

EeasyStore

Généré par Doxygen 1.9.3

1 Index des classes	1
1.1 Liste des classes	1
2 Index des fichiers	3
2.1 Liste des fichiers	3
3 Documentation des classes	5
3.1 Référence de la classe Client	5
3.1.1 Description détaillée	6
3.1.2 Documentation des fonctions membres	6
3.1.2.1 deleteProductFromBasket()	6
3.1.2.2 getClientId()	6
3.1.2.3 setBasket()	6
3.1.2.4 setQuantityForAProductInBasket()	7
3.1.2.5 toStringWithBasket()	7
3.2 Référence de la classe Magasin	7
3.2.1 Description détaillée	9
3.2.2 Documentation des constructeurs et destructeur	10
3.2.2.1 Magasin()	10
3.2.3 Documentation des fonctions membres	10
3.2.3.1 AddProductToBasket()	10
3.2.3.2 clearBasket()	11
3.2.3.3 clients()	11
3.2.3.4 deleteProductFromBasket()	11
3.2.3.5 diplayAllOrdersByState()	12
3.2.3.6 diplayProductsByName()	12
3.2.3.7 displayOrderOfAClient()	12
3.2.3.8 displayProductOfAClient()	13
3.2.3.9 findClientById()	13
3.2.3.10 findClientByName()	13
3.2.3.11 findOrderByCientIdAndOrderId()	14
3.2.3.12 findOrderById()	14
3.2.3.13 findOrdersByClientId()	15
3.2.3.14 findProductById()	15
3.2.3.15 findProductKeyByName()	15
3.2.3.16 modifyProductQuantityInBasket()	16
3.2.3.17 newClient()	16
3.2.3.18 newOrder()	16
3.2.3.19 newProduct()	17
3.2.3.20 nextClientKey()	17
3.2.3.21 nextOrderKey()	17
3.2.3.22 nextProductKey()	17
3.2.3.23 productNumberInOrder()	17

3.2.3.24 products()	18
3.2.3.25 setClients()	18
3.2.3.26 setOrders()	18
3.2.3.27 setProducts()	18
3.2.3.28 updateAnOrderState()	19
3.2.3.29 updateOrders()	19
3.2.3.30 updateProductsByQuantityByName()	19
3.2.3.31 validateAnOrder()	20
3.3 Référence de la classe Menu	20
3.3.1 Description détaillée	21
3.3.2 Documentation des fonctions membres	21
3.3.2.1 menu_gestion_des_utilisateurs()	21
3.4 Référence de la classe Order	22
3.4.1 Documentation des constructeurs et destructeur	22
3.4.1.1 Order()	22
3.4.2 Documentation des fonctions membres	23
3.4.2.1 addProduct()	23
3.4.2.2 getCilent()	23
3.4.2.3 getOrderId()	23
3.4.2.4 getProductOrderId()	23
3.4.2.5 getProducts()	24
3.4.2.6 getStatus()	24
3.4.2.7 setCilent()	24
3.4.2.8 setOrderId()	25
3.4.2.9 setProducts()	25
3.4.2.10 setStatus()	25
3.4.2.11 toFile()	25
3.4.2.12 toString()	26
3.5 Référence de la classe Product	26
3.5.1 Description détaillée	26
3.6 Référence de la classe Produit	27
4 Documentation des fichiers	29
4.1 client.h	29
4.2 magasin.h	29
4.3 main.h	30
4.4 menu.h	31
4.5 order.h	31
4.6 product.h	31
4.7 produit.h	32
Index	33

Chapitre 1

Index des classes

1.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

Client	La classe Client nous permet de répondre à la question: 4.a) Créer la classe Client avec ses variables membres (identifiant, prénom, nom, panier d'achat) et ses fonctions getters. La classe Client est divisée en trois parties:	5
Magasin	: réponse à la question 1.a) Créer la classe Magasin avec ses variables membres, son constructeur La classe Magasin est divisée en trois parties:	7
Menu	: réponse à la question 8.b)8.b) Ecrire le code nécessaire à l'enregistrement et à la lecture des données (produits, clients et commandes) dans des fichiers. La classe Menu contient: les déclarations public	20
Order	22
Product	Permet de répondre à la question: 2.a) Créer la classe Produit avec ses variables membres, son constructeur et ses fonctions getters. La classe Product est divisée en trois parties:	26
Produit	27

Chapitre 2

Index des fichiers

2.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

C:/temp/mootez/tp_3easystore/ client.h	29
C:/temp/mootez/tp_3easystore/ magasin.h	29
C:/temp/mootez/tp_3easystore/ main.h	30
C:/temp/mootez/tp_3easystore/ menu.h	31
C:/temp/mootez/tp_3easystore/ order.h	31
C:/temp/mootez/tp_3easystore/ product.h	31
C:/temp/mootez/tp_3easystore/ produit.h	32

Chapitre 3

Documentation des classes

3.1 Référence de la classe Client

La classe `Client` nous permet de répondre à la question: 4.a) Créer la classe `Client` avec ses variables membres (identifiant, prénom, nom, panier d'achat) et ses fonctions getters. La classe `Client` est divisée en trois parties:

```
#include <client.h>
```

Fonctions membres publiques

- **Client** (unsigned int _clientId, string _FirstName, string _lastName)
- void **setClientId** (unsigned int value)
Client 4.a) Créer la classe `Client` avec ses variables membres (identifiant, prénom, nom, panier d'achat) et ses fonctions getters.
- void **setFirstName** (const string &value)
- void **setLastName** (const string &value)
- void **setBasket** (const `Product` &value)
Client::setBasket permet de modifier le contenu du panier d'achat.
- std::vector< `Product` > **getBasket** () const
- unsigned int **getClientId** () const
Client::Client :ce constructeur permet de créer une instance de la classe `Client`.
- string **getFirstName** () const
- string **getLastName** () const
- void **clearBasket** ()
Client::clearBasket est une méthode helper permettant de vider le panier d'achat.
- void **setQuantityForAProductInBasket** (const unsigned int &_productId, const unsigned int &quantity)
Client::setQuantityForAProductInBasket est une méthode permettant de modifier la quantité d'un produit ajouté au panier d'achat.
- void **deleteProductFromBasket** (const unsigned int &_productId)
Client::deleteProductFromBasket cette méthode permettant de supprimer un produit du panier d'achat.
- string **toStringWithBasket** ()
Client::toString permet de surcharger l'opérateur << pour pouvoir afficher toutes les informations du client (incluant les produits du panier d'achat.
- string **toString** ()

3.1.1 Description détaillée

La classe [Client](#) nous permet de répondre à la question: 4.a) Créer la classe [Client](#) avec ses variables membres (identifiant, prénom, nom, panier d'achat) et ses fonctions getters. La classe [Client](#) est divisée en trois parties:

- partie des déclarations privées (clientId, firstName, lastName et basket)
- partie des déclarations publiques
 - contenant les constructeurs (deux constructeurs [Client\(\)](#) et [Client\(unsigned int _clientId, string _firstName, string _lastName\)](#))
 - les getters et les setters
 - et les fonctions membres (clearBasket, setQuantityForAProductInBasket, deleteProductFromBasket, to← StringWithBasket et toString)

3.1.2 Documentation des fonctions membres

3.1.2.1 deleteProductFromBasket()

```
void Client::deleteProductFromBasket (
    const unsigned int & _productId )
```

[Client::deleteProductFromBasket](#) cette méthode permettant de supprimer un produit du panier d'achat.

Paramètres

<code>← productId</code>	paramètre d'entrée permettant d'identifier le produit à effacer
------------------------------	---

3.1.2.2 getClientId()

```
unsigned int Client::getClientId ( ) const
```

`Client::Client` : ce constructeur permet de créer une instance de la classe [Client](#).

Paramètres

<code>_clientId</code>	: identifiant du client
<code>_FirstName</code>	: nom du client
<code>_lastName</code>	: prénom du client

3.1.2.3 setBasket()

```
void Client::setBasket (
```

```
const Product & value )
```

`Client::setBasket` permet de modifier le contenu du panier d'achat.

Paramètres

<code>value</code>	
--------------------	--

3.1.2.4 setQuantityForAProductInBasket()

```
void Client::setQuantityForAProductInBasket (
    const unsigned int & _productId,
    const unsigned int & quantity )
```

`Client::setQuantityForAProductInBasket` est une méthode permettant de modifier la quantité d'un produit ajouté au panier d'achat.

Paramètres

<code>_↔ productId</code>	identifiant du produit dans le magasin
<code>quantity</code>	quantite a modifier dans le panier du client

3.1.2.5 toStringWithBasket()

```
string Client::toStringWithBasket ( )
```

`Client::toString` permet de surcharger l'opérateur << pour pouvoir afficher toutes les informations du client (incluant les produits du panier d'achat).

Renvoie

retourne une chaine de caractere permettant de la sauvegrader dans un fichier externe.

La documentation de cette classe a été générée à partir du fichier suivant :

- C:/temp/mootez/tp_3easystore/client.h
- C:/temp/mootez/tp_3easystore/client.cpp

3.2 Référence de la classe Magasin

The `Magasin` class : reponse a la question 1.a) Créer la classe `Magasin` avec ses variables membres, son constructeur La classe `Magasin` est divisée en trois parties:

```
#include <magasin.h>
```

Fonctions membres publiques

- **Magasin** (**Product** p, **Client** c, **Order** o)
Magasin::Magasin constructeur de la classe **Magasin** reponse a la question 1.a) Créer la classe **Magasin** avec ses variables membres, son constructeur.
- void **setOrders** (const **Order** &order)
Magasin::setOrders methode setter de variable membre de la class magasin.
- void **setClients** (const **Client** &client)
Magasin::setClients methode setter de variable membre de la class magasin.
- void **setProducts** (const **Product** &product)
Magasin::setProducts methode setter de variable membre de la class magasin.
- std::vector< **Product** > **products** () const
Magasin::products methode getter de retourner le tableau de _produits dans le magasin.
- std::vector< **Client** > **clients** () const
Magasin::clients cette methode est un getter de la classe permettant de retourner le tableau de clients.
- std::vector< **Order** > **orders** () const
Magasin::Magasin constructeur vide de la classe **Magasin**.
- void **newClient** (**Client** *c)
Magasin::newClient cette fonction permet d'ajouter un nouveau client au magasin, pour garentir l'unicite nous devons comparer l'identifiant du client a celui de dernier client enregistre au magasin.
- int **AddProductToBasket** (unsigned int _clientId, unsigned int _productId, unsigned int quantity)
Magasin::AddProductToBasket cette methode permet d'ajouter un produit a un client la fonction commence par chercher le client dans le magasin en utilisant une boucle for, une fois trouve, elle cherche le produit dans une boucle imbriquee pour selection l'objet produit. dans la derniere etape en ajoute le produit trouve dans le panier du client par fonction **setBasket** membre de la classe **Client**.
- int **clearBasket** (unsigned int _clientId)
Magasin::clearBasket est une méthode helper permettant de vider le panier d'achat d'un client.
- void **displayAllProducts** ()
Magasin::displayAllProducts: Reponse a la question 3.b) Ecrire une méthode ou une fonction helper permettant d'afficher à l'écran tous les produits référencés dans le magasin. cette fonction permet de parcourir le tableau produits et d'afficher son contenu.
- void **displayProductsByName** (string pname)
Magasin::displayProductsByName: 3.c) Ecrire une méthode ou une fonction helper permettant d'afficher à l'écran un produit sélectionné par son nom.
- void **updateProductsByQuantityByName** (string ptitle, unsigned int new_quantity)
Magasin::updateProductsByQuantityByName: 4.c) Ecrire une méthode ou une fonction helper permettant de mettre à jour la quantité d'un produit sélectionné par son nom.
- unsigned int **nextProductKey** () const
Magasin::nextProductKey cette methode permet de generer un nouveau identifiant du **Product**.
- unsigned int **nextClientKey** () const
Magasin::nextClientKey cette methode permet de generer un nouveau identifiant du **Client**.
- unsigned int **findClientByName** (string fname, string lname) const
Magasin::nextClientKey cette fonction permet de verifier si le client est deja enregistre au magasin ou non si oui la fonction retourne son identifiant autrement la fonction genere un nouveau identifiant pour garantir l'unicite des clients dans le magasin.
- unsigned int **findProductKeyByName** (string _title) const
Magasin::findProductKeyByName cette methode permet de chercher un produit dans le magasin par son nom (title)
- void **newProduct** (**Product** *p)
Magasin::newProduct cette fonction permet d'ajouter un nouveau **Produit** au magasin, pour garentir l'unicite nous devons compare l'identifiant s'il est a zeor cela veut dire que le produit est nouveau au magasin.
- int **modifyProductQuantityInBasket** (unsigned int _clientId, unsigned int _productId, unsigned int quantity)
Magasin::modifyProductQuantityInBasket Cette méthode permet de modifier la quantité d'un produit existant dans le panier d'achat d'un client. dans un premier temps la methode cherche le client dans le magasin par son identifiant, en suite elle appelle une fonction membre de la classe client pour chercher le produit par son identifiant et modifier sa quantite.
- int **deleteProductFromBasket** (unsigned int _clientId, unsigned int _productId)
Magasin::deleteProductFromBasket Cette méthode permet de supprimer un produit au panier d'achat d'un client. dans un premier temps la methode cherche le client dans le magasin par son identifiant, en suite elle appelle une fonction membre de la classe client pour chercher le produit par son identifiant pour le retirer.
- int **displayProductOfAClient** (unsigned int _clientId)
Magasin::displayProductOfAClient cette methode permet d'afficher les produits selectionees dans le panier de client.

- void **diplayAllClients** ()
Magasin::diplayAllClients Cette méthode permet d'afficher à l'écran tous les clients du magasin.
- Client **findClientById** (unsigned int id) const
Magasin::findClientById cette methode permet d'afficher à l'écran un client sélectionné par son identifiant.
- int **validateAnOrder** (unsigned int _clientId)
Magasin::validateAnOrder 7.a) cette méthode permet de valider une commande et la sauvegrde de celle ci dans le tableau orders apres la recherche du client par son identifiant elle copie le contenu dans son panier dans le tableau de produit de la commande.
- unsigned int **nextOrderKey** () const
Magasin::nextOrderKey cette methode permet de generer un nouveau identifiant du **Order**.
- Order **findOrderByCientIdAndOrderId** (unsigned int orderId, unsigned int clientId) const
Magasin::findOrderByCientIdAndOrderId cette methode permet d'afficher à l'écran une commande sélectionné par son identifiant.
- int **updateAnOrderState** (unsigned int _clientId, unsigned int _orderId, string newSatate)
Magasin::updateAnOrderState nous permet de reponde a la question 7.b) Ajouter une méthode ou une fonction helper permettant de mettre à jour le statut d'une commande.
- void **diplayAllOrders** ()
Magasin::diplayAllOrders cette méthode permet d'afficher toutes les commandes passées.
- std::vector< Order > **findOrdersByClientId** (unsigned int clientId) const
Magasin::findOrdersByCientId cette methode permet de chercher tous les commandes d'un client.
- void **displayOrderOfAClient** (unsigned int _clientId)
Magasin::displayProductOfAClient affiche des information sur le client ainssi que le contenu de son panier.
- void **diplayAllOrdersByState** (string state)
Magasin::diplayAllOrdersByState reponse a la question 7.c) Ajouter une méthode ou une fonction helper permettant d'afficher toutes les commandes passées.
- void **newOrder** (Order *o)
Magasin::newOrder cette fonction permet d'ajouter une nouvelle commande au magasin, pour garentir l'unicite nous devons comparer l'identifiant du client a celui de dernier client enregistre au magasin.
- Product **findProductById** (unsigned int id) const
Magasin::findProductById cette methode permet d'afficher à l'écran un produit sélectionné par son identifiant.
- Order **findOrderById** (unsigned int orderId) const
Magasin::findOrderByCientIdAndOrderId cette methode permet d'afficher à l'écran une commande sélectionné par son identifiant.
- void **updateOrders** (unsigned int key, const Order &order)
Magasin::setOrders methode setter de variable membre de la class magasin.
- unsigned int **productNumberInOrder** (unsigned int key)
Magasin::productNumberInOrder methode permettant de chercher le nombre d'elements d'un article d'une commande.

3.2.1 Description détaillée

The **Magasin** class : reponse a la question 1.a) Créer la classe **Magasin** avec ses variables membres, son constructeur La classe **Magasin** est divisee en trois parties:

- partie des declarations privees(_products, _clients et _orders)
- parite des declarations public
 - contenant les constructeurs (**Magasin()** et **Magasin(Product p, Client c, Order o)**)
 - les getters et les setters
 - et les fonction membres: newClient, AddProductToBasket, clearBasket, diplayAllProducts, diplayProductsByName, updateProductsByQuantityByName, nextProductKey, nextClientKey, findClientByName, findProductKeyByName, newProduct, modifyProductQuantityInBasket, deleteProductFromBasket, displayProductOfAClient, diplayAllClients, findClientById, validateAnOrder, nextOrderKey, findOrderByCientIdAndOrderId, updateAnOrderState, diplayAllOrders, findOrdersByClientId, displayOrderOfAClient, diplayAllOrdersByState, newOrder, findProductById, findOrderById, updateOrders, productNumberInOrder

3.2.2 Documentation des constructeurs et destructeur

3.2.2.1 Magasin()

```
Magasin::Magasin (
    Product p,
    Client c,
    Order o )
```

Magasin::Magasin constructeur de la classe [Magasin](#) reponse a la question 1.a) Créer la classe [Magasin](#) avec ses variables membres, son constructeur.

Paramètres

<i>p</i>	parametre d'entree de type Produit pour etre sauvegarde dans le vecteur membre de <code>_products</code>
<i>c</i>	parametre d'entree de type Client pour etre sauvegarde dans le vecteur membre de <code>_clients</code>
<i>o</i>	parametre d'entree de type Order pour etre sauvegarde dans le vecteur membre de <code>_orders</code>

3.2.3 Documentation des fonctions membres

3.2.3.1 AddProductToBasket()

```
int Magasin::AddProductToBasket (
    unsigned int _clientId,
    unsigned int _productId,
    unsigned int quantity )
```

[Magasin::AddProductToBasket](#) cette methode permet d'ajouter un produit a un client la fonction commence par chercher le client dans le magasin en utilisant une boucle for, une fois trouve, elle cherche le produit dans une boucle imbriquee pour selection l'objet produit. dans la derniere etape en ajoute le produit trouve dans le panier du client par fonction `setBasket` membre de la classe [Client](#).

Paramètres

<i>_clientId</i>	parametre de recherche contenant l'identifiant du client
<i>↔ productId</i>	parametre de recherche contenant l'identifiant du client

Renvoie


la valeur de retour nous permet d'afficher un message a l'utilisateur

3.2.3.2 clearBasket()

```
int Magasin::clearBasket (
    unsigned int _clientId )
```

[Magasin::clearBasket](#) est une méthode helper permettant de vider le panier d'achat d'un client.

Paramètres

 <i>clientId</i>	parametre de selection du client a qui nous allons vider son panier
--	---

Renvoie

la valeur de retour permet d'informer l'utilisateur sur l'etat de l'operation

3.2.3.3 clients()

```
std::vector< Client > Magasin::clients ( ) const
```

[Magasin::clients](#) cette methode est un getter de la classe permettant de retourner le tableau de clients.

Renvoie


la valeur de retour est le tableau des clients dans le magasin

3.2.3.4 deleteProductFromBasket()

```
int Magasin::deleteProductFromBasket (
    unsigned int _clientId,
    unsigned int _productId )
```

[Magasin::deleteProductFromBasket](#) Cette méthode permet de supprimer un produit au panier d'achat d'un client. dans un premier temps la methode cherche le client dans le magasin par son identifiant, en suite elle appelle une fonction membre de la classe client pour chercher le produit par son identifiant pour le retirer.

Paramètres

<i>_clientId</i>	Identifiant du client
 <i>productId</i>	Identifiant du produit a retirer du panier

Renvoie

la methode retourne un entier pour la bonne gestion des messages de l'utilisateur

3.2.3.5 diplayAllOrdersByState()

```
void Magasin::diplayAllOrdersByState (
    string state )
```

[Magasin::diplayAllOrdersByState](#) reponse a la question 7.c) Ajouter une méthode ou une fonction helper permettant d'afficher toutes les commandes passées.

Paramètres

<i>state</i>	parametre specifiant l'etat des commandes a afficher
--------------	--

3.2.3.6 diplayProductsByName()

```
void Magasin::diplayProductsByName (
    string ptitle )
```

[Magasin::diplayProductsByName](#): 3.c) Ecrire une méthode ou une fonction helper permettant d'afficher à l'écran un produit sélectionné par son nom.

Paramètres

<i>ptitle</i>	c'est le nom du produit qui servira comme critere de selection du produit a afficher s'il existe
---------------	--

3.2.3.7 displayOrderOfAClient()

```
void Magasin::displayOrderOfAClient (
    unsigned int _clientId )
```

[Magasin::displayProductOfAClient](#) affiche des information sur le client ainssi que le contenu de son panier.

Paramètres

<i>_↔ clientId</i>	Identifiant du client
------------------------	-----------------------

Renvoie

retourne un entier permettant de savoir le reussite de l'operation

[Magasin::displayOrderOfAClient](#) cette methode permet d'afficher toutes les commandes d'un client donné.

Paramètres


<i>_↔ clientId</i>	parametre d'entree permettant de selectionner tous les commandes d'un client
------------------------	--

3.2.3.8 displayProductOfAClient()

```
int Magasin::displayProductOfAClient (
    unsigned int _clientId )
```

[Magasin::displayProductOfAClient](#) cette methode permet d'afficher les produits selectionees dans le panier de client.

Paramètres

 <i>clientId</i>	identifiant du client
---	-----------------------

Renvoie

valeur de retour qui nous permet d'afficher un message a l'utilisateur

3.2.3.9 findClientById()

```
Client Magasin::findClientById (
    unsigned int id ) const
```

[Magasin::findClientById](#) cette methode permet d'afficher à l'écran un client sélectionné par son identifiant.

Paramètres

<i>id</i>	
-----------	--

Renvoie

un objet de type [Client](#) resultat de la recherche

3.2.3.10 findClientByName()

```
unsigned int Magasin::findClientByName (
    string fname,
    string lname ) const
```

[Magasin::nextClientKey](#) cette fonction permet de verifier si le client est deja enregistre au magasin ou non si oui la fonction retourne son identifiant autrement la fonction genere un nouveau identifiant pour garantir l'unicite des clients dans le magasin.

Paramètres

<i>fname</i>	premier critere de recherche par le nom
<i>lname</i>	premier critere de recherche par le prenom

Renvoie

retourne l'identifiant du client s'il existe dans le magasin ou zero s'il n'existe pas.

3.2.3.11 findOrderByCientIdAndOrderId()

```
Order Magasin::findOrderByCientIdAndOrderId (
    unsigned int orderId,
    unsigned int clientId ) const
```

[Magasin::findOrderByCientIdAndOrderId](#) cette methode permet d'afficher à l'écran une commande sélectionné par son identifiant.

Paramètres

<i>orderId</i>	critere de recherche par identifiant de la commande
<i>clientId</i>	critere de recherche par identifiant du client

Renvoie

la valeur de retour est objet de type [Order](#) resultat de la recherche

3.2.3.12 findOrderById()

```
Order Magasin::findOrderById (
    unsigned int orderId ) const
```

[Magasin::findOrderByCientIdAndOrderId](#) cette methode permet d'afficher à l'écran une commande sélectionné par son identifiant.

Paramètres

<i>orderId</i>	critere de recherche par identifiant de la commande
----------------	---

Renvoie

la valeur de retour est objet de type [Order](#) resultat de la recherche

3.2.3.13 findOrdersByClientId()

```
std::vector< Order > Magasin::findOrdersByClientId (
    unsigned int clientId ) const
```

Magasin::findOrdersByClientId cette methode permet de chercher tous les commandes d'un client.

Paramètres

<i>clientId</i>	parametre d'entree permettant de selection le client a l'interieur du tableau des commandes
-----------------	---

Renvoie

la valeur de retour est un tableau de commande verifiant le critere de selection ou 0

3.2.3.14 findProductById()

```
Product Magasin::findProductById (
    unsigned int id ) const
```

Magasin::findProductById cette methode permet d'afficher à l'écran un produit sélectionné par son identifiant.

Paramètres

<i>id</i>	
-----------	--

Renvoie

un objet de type [Product](#) resultat de la recherche

3.2.3.15 findProductKeyByName()

```
unsigned int Magasin::findProductKeyByName (
    string _title ) const
```

Magasin::findProductKeyByName cette methode permet de chercher un produit dans le magasin par son nom (title)

Paramètres

<i>_title</i>	est un parametre d'entree pour nom du produit
---------------	---

Renvoie

retourne l'identifiant du produit dans le magasin

3.2.3.16 modifyProductQuantityInBasket()

```
int Magasin::modifyProductQuantityInBasket (
    unsigned int _clientId,
    unsigned int _productId,
    unsigned int quantity )
```

[Magasin::modifyProductQuantityInBasket](#) Cette méthode permet de modifier la quantité d'un produit existant dans le panier d'achat d'un client. dans un premier temps la methode cherche le client dans le magasin par son identifiant, en suite elle appelle une fonction membre de la classe client pour chercher le produit par son identifiant et modifier sa quantité.

Paramètres

<i>_clientId</i>	l'identifiant du client
<i>_↔ productId</i>	identifiant du produit
<i>quantity</i>	la quantité de produit

Renvoie

la valeur de retour nous permet d'afficher un message a l'utilisateur

3.2.3.17 newClient()

```
void Magasin::newClient (
    Client * c )
```

[Magasin::newClient](#) cette fonction permet d'ajouter un nouveau client au magasin, pour garantir l'unicite nous devons comparer l'identifiant du client a celui de dernier client enregistre au magasin.

Paramètres

<i>c</i>	
----------	--

3.2.3.18 newOrder()

```
void Magasin::newOrder (
    Order * o )
```

[Magasin::newOrder](#) cette fonction permet d'ajouter une nouvelle commande au magasin, pour garantir l'unicite nous devons comparer l'identifiant du client a celui de dernier client enregistre au magasin.

Paramètres

<i>c</i>	
----------	--

3.2.3.19 newProduct()

```
void Magasin::newProduct (
    Product * p )
```

[Magasin::newProduct](#) cette fonction permet d'ajouter un nouveau [Produit](#) au magasin, pour garantir l'unicite nous devons compare l'identifiant s'il est a zero cela veut dire que le produit est nouveau au magasin.

Paramètres

<i>p</i>	c'est un objet de type Product
----------	--

3.2.3.20 nextClientKey()

```
unsigned int Magasin::nextClientKey ( ) const
```

[Magasin::nextClientKey](#) cette methode permet de generer un nouveau identifiant du [Client](#).

Renvoie

la valeur de retour est un entier garantissant l'unicite de l'identifiant du [Client](#) dans le tableau des clients

3.2.3.21 nextOrderKey()

```
unsigned int Magasin::nextOrderKey ( ) const
```

[Magasin::nextOrderKey](#) cette methode permet de generer un nouveau identifiant du [Order](#).

Renvoie

la valeur de retour est un entier garantissant l'unicite de l'identifiant du commande dans le tableau des orders

3.2.3.22 nextProductKey()

```
unsigned int Magasin::nextProductKey ( ) const
```

[Magasin::nextProductkey](#) cette methode permet de generer un nouveau identifiant du [Product](#).

Renvoie

la valeur de retour est un entier garantissant l'unicite de l'identifiant du produit dans le tableau des produits

3.2.3.23 productNumberInOrder()

```
unsigned int Magasin::productNumberInOrder (
    unsigned int key )
```

[Magasin::productNumberInOrder](#) methode permettant de chercher le nombre d'elements d'un article d'une commande.

Paramètres

<i>key</i>	est l'index de la commande
------------	----------------------------

3.2.3.24 products()

```
std::vector< Product > Magasin::products ( ) const
```

[Magasin::products](#) methode getter de retourner le tableau de `_produits` dans le magasin.

Paramètres

<i>product</i>	est un table de products
----------------	--------------------------

3.2.3.25 setClients()

```
void Magasin::setClients (
    const Client & client )
```

[Magasin::setClients](#) methode setter de variable membre de la class magasin.

Paramètres

<i>order</i>	est la variable membre de type Client
--------------	---

3.2.3.26 setOrders()

```
void Magasin::setOrders (
    const Order & order )
```

[Magasin::setOrders](#) methode setter de variable membre de la class magasin.

Paramètres

<i>order</i>	est la variable membre de type Order
--------------	--

3.2.3.27 setProducts()

```
void Magasin::setProducts (
```

```
const Product & product )
```

[Magasin::setProducts](#) methode setter de variable membre de la class magasin.

Paramètres

<i>product</i>	est la variable membre de type Product
----------------	--

3.2.3.28 updateAnOrderState()

```
int Magasin::updateAnOrderState (
    unsigned int _clientId,
    unsigned int _orderId,
    string newSatate )
```

[Magasin::updateAnOrderState](#) nous permet de reponde a la question 7.b) Ajouter une méthode ou une fonction helper permettant de mettre à jour le statut d'une commande.

Paramètres

<i>_clientId</i>	identifiant du client
<i>_orderId</i>	identifiant de la commande
<i>newSatate</i>	nouvelle etat de la commande

Renvoie

la valeur de retour est un entier permettant d'afficher a l'utilisateur d'etat de l'operation

3.2.3.29 updateOrders()

```
void Magasin::updateOrders (
    unsigned int key,
    const Order & order )
```

[Magasin::setOrders](#) methode setter de variable membre de la class magasin.

Paramètres

<i>order</i>	est la variable membre de type Order
--------------	--

3.2.3.30 updateProductsByQuantityByName()

```
void Magasin::updateProductsByQuantityByName (
```

```
string ptitle,
unsigned int new_quantity )
```

[Magasin::updateProductsByQuantityByName](#): 4.c) Ecrire une méthode ou une fonction helper permettant de mettre à jour la quantité d'un produit sélectionné par son nom.

Paramètres


<i>ptitle</i>	nom du produit
<i>new_quantity</i>	nouvelle quantite a mettre a jour dans le tableau de produits

3.2.3.31 validateAnOrder()

```
int Magasin::validateAnOrder (
    unsigned int _clientId )
```

[Magasin::validateAnOrder](#) 7.a) cette méthode permet de valider une commande et la sauvegrde de celle ci dans le tableau orders apres la recherche du client par son identifiant elle copie le contenu dans son panier dans le tableau de produit de la commande.

Paramètres

 <i>clientId</i>	identifiant du client binificiaire de la commande.
--	--

Renvoie

la valeur de retour est entier perettant de gerer les message a le l'utilisateur

La documentation de cette classe a été générée à partir du fichier suivant :

- C:/temp/mootez/tp_3easystore/magasin.h
- C:/temp/mootez/tp_3easystore/magasin.cpp

3.3 Référence de la classe Menu

The [Menu](#) class : reponse a la question 8.b)8.b) Ecrire le code nécessaire à l'enregistrement et à la lecture des données (produits, clients et commandes) dans des fichiers. La classe [Menu](#) contient: les declarations public.

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

Fonctions membres publiques

- void **menu** ()
- void **menu_gestion_du_magasin** ()

[Menu::menu_gestion_du_magasin](#) permet de tester les methodes suivantes: 2.a) modifier la quantité disponible d'un produit. 2.b) afficher un produit. 3.a) ajouter un nouveau produit au magasin. 3.b) afficher tous les produits référencés dans le magasin. 3.c) afficher un produit sélectionné par son nom. 3.c) Mettre à jour la quantité d'un produit sélectionné par son nom.

— void `menu_gestion_des_utilisateurs` ()

`Menu::menu_gestion_des_utilisateurs` permet de tester les methodes suivantes: 4.a) Créer la classe `Client` avec ses variables membres (identifiant, prenom, nom, panier d'achat) et ses fonctions getters. 4.b) ajouter un produit au panier d'achat. 4.c) vider le panier d'achat. 4.d) modifier la quantité d'un produit ajouté au panier d'achat 4.e) supprimer un produit du panier d'achat. 4.f) afficher toutes les informations du client (incluant les produits du panier d'achat.

— void `menu_gestion_des_commandes` ()

`Menu::menu_gestion_des_commandes` permet de tester les methodes suivantes: 6.a) Créer la classe `Commande` avec ses variables membres (client, produits achetés, statut). 6.f) afficher toutes les informations de la commande. 7.a) valider une commande 7.b) mettre à jour le statut d'une commande 7.c) afficher toutes les commandes passées. 7.d) afficher toutes les commandes d'un client donné.

3.3.1 Description détaillée

The `Menu` class : reponse a la question 8.b)8.b) Ecrire le code nécessaire à l'enregistrement et à la lecture des données (produits, clients et commandes) dans des fichiers. La classe `Menu` contient: les declarations public.

- contenant les constructeurs
- et les fonction membres

3.3.2 Documentation des fonctions membres

3.3.2.1 `menu_gestion_des_utilisateurs()`

```
void Menu::menu_gestion_des_utilisateurs ( )
```

`Menu::menu_gestion_des_utilisateurs` permet de tester les methodes suivantes: 4.a) Créer la classe `Client` avec ses variables membres (identifiant, prenom, nom, panier d'achat) et ses fonctions getters. 4.b) ajouter un produit au panier d'achat. 4.c) vider le panier d'achat. 4.d) modifier la quantité d'un produit ajouté au panier d'achat 4.e) supprimer un produit du panier d'achat. 4.f) afficher toutes les informations du client (incluant les produits du panier d'achat.

5.a) ajouter un nouveau client au magasin. 5.b) afficher à l'écran tous les clients du magasin. 5.c) afficher à l'écran un client sélectionné par son nom ou son identifiant. 5.d) ajouter un produit au panier d'achat d'un client. 5.e) supprimer un produit au panier d'achat d'un client. 5.f) modifier la quantité d'un produit du panier d'achat d'un client.

La documentation de cette classe a été générée à partir du fichier suivant :

- `C:/temp/mootez/tp_3easystore/menu.h`
- `C:/temp/mootez/tp_3easystore/menu.cpp`

3.4 Référence de la classe Order

Fonctions membres publiques

- **Order** ()
Order::Order constructeur vide de la classe *Order*.
- **Order** (unsigned int orderId, *Client* _client, std::vector< *Product* > _products, string _status)
Order::Oder c'est un constructeur permettant de creer une commande.
- void **setOrderId** (unsigned int value)
Order::setOrderId fonction setter permettant de mettre a jours l'identifiant.
- void **setClient** (const *Client* &value)
Order::setClient fonction seeter permettant de mettre a jours un client.
- void **setStatus** (const string &value)
Order::setStatus fonction setter permettant de mettre a jours le status d'une commande.
- void **setProducts** (const std::vector< *Product* > &value)
Order::setProducts fonction membre permettant de mettre a jours une liste de produits.
- unsigned int **getOrderId** () const
Order::getOrderId fonction getter permettant de retourner l'idnetifiant.
- *Client* **getClient** () const
Order::getClient fonction getter permettant de retourner un objet client.
- string **getStatus** () const
Order::getStatus fonction getter permettant de retourner le status de la commande.
- std::vector< *Product* > **getProducts** () const
Order::getProducts fonction getter permettant de retourner une liste de produits.
- void **addProduct** (*Product* &p)
Order::addProduct fonction membre permettant de d'ajouter un produit a la commande.
- std::string **toString** ()
Order::toString cette fonction permet de surcharger l'opérateur << pour pouvoir afficher toutes les informations de la commande.
- unsigned int **getProductOrderId** (unsigned int key)
Order::getProductOrderId cette methode permet de chercher l'identifiant d'un produit dans une commande.
- string **toFile** ()
Order::toString cette fonction permet de surcharger l'opérateur << pour pouvoir le sauvegarder toutes les informations de la commande dans un fichier.

3.4.1 Documentation des constructeurs et destructeur

3.4.1.1 Order()

```
Order::Order (
    unsigned int orderId,
    Client _client,
    std::vector< Product > _products,
    string _status )
```

Order::Oder c'est un constructeur permettant de creer une commande.

Paramètres

_orderId	Identifiant de la commande
_client	l'objet client de la commande
_products	un tableau de produits achetes par le client
_status	l'etat de la commande pour voir si la commande (Ex : livrée / pas livrée).

3.4.2 Documentation des fonctions membres

3.4.2.1 addProduct()

```
void Order::addProduct (
    Product & p )
```

[Order::addProduct](#) fonction membre permettant de d'ajouter un produit a la commande.

Paramètres

<i>p</i>	
----------	--

3.4.2.2 getCilent()

```
Client Order::getCilent ( ) const
```

[Order::getCilent](#) fonction getter permettant de retourner un objet client.

Renvoie

3.4.2.3 getOrderId()

```
unsigned int Order::getOrderId ( ) const
```

[Order::getOrderId](#) fonction getter permettant de retourner l'identifiant.

Renvoie

identifiant

3.4.2.4 getProductOrderId()

```
unsigned int Order::getProductOrderId (
    unsigned int key )
```

[Order::getProductOrderId](#) cette methode permet de chercher l'identifiant d'un produit dans une commande.

Paramètres

<i>key</i>	l'index de l'article dans la commande
------------	---------------------------------------

Renvoie

l'identifiant du produit dans le magasin

3.4.2.5 getProducts()

```
std::vector< Product > Order::getProducts ( ) const
```

[Order::getProducts](#) fonction getter permettant de retourner une liste de produits.

Renvoie**3.4.2.6 getStatus()**

```
string Order::getStatus ( ) const
```

[Order::getStatus](#) fonction getter permettant de retourner le status de la commande.

Renvoie**3.4.2.7 setCilent()**

```
void Order::setCilent (
    const Client & value )
```

[Order::setCilent](#) fonction setter permettant de mettre a jours un client.

Paramètres

<i>value</i>	
--------------	--

3.4.2.8 setOrderId()

```
void Order::setOrderId (
    unsigned int value )
```

[Order::setOrderId](#) fonction setter permettant de mettre a jours l'identifiant.

Paramètres

<i>value</i>	
--------------	--

3.4.2.9 setProducts()

```
void Order::setProducts (
    const std::vector< Product > & value )
```

[Order::setProducts](#) fonction membre permettant de mettre a jours une liste de produits.

Paramètres

<i>value</i>	
--------------	--

3.4.2.10 setStatus()

```
void Order::setStatus (
    const string & value )
```

[Order::setStatus](#) fonction setter permettant de mettre a jours le status d'une commande.

Paramètres

<i>value</i>	
--------------	--

3.4.2.11 toFile()

```
string Order::toFile ( )
```

[Order::toString](#) cette fonction permet de surcharger l'opérateur << pour pouvoir le sauvegarder toutes les informations de la commande dans un fichier.

Renvoie

3.4.2.12 toString()

```
string Order::toString ( )
```

[Order::toString](#) cette fonction permet de surcharger l'opérateur << pour pouvoir afficher toutes les informations de la commande.

Renvoie

La documentation de cette classe a été générée à partir du fichier suivant :

- C:/temp/mootez/tp_3easystore/order.h
- C:/temp/mootez/tp_3easystore/order.cpp

3.5 Référence de la classe Product

The [Product](#) class permet de répondre à la question: 2.a) Créer la classe [Produit](#) avec ses variables membres, son constructeur et ses fonctions getters. La classe [Product](#) est divisée en trois parties:

```
#include <product.h>
```

Fonctions membres publiques

- **Product** (unsigned int productId, string _titre, string _description, unsigned int _quantiteDisponible, double _prix)
- void **setQuantiteDisponible** (unsigned int quantiteDisponible)
- void **setProductId** (unsigned int value)
- unsigned int **getProductId** () const
- string **getTitre** () const
- string **getDescription** () const
- unsigned int **getQuantiteDisponible** () const
- double **getPrix** () const
- string **toString** ()

3.5.1 Description détaillée

The [Product](#) class permet de répondre à la question: 2.a) Créer la classe [Produit](#) avec ses variables membres, son constructeur et ses fonctions getters. La classe [Product](#) est divisée en trois parties:

- partie des déclarations privées(productId, _titre, _description, _quantiteDisponible et _prix)
- partie des déclarations public
 - contenant les constructeurs (deux constructeurs [Product\(\)](#) et [Product\(unsigned int productId, string _titre, string _description\)](#))
 - les getters et les setters
 - et les fonctions membres (toString)

La documentation de cette classe a été générée à partir du fichier suivant :

- C:/temp/mootez/tp_3easystore/product.h
- C:/temp/mootez/tp_3easystore/product.cpp

3.6 Référence de la classe Produit

Fonctions membres publiques

- void **setTitre** ()
- void **setDescription** ()
- void **setID** ()
- void **setQuantiteDisponible** ()
- void **setPrix** ()
- string **getTitre** ()
- string **getDescription** ()
- string **getID** ()
- int **getQuantiteDisponible** ()
- double **getPrix** ()
- std::string **toString** ()

La documentation de cette classe a été générée à partir du fichier suivant :

- C:/temp/mootez/tp_3easystore/produit.h

Chapitre 4

Documentation des fichiers

4.1 client.h

```
1 #ifndef CLIENT_H
2 #define CLIENT_H
3
4 #include <string>
5 #include <vector>
6 #include "product.h"
7
8 using namespace std;
9
10 class Client
11 {
12 private:
13     unsigned int clientId; // identifiant unique
14     string  FirstName;
15     string  lastName ;
16     std::vector<Product> basket;
17
18 public:
19     //Constructeurs
20     Client();
21     Client(unsigned int _clientId, string _FirstName, string _lastName);
22
23     //Setters
24     void setClientId(unsigned int value);
25     void setFirstName(const string &value);
26     void setLastName(const string &value);
27     void setBasket(const Product &value);
28
29     //Getters
30     std::vector<Product> getBasket() const;
31     unsigned int getClientId() const;
32     string getFirstName() const;
33     string getLastName() const;
34
35     //Fonction membres
36     void clearBasket();
37     void setQuantityForAProductInBasket(const unsigned int &productId, const unsigned int &quantity);
38     void deleteProductFromBasket(const unsigned int &productId);
39     string toStringWithBasket();
40     string toString();
41 };
42
43 #endif // CLIENT_H
```

4.2 magasin.h

```
1 #ifndef MAGASIN_H
2 #define MAGASIN_H
3
```

```

4 #include "product.h"
5 #include "order.h"
6 #include "client.h"
7
8
9 #include <vector>
27 class Magasin
28 {
29
30 private:
31
32     std::vector<Product> _products;
33     std::vector<Client> _clients;
34     std::vector<Order> _orders;
35
36
37 public:
38
39     //Constructeurs
40
41     Magasin();
42     Magasin(Product p, Client c, Order o);
43
44     //Setters
45     void setOrders(const Order &order);
46     void setClients(const Client &client);
47     void setProducts(const Product &product);
48
49
50     //Getters
51     std::vector<Product> products() const;
52     std::vector<Client> clients() const;
53     std::vector<Order> orders() const;
54
55     //Fonction membres
56
57     void newClient(Client *c);
58     int AddProductToBasket(unsigned int _clientId, unsigned int _productId, unsigned int quantity);
59     int clearBasket(unsigned int _clientId);
60     void diplayAllProducts();
61     void diplayProductsByName(string pname);
62     void updateProductsByQuantityByName(string ptitle, unsigned int new_quantity);
63     unsigned int nextProductKey() const;
64     unsigned int nextClientKey() const;
65     unsigned int findClientByName(string fname, string lname) const;
66     unsigned int findProductKeyByName(string _title) const;
67     void newProduct(Product *p);
68     int modifyProductQuantityInBasket(unsigned int _clientId, unsigned int _productId, unsigned int
quantity);
69     int deleteProductFromBasket(unsigned int _clientId, unsigned int _productId);
70     int displayProductOfAClient(unsigned int _clientId);
71     void diplayAllClients();
72     Client findClientById(unsigned int id) const;
73     int validateAnOrder(unsigned int _clientId);
74     unsigned int nextOrderKey() const;
75     Order findOrderByClientIdAndOrderId(unsigned int orderId, unsigned int clientId) const;
76     int updateAnOrderState(unsigned int _clientId, unsigned int _orderId, string newSatate);
77     void diplayAllOrders();
78     std::vector<Order> findOrdersByClientId(unsigned int clientId) const;
79     void displayOrderOfAClient(unsigned int _clientId);
80     void diplayAllOrdersByState(string state);
81     void newOrder(Order *o);
82     Product findProductById(unsigned int id) const;
83     Order findOrderById(unsigned int orderId) const;
84     void updateOrders(unsigned int key, const Order &order);
85     unsigned int productNumberInOrder(unsigned int key);
86 };
87
88 #endif // MAGASIN_H

```

4.3 main.h

```

1 #ifndef MAIN_H
2 #define MAIN_H
3
4 #endif // MAIN_H
5 #include <string>
6 #include "magasin.h"
7
8 using namespace std;
9
10 Magasin *magasin;

```

4.4 menu.h

```
1 #ifndef MENU_H
2 #define MENU_H
3
12 class Menu
13 {
14 public:
15     Menu();
16     void menu();
17     void menu_gestion_du_magasin();
18     void menu_gestion_des_utilisateurs();
19     void menu_gestion_des_commandes();
20 };
21
22 #endif // MENU_H
```

4.5 order.h

```
1 #ifndef ORDER_H
2 #define ORDER_H
3 // #include "product.h"
4 #include "product.h"
5 #include <vector>
6 #include "client.h"
7
8 #include <string>
9
10 class Order
11 {
12 private:
13     unsigned int orderId;
14     Client cilent;
15     std::vector<Product> products;
16     string status;
17
18
19 public:
20     //Constructeurs
21     Order();
22     Order(unsigned int orderId, Client _cilent, std::vector<Product> _products, string _status);
23
24
25     //Setters
26     void setOrderId(unsigned int value);
27     void setCilent(const Client &value);
28     void setStatus(const string &value);
29     void setProducts(const std::vector<Product> &value);
30
31     //Getters
32
33     unsigned int getOrderId() const;
34     Client getCilent() const;
35     string getStatus() const;
36     std::vector<Product> getProducts() const;
37
38
39     //Fonction membres
40     void addProduct(Product &p);
41
42     std::string toString();
43     unsigned int getProductOrderId(unsigned int key);
44     string toFile();
45 };
46
47
48 #endif // ORDER_H
```

4.6 product.h

```
1 #ifndef PRODUCT_H
2 #define PRODUCT_H
3 #include <string>
4
5 using namespace std;
19 class Product
20 {
21 private:
22     unsigned int productId; // identifiant unique
```

```

23     string _titre;
24     string _description;
25     unsigned int _quantiteDisponible;
26     double _prix;
27
28 public:
29     //Constructeurs
30     Product();
31     Product(unsigned int productId, string _titre, string _description,
32             unsigned int _quantiteDisponible, double _prix );
33
34     //Setters
35
36     void setQuantiteDisponible(unsigned int quantiteDisponible);
37     void setProductId(unsigned int value);
38
39     //Getters
40
41     unsigned int getProductId() const;
42     string getTitre() const;
43     string getDescription() const;
44     unsigned int getQuantiteDisponible() const;
45     double getPrix() const;
46
47     //Fonction membres
48
49     string toString();
50
51
52 };
53
54 #endif // PRODUCT_H

```

4.7 produit.h

```

1  #ifndef PRODUIT_H
2  #define PRODUIT_H
3
4  #include <string>
5
6  using namespace std;
7
8  class Produit
9  {
10 private:
11     string _titre;
12     string _description;
13     string _ID;
14     int _quantiteDisponible;
15     double _prix;
16 public:
17     Produit();
18     //Setters
19     void setTitre();
20     void setDescription();
21     void setID();
22     void setQuantiteDisponible();
23     void setPrix();
24     //Getters
25     string getTitre();
26     string getDescription();
27     string getID();
28     int getQuantiteDisponible();
29     double getPrix();
30     //ToString
31     std::string toString();
32
33
34 };
35
36 #endif // PRODUIT_H

```

Index

- addProduct
 - Order, [23](#)
- AddProductToBasket
 - Magasin, [10](#)
- C:/temp/mootez/tp_3easystore/client.h, [29](#)
- C:/temp/mootez/tp_3easystore/magasin.h, [29](#)
- C:/temp/mootez/tp_3easystore/main.h, [30](#)
- C:/temp/mootez/tp_3easystore/menu.h, [31](#)
- C:/temp/mootez/tp_3easystore/order.h, [31](#)
- C:/temp/mootez/tp_3easystore/product.h, [31](#)
- C:/temp/mootez/tp_3easystore/produit.h, [32](#)
- clearBasket
 - Magasin, [10](#)
- Client, [5](#)
 - deleteProductFromBasket, [6](#)
 - getClientId, [6](#)
 - setBasket, [6](#)
 - setQuantityForAProductInBasket, [7](#)
 - toStringWithBasket, [7](#)
- clients
 - Magasin, [11](#)
- deleteProductFromBasket
 - Client, [6](#)
 - Magasin, [11](#)
- displayAllOrdersByState
 - Magasin, [11](#)
- displayProductsByName
 - Magasin, [12](#)
- displayOrderOfAClient
 - Magasin, [12](#)
- displayProductOfAClient
 - Magasin, [13](#)
- findClientById
 - Magasin, [13](#)
- findClientByName
 - Magasin, [13](#)
- findOrderByClientIdAndOrderId
 - Magasin, [14](#)
- findOrderById
 - Magasin, [14](#)
- findOrdersByClientId
 - Magasin, [14](#)
- findProductById
 - Magasin, [15](#)
- findProductKeyByName
 - Magasin, [15](#)
- getCilent
 - Order, [23](#)
- getClientId
 - Client, [6](#)
- getOrderId
 - Order, [23](#)
- getProductOrderId
 - Order, [23](#)
- getProducts
 - Order, [24](#)
- getStatus
 - Order, [24](#)
- Magasin, [7](#)
 - AddProductToBasket, [10](#)
 - clearBasket, [10](#)
 - clients, [11](#)
 - deleteProductFromBasket, [11](#)
 - displayAllOrdersByState, [11](#)
 - displayProductsByName, [12](#)
 - displayOrderOfAClient, [12](#)
 - displayProductOfAClient, [13](#)
 - findClientById, [13](#)
 - findClientByName, [13](#)
 - findOrderByClientIdAndOrderId, [14](#)
 - findOrderById, [14](#)
 - findOrdersByClientId, [14](#)
 - findProductById, [15](#)
 - findProductKeyByName, [15](#)
 - Magasin, [10](#)
 - modifyProductQuantityInBasket, [15](#)
 - newClient, [16](#)
 - newOrder, [16](#)
 - newProduct, [17](#)
 - nextClientKey, [17](#)
 - nextOrderKey, [17](#)
 - nextProductKey, [17](#)
 - productNumberInOrder, [17](#)
 - products, [18](#)
 - setClients, [18](#)
 - setOrders, [18](#)
 - setProducts, [18](#)
 - updateAnOrderState, [19](#)
 - updateOrders, [19](#)
 - updateProductsByQuantityByName, [19](#)
 - validateAnOrder, [20](#)
- Menu, [20](#)
 - menu_gestion_des_utilisateurs, [21](#)
- menu_gestion_des_utilisateurs
 - Menu, [21](#)
- modifyProductQuantityInBasket

- Magasin, [15](#)
- newClient
 - Magasin, [16](#)
- newOrder
 - Magasin, [16](#)
- newProduct
 - Magasin, [17](#)
- nextClientKey
 - Magasin, [17](#)
- nextOrderKey
 - Magasin, [17](#)
- nextProductKey
 - Magasin, [17](#)
- Order, [22](#)
 - addProduct, [23](#)
 - getCilent, [23](#)
 - getOrderId, [23](#)
 - getProductOrderId, [23](#)
 - getProducts, [24](#)
 - getStatus, [24](#)
 - Order, [22](#)
 - setCilent, [24](#)
 - setOrderId, [24](#)
 - setProducts, [25](#)
 - setStatus, [25](#)
 - toFile, [25](#)
 - toString, [25](#)
- Product, [26](#)
- productNumberInOrder
 - Magasin, [17](#)
- products
 - Magasin, [18](#)
- Produit, [27](#)
- setBasket
 - Client, [6](#)
- setCilent
 - Order, [24](#)
- setClients
 - Magasin, [18](#)
- setOrderId
 - Order, [24](#)
- setOrders
 - Magasin, [18](#)
- setProducts
 - Magasin, [18](#)
 - Order, [25](#)
- setQuantityForAProductInBasket
 - Client, [7](#)
- setStatus
 - Order, [25](#)
- toFile
 - Order, [25](#)
- toString
 - Order, [25](#)
- toStringWithBasket
 - Client, [7](#)
- updateAnOrderState
 - Magasin, [19](#)
- updateOrders
 - Magasin, [19](#)
- updateProductsByQuantityByName
 - Magasin, [19](#)
- validateAnOrder
 - Magasin, [20](#)