CS 474 Homework 5 Documentation

Author:
William Montgomery

University of Illinois at Chicago

CS 474 - Object Oriented Languages and Environments

Spring 2014

This was originally generated by Doxygen as a RTF document. A HTML version of the documentation is included in the docs/html directory.

Source Code

The source code for the project is included in the "hw5" subdirectory.

Running the Project

A Makefile is provided to automate the building of the project. It can be found in the "build" directory. Running `make` in the "build" directory will build an executable named "hw5".

Running the Project

Once the project is built it can be run from the command line with "./h5". It prints a menu of options to standard error and then reads in commands from standard input. The menu is printed to standard error so that the program can be run in batch mode, i.e. `./hw4 2> /dev/null`.

Functionality

You are still working for Talk Small & Hack Big, the company that rolled the EROS product earlier on. This time the company wants to develop a Customer Order System (COS), which allows waiters to enter information about dishes ordered by patrons in a restaurant. Customers can only order food items from a limited selection listed below. However, each patron can order the same item multiple times, possibly prepared in different ways, and you must keep track of each item individually. Thus, the food order list of each customer can potentially have unlimited length. In addition, you must keep track of multiple order lists simultaneously. Your restaurant has a capacity of 10 patrons; your COS must be able to handle that many lists at all times. Finally, food items in each list should be kept in the order in which they will be served. Here is a description of food items. Each food item will have a name (a C-style string), a price (a double), and a calorie count (an integer). **Restaurant** seats are numbered, and each food item list should indicate the seat number of the patron who ordered the list (i.e., an integer from 1 to 10). In addition, each of the food items adds the following items of information:

Salmon — This food item adds an enumerated type describing how it should be cooked (i.e., rare, medium, well-done), and another C-style string describing the side dish to included (e.g., mashed pota- toes, baked potato, rice pilaf, etc.)

Turkey Sandwich (**TurkeySandwich**) — This food item adds an enumerated type describing the kind of bread for the sand- wich (i.e., white, whole wheat, or rye), and a list of condiments to be added to the sandwich (e.g., lettuce, tomato, mayonnaise, mustard, provolone cheese, etc.) The list of condiments may have unlimited length.

Eggplant Casserole (**EggplantCasserole**) — This food item adds an integer describing the number of ounces in the serving and a C-style string describing the side dish, similar to the case of the salmon above.

Use a command line interface for entering the commands below. Your command line interface will prompt

the user for a command, and then execute the command. Here is a list of commands. Make sure not to cause any memory leaks or dangling pointers in the implementation of these commands.

n — New List. This operation allows a user (i.e., the restaurant owner) to create a new food order list. COS prompts the user (i.e., a waiter) for the seat number of the patron requesting the list and an empty list is displayed on a the user's screen. An error is returned if the input seat number was already assigned to a patron, that is, the seat is not empty.

a — Add List Entry. This operation allows a user to add a new food item to a food item list. The user is prompted for the patron number, and the type of food item (from the 3 above possibilities). Next, the user is prompted for the additional items pertaining to each food type. Upon completion of this interaction, the updated list is displayed on the user's screen. The new entry is assumed to be added at the end of the list.

c — Copy List. This operation allows a user to copy a food list from a patron to another patron. The user is prompted for the numbers of the two patrons. Next, the original food item list is copied to the destination patron. The two lists should not share any data structures at all. (This means that the first list is deep copied into the second list.) The two lists are displayed on the user's screen.• d — Delete List. When a patron leaves the restaurant, the corresponding food item list is deleted com- pletely. Make sure not to cause any memory leaks. The seat of the patron is made available for new patrons.

l — List patron. The food order list of a given patron is displayed. The user is prompted for the patron number. An appropriate message is displayed if the input number does not have a food item list (e.g., because that seat is empty).

s — Swap orders. The command swaps the orders of two restaurant patrons. The user is prompted for the seat numbers of the two patrons and the corresponding lists are swapped. The new orders for the two patrons are displayed.

q — quit. The COS is exited.

You must make use of an abstract superclass FoodItem with three concrete subclasses corresponding to the

three foods above. These classes must implement a virtual deletion and virtual copying schemes. Make sure to code the big-three for each of the classes. You are at liberty to choose an appropriate data structure to hold the 10 food item lists.

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

File Index

File List

Here is a list of all files with brief descriptions:

Class Documentation

BreadChoice Class Reference

The **BreadChoice** class Represents a choice in breads.

`#include <breadchoice.h>`

Public Member Functions

BreadChoice ()

BreadChoice::BreadChoice *Default Constructor.*

Protected Member Functions

bool **isValid** (int)

BreadChoice::isValid *Returns whether an enumeration is a valid enumeration.*

Protected Attributes

Bread bread

bread The bread chosen

Static Protected Attributes

static char **breadString** [**NUMBREADS**][32] = {"White", "Whole Wheat", "Rye"}

breadString Maps enums to strings

Friends

ostream & **operator<<** (ostream &, const **BreadChoice** &)

operator << Overloading the << operator for streams

Detailed Description

The **BreadChoice** class Represents a choice in breads.

Definition at line 19 of file breadchoice.h.

Constructor & Destructor Documentation

BreadChoice::BreadChoice ()

BreadChoice::BreadChoice Default Constructor.

Get the selection

Set the selection

Definition at line 8 of file breadchoice.cpp.

Here is the call graph for this function:

IMAGE

Member Function Documentation

bool BreadChoice::isValid (int *enumeration*) [protected]

BreadChoice::isValid Returns whether an enumeration is a valid enumeration.

Parameters:

enumeration	The enumeration to check

Returns:

true if valid, false otherwise

Definition at line 35 of file breadchoice.cpp.

Here is the caller graph for this function:

IMAGE

Friends And Related Function Documentation

ostream& operator<< (ostream & *stream*, const BreadChoice & *instance*) [friend]

operator << Overloading the << operator for streams

Parameters:

stream	The output stream
instance	The instance to output

Returns:

The output stream

Definition at line 46 of file breadchoice.cpp.

Member Data Documentation

Bread BreadChoice::bread [protected]

bread The bread chosen

Definition at line 30 of file breadchoice.h.

char BreadChoice::breadString = {"White", "Whole Wheat", "Rye"} `[static], [protected]`

breadString Maps enums to strings

Definition at line 25 of file breadchoice.h.

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

hw5/**breadchoice.h**
hw5/**breadchoice.cpp**

CondimentList Class Reference

The **CondimentList** class Represents a list of condiments.

`#include <condimentlist.h>`

Public Member Functions

CondimentList ()
CondimentList::CondimentList *Default constructor.*

~CondimentList ()
CondimentList::~CondimentList *Destructor.*

CondimentList (const **CondimentList** &)
CondimentList::CondimentList *Copy Constructor.*

CondimentList & **operator=** (**CondimentList** &)
CondimentList::operator = *The copy assignment operator.*

Protected Attributes

vector< char * > **condiments**
condiments Holds the condiments

Friends

ostream & **operator<<** (ostream &, const **CondimentList** &)
operator << Overloads the << operator for streams

Detailed Description

The **CondimentList** class Represents a list of condiments.

Definition at line 12 of file condimentlist.h.

Constructor & Destructor Documentation

CondimentList::CondimentList ()

CondimentList::CondimentList Default constructor.

Read in values

A blank line exits the loop

Otherwise we push it into the condiments

Definition at line 8 of file condimentlist.cpp.

CondimentList::~CondimentList ()

CondimentList::~CondimentList Destructor.

Definition at line 39 of file condimentlist.cpp.

CondimentList::CondimentList (const CondimentList & *other*)

CondimentList::CondimentList Copy Constructor.

Parameters:

other	The other **CondimentList**

Definition at line 52 of file condimentlist.cpp.

Member Function Documentation

CondimentList & CondimentList::operator= (CondimentList & *other*)

CondimentList::operator = The copy assignment operator.

Parameters:

other	The other list

Returns:

*this

Definition at line 69 of file condimentlist.cpp.

Friends And Related Function Documentation

ostream& operator<< (ostream & *stream*, const CondimentList & *list*) [friend]

operator << Overloads the << operator for streams

Parameters:

stream	The output stream
list	The **CondimentList**

Returns:

The output stream

Definition at line 96 of file condimentlist.cpp.

Member Data Documentation

vector<char*> CondimentList::condiments [protected]

condiments Holds the condiments

Definition at line 18 of file condimentlist.h.

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

hw5/**condimentlist.h**
hw5/**condimentlist.cpp**

EggplantCasserole Class Reference

The **EggplantCasserole** class represents an Eggplant Casserole.

`#include <eggplantcasserole.h>`

Inheritance diagram for EggplantCasserole:

IMAGE

Collaboration diagram for EggplantCasserole:

IMAGE

Public Member Functions

EggplantCasserole ()

EggplantCasserole::EggplantCasserole *Constructor.*

virtual **~EggplantCasserole** ()

EggplantCasserole::~EggplantCasserole *Destructor.*

virtual **EggplantCasserole * copy** ()

EggplantCasserole::copy *Copies an* ***EggplantCasserole****, be sure to deallocate returned instance.*

virtual void **print** ()

EggplantCasserole::print *Prints to stdout.*

Protected Attributes

int **ounces**

ounces The number of ounces

char **side** [256]

side The side dish

Additional Inherited Members

Detailed Description

The **EggplantCasserole** class represents an Eggplant Casserole.

Definition at line 13 of file eggplantcasserole.h.

Constructor & Destructor Documentation

EggplantCasserole::EggplantCasserole ()

EggplantCasserole::EggplantCasserole Constructor.

Definition at line 7 of file eggplantcasserole.cpp.

Here is the caller graph for this function:

IMAGE

EggplantCasserole::~EggplantCasserole () [virtual]

EggplantCasserole::~EggplantCasserole Destructor.

Definition at line 21 of file eggplantcasserole.cpp.

Member Function Documentation

EggplantCasserole * EggplantCasserole::copy () [virtual]

EggplantCasserole::copy Copies an **EggplantCasserole**, be sure to deallocate returned instance.

Returns:

A new instance

Implements **FoodItem** (*p.21*).

Definition at line 31 of file eggplantcasserole.cpp.

Here is the call graph for this function:

IMAGE

void EggplantCasserole::print () [virtual]

EggplantCasserole::print Prints to stdout.

Reimplemented from **FoodItem** (*p.21*).

Definition at line 41 of file eggplantcasserole.cpp.

Here is the call graph for this function:

IMAGE

Member Data Documentation

int EggplantCasserole::ounces [protected]

ounces The number of ounces

Definition at line 19 of file eggplantcasserole.h.

char EggplantCasserole::side[256] [protected]

side The side dish

Definition at line 24 of file eggplantcasserole.h.

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

hw5/**eggplantcasserole.h**
hw5/**eggplantcasserole.cpp**

FoodItem Class Reference

`#include <fooditem.h>`
Inheritance diagram for FoodItem:

IMAGE

Public Member Functions

virtual **~FoodItem** ()
FoodItem::~FoodItem *Destructor.*

virtual **FoodItem** * **copy** ()=0
virtual void **print** ()
FoodItem::print *Prints to stdout.*

Protected Member Functions

FoodItem ()
FoodItem::FoodItem *Default Constructor.*

Protected Attributes

char **name** [32]
name The name of the item

double **price**
price The price of the item

int **calories**
calories The number of calories

Detailed Description

Definition at line 8 of file fooditem.h.

Constructor & Destructor Documentation

FoodItem::FoodItem () `[protected]`

FoodItem::FoodItem Default Constructor.

Definition at line 9 of file fooditem.cpp.

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

FoodItem::~FoodItem Destructor.

Definition at line 21 of file fooditem.cpp.

Member Function Documentation

virtual FoodItem* FoodItem::copy () [pure virtual]

Implemented in **Salmon** (*p.31*), **EggplantCasserole** (*p.18*), and **TurkeySandwich** (*p.50*).

void FoodItem::print () [virtual]

FoodItem::print Prints to stdout.

Reimplemented in **Salmon** (*p.31*), **EggplantCasserole** (*p.18*), and **TurkeySandwich** (*p.50*).

Definition at line 29 of file fooditem.cpp.

Here is the caller graph for this function:

IMAGE

Member Data Documentation

int FoodItem::calories [protected]

calories The number of calories

Definition at line 24 of file fooditem.h.

char FoodItem::name[32] [protected]

name The name of the item

Definition at line 14 of file fooditem.h.

double FoodItem::price [protected]

price The price of the item

Definition at line 19 of file fooditem.h.

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

hw5/**fooditem.h**
hw5/**fooditem.cpp**

MenuList Class Reference

The **MenuList** class Represents an order containing a number of FoodItems.

`#include <menulist.h>`

Public Member Functions

MenuList ()
__MenuList::MenuList__ Constructor.

~MenuList ()
__MenuList::~MenuList__ Destructor.

MenuList (const **MenuList** &)
__MenuList::MenuList__ Copy Constructor.

MenuList & **operator=** (**MenuList** &)
__MenuList::operator =__ Copy Assignment Constructor.

MenuList * **add** ()
__MenuList::add__ Adds a __FoodItem__ to the __MenuList__.

void **printOptions** ()
__MenuList::printOptions__ Prints a list of options to stdout.

Protected Attributes

vector< **FoodItem** * > **orders**
orders The list of FoodItems

Friends

ostream & **operator<<** (ostream &, const **MenuList** &)
operator << Overloading the << operator for streams

Detailed Description

The **MenuList** class Represents an order containing a number of FoodItems.

Definition at line 16 of file menulist.h.

Constructor & Destructor Documentation

MenuList::MenuList ()

MenuList::MenuList Constructor.

Definition at line 6 of file menulist.cpp.

MenuList::~MenuList ()

MenuList::~MenuList Destructor.

Definition at line 14 of file menulist.cpp.

MenuList::MenuList (const MenuList & *other*)

MenuList::MenuList Copy Constructor.

Parameters:

other	The other **MenuList**

Make copies of the orders

Definition at line 27 of file menulist.cpp.

Member Function Documentation

MenuList * MenuList::add ()

MenuList::add Adds a **FoodItem** to the **MenuList**.

Returns:

this

Definition at line 76 of file menulist.cpp.

Here is the call graph for this function:

IMAGE

Here is the caller graph for this function:

IMAGE

MenuList & MenuList::operator= (MenuList & *other*)

MenuList::operator = Copy Assignment Constructor.

Parameters:

other	The other **MenuList**

Returns:

*this

Delete all the current orders

Make copies of the orders

Definition at line 42 of file menulist.cpp.

void MenuList::printOptions ()

MenuList::printOptions Prints a list of options to stdout.

Definition at line 64 of file menulist.cpp.

Here is the caller graph for this function:

IMAGE

Friends And Related Function Documentation

ostream& operator<< (ostream & *stream*, const MenuList & *list*) [friend]

operator << Overloading the << operator for streams

Parameters:

stream	The output stream
list	The **MenuList** to print

Returns:

The output stream

Definition at line 117 of file menulist.cpp.

Member Data Documentation

vector<FoodItem*> MenuList::orders [protected]

orders The list of FoodItems

Definition at line 22 of file menulist.h.

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

hw5/**menulist.h**
hw5/**menulist.cpp**

Restaurant Class Reference

The **Restaurant** class Represents a **Restaurant**.

`#include <restaurant.h>`

Collaboration diagram for Restaurant:

IMAGE

Public Member Functions

Restaurant ()
Restaurant::Restaurant *Constructor.*

~Restaurant ()
Restaurant::~Restaurant *Destructor.*

Restaurant (const **Restaurant** &)
Restaurant::Restaurant *Copy Constructor.*

Restaurant & **operator=** (**Restaurant** &)
Restaurant::operator = *Copy Assignment Constructor.*

Restaurant * run ()
Restaurant::run *Executes the POS.*

Restaurant * printMenu ()
Restaurant::printMenu *Prints the list of options.*

Restaurant * getChoice (char)
Restaurant::getChoice *Executes a method based on a given input.*

void **doNewList** ()
Restaurant::doNewList *Creates a new list for a specific seat.*

void **doDelete** ()
Restaurant::doDelete *Deletes a specific list.*

void **doAdd** ()
Restaurant::doAdd *Adds a* ***FoodItem*** *to a* ***MenuList****.*

void **doCopy** ()
Restaurant::doCopy *Copies a* ***MenuList*** *to another seat.*

void **doList** ()
Restaurant::doList *Lists the current order for a specific seat.*

void **doSwitch** ()
Restaurant::doSwitch *Switches two orders.*

Protected Attributes

MenuList * seats [**NUMSEATS**]
seats The seats in the ***Restaurant***

Detailed Description

The **Restaurant** class Represents a **Restaurant**.

Definition at line 17 of file restaurant.h.

Constructor & Destructor Documentation

Restaurant::Restaurant ()

Restaurant::Restaurant Constructor.

Definition at line 7 of file restaurant.cpp.

Restaurant::~Restaurant ()

Restaurant::~Restaurant Destructor.

Definition at line 18 of file restaurant.cpp.

Restaurant::Restaurant (const Restaurant & *otherRestaurant*)

Restaurant::Restaurant Copy Constructor.

Parameters:

otherRestaurant	The other **Restaurant**

Definition at line 30 of file restaurant.cpp.

Member Function Documentation

void Restaurant::doAdd ()

Restaurant::doAdd Adds a **FoodItem** to a **MenuList**.

Definition at line 137 of file restaurant.cpp.

Here is the call graph for this function:

IMAGE

Here is the caller graph for this function:

IMAGE

void Restaurant::doCopy ()

Restaurant::doCopy Copies a **MenuList** to another seat.

Definition at line 162 of file restaurant.cpp.

Here is the caller graph for this function:

IMAGE

void Restaurant::doDelete ()

Restaurant::doDelete Deletes a specific list.

Definition at line 116 of file restaurant.cpp.

Here is the caller graph for this function:

IMAGE

void Restaurant::doList ()

Restaurant::doList Lists the current order for a specific seat.

Definition at line 197 of file restaurant.cpp.

Here is the caller graph for this function:

IMAGE

void Restaurant::doNewList ()

Restaurant::doNewList Creates a new list for a specific seat.

Definition at line 92 of file restaurant.cpp.

Here is the caller graph for this function:

IMAGE

void Restaurant::doSwitch ()

Restaurant::doSwitch Switches two orders.

Definition at line 224 of file restaurant.cpp.

Here is the caller graph for this function:

IMAGE

Restaurant * Restaurant::getChoice (char *input*)

Restaurant::getChoice Executes a method based on a given input.

Parameters:

input	The input character

Returns:

this

Definition at line 257 of file restaurant.cpp.

Here is the call graph for this function:

IMAGE

Here is the caller graph for this function:

IMAGE

Restaurant & Restaurant::operator= (Restaurant & *otherRestaurant*)

Restaurant::operator = Copy Assignment Constructor.

Parameters:

otherRestaurant	The other **Restaurant**

Returns:

*this

Definition at line 43 of file restaurant.cpp.

Restaurant * Restaurant::printMenu ()

Restaurant::printMenu Prints the list of options.

Returns:

this

Definition at line 74 of file restaurant.cpp.

Here is the caller graph for this function:

IMAGE

Restaurant * Restaurant::run ()

Restaurant::run Executes the POS.

Returns:

this

Definition at line 57 of file restaurant.cpp.

Here is the call graph for this function:

IMAGE

Here is the caller graph for this function:

IMAGE

Member Data Documentation

MenuList* Restaurant::seats[NUMSEATS] [protected]

seats The seats in the **Restaurant**

Definition at line 23 of file restaurant.h.

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

hw5/**restaurant.h**
hw5/**restaurant.cpp**

Salmon Class Reference

The **Salmon** class Represents a **Salmon** dish.

`#include <salmon.h>`

Inheritance diagram for Salmon:

IMAGE

Collaboration diagram for Salmon:

IMAGE

Public Member Functions

Salmon ()

Salmon::Salmon *Constructor.*

virtual ~**Salmon** ()

Salmon::~Salmon *Destructor.*

virtual **Salmon * copy** ()

Salmon::copy *Copies the current instance. Caller is responsible for deallocating returned object.*

virtual void **print** ()

Salmon::print *Prints the current instance.*

Protected Attributes

Temperature done

done The desired doneness

char **side** [256]

side The side dish

Additional Inherited Members

Detailed Description

The **Salmon** class Represents a **Salmon** dish.

Definition at line 14 of file salmon.h.

Constructor & Destructor Documentation

Salmon::Salmon ()

Salmon::Salmon Constructor.

Definition at line 7 of file salmon.cpp.

Here is the caller graph for this function:

IMAGE

Salmon::~Salmon () [virtual]

Salmon::~Salmon Destructor.

Definition at line 20 of file salmon.cpp.

Member Function Documentation

Salmon * Salmon::copy () [virtual]

Salmon::copy Copies the current instance. Caller is responsible for deallocating returned object.

Returns:

A new **Salmon** instance

Implements **FoodItem** (*p.21*).

Definition at line 30 of file salmon.cpp.

Here is the call graph for this function:

IMAGE

void Salmon::print () [virtual]

Salmon::print Prints the current instance.

Reimplemented from **FoodItem** (*p.21*).

Definition at line 39 of file salmon.cpp.

Here is the call graph for this function:

IMAGE

Member Data Documentation

Temperature Salmon::done [protected]

done The desired doneness

Definition at line 20 of file salmon.h.

char Salmon::side[256] [protected]

side The side dish

Definition at line 25 of file salmon.h.

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

hw5/**salmon.h**
hw5/**salmon.cpp**

Temperature Class Reference

`#include <temperature.h>`

Public Member Functions

Temperature ()
Temperature::Temperature *Constructor.*

Protected Member Functions

bool **isValid** (int)
Temperature::isValid *Evaluates whether the enumeration is valid.*

Protected Attributes

Doneness doneness
doneness The selected doneness

Static Protected Attributes

static char **donesnessString** [**NUMTEMPS**][32] = {"Rare", "Medium Rare", "Medium", "Medium Well", "Well Done"}
donesnessString Strings representing all of the enumeration values

Friends

ostream & **operator<<** (ostream &, const **Temperature** &)
operator << Overloads the << operator for streams

Detailed Description

Definition at line 19 of file temperature.h.

Constructor & Destructor Documentation

Temperature::Temperature ()

Temperature::Temperature Constructor.

Definition at line 8 of file temperature.cpp.

Here is the call graph for this function:

IMAGE

Member Function Documentation

bool Temperature::isValid (int *enumeration*) [protected]

Temperature::isValid Evaluates whether the enumeration is valid.

Parameters:

enumeration	The enumeration to evaluate

Returns:

true if valid, false otherwise

Definition at line 32 of file temperature.cpp.

Here is the caller graph for this function:

IMAGE

Friends And Related Function Documentation

ostream& operator<< (ostream & *stream*, const Temperature & *temperature*) [friend]

operator << Overloads the << operator for streams

Parameters:

stream	The output stream
temperature	The **Temperature**

Returns:

The output stream

Definition at line 43 of file temperature.cpp.

Member Data Documentation

Doneness Temperature::doneness [protected]

doneness The selected doneness

Definition at line 30 of file temperature.h.

char Temperature::donesnessString = {"Rare", "Medium Rare", "Medium", "Medium Well", "Well Done"} [static], [protected]

donesnessString Strings representing all of the enumeration values

Definition at line 25 of file temperature.h.

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

hw5/**temperature.h**
hw5/**temperature.cpp**

TurkeySandwich Class Reference

The **TurkeySandwich** class Represents a Turkey Sandwich.

`#include <turkeysandwich.h>`

Inheritance diagram for TurkeySandwich:

IMAGE

Collaboration diagram for TurkeySandwich:

IMAGE

Public Member Functions

TurkeySandwich ()

TurkeySandwich::TurkeySandwich *Constructor.*

~TurkeySandwich ()

TurkeySandwich::~TurkeySandwich *Destructor.*

virtual **TurkeySandwich * copy** ()

TurkeySandwich::copy *Copies the current instance. Caller is responsible for deallocating the object.*

virtual void **print** ()

TurkeySandwich::print *Prints out the* ***TurkeySandwich*** *to stdout.*

Protected Attributes

BreadChoice bread

bread The choice of bread

CondimentList condiments

condiments The selected condiments

Additional Inherited Members

Detailed Description

The **TurkeySandwich** class Represents a Turkey Sandwich.

Definition at line 13 of file turkeysandwich.h.

Constructor & Destructor Documentation

TurkeySandwich::TurkeySandwich ()

TurkeySandwich::TurkeySandwich Constructor.

Definition at line 7 of file turkeysandwich.cpp.

Here is the caller graph for this function:

IMAGE

TurkeySandwich::~TurkeySandwich ()

TurkeySandwich::~TurkeySandwich Destructor.

Definition at line 15 of file turkeysandwich.cpp.

Member Function Documentation

TurkeySandwich * TurkeySandwich::copy () [virtual]

TurkeySandwich::copy Copies the current instance. Caller is responsible for deallocating the object.

Returns:

A new instance of **TurkeySandwich**

Implements **FoodItem** (*p.21*).

Definition at line 24 of file turkeysandwich.cpp.

Here is the call graph for this function:

IMAGE

void TurkeySandwich::print () [virtual]

TurkeySandwich::print Prints out the **TurkeySandwich** to stdout.

Reimplemented from **FoodItem** (*p.21*).

Definition at line 33 of file turkeysandwich.cpp.

Here is the call graph for this function:

IMAGE

Member Data Documentation

BreadChoice TurkeySandwich::bread [protected]

bread The choice of bread

Definition at line 19 of file turkeysandwich.h.

CondimentList TurkeySandwich::condiments [protected]

condiments The selected condiments

Definition at line 24 of file turkeysandwich.h.

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

hw5/**turkeysandwich.h**
hw5/**turkeysandwich.cpp**

File Documentation

hw5/breadchoice.cpp File Reference

```
#include "breadchoice.h"
```

Include dependency graph for breadchoice.cpp:

IMAGE

Functions

ostream & **operator<<** (ostream &stream, const **BreadChoice** &instance)

operator << Overloading the << operator for streams

Function Documentation

ostream& operator<< (ostream & *stream*, const BreadChoice & *instance*)

operator << Overloading the << operator for streams

Parameters:

stream	The output stream
instance	The instance to output

Returns:

The output stream

Definition at line 46 of file breadchoice.cpp.

hw5/breadchoice.h File Reference

`#include <iostream>`

Include dependency graph for breadchoice.h:

IMAGE

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

IMAGE

Classes

class **BreadChoice**

*The **BreadChoice** class Represents a choice in breads.* Enumerations

enum **Bread** { **WHITE** = 0, **WHOLEWHEAT** = 1, **RYE** = 3 }

The Bread enum Maps the possible types of bread. Variables

static const int **NUMBREADS** = 3

Enumeration Type Documentation

enum Bread

The Bread enum Maps the possible types of bread.

Enumerator

WHITE
WHOLEWHEAT
RYE

Definition at line 13 of file breadchoice.h.

Variable Documentation

const int NUMBREADS = 3 `[static]`

Definition at line 8 of file breadchoice.h.

hw5/condimentlist.cpp File Reference

```
#include <limits.h>
#include <string.h>
#include "condimentlist.h"
```

Include dependency graph for condimentlist.cpp:

IMAGE

Functions

ostream & **operator<<** (ostream &stream, const **CondimentList** &list)

operator << Overloads the << operator for streams

Function Documentation

ostream& operator<< (ostream & *stream*, const CondimentList & *list*)

operator << Overloads the << operator for streams

Parameters:

stream	The output stream
list	The **CondimentList**

Returns:

The output stream

Definition at line 96 of file condimentlist.cpp.

hw5/condimentlist.h File Reference

```
#include <vector>
#include <iostream>
```

Include dependency graph for condimentlist.h:

IMAGE

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

IMAGE

Classes

class **CondimentList**

*The **CondimentList** class Represents a list of condiments.*

hw5/eggplantcasserole.cpp File Reference

```
#include "eggplantcasserole.h"
#include <string.h>
```

Include dependency graph for eggplantcasserole.cpp:

IMAGE

hw5/eggplantcasserole.h File Reference

```
#include <stdlib.h>
#include <iostream>
#include "fooditem.h"
```

Include dependency graph for eggplantcasserole.h:

IMAGE

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

IMAGE

Classes

class **EggplantCasserole**

*The **EggplantCasserole** class represents an Eggplant Casserole.*

hw5/fooditem.cpp File Reference

```
#include <iostream>
#include <string>
#include <iomanip>
#include "fooditem.h"
```

Include dependency graph for fooditem.cpp:

IMAGE

hw5/fooditem.h File Reference

```
#include <iostream>
```

Include dependency graph for fooditem.h:

IMAGE

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

IMAGE

Classes

class **FoodItem**

hw5/main.cpp File Reference

```
#include <stdio.h>
#include "restaurant.h"
```

Include dependency graph for main.cpp:

IMAGE

Functions

int **main** (void)

*main Creates a **Restaurant** object and then runs it*

Function Documentation

int main (void)

main Creates a **Restaurant** object and then runs it

Returns:

0 if no errors

Definition at line 8 of file main.cpp.

Here is the call graph for this function:

IMAGE

hw5/mainpage.dox File Reference

hw5/menulist.cpp File Reference

```
#include "menulist.h"
```

Include dependency graph for menulist.cpp:

IMAGE

Functions

ostream & **operator<<** (ostream &stream, const **MenuList** &list)
operator << Overloading the << operator for streams

Function Documentation

ostream& operator<< (ostream & *stream*, const MenuList & *list*)

operator << Overloading the << operator for streams

Parameters:

stream	The output stream
list	The **MenuList** to print

Returns:

The output stream

Definition at line 117 of file menulist.cpp.

hw5/menulist.h File Reference

```
#include <vector>
#include <iostream>
#include "fooditem.h"
#include "salmon.h"
#include "turkeysandwich.h"
#include "eggplantcasserole.h"
```

Include dependency graph for menulist.h:

IMAGE

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

IMAGE

Classes

class **MenuList**

*The **MenuList** class Represents an order containing a number of FoodItems.*

hw5/restaurant.cpp File Reference

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

Include dependency graph for restaurant.cpp:

IMAGE

hw5/restaurant.h File Reference

`#include "menulist.h"`

Include dependency graph for restaurant.h:

IMAGE

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

IMAGE

Classes

class **Restaurant**

*The **Restaurant** class Represents a **Restaurant**.* Variables

static const int **NUMSEATS** = 11

NUMSEATS The number of seats + 1. We use an extra dummy seat at the beginning of the array so we can start the index at 1.

Variable Documentation

const int NUMSEATS = 11 `[static]`

NUMSEATS The number of seats + 1. We use an extra dummy seat at the beginning of the array so we can start the index at 1.

Definition at line 12 of file restaurant.h.

hw5/salmon.cpp File Reference

```
#include <string.h>
#include "salmon.h"
```

Include dependency graph for salmon.cpp:

IMAGE

hw5/salmon.h File Reference

```
#include <stdlib.h>
#include <iostream>
#include "fooditem.h"
#include "temperature.h"
```

Include dependency graph for salmon.h:

IMAGE

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

IMAGE

Classes

class **Salmon**

*The **Salmon** class Represents a **Salmon** dish.*

hw5/temperature.cpp File Reference

`#include "temperature.h"`

Include dependency graph for temperature.cpp:

IMAGE

Functions

ostream & **operator<<** (ostream &stream, const **Temperature** &temperature)

operator << Overloads the << operator for streams

Function Documentation

ostream& operator<< (ostream & *stream*, const Temperature & *temperature*)

operator << Overloads the << operator for streams

Parameters:

stream	The output stream
temperature	The **Temperature**

Returns:

The output stream

Definition at line 43 of file temperature.cpp.

hw5/temperature.h File Reference

`#include <iostream>`

Include dependency graph for temperature.h:

IMAGE

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

IMAGE

Classes

class **Temperature**

Enumerations

enum **Doneness** { **RARE** = 0, **MEDIUMRARE** = 1, **MEDIUM** = 2, **MEDIUMWELL** = 3, **WELLDONE** = 4 }

The Doneness enum An enumeration with all possible values of the Doneness. Variables

static const int **NUMTEMPS** = 5

NUMTEMPS The number of temps in the enum.

Enumeration Type Documentation

enum Doneness

The Doneness enum An enumeration with all possible values of the Doneness.

Enumerator

RARE
MEDIUMRARE
MEDIUM
MEDIUMWELL
WELLDONE

Definition at line 16 of file temperature.h.

Variable Documentation

const int NUMTEMPS = 5 `[static]`

NUMTEMPS The number of temps in the enum.

Definition at line 11 of file temperature.h.

hw5/turkeysandwich.cpp File Reference

```
#include "turkeysandwich.h"
#include <string.h>
```

Include dependency graph for turkeysandwich.cpp:

IMAGE

hw5/turkeysandwich.h File Reference

```
#include "fooditem.h"
#include "breadchoice.h"
#include "condimentlist.h"
```

Include dependency graph for turkeysandwich.h:

IMAGE

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

IMAGE

Classes

class **TurkeySandwich**

*The **TurkeySandwich** class Represents a Turkey Sandwich.* Index
INDEX