PIZZMAN – felhasználói dokumentáció

Adminisztrációs szoftver pizzasütödék számára

Program:

A szoftver alapvető feladata a pizzarendelések adatbázisának nyilvántartása, használata. Lehetőséget kínál adminisztrátori, szállítói és felhasználói kezelésre, azaz háromféle profilkör létezik: Admin, Deliverer és User. Az Adminoknak van csak teljeskörű hozzáférése a funkciókhoz. A program elején lehet Customerként regisztrálni, avagy bejelentkezni más minőségben.

Profilkör: User (A következők menüpontok is az alkalmáson belül)

- Felhasználónév és jelszó segítségével bejelentkezik/regisztrál.
- Lekérheti a rendeléseinek állapotát.
- Elindíthat rendelést.
    - Itt lekérheti a pizzákat.
    - Hozzáadhat pizzát. / Eltávolíthat felvett pizzát.
    - Hozzáadhat egy pizzához plusz feltétet.
    - Saját címére, vagy máshova kéri. (Default: saját)
    - Választhat fizetési módot (Default: készpénz)
    - (Leadás dátumát az alkalmazás generálja)
    - Megjegyzés hozzáadása.
    - Elküldheti a rendelést.

Profilkör: Deliverer (Az ő adatait kézzel visszük be az adatbázisba)

- Felhasználónév és jelszó segítségével bejelentkezik.
- Magához vehet leadott rendeléseket.
    - Itt lekérheti a „szállítható” rendeléseket.
    - Rendelésazonosítóval lefoglalhatja a rendelést.
- Visszajelezhet kiszállított rendelésekről.
    - Itt lekérheti a saját kiszállítás alatt lévő rendeléseit.
    - Rendelésazonosítóval visszajelezheti, hogy sikeres volt-e a kiszállítás.

Profilkör: Admin (Az ő adatai be vannak égetve adatbázisba)

- Felhasználónév és jelszó segítségével bejelentkezik.
- Lekérheti az összes létező pizzát.
- Lekérheti az összes kiszállítatlan rendelést.
- Átállíthat rendelést „frissen felvett”-ről „szállítható”-ra.
- Elindíthat rendelést. (Ugyanúgy, mint User)
- Lekérheti egy nap bevételét.
- Létrehozhat pizzát.
    - Elnevezheti a pizzát.
    - Létrehozhat itt új feltétet.

PIZZMAN – programozói dokumentáció

Adminisztrációs szoftver pizzasütödék számára

UML diagram:

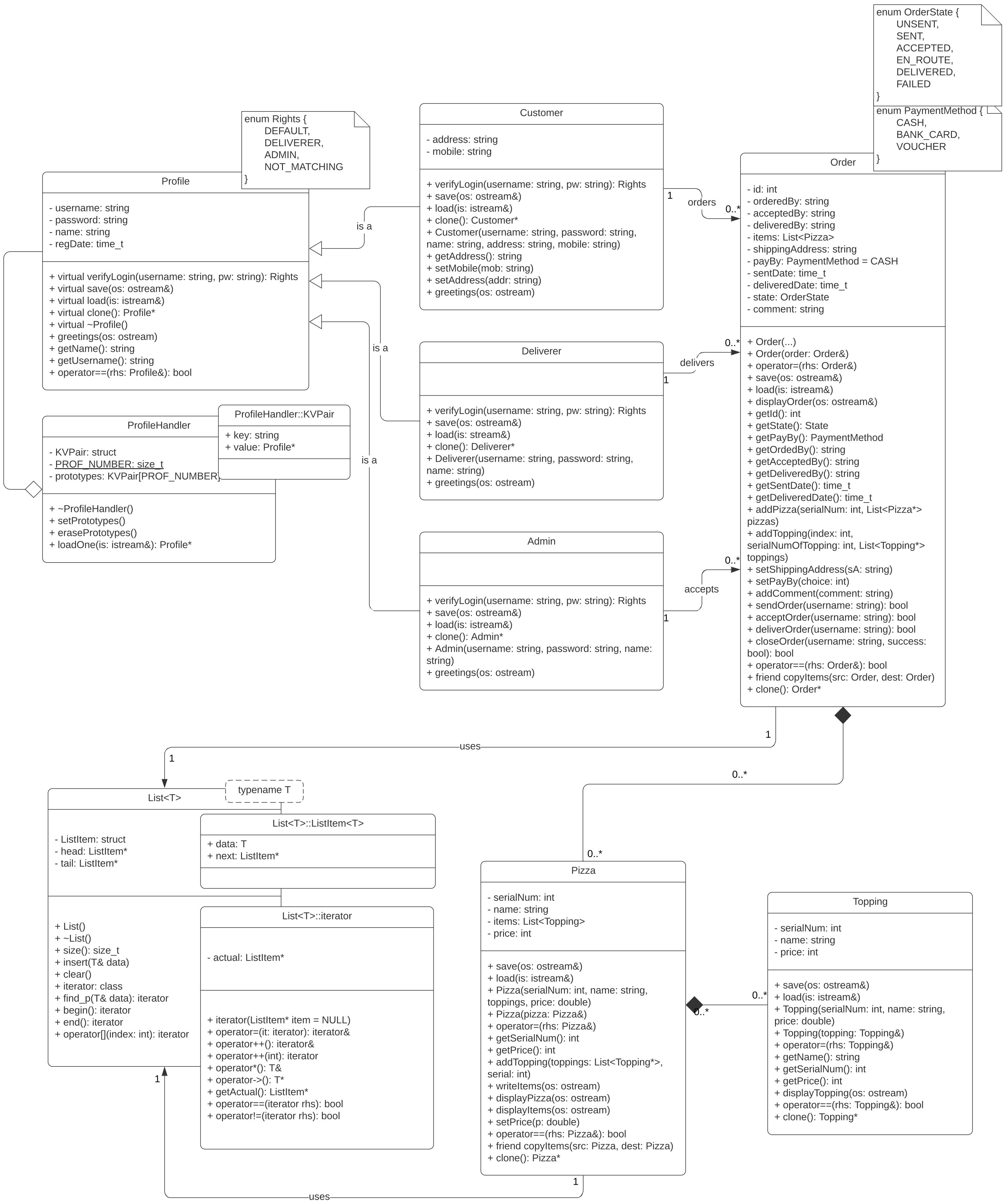
Lásd következő oldal, bele lehet nagyítani…

Megvalósítási ötletek:

- Teszteléshez a memtrace környezet bevetésre kerül.
- A profilok tárolásához és kezeléséhez – mivel azok heterogén kollekcióban tárolódnak – külső forrásból merítek ötletet (https://prog2.cppftw.org/extra_heterogen/). Csupán az én programom statikusan tartalmazza a prototípusokat.
- Bejelentkezés alkalmával a bejelentkezési adatokkal generálunk egy dummy Profile-t, a Profilok listájának find() metódusa pedig a lineáris keresés algoritmusával megszerzi a megfelelő Profile-t, amire ráeresztve az ő (virtuálisan megszerzett) verifyLogin(username, pw) metódusa fog visszatérni azzal (Rights), ami eldönti, milyen hozzáférési jogokat érhet el a bejelentkezett felhasználó.
- A főprogramban globálisan lesz lementve a bejelentkezett felhasználó jogköre (Rights), és listákban a beolvasott profilok (Profile), rendelések (Order), pizzák (Pizza), valamint feltétek (Topping).
- Minden fontos osztálynak van komparátor operátora, főleg, hogy a List find_p() metódusa jól működhessen.
- List néven implementált egy láncolt lista saját bejáró osztállyal, amelyet indexelni is lehet (az indexelés a megfelelő pizza kiválasztásához szükséges új feltét hozzáadásakor).

Fájlkezelés:

Külön fájlokba (táblákba) vannak rendezve: profilok, rendelések, pizzák, feltétek. Minden osztálynak van save() és load() metódusa pontosan ezen fájlokból való kiolvasásra és beolvasásra. A beolvasások az alkalmazás indításakor futnak le, a kiírások pedig a bezáráskor.


enum OrderState {
UNSENT,
SENT,
ACCEPTED,
EN_ROUTE,
DELIVERED,
FAILED
}
enum PaymentMethod {
CASH,
BANK_CARD,
VOUCHER
}
enum Rights {
DEFAULT,
DELIVERER,
ADMIN,
NOT_MATCHING
}
Profile
- username: string
- password: string
- name: string
- regDate: time_t
+ virtual verifyLogin(username: string, pw: string): Rights
+ virtual save(os: ostream&)
+ virtual load(is: istream&)
+ virtual clone(): Profile*
+ virtual ~Profile()
+ greetings(os: ostream)
+ getName(): string
+ getUsername(): string
+ operator==(rhs: Profile&): bool
Customer
- address: string
- mobile: string
+ verifyLogin(username: string, pw: string): Rights
+ save(os: ostream&)
+ load(is: istream&)
+ clone(): Customer*
+ Customer(username: string, password: string, name: string, address: string, mobile: string)
+ getAddress(): string
+ setMobile(mob: string)
+ setAddress(addr: string)
+ greetings(os: ostream)
is a
is a
is a
orders
1
0..*
Order
- id: int
- orderedBy: string
- acceptedBy: string
- deliveredBy: string
- items: List<Pizza>
- shippingAddress: string
- payBy: PaymentMethod = CASH
- sentDate: time_t
- deliveredDate: time_t
- state: OrderState
- comment: string
+ Order(...)
+ Order(order: Order&)
+ operator=(rhs: Order&)
+ save(os: ostream&)
+ load(is: istream&)
+ displayOrder(os: ostream&)
+ getId(): int
+ getState(): State
+ getPayBy(): PaymentMethod
+ getOrdedBy(): string
+ getAcceptedBy(): string
+ getDeliveredBy(): string
+ getSentDate(): time_t
+ getDeliveredDate(): time_t
+ addPizza(serialNum: int, List<Pizza*> pizzas)
+ addTopping(index: int, serialNumOfTopping: int, List<Topping*> toppings)
+ setShippingAddress(sA: string)
+ setPayBy(choice: int)
+ addComment(comment: string)
+ sendOrder(username: string): bool
+ acceptOrder(username: string): bool
+ deliverOrder(username: string): bool
+ closeOrder(username: string, success: bool): bool
+ operator==(rhs: Order&): bool
+ friend copyItems(src: Order, dest: Order)
+ clone(): Order*
Deliverer
+ verifyLogin(username: string, pw: string): Rights
+ save(os: ostream&)
+ load(is: stream&)
+ clone(): Deliverer*
+ Deliverer(username: string, password: string, name: string)
+ greetings(os: ostream)
delivers
1
0..*
Admin
+ verifyLogin(username: string, pw: string): Rights
+ save(os: ostream&)
+ load(is: istream&)
+ clone(): Admin*
+ Admin(username: string, password: string, name: string)
+ greetings(os: ostream)
accepts
1
0..*
ProfileHandler
- KVPair: struct
- PROF_NUMBER: size_t
- prototypes: KVPair[PROF_NUMBER]
+ ~ProfileHandler()
+ setPrototypes()
+ erasePrototypes()
+ loadOne(is: istream&): Profile*
ProfileHandler::KVPair
+ key: string
+ value: Profile*
uses
1
1
0..*
List<T>
typename T
- ListItem: struct
- head: ListItem*
- tail: ListItem*
+ List()
+ ~List()
+ size(): size_t
+ insert(T& data)
+ clear()
+ iterator: class
+ find_p(T& data): iterator
+ begin(): iterator
+ end(): iterator
+ operator[](index: int): iterator
List<T>::ListItem<T>
+ data: T
+ next: ListItem*
List<T>::iterator
- actual: ListItem*
+ iterator(ListItem* item = NULL)
+ operator=(it: iterator): iterator&
+ operator++(): iterator&
+ operator++(int): iterator
+ operator*(): T&
+ operator->(): T*
+ getActual(): ListItem*
+ operator==(iterator rhs): bool
+ operator!=(iterator rhs): bool
0..*
Pizza
- serialNum: int
- name: string
- items: List<Topping>
- price: int
+ save(os: ostream&)
+ load(is: istream&)
+ Pizza(serialNum: int, name: string, toppings, price: double)
+ Pizza(pizza: Pizza&)
+ operator=(rhs: Pizza&)
+ getSerialNum(): int
+ getPrice(): int
+ addTopping(toppings: List<Topping*>, serial: int)
+ writeItems(os: ostream)
+ displayPizza(os: ostream)
+ displayItems(os: ostream)
+ setPrice(p: double)
+ operator==(rhs: Pizza&): bool
+ friend copyItems(src: Pizza, dest: Pizza)
+ clone(): Pizza*
0..*
0..*
Topping
- serialNum: int
- name: string
- price: int
+ save(os: ostream&)
+ load(is: istream&)
+ Topping(serialNum: int, name: string, price: double)
+ Topping(topping: Topping&)
+ operator=(rhs: Topping&)
+ getName(): string
+ getSerialNum(): int
+ getPrice(): int
+ displayTopping(os: ostream)
+ operator==(rhs: Topping&): bool
+ clone(): Topping*
1
1
uses

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

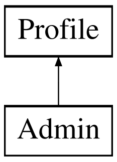
Class Documentation

4.1 Admin Class Reference

Child class of a Profile, defines an administrator's attributes.

```
#include <admin.h>
```

Inheritance diagram for Admin:



Public Member Functions

- **Admin** (const std::string &username, const std::string &pw="", const std::string &name="")
- Rights verifyLogin (const std::string &username, const std::string &pw) const

    Grants the ADMIN rights if the proper username and password is given.
- void greetings (std::ostream &os) const

    Greets logged in user.
- void save (std::ostream &os) const

    ------------------— Persistence -----------------—
- void load (std::istream &is)

    Simple I/O function for object storing.
- Admin ∗ **clone** () const

Additional Inherited Members

4.1.1 Detailed Description

Child class of a Profile, defines an administrator's attributes.

4.1.2 Member Function Documentation

4.1.2.1 save()

```
void Admin::save (
            std::ostream & os ) const  [virtual]
```

------------------– Persistence -----------------–

Simple I/O function for object storing

Implements Profile.

4.1.2.2 verifyLogin()

```
Rights Admin::verifyLogin (
            const std::string & usern,
            const std::string & pw ) const  [virtual]
```

Grants the ADMIN rights if the proper username and password is given.

Parameters

usern	- the username input
pw	- the password input

Returns

ADMIN if the credentials are identical, NOT_MATCHING otherwise

Implements Profile.

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

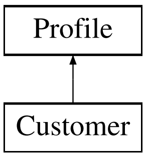
- admin.h
- admin.cpp

4.2 Customer Class Reference

Child class of a Profile, defines a customer's attributes.

```
#include <customer.h>
```

Inheritance diagram for Customer:



Public Member Functions

- **Customer** (std::string username, std::string pw="", std::string name="", std::string addr="", std::string mob="")
- std::string getAddress ()

  Simple getter for getting the address of object.
- void setMobile (const std::string &mob)

  Simple setter for mobile phone number.
- void setAddress (const std::string &addr)

  -------------------— Setters -------------------—
- Rights verifyLogin (const std::string &username, const std::string &pw) const

  Grants the DEFAULT rights if the proper username and password is given.
- void greetings (std::ostream &os) const

  Greets logged in user.
- void save (std::ostream &os) const

  ----------------— End of Setters ----------------—
- void load (std::istream &is)

  Simple I/O function for object storing.
- Customer ∗ **clone** () const

Additional Inherited Members

4.2.1 Detailed Description

Child class of a Profile, defines a customer's attributes.

4.2.2 Member Function Documentation

4.2.2.1 getAddress()

```
std::string Customer::getAddress ( )
```

Simple getter for getting the address of object.

Returns

address

4.2.2.2 save()

```
void Customer::save (
             std::ostream & os ) const  [virtual]
```

----------------— End of Setters ----------------—

-------------------— Persistence -----------------—

Simple I/O function for object storing

Implements Profile.

4.2.2.3 setAddress()

```
void Customer::setAddress (
             const std::string & addr )
```

-------------------— Setters --------------------—

Simple setter for address

Parameters

addr	- the new address

4.2.2.4 setMobile()

```
void Customer::setMobile (
             const std::string & mob )
```

Simple setter for mobile phone number.

Parameters

mob	- the new mobile phone number

4.2.2.5 verifyLogin()

```
Rights Customer::verifyLogin (
             const std::string & usern,
             const std::string & pw ) const  [virtual]
```

Grants the DEFAULT rights if the proper username and password is given.

Parameters

usern	- the username input
pw	- the password input

Returns

DEFAULT if the credentials are identical, NOT_MATCHING otherwise

Implements Profile.

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

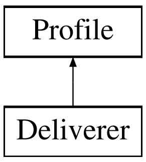
- customer.h
- customer.cpp

4.3 Deliverer Class Reference

Child class of a Profile, defines a deliverer's attributes.

`#include <deliverer.h>`

Inheritance diagram for Deliverer:



Public Member Functions

- **Deliverer** (std::string username, std::string pw="", std::string name="")
- Rights verifyLogin (const std::string &username, const std::string &pw) const

  Grants the DELIVERER rights if the proper username and password is given.
- void greetings (std::ostream &os) const

  Greets logged in user.
- void save (std::ostream &os) const

  ------------------— Persistence -----------------—
- void load (std::istream &is)

  Simple I/O function for object storing.
- Deliverer ∗ **clone** () const

Additional Inherited Members

4.3.1 Detailed Description

Child class of a Profile, defines a deliverer's attributes.

4.3.2 Member Function Documentation

4.3.2.1 save()

```
void Deliverer::save (
            std::ostream & os ) const  [virtual]
```

------------------— Persistence -----------------—

Simple I/O function for object storing

Implements Profile.

4.3.2.2 verifyLogin()

```
Rights Deliverer::verifyLogin (
            const std::string & usern,
            const std::string & pw ) const  [virtual]
```

Grants the DELIVERER rights if the proper username and password is given.

Parameters

usern	- the username input
pw	- the password input

Returns

DELIVERER if the credentials are identical, NOT_MATCHING otherwise

Implements Profile.

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

- deliverer.h
- deliverer.cpp

4.4 List< T >::iterator Class Reference

lovely iterator for our List

```
#include <list.hpp>
```

Public Member Functions

- **iterator** (const List &l)
- **iterator** (const iterator &iter)
- void **operator=** (const iterator &rhs)
- ListItem ∗ **getActual** () const
- iterator & **operator++** ()
- iterator **operator++** (int)
- T & **operator∗** ()
- T ∗ **operator->** ()
- bool **operator==** (const iterator &rhs) const
- bool **operator!=** (const iterator &rhs) const

4.4.1 Detailed Description

template<typename T>
class List< T >::iterator

lovely iterator for our List

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

- list.hpp

4.5 List< T > Class Template Reference

Reimplementing the template of a guarded list with its typical methods Other spicy methods included.

Classes

- class iterator

  lovely iterator for our List

Public Member Functions

- size_t size () const

  returns the number of list items in list
- void insert (const T &data)

  inserts an item into the list (at the tail)
- void clear ()

  frees all the items in list
- iterator find_p (const T &data)

  method is useful only for heterogeneous stores (List<X∗>) Upon comparing, the method dereferences both of the items in order to Work with the dedicated operator== between the two items.
- iterator begin () const

  Typical begin()
- iterator end () const

  Typical end()
- iterator operator[ ] (const size_t &index)

  an interesting method that works similarly as find_p Iterates through list to find the searched "indexed" item

4.5.1 Detailed Description

template<typename T>
class List< T >

Reimplementing the template of a guarded list with its typical methods Other spicy methods included.

4.5.2 Member Function Documentation

4.5.2.1 begin()

```
template<typename T >
iterator List< T >::begin ( ) const  [inline]
```

Typical begin()

Returns

iterator with head item

4.5.2.2 end()

```
template<typename T >
iterator List< T >::end ( ) const  [inline]
```

Typical end()

Returns

iterator with sentinel (blank) item

4.5.2.3 find_p()

```
template<typename T >
iterator List< T >::find_p (
             const T & data )  [inline]
```

method is useful only for heterogeneous stores (List<X∗>) Upon comparing, the method dereferences both of the items in order to Work with the dedicated operator== between the two items.

See also

pizza.h, order.h, topping.h, profile.h's operator==

Returns

an iterator with the found item

4.5.2.4 operator[]()

```
template<typename T >
iterator List< T >::operator[] (
             const size_t & index )  [inline]
```

an interesting method that works similarly as find_p Iterates through list to find the searched "indexed" item

Returns

an iterator with the found item

Exceptions

if	overindexing happens

4.5.2.5 size()

```
template<typename T >
size_t List< T >::size ( ) const  [inline]
```

returns the number of list items in list

Returns

number

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

- list.hpp

4.6 Order Class Reference

Model for order.

```
#include <order.h>
```

Public Member Functions

- **Order** (int id=-1, std::string orderedBy="", std::string acceptedBy="", std::string deliveredBy="", std::string shippingAddress="", PaymentMethod payBy=CASH, time_t sentDate=0, time_t deliveredDate=0, OrderState state=UNSENT, std::string comment="")
- Order (const Order &order)

  Copy constructor Highly dependent on assign operator.
- Order & operator= (const Order &order)

  Assign operator for Order.
- void save (std::ostream &os) const

  persistence
- bool load (std::istream &is, List$<$ Topping $*$ $>$ &toppings)

  Simple I/O function for object storing.
- int getId () const

  getters
- std::string getOrderedBy () const

  Simple getter for getting username of orderer.
- std::string getAcceptedBy () const

  Simple getter for getting username of admin that accepted order.
- std::string getDeliveredBy () const

  Simple getter for getting username of deliverer.
- std::string getShippingAddress () const

  Simple getter for getting shipping address of object.
- PaymentMethod getPayBy () const

  Simple getter for getting payment method of object.
- time_t getSentDate () const

  Simple getter for getting date of sending in.
- time_t getDeliveredDate () const

  Simple getter for getting date of delivery.
- OrderState getState () const

  Simple getter for getting state of object.
- std::string getComment () const

  Simple getter for getting comment of object.
- void displayOrder (std::ostream &os) const

  Displays every important information of the ordering onto a stream.
- void addPizza (const int &serialNum, List$<$ Pizza $*$ $>$ &pizzas)

  setters
- void addTopping (const size_t &index, const int &serialNumOfTopping, List$<$ Topping $*$ $>$ &toppings)

  Adding a topping into a pizza item And increments price of the pizza.
- void setShippingAddress (const std::string &sA)

  Simple setter for shipping address.
- void setPayBy (const int &choice)

  Simple setter for payment method accepts a simple int, will cast it.
- void setComment (const std::string &comment)

  Simple setter for comment.
- void sendOrder (const std::string &username)

  state modifiers
- void acceptOrder (const std::string &username)

  Sets state to ACCEPTED, ready for delivery, accepter's username.
- void deliverOrder (const std::string &username)

  Sets state to EN_ROUTE, deliverer's username.

- void closeOrder (bool success, const std::string &comment="")

  Sets state to DELIVERED or FAILED, delivery arrival's date.
- bool operator== (const Order &rhs) const

  Comparator for orders.
- Order ∗ clone () const

  Cloning method.

Friends

- void copyItems (const Order &orderSource, Order &orderDest)

  Copying object's items into other one.

4.6.1 Detailed Description

Model for order.

See also

private data items is a List of Pizza

4.6.2 Member Function Documentation

4.6.2.1 acceptOrder()

```
void Order::acceptOrder (
             const std::string & username )
```

Sets state to ACCEPTED, ready for delivery, accepter's username.

Parameters

username	- admin's username that accepted order

4.6.2.2 addPizza()

```
void Order::addPizza (
             const int & serial,
             List< Pizza * > & pizzas )
```

setters

------------------— End of Getters ------------------—

---------------------— Setters ----------------------—

Appends a pizza from the global pizzas to the order list

Parameters

allItems	- points to the List of all the global pizzas
serialNum	- identifies the pizza, which should be appended to order list from global pizzas

Exceptions

if	there is indexing problem

4.6.2.3 addTopping()

```
void Order::addTopping (
            const size_t & index,
            const int & serialNumOfTopping,
            List< Topping * > & toppings )
```

Adding a topping into a pizza item And increments price of the pizza.

Exceptions

if	there is indexing problem

4.6.2.4 clone()

```
Order * Order::clone ( ) const
```

Cloning method.

Returns

a pointer with clone object

4.6.2.5 closeOrder()

```
void Order::closeOrder (
            bool success,
            const std::string & comment = "" )
```

Sets state to DELIVERED or FAILED, delivery arrival's date.

Parameters

success	- true if successful delivery, false if failed
comment	- sets comment

4.6.2.6 deliverOrder()

```
void Order::deliverOrder (
             const std::string & username )
```

Sets state to EN_ROUTE, deliverer's username.

Parameters

username	- deliverer's username

4.6.2.7 displayOrder()

```
void Order::displayOrder (
             std::ostream & os ) const
```

Displays every important information of the ordering onto a stream.

Parameters

os	- the stream to write onto

4.6.2.8 getId()

```
int Order::getId ( ) const
```

getters

---------------— End of Persistence ---------------—

--------------------— Getters ---------------------—

Simple getter for getting id of object

4.6.2.9 getPayBy()

```
PaymentMethod Order::getPayBy ( ) const
```

Simple getter for getting payment method of object.

Returns

a PaymentMethod (will have to static_cast)

4.6.2.10 getState()

```
OrderState Order::getState ( ) const
```

Simple getter for getting state of object.

Returns

a OrderState (will have to static_cast)

4.6.2.11 operator=()

```
Order & Order::operator= (
             const Order & rhs )
```

Assign operator for Order.

See also

Used mainly by copy constructor

Returns

this object by reference

4.6.2.12 operator==()

```
bool Order::operator== (
             const Order & rhs ) const
```

Comparator for orders.

Returns

true if their ids are identical (it's their unique key)

See also

List<T>.find(const T& data)

4.6.2.13 save()

```
void Order::save (
            std::ostream & os ) const
```

persistence

-------------------— Persistence -------------------—

Simple I/O function for object storing

4.6.2.14 sendOrder()

```
void Order::sendOrder (
            const std::string & username )
```

state modifiers

------------------— End of Setters ------------------—

------------------— State modifiers ------------------—

Sets state to SENT, date of sending and sender username

Parameters

username	- sender's username

4.6.3 Friends And Related Function Documentation

4.6.3.1 copyItems

```
void copyItems (
            const Order & orderSource,
            Order & orderDest )  [friend]
```

Copying object's items into other one.

Parameters

orderSource	- source, that's items will get copied
orderDest	- destination, into the items will get copied

See also

Assign operator for use

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

- order.h
- order.cpp

4.7 Pizza Class Reference

Model for pizza.

`#include <pizza.h>`

Public Member Functions

- void save (std::ostream &os) const

  -------------------— Persistence -------------------—
- bool load (std::istream &is, List< Topping * > &toppings)

  Simple I/O function for object storing.

- **Pizza** (const int &serialNum=-1, const std::string &name="", const int &price=0)
- Pizza (const Pizza &pizza)

  Copy constructor Depends highly on assign operator.
- Pizza & operator= (const Pizza &rhs)

  Assign operator for Pizza.
- int getSerialNum () const

  ---------------— End of Persistence ---------------—
- std::string getName () const

  Getting name of object.
- int getPrice () const

  Getting price of object.
- void displayItems (std::ostream &os) const

  Displays pizza's toppings onto a stream.
- void displayPizza (std::ostream &os) const

  Displays pizza's all details onto a stream.
- void writeItems (std::ostream &os) const

  Writes items onto a file stream.
- bool addTopping (List< Topping ∗ > &toppings, const int &serial)

  -----------------— End of Getters -----------------—
- void setPrice (const int &p)

  Admin can reset the price to an amount they want.
- bool operator== (const Pizza &rhs) const

  -----------------— End of Setters -----------------—
- Pizza ∗ clone () const

  Cloning method.

Friends

- void copyItems (const Pizza &pizzaSource, Pizza &pizzaDest)

  Copying object's items into other one (overload for Pizza)

4.7.1 Detailed Description

Model for pizza.

See also

private data items is a List of Toppings

4.7.2 Member Function Documentation

4.7.2.1 addTopping()

```
bool Pizza::addTopping (
              List< Topping ∗ > & toppings,
              const int & serial )
```

-----------------— End of Getters -----------------—
--------------------— Setters --------------------—
Inserts the serial of topping to the list of toppings in the pizza Also, the price is incremented by the price of the new topping.

Parameters

serial	- the serialNum of topping to be inserted

Returns

true, if insertion was successful

4.7.2.2 clone()

```
Pizza * Pizza::clone ( ) const
```

Cloning method.

Returns

a pointer with clone object

4.7.2.3 getSerialNum()

```
int Pizza::getSerialNum ( ) const
```

----------------— End of Persistence ----------------—

---------------------— Getters ----------------------—

Getting serialNum of object

4.7.2.4 operator=()

```
Pizza & Pizza::operator= (
                const Pizza & rhs )
```

Assign operator for Pizza.

See also

Used by order.h

4.7.2.5 operator==()

```
bool Pizza::operator== (
                const Pizza & rhs ) const
```

-------------------— End of Setters -------------------—

Comparator for pizzas

Returns

true if their serialNums are identical (it's their unique key)

See also

List<T>.find(const T& data)

4.7.2.6 save()

```
void Pizza::save (
                std::ostream & os ) const
```

--------------------— Persistence --------------------—

Simple I/O function for object storing

4.7.2.7 setPrice()

```
void Pizza::setPrice (
                const int & p )
```

Admin can reset the price to an amount they want.

Parameters

p	- price to be set to

4.7.3 Friends And Related Function Documentation

4.7.3.1 copyItems

```
void copyItems (
            const Pizza & pizzaSource,
            Pizza & pizzaDest )  [friend]
```

Copying object's items into other one (overload for Pizza)

Parameters

pizzaSource	- source, that's items will get copied
pizzaDest	- destination, into the items will get copied

See also

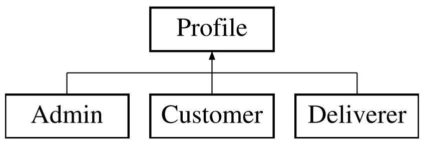
Used by assign operator ...

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

- pizza.h
- pizza.cpp

4.8 Profile Class Reference

Abstract parent class for profiles.
`#include <profile.h>`
Inheritance diagram for Profile:



Public Member Functions

- **Profile** (const std::string &username, const std::string &pw="", const std::string &name="")
- std::string getUsername () const

  Simple getter for getting the username of object.
- std::string getName () const

  Simple getter for getting the name of object.
- bool operator== (const Profile &rhs) const

  Comparator for profiles.
- virtual Rights **verifyLogin** (const std::string &username, const std::string &pw) const =0
- virtual void **greetings** (std::ostream &os) const =0
- virtual void **save** (std::ostream &os) const =0
- virtual void **load** (std::istream &is)=0
- virtual Profile ∗ **clone** () const =0

Protected Attributes

- std::string **username**
- std::string **password**
- std::string **name**
- time_t regDate

  Will be assigned automatically upon creation.

4.8.1 Detailed Description

Abstract parent class for profiles.

4.8.2 Member Function Documentation

4.8.2.1 getName()

```
std::string Profile::getName ( ) const
```
Simple getter for getting the name of object.

Returns

name

4.8.2.2 getUsername()

```
std::string Profile::getUsername ( ) const
```
Simple getter for getting the username of object.

Returns

username

4.8.2.3 operator==()

```
bool Profile::operator== (
             const Profile & rhs ) const
```
Comparator for profiles.

Returns

true if their usernames are identical (it's their unique key)

See also

List<T>.find(const T& data)

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

- profile.h
- profile.cpp

4.9 ProfileHandler Class Reference

A helping class to handle persistence of heterogeneous store of Profile∗-s Should need refactor upon new Profile child implementation.

```
#include <profile_handler.h>
```

Public Member Functions

- Profile ∗ loadOne (std::istream &is)

  Finds the prototype by the key read from the stream.
- void setPrototypes ()

  Loading up static items of prototypes.
- void erasePrototypes ()

  Deletes allocated memory for static members.

4.9.1 Detailed Description

A helping class to handle persistence of heterogeneous store of Profile∗-s Should need refactor upon new Profile child implementation.

4.9.2 Member Function Documentation

4.9.2.1 loadOne()

```
Profile ∗ ProfileHandler::loadOne (
            std::istream & is )
```

Finds the prototype by the key read from the stream.

Returns

a pointer to the found prototype's value

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

- profile_handler.h
- profile_handler.cpp

4.10 Topping Class Reference

Model for topping.

```
#include <topping.h>
```

Public Member Functions

- void save (std::ostream &os)

  -------------------— Persistence -------------------—
- bool load (std::istream &is)

  Simple I/O function for object storing.
- **Topping** (const int &serialNum=-1, const std::string &name="", const int &price=0)
- Topping (const Topping &topping)

  Copy constructor for Topping Highly dependent of assign operator.
- Topping & operator= (const Topping &topping)

  ----------------— End of Persistence ----------------—
- void displayTopping (std::ostream &os, bool showPrice) const

  --------------------— Getters --------------------—
- std::string getName () const

  Simple getter for name of topping.
- int getSerialNum () const

  Simple getter for serialNum.
- int getPrice () const

Simple getter for price of topping.
- bool operator== (const Topping &rhs) const

  -----------------—— End of Getters ------------------——
- Topping ∗ clone () const

  Cloning method.

4.10.1 Detailed Description

Model for topping.

4.10.2 Member Function Documentation

4.10.2.1 clone()

```
Topping * Topping::clone ( ) const
```
Cloning method.

Returns

a pointer with clone object

4.10.2.2 displayTopping()

```
void Topping::displayTopping (
             std::ostream & os,
             bool showPrice ) const
```
---------------------—— Getters ----------------------——
Displays topping's details onto a stream

Parameters

showPrice	- if true, write the price to the stream as well

4.10.2.3 getName()

```
std::string Topping::getName ( ) const
```
Simple getter for name of topping.

Returns

name

4.10.2.4 getPrice()

```
int Topping::getPrice ( ) const
```
Simple getter for price of topping.

Returns

price

4.10.2.5 getSerialNum()

```
int Topping::getSerialNum ( ) const
```

Simple getter for serialNum.

Returns

serialNum

4.10.2.6 load()

```
bool Topping::load (
              std::istream & is )
```

Simple I/O function for object storing.

Returns

true if stream reading was a success

4.10.2.7 operator=()

```
Topping & Topping::operator= (
              const Topping & rhs )
```

----------------— End of Persistence ----------------—

Assign operator for Topping

Returns

this object by reference

See also

Used by copy constructor

4.10.2.8 operator==()

```
bool Topping::operator== (
              const Topping & rhs ) const
```

-------------------— End of Getters ------------------—

Comparator for toppings

Returns

true if their serialNum are identical (it's their unique key)

See also

List<T>.find(const T& data)

4.10.2.9 save()

```
void Topping::save (
              std::ostream & os )
```

--------------------— Persistence --------------------—

Simple I/O function for object storing

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

- topping.h
- topping.cpp

Chapter 5

File Documentation

5.1 admin.cpp File Reference

Admin class definitions.

```
#include "admin.h"
#include "profile.h"
#include "usefulio.hpp"
#include <cstdlib>
#include <iostream>
#include <string>
```

5.1.1 Detailed Description

Admin class definitions.

5.2 admin.h File Reference

Admin class declaration.

```
#include <string>
#include <iostream>
#include "profile.h"
```

Classes

- class Admin

  Child class of a Profile, defines an administrator's attributes.

5.2.1 Detailed Description

Admin class declaration.

5.3 customer.cpp File Reference

Customer class definitions.

```
#include "customer.h"
#include "profile.h"
#include "usefulio.hpp"
#include <cstdlib>
#include <iostream>
#include <string>
```

5.3.1 Detailed Description

Customer class definitions.

5.4 customer.h File Reference

Customer class declaration.

```
#include <string>
#include <iostream>
#include "profile.h"
```

Classes

- class Customer

  Child class of a Profile, defines a customer's attributes.

5.4.1 Detailed Description

Customer class declaration.

5.5 deliverer.cpp File Reference

Deliverer class definitions.

```
#include "deliverer.h"
#include "profile.h"
#include "usefulio.hpp"
#include <cstdlib>
#include <iostream>
#include <string>
```

5.5.1 Detailed Description

Deliverer class definitions.

5.6 deliverer.h File Reference

Deliverer class declaration.

```
#include <string>
#include <iostream>
#include "profile.h"
```

Classes

- class Deliverer

  Child class of a Profile, defines a deliverer's attributes.

5.6.1 Detailed Description

Deliverer class declaration.

5.7 list.hpp File Reference

List class declaration.

```
#include <cstdlib>
#include <stdexcept>
```

Classes

- class List< T >

  Reimplementing the template of a guarded list with its typical methods Other spicy methods included.
- class List< T >::iterator

  lovely iterator for our List

5.7.1 Detailed Description

List class declaration.

5.8 order.cpp File Reference

Order class definitions Contains: +copyItems() friend function.

```
#include "order.h"
#include "list.hpp"
#include "usefulio.hpp"
#include <cstdlib>
#include <iostream>
#include <string>
#include <ctime>
#include <stdexcept>
```

Typedefs

- typedef List< Pizza >::iterator ItemIter

  Used for items' List's iterator.
- typedef List< Topping $*$ >::iterator TopIter

  Used for toppings (heterostore) list's iterator.
- typedef List< Pizza $*$ >::iterator PizzaIter

  Used for pizzas (heterostore) list's iterator.

Functions

- void copyItems (const Order &orderSource, Order &orderDest)

  Copying object's items into other one.

5.8.1 Detailed Description

Order class definitions Contains: +copyItems() friend function.

5.8.2 Typedef Documentation

5.8.2.1 ItemIter

`ItemIter`
Used for items' List's iterator.
Used for Pizza's own items List iterator.

5.8.3 Function Documentation

5.8.3.1 copyItems()

```
void copyItems (
            const Order & orderSource,
            Order & orderDest )
```

Copying object's items into other one.

Parameters

orderSource	- source, that's items will get copied
orderDest	- destination, into the items will get copied

See also

Assign operator for use

5.9 order.h File Reference

Order class declaration.

```
#include <iostream>
#include <string>
#include <ctime>
#include "pizza.h"
#include "list.hpp"
```

Classes

- class Order

  Model for order.

Enumerations

- enum OrderState {
  UNSENT, **SENT**, **ACCEPTED**, **EN_ROUTE**,
  DELIVERED, **FAILED** }

  Enumerator for defining the state of the order.
- enum **PaymentMethod** { **CASH**, **BANK_CARD**, **VOUCHER** }

5.9.1 Detailed Description

Order class declaration.

5.10 pizza.cpp File Reference

Pizza class definitions Contains: +copyItems() friend function, +loadPizzas() function.

```
#include "pizza.h"
#include "list.hpp"
#include "topping.h"
#include "usefulio.hpp"
#include <cstdlib>
#include <iostream>
#include <string>
#include <stdexcept>
```

Typedefs

- typedef List< Topping >::iterator **ItemIter**
- typedef List< Topping ∗ >::iterator **TopIter**

Functions

- void copyItems (const Pizza &pizzaSource, Pizza &pizzaDest)

  Copying object's items into other one (overload for Pizza)
- bool loadPizzas (List< Pizza ∗ > &pizzas, List< Topping ∗ > &toppings, std::istream &is)

  loading up pizzas with data from give file

5.10.1 Detailed Description

Pizza class definitions Contains: +copyItems() friend function, +loadPizzas() function.

5.10.2 Function Documentation

5.10.2.1 copyItems()

```
void copyItems (
            const Pizza & pizzaSource,
            Pizza & pizzaDest )
```

Copying object's items into other one (overload for Pizza)

Parameters

pizzaSource	- source, that's items will get copied
pizzaDest	- destination, into the items will get copied

See also

Used by assign operator ...

5.10.2.2 loadPizzas()

```
bool loadPizzas (
            List< Pizza ∗ > & pizzas,
            List< Topping ∗ > & toppings,
            std::istream & is )
```

loading up pizzas with data from give file

Returns

true if loading was successful

5.11 pizza.h File Reference

Pizza class declaration.

```
#include <iostream>
#include <string>
#include "list.hpp"
#include "topping.h"
```

Classes

- class Pizza

  Model for pizza.

Functions

- bool loadPizzas (List< Pizza ∗ > &pizzas, List< Topping ∗ > &toppings, std::istream &is)

  loading up pizzas with data from give file

5.11.1 Detailed Description

Pizza class declaration.

5.11.2 Function Documentation

5.11.2.1 loadPizzas()

```
bool loadPizzas (
            List< Pizza * > & pizzas,
            List< Topping * > & toppings,
            std::istream & is )
```

loading up pizzas with data from give file

Returns

true if loading was successful

5.12 pizzman_main.cpp File Reference

Realizing the use of program and/or testing.

```
#include <cstdlib>
#include <string>
#include <iostream>
#include <vector>
#include <stdexcept>
#include "memtrace.h"
#include "admin.h"
#include "profile.h"
#include "customer.h"
#include "deliverer.h"
#include "list.hpp"
#include "pizza.h"
```

```
#include "order.h"
#include "topping.h"
#include "motor_functions.hpp"
#include "menus.hpp"
#include "catch.hpp"
```

Macros

- #define TESTING

  Comment these to run live mode.
- #define **CATCH_CONFIG_MAIN**

Functions

- SCENARIO ("Lists have size and can be iterated and indexed")

  The memtrace error comes from catch.hpp when testing :(.

5.12.1 Detailed Description

Realizing the use of program and/or testing.

5.13 profile.cpp File Reference

Profile abstract class definitions.

```
#include "profile.h"
#include <cstdlib>
#include <iostream>
#include <string>
```

5.13.1 Detailed Description

Profile abstract class definitions.

5.14 profile.h File Reference

Profile abstract class declaration.

```
#include <string>
#include <iostream>
#include <ctime>
```

Classes

- class Profile

  Abstract parent class for profiles.

Enumerations

- enum Rights { **DEFAULT**, **DELIVERER**, **ADMIN**, **NOT_MATCHING** }

  Enumerator for defining the payment method of the order.

5.14.1 Detailed Description

Profile abstract class declaration.

5.14.2 Enumeration Type Documentation

5.14.2.1 Rights

```
enum Rights
```

Enumerator for defining the payment method of the order.
Enumerator for verifyLogin to return the result of found profile.

5.15 profile_handler.cpp File Reference

ProfileHandler class definitions.

```
#include "profile_handler.h"
#include "admin.h"
#include "customer.h"
#include "deliverer.h"
#include <cstdlib>
#include <iostream>
#include <string>
```

5.15.1 Detailed Description

ProfileHandler class definitions.

5.16 profile_handler.h File Reference

ProfileHandler class declaration.

```
#include <iostream>
#include "profile.h"
```

Classes

- class ProfileHandler

  A helping class to handle persistence of heterogeneous store of Profile∗-s Should need refactor upon new Profile child implementation.

5.16.1 Detailed Description

ProfileHandler class declaration.

5.17 topping.cpp File Reference

Topping class definitions.

```
#include "topping.h"
#include "usefulio.hpp"
#include <stdexcept>
```

5.17.1 Detailed Description

Topping class definitions.

5.18 topping.h File Reference

Topping class declaration.

```
#include <iostream>
#include <string>
```

Classes

- class Topping

  Model for topping.

5.18.1 Detailed Description

Topping class declaration.

Index