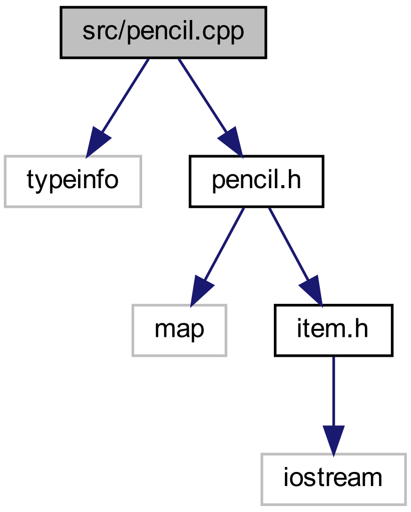
6.38 pen.h

```
#ifndef PEN_H
#define PEN_H

#include "item.h"

using namespace std;

class Pen final : public Item
{
public:
    Pen(double tipSize, string color, int stock, double price, string name, string desc);

    void   setTipSize(double tipSize);
    double getTipSize();

    void   setColor(string color);
    string getColor();

    void   setId() override;
    string getDetails() override;

private:
    double _tipSize;
    string _color;
};

#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

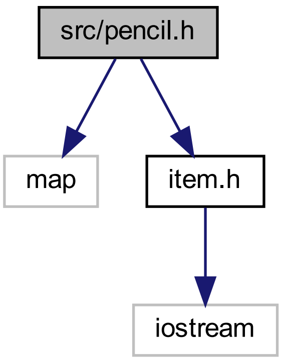
```
#include <typeinfo>
```

```
#include "pencil.h"

Pencil::Pencil(double tipSize, Pencil::Type type, int stock, double price, string name, string desc) :
 Item(stock, price, name, desc)
{
    setTipSize(tipSize);
    setType(type);
    setCategory(typeid(*this).name());
    setId();
}

void
Pencil::setId()
{
    size_t h_obj = hash<string>{}(typeid(*this).name());
    size_t h_typ = hash<string>{}(_typeMap[_type]);
    size_t h_tip = hash<double>{}(_tipSize);
    Item::setId(h_obj ^ ((h_typ ^ (h_tip « 1)) « 1));
}

string
Pencil::getDetails()
{
    string ret;
    ret += to_string(_tipSize);
    ret += ", ";
    ret += _typeMap[_type];
    ret += "\n";
    return ret;
}

void
Pencil::setTipSize(double size) { _tipSize = size; }
double
Pencil::getTipSize() { return _tipSize; }

void
Pencil::setType(Type type) { _type = type; }
Pencil::Type
Pencil::getType() { return _type; }
```

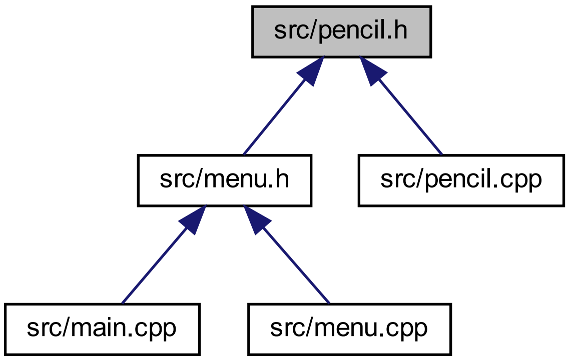
6.41 src/pencil.h File Reference

```
#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.

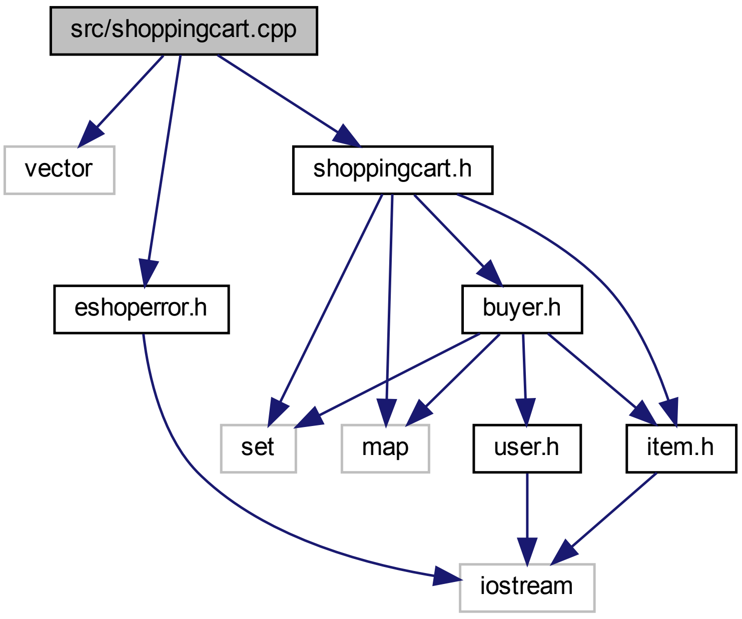
6.42 pencil.h

```
#ifndef PENCIL_H
#define PENCIL_H

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

class Pencil final : public Item
{
public:

    enum Type {
        H,
        B,
        HB,
    };

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

    void   setTipSize(double tipSize);
    double getTipSize();

    void setType(Type type);
    Type getType();

    void   setId() override;
    string getDetails() override;

private:
    double _tipSize;
    Type _type;
    map<Type, string> _typeMap { {H, "H"}, { B, "B" }, {HB, "HB" } };
};

#endif // PENCIL_H
```

6.43 src/shoppingcart.cpp File Reference

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

Include dependency graph for shoppingcart.cpp:



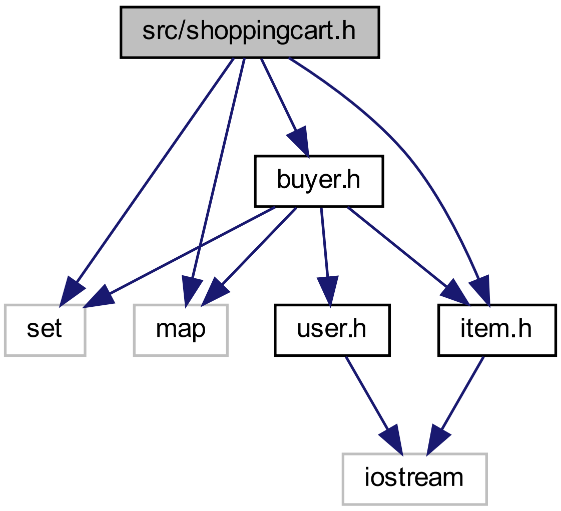
6.44 shoppingcart.cpp

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

ShoppingCart::ShoppingCart(Buyer* buyer)
{
    _buyer = buyer;
}

void
ShoppingCart::addItemOrder(Item* item, int quantity)
{
    //TODO: exception
    if (item->getStock() >= quantity) {
        if (_order.find(item) != _order.end())
            changeItemOrderQuantity(item, quantity);
        else
            _order.emplace(item, quantity);
        item->setStock(item->getStock() - quantity);
    } else {
        throw EShopError("Item " + item->getName() + " does not have enough stock.");
    }
}

void
ShoppingCart::removeItemOrder(Item* item)
{
    if (_order.find(item) != _order.end()) {
        item->setStock(item->getStock() + _order[item]);
        _order.erase(item);
    }
}

void
ShoppingCart::changeItemOrderQuantity(Item* item, int quantity)
{
    _order[item] += quantity;
    if (_order[item] <= 0) removeItemOrder(item);
}

pair<Item*, int>
ShoppingCart::getItemOrder(Item* item)
{
    if (_order.find(item) != _order.end())
        return make_pair(item, _order[item]);
    else {
        throw EShopError("Item " + item->getName() + " not found in the cart");
        return make_pair(nullptr, 0);
    }
}

double
ShoppingCart::calculateNet()
{
    double value = 0;
    for(auto o: _order) value += o.first->getPrice()*o.second;
    return value;
}

double
ShoppingCart::calculateCourier()
{
    if (_buyer->getCategory() == Buyer::Gold) return 0;
    else {
        double value = calculateNet();
        if (_buyer->getCategory() == Buyer::Silver) return value*0.01;
        else return (value*0.02 >= 3 ? value*0.02 : 3);
    }
}

set<int>
ShoppingCart::showCart()
{
    set<int> ids;
    if (_order.empty()) throw EShopError("Shopping cart is empty.");
    for (auto o: _order) {
        ids.insert(o.first->getId());
        cout « o.first->getId() « ", " « o.first->getName() « ", " « o.second « ", "«
  o.first->getPrice()*o.second « endl;
    }
    cout « "Order value: " « calculateNet() « endl;
    cout « "Courier value: " « calculateCourier() « endl;
    return ids;
}

void
ShoppingCart::clearCart()
{
    for (auto o: _order)
        o.first->setStock(o.first->getStock() + o.second);
    _order.clear();
}

void
ShoppingCart::checkout()
{
    showCart();

    string ans;
    do {
        cout « "Do you want to continue the checkout? [y/n]: ";
        cin » ans;
    }
    while( (!cin.fail() && ans!="y" && ans!="n") );

    if (ans == "y") {
        _buyer->awardBonus(calculateNet());
        _order.clear();
    }
}

```

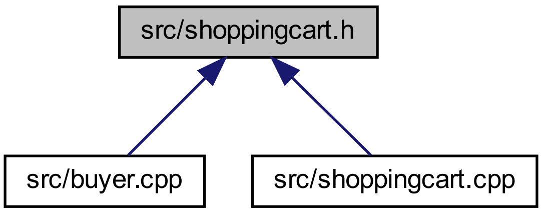
6.45 src/shoppingcart.h File Reference

```
#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.

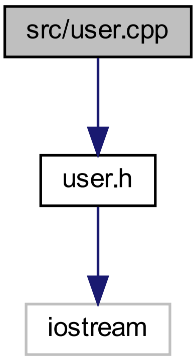
6.46 shoppingcart.h

```
#ifndef SHOPPINGCART_H
#define SHOPPINGCART_H

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

using namespace std;

class ShoppingCart
{
public:
    explicit ShoppingCart(Buyer* buyer);

    void addItemOrder(Item* item, int quantity);
    void removeItemOrder(Item* item);
    void changeItemOrderQuantity(Item* item, int quantity);
    pair<Item*, int> getItemOrder(Item*);

    set<int> showCart();
    void clearCart();

    void checkout();

    double calculateNet();
    double calculateCourier();

private:
    map<Item*, int> _order;
    Buyer* _buyer;
};

#endif // SHOPPINGCART_H
```

6.47 src/user.cpp File Reference

`#include "user.h"`

Include dependency graph for user.cpp:



6.48 user.cpp

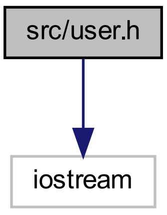
```
#include "user.h"

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

```
{
    setName(name);
    setEmail(email);
}

void
User::setName(string name) { _name = name; }

string
User::getName() { return _name; }

void
User::setEmail(string email) { _email = email; }

string
User::getEmail() { return _email; }
```

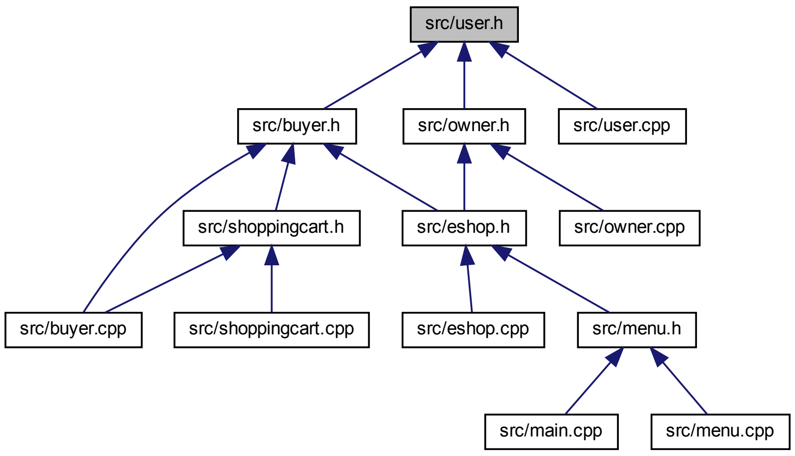
6.49 src/user.h File Reference

`#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

```
#ifndef USER_H
#define USER_H

#include <iostream>

using namespace std;

class User {
public:
    User(string name, string email);

    string getName();
    string getEmail();

    virtual bool isAdmin() = 0;

private:
    void setName(string name);
    void setEmail(string email);

    string _name;
    string _email;
};

#endif // USER_H
```

Index