MainAssignment Reference Manual

Generated by Doxygen 1.4.5

Thu Dec 14 07:45:07 2017

Contents

Chapter 1

MainAssignment Hierarchical Index

1.1 MainAssignment Class Hierarchy

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

Chapter 2

MainAssignment Data Structure Index

2.1 MainAssignment Data Structures

Here are the data structures with brief descriptions:

Chapter 3

MainAssignment Data Structure Documentation

3.1 Afhending Class Reference

Functional class for AfhendingUI.

`#include <Afhending.h>`

Public Member Functions

- Afhending ()

  Load a directory containing all customers waiting for and order.

- void setAfhendingLocation (char currentLocation[32])

  Set location of baker, and filter customers vector.

- vector< PizzaLocations > **getPizzaLocations** ()
- vector< Client > **getCustomerVec** ()
- string **getAfhendingLocation** ()
- vector< Pizza > getOrderVec (unsigned int customersVecNumber)
- void deliverOrder (int customersVecNumber)

3.1.1 Detailed Description

Functional class for AfhendingUI.

3.1.2 Member Function Documentation

3.1.2.1 void Afhending::deliverOrder (int *customersVecNumber*)

EFTIR AD COMMENTA

Parameters:
customersVecNumber is the position of customer in customersVec

3.1.2.2 vector< Pizza > Afhending::getOrderVec (unsigned int *customersVecNumber*)

Read orders from file for a customer at specific position in customersVec

Parameters:
customersVecNumber is the position of customer in customersVec

3.1.2.3 void Afhending::setAfhendingLocation (char *currentLocation*[32])

Set location of baker, and filter customers vector.

Find all customers from customerlist.dat that are from a specific location and whose orders have been finished, and are ready for a delivery

Parameters:
currentLocation is the location of delivery

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

- Afhending.h
- Afhending.cpp

3.2 AfhendingUI Class Reference

User interface for delivering finished orders.

`#include <AfhendingUI.h>`

Public Member Functions

- void **main** ()
- bool pickLocation ()

  choose location of baker

- void **displayAllCustomers** (bool show)
- void **displayCustomerOrder** (unsigned int customerNumber)

3.2.1 Detailed Description

User interface for delivering finished orders.

3.2.2 Member Function Documentation

3.2.2.1 bool AfhendingUI::pickLocation ()

choose location of baker

Returns:
true if there are customers available, else false

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

- AfhendingUI.h
- AfhendingUI.cpp

3.3 Baker Class Reference

Functional class for BakerUI.

`#include <Baker.h>`

Public Member Functions

- void setBakerLocation (char currentLocation[32])

  Set baker location and customize vectors.

- vector< Client > **getCustomerVec** ()
- vector< Client > **getCustomersVecInProgress** ()
- vector< Client > **getCustomersVecDueProgress** ()
- vector< Pizza > **getOrderVec** (unsigned int customersVecNumber)
- vector< Pizza > **getCustomersOrderDueProgress** (unsigned int customerNumber)
- vector< Pizza > **getCustomersOrderInProgress** (unsigned int customerNumber)
- vector< PizzaLocations > **getPizzaLocations** ()
- string **getBakerLocation** ()
- void workOnOrder (unsigned int customersVecNumber)

  Set customer order as in progress.

- void finishOrder (unsigned int customerID)

  Set customer order as.

3.3.1 Detailed Description

Functional class for BakerUI.

Class can set customer order as being worked on and finished working on.

3.3.2 Member Function Documentation

3.3.2.1 void Baker::finishOrder (unsigned int *customerID*)

Set customer order as.

Move customer from customersVecInProgress and set status as finished. Rewrite file containing customer order with updated client class

Parameters:

customerID is the position of customer to update in vector

3.3.2.2 void Baker::setBakerLocation (char *currentLocation*[32])

Set baker location and customize vectors.

Function that set's baker location and sort's orders into vector's depending on order status 'in progress' or 'due progress'

Parameters:

currentLocation is baker's location

3.3.2.3 void Baker::workOnOrder (unsigned int *customersVecNumber*)

Set customer order as in progress.

Move customer from customersVecDueProgress to customersVecInProgress and rewrite file containing customer order with updated client class

Parameters:

customersVecNumber is the position of customer to update in vector

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

- Baker.h
- Baker.cpp

3.4 BakerUI Class Reference

User interface for setting customers order as in progress and/or finished.

`#include <BakerUI.h>`

Public Member Functions

- void **main** ()
- void **displayAllOrders** ()
- void **displayCustomerDueProgress** ()
- void **displayCustomerInProgress** ()
- void **displayCustomerDueProgressOrder** (unsigned int customerNumber)
- void **displayCustomerInProgressOrder** (unsigned int customerNumber)
- void **chooseSeeAllOrders** ()
- void **chooseSeeDueOrders** ()
- void **chooseSeeInProgressOrders** ()
- bool pickLocation ()

  Choose location of baker.

3.4.1 Detailed Description

User interface for setting customers order as in progress and/or finished.

3.4.2 Member Function Documentation

3.4.2.1 bool BakerUI::pickLocation ()

Choose location of baker.

Returns:
false if there are no pending customers, else true

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

- BakerUI.h
- BakerUI.cpp

3.5 Client Class Reference

All information about customer.

`#include <Client.h>`

Data Fields

- char **name** [64]
- char **address** [32]
- int **addressNumber**
- char **comment** [128]
- bool **inProgress**
- bool **finished**
- bool **orderPaid**
- bool **orderDelivered**
- bool **deliverOrder**
- int **sumOfOrder**
- unsigned int **orderCounter**

Friends

- ostream & **operator**<< (ostream &outs, Client &customer)

3.5.1 Detailed Description

All information about customer.

A class to store customer information, and how many orders there are for a particular customer

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

- Client.h
- Client.cpp

3.6 ErrorUI Class Reference

User interface for viewing exception log.

`#include <ErrorUI.h>`

Public Member Functions

- void **displayErrorCount** ()
- void **mainUI** ()

3.6.1 Detailed Description

User interface for viewing exception log.

A user can view count of exceptions and then choose to view a particular exception and its message

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

- ErrorUI.h
- ErrorUI.cpp

3.7 PizzaHelper Class Reference

A helper class to store count of items on a single pizza order.

`#include <PizzaHelper.h>`

Data Fields

- unsigned int **crustCounter**
- unsigned int **toppingsCounter**
- unsigned int **extrasCounter**
- unsigned int **menuCounter**
- unsigned int **locationCounter**
- unsigned int **sizeCounter**

3.7.1 Detailed Description

A helper class to store count of items on a single pizza order.

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

- PizzaHelper.h
- PizzaHelper.cpp

3.8 ReadWriteClass Class Reference

Data class that writes/reads/removes classes from files.

`#include <ReadWriteClass.h>`

Public Member Functions

- template<class pizzaClass> void **writeClassToFile** (pizzaClass &classToWrite, const char *fname)
- void loadAllVectors (Pizza &p)

  Function to load vectors containing all available items for sale, and locations of pizza places.

- void **removeAllContentsOfFile** (const char *fname)
- bool **loadCustomer** (Client &customer, vector< Pizza > &order, vector< PizzaHelper > &pHelper, const char *fname)
- template<class pizzaClass> bool loadSpecificVector (vector< pizzaClass > &loadVector, const char *fileName, pizzaClass &pClass)

  load a vector from file

3.8.1 Detailed Description

Data class that writes/reads/removes classes from files.

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

- ReadWriteClass.h
- ReadWriteClass.cpp

3.9 Sala Class Reference

Sala handles functionality for SalaUI.

`#include <Sala.h>`

Public Member Functions

- Sala ()

  Constructor initiates vector variable lager with everything available for ordering on a pizza.

- void enterCrust (unsigned int input)

  All enter functions push items selected by customer to order vector.

- void **enterExtras** (unsigned int input)
- void **enterLocation** (unsigned int input)
- void **enterMenu** (unsigned int input)
- void **enterToppings** (unsigned int input)
- void **enterPizzaSize** (unsigned int input)
- void newPizza ()

  Initiate a new Pizza order and push back former order to vector order.

- void **calculateSumOfOrder** ()
- void createOrder (string name, string address, int number, bool paid, bool delivery, string comment)

  Write client and pizza order to file, and client to a directory containing a list of customers.

- Client **getCustomerOrdersVector** ()
- vector< PizzaCrust > **getLagerpcrust** ()
- vector< PizzaExtras > **getLagerpextras** ()
- vector< PizzaLocations > **getLagerplocations** ()
- vector< PizzaMenu > **getLagerpMenu** ()
- vector< PizzaSize > **getLagerpsize** ()
- vector< PizzaToppings > **getLagerptoppings** ()
- vector< Pizza > **getOrder** ()
- Client **getClient** ()

Friends

- ostream & **operator**<< (ostream &outs, Sala &s)

3.9.1 Detailed Description

Sala handles functionality for SalaUI.

3.9.2 Member Function Documentation

3.9.2.1 void Sala::createOrder (string *name*, string *address*, int *number*, bool *paid*, bool *delivery*, string *comment*)

Write client and pizza order to file, and client to a directory containing a list of customers.

Parameters:

name is customers name

address is customers address

number is customers address number

paid is true if order is paid, else false

delivery is true if customer wants a delivery, else false

comment is customer comment to follow with the order

3.9.2.2 void Sala::enterCrust (unsigned int *input*)

All enter functions push items selected by customer to order vector.

Parameters:

input,: is the location of item in order vector for a specific item ordered

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

- Sala.h
- Sala.cpp

3.10 SalaUI Class Reference

Class that handles all user interface interactions due to selling a pizza.

`#include <SalaUI.h>`

Public Member Functions

- void **mainOrder** ()

3.10.1 Detailed Description

Class that handles all user interface interactions due to selling a pizza.

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

- SalaUI.h
- SalaUI.cpp

3.11 UmsjonUI Class Reference

User interface for creating new Pizza items for sale and new Pizza locations.

`#include <UmsjonUI.h>`

Public Member Functions

- void **main** ()

3.11.1 Detailed Description

User interface for creating new Pizza items for sale and new Pizza locations.

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

- UmsjonUI.h
- UmsjonUI.cpp