Cheat GBT

AUTHOR
Version

Table of Contents

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

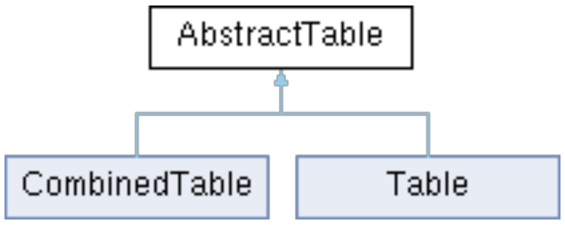
Here is a list of all files with brief descriptions:

Class Documentation

AbstractTable Class Reference

`#include <AbstractTable.h>`

Inheritance diagram for AbstractTable:



Public Member Functions

- int **getTableID** ()
  *Get the **Table** I D object.*

- void **setTableID** (int ID)
  *Set the **Table** ID.*

- void **setOccupied** (bool o)
  Set the Occupied object.

- bool **getOccupied** ()
  Get the Occupied object.

- int **getMaxPeople** ()
  Get the MaxPeople allowed in on the table.

- bool **visitTable** ()
  Set the Max People object.

- void **setMaxPeople** (int **maxPeople**)
- **TableState** * **getTableState** ()
- void **setTableState** (**TableState** *state)
- **CustomerGroup** * **getCustomerGroup** ()
- void **setCustomerGroup** (**CustomerGroup** ***customerGroup**)
- virtual bool **AddTable** (**AbstractTable** *table)=0

- virtual **AbstractTable** * **SeparateTable** ()=0
- int **getCurrentPeople** ()
- void **setCurrentPeople** (int **currentPeople**)
- virtual vector< **Order** * > **PlaceOrder** ()
- void **ReceiveOrder** (vector< **Order** * > orders)
- int **getRandomState** ()
- void **setRandomState** (int **RandomState**)
- string **EnquireState** ()
- **AbstractTable** ()
- virtual **~AbstractTable** ()
- bool **payBill** ()
- vector< **Review** > **ReviewFood** ()
- vector< **Review** > **ReviewService** ()

Protected Attributes

- int **maxPeople**
- **TableState** * **tableState**
- **CustomerGroup** * **customerGroup**
- int **currentPeople**
- int **RandomState**
- int **tableID**
- bool **occupied** =false

Static Protected Attributes

- static int **counter**

Constructor & Destructor Documentation

AbstractTable::AbstractTable ()

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

Member Function Documentation

virtual bool AbstractTable::AddTable (AbstractTable * *table*) `[pure virtual]`

Implemented in **CombinedTable** (*p.54*), and **Table** (*p.146*).

string AbstractTable::EnquireState ()

int AbstractTable::getCurrentPeople () `[inline]`

CustomerGroup * AbstractTable::getCustomerGroup () `[inline]`

int AbstractTable::getMaxPeople () `[inline]`

Get the MaxPeople allowed in on the table.

Returns

int

bool AbstractTable::getOccupied ()`[inline]`

Get the Occupied object.

Returns

true

false

int AbstractTable::getRandomState ()

int AbstractTable::getTableID ()`[inline]`

Get the **Table** I D object.

Returns

int

TableState * AbstractTable::getTableState ()`[inline]`

bool AbstractTable::payBill ()

virtual vector< Order * > AbstractTable::PlaceOrder ()`[virtual]`

Reimplemented in **CombinedTable** (*p.54*).

void AbstractTable::ReceiveOrder (vector< Order * > *orders*)

vector< Review > AbstractTable::ReviewFood ()`[inline]`

vector< Review > AbstractTable::ReviewService ()`[inline]`

virtual AbstractTable * AbstractTable::SeparateTable ()`[pure virtual]`

Implemented in **CombinedTable** (*p.54*), and **Table** (*p.147*).

void AbstractTable::setCurrentPeople (int *currentPeople*)`[inline]`

void AbstractTable::setCustomerGroup (CustomerGroup * *customerGroup*)`[inline]`

void AbstractTable::setMaxPeople (int *maxPeople*)`[inline]`

void AbstractTable::setOccupied (bool *o*)`[inline]`

Set the Occupied object.

Parameters

o	set occupied attribute
o	

void AbstractTable::setRandomState (int *RandomState*)`[inline]`

void AbstractTable::setTableID (int *ID*)`[inline]`

Set the **Table** ID.

Parameters

ID	seting tableID
ID	

void AbstractTable::setTableState (TableState * *state*)`[inline]`

bool AbstractTable::visitTable ()

Set the Max People object.

Parameters

maxPeople	

Member Data Documentation

int AbstractTable::counter`[static], [protected]`

int AbstractTable::currentPeople`[protected]`

CustomerGroup* AbstractTable::customerGroup`[protected]`

int AbstractTable::maxPeople`[protected]`

bool AbstractTable::occupied =false`[protected]`

int AbstractTable::RandomState`[protected]`

int AbstractTable::tableID`[protected]`

TableState* AbstractTable::tableState`[protected]`

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

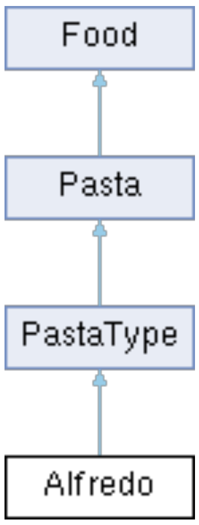
AbstractTable.h

Alfredo Class Reference

The **Alfredo** class represents **Alfredo** pasta, which is a specific type of **PastaType**.

```
#include <Alfredo.h>
```

Inheritance diagram for Alfredo:



Public Member Functions

- **Alfredo** ()
  *Constructor for **Alfredo** to set its name and cost.*

- **~Alfredo** ()
  *Destructor for **Alfredo**.*

Public Member Functions inherited from PastaType

- **PastaType** ()
  *Constructor for **PastaType** to set cost to 0.0.*

- virtual double **total** ()
  Virtual method to get the total cost of the pasta.

- virtual void **decorate** (**Pasta** *pastaType)
  Virtual method to decorate the pasta.

Public Member Functions inherited from Pasta

- **Pasta** ()
  *Constructor for **Pasta** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pasta.

- void **setCost** (double cost)
  Set the cost of the pasta.

- virtual **~Pasta** ()
  *Virtual destructor for **Pasta**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pizza** *)

Additional Inherited Members

Protected Member Functions inherited from PastaType

- **~PastaType** ()
  *Destructor for **PastaType**.*

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **Alfredo** class represents **Alfredo** pasta, which is a specific type of **PastaType**.

Constructor & Destructor Documentation

Alfredo::Alfredo ()

Constructor for **Alfredo** to set its name and cost.

Alfredo::~Alfredo ()

Destructor for **Alfredo**.

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

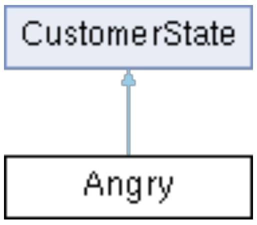
Alfredo.h

Angry Class Reference

Represents the "Angry" state of a customer.

`#include <Angry.h>`

Inheritance diagram for Angry:



Public Member Functions

- string **getStatus** ()
  Get the status of the customer state.

- void **action** ()
  Perform an action associated with the ***Angry*** *state. In this case, it prints a message indicating dissatisfaction with the food.*

Detailed Description

Represents the "Angry" state of a customer.

Member Function Documentation

void Angry::action () `[inline]`, `[virtual]`

Perform an action associated with the **Angry** state. In this case, it prints a message indicating dissatisfaction with the food.

Implements **CustomerState** (*p.64*).

string Angry::getStatus () `[inline]`, `[virtual]`

Get the status of the customer state.

Returns

A string representing the status, which is "ANGRY" for **Angry** state.

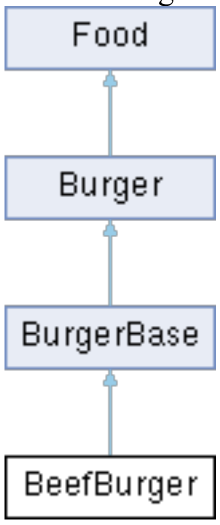
Implements **CustomerState** (*p.64*).

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

Angry.h

BeefBurger Class Reference

The **BeefBurger** class represents a beef burger, which is a specific type of **BurgerBase**.
`#include <BeefBurger.h>`
Inheritance diagram for BeefBurger:



Public Member Functions

- **BeefBurger** ()
  *Constructor for **BeefBurger** to set its name and cost.*

Public Member Functions inherited from BurgerBase

- **BurgerBase** ()
  *Constructor for **BurgerBase**.*

- virtual double **total** ()
  Virtual method to get the total cost of the burger.

- virtual void **decorate** (**Burger** *)
  Virtual method to decorate the burger.

- **~BurgerBase** ()
  *Destructor for **BurgerBase**.*

Public Member Functions inherited from Burger

- **Burger** ()
  *Constructor for **Burger** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the burger.

- void **setCost** (double cost)
  Set the cost of the burger.

- virtual **~Burger** ()
  *Virtual destructor for **Burger**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **BeefBurger** class represents a beef burger, which is a specific type of **BurgerBase**.

Constructor & Destructor Documentation

BeefBurger::BeefBurger ()

Constructor for **BeefBurger** to set its name and cost.

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

BeefBurger.h

Bill Class Reference

Represents a bill associated with a customer's order.
`#include <Bill.h>`

Public Member Functions

- **Bill** ()
  *Constructor for the **Bill** class.*
- **Order** * **getCopyOrder** ()
  Get a copy of the order associated with the bill.
- void **setCopyOrder** (**Order** *order)
  Set a copy of the order associated with the bill.
- float **getCost** ()
  Get the cost of the bill.
- void **setCost** (float orderCost)
  Set the cost of the bill.
- **BillState** * **getBillState** ()
  Get the current state of the bill.
- bool **getBillStatus** ()
  Get the status of the bill.
- void **setBillStatus** (bool BillStatus)
  Set the status of the bill.
- void **setTableID** (int ID)
  Set the table ID associated with the bill.
- void **setCustomerID** (string ID)
  Set the customer ID associated with the bill.
- std::string **getCustomerID** ()
  Get the customer ID associated with the bill.
- int **getTableID** ()
  Get the table ID associated with the bill.
- void **recoverBill** (**BillMemento** *mem)
  Recover the state of the bill from a Memento object.
- **BillMemento * saveState** ()

Save the current state of the bill to a Memento object.

- void **print** ()
  Print information about the bill, including cost, payment status, table ID, and customer ID.

Detailed Description

Represents a bill associated with a customer's order.

Constructor & Destructor Documentation

Bill::Bill ()

Constructor for the **Bill** class.

Member Function Documentation

BillState * Bill::getBillState ()

Get the current state of the bill.

Returns

A pointer to the **BillState** object representing the current state of the bill.

bool Bill::getBillStatus ()

Get the status of the bill.

Returns

true if the bill is paid, false otherwise.

Order * Bill::getCopyOrder ()

Get a copy of the order associated with the bill.

Returns

A pointer to the copy of the order.

float Bill::getCost ()

Get the cost of the bill.

Returns

The cost of the bill.

std::string Bill::getCustomerID ()

Get the customer ID associated with the bill.

Returns

The customer ID as a string.

int Bill::getTableID ()

Get the table ID associated with the bill.

Returns

The table ID.

void Bill::print ()

Print information about the bill, including cost, payment status, table ID, and customer ID.

void Bill::recoverBill (BillMemento * *mem*)

Recover the state of the bill from a Memento object.

Parameters

mem	A pointer to the Memento object containing the saved state.

BillMemento * Bill::saveState ()

Save the current state of the bill to a Memento object.

Returns

A pointer to the Memento object containing the saved state.

void Bill::setBillStatus (bool *BillStatus*)

Set the status of the bill.

Parameters

BillStatus	true if the bill is paid, false otherwise.

void Bill::setCopyOrder (Order * *order*)

Set a copy of the order associated with the bill.

Parameters

order	A pointer to the order to be copied and associated with the bill.

void Bill::setCost (float *orderCost*)

Set the cost of the bill.

Parameters

orderCost	The cost of the bill to be set.

void Bill::setCustomerID (string *ID*)

Set the customer ID associated with the bill.

Parameters

ID	The customer ID to be set.

void Bill::setTableID (int *ID*)

Set the table ID associated with the bill.

Parameters

ID	The table ID to be set.

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

Bill.h

BillCaretaker Class Reference

Manages the storage and retrieval of bill Memento objects.
`#include <BillCaretaker.h>`

Public Member Functions

- **BillCaretaker** ()
  *Constructor for the **BillCaretaker** class.*

- void **storeMemento** (**BillMemento** *mem)
  Store a Memento object in the caretaker.

- **BillMemento** * **retrieveMemento** (string customerID)
  Retrieve a Memento object by customer ID.

Detailed Description

Manages the storage and retrieval of bill Memento objects.

Constructor & Destructor Documentation

BillCaretaker::BillCaretaker ()

Constructor for the **BillCaretaker** class.

Member Function Documentation

BillMemento * BillCaretaker::retrieveMemento (string *customerID*)

Retrieve a Memento object by customer ID.

Parameters

customerID	The customer ID for which to retrieve the Memento.

Returns

A pointer to the **BillMemento** object matching the customer ID, or nullptr if not found.

void BillCaretaker::storeMemento (BillMemento * *mem*)

Store a Memento object in the caretaker.

Parameters

mem	A pointer to the **BillMemento** object to be stored.

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

BillCaretaker.h

BillMemento Class Reference

Represents a Memento for storing the state of a bill.
`#include <BillMemento.h>`

Public Member Functions

- **BillMemento** ()
  *Constructor for the **BillMemento** class.*

- **BillState** * **getState** ()
  Get the state associated with the Memento.

- void **setState** (**BillState** *bs)
  Set the state associated with the Memento.

Detailed Description

Represents a Memento for storing the state of a bill.

Constructor & Destructor Documentation

BillMemento::BillMemento () `[inline]`

Constructor for the **BillMemento** class.

Member Function Documentation

BillState * BillMemento::getState ()

Get the state associated with the Memento.

Returns

A pointer to the **BillState** object representing the state.

void BillMemento::setState (BillState * *bs*)

Set the state associated with the Memento.

Parameters

bs	A pointer to the **BillState** object to be set as the state.

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

BillMemento.h

BillState Class Reference

Represents the state of a bill associated with an order.
`#include <BillState.h>`

Public Member Functions

- void **loadFromFile** (string filename)
  Load the state from a file.
- void **saveToFile** (string filename)
  Save the state to a file.
- **Order** * **getCopyOrder** ()
  Get a copy of the order associated with the bill state.
- float **getCost** ()
  Get the cost of the bill state.
- bool **getPaidStatus** ()
  Get the payment status of the bill state.
- std::string **getCustomerID** ()
  Get the customer ID associated with the bill state.
- int **getTableID** ()
  Get the table ID associated with the bill state.
- void **setCopyOrder** (**Order** *order)
  Set a copy of the order associated with the bill state.
- void **setCost** (float newCost)
  Set the cost of the bill state.
- void **setPaid** (bool pay)
  Set the payment status of the bill state.
- void **setCustomerID** (string custID)
  Set the customer ID associated with the bill state.
- void **setTableID** (int tabID)
  Set the table ID associated with the bill state.

Detailed Description

Represents the state of a bill associated with an order.

Member Function Documentation

Order * BillState::getCopyOrder ()

Get a copy of the order associated with the bill state.

Returns

A pointer to the copy of the order.

float BillState::getCost ()

Get the cost of the bill state.

Returns

The cost of the bill state.

std::string BillState::getCustomerID ()

Get the customer ID associated with the bill state.

Returns

The customer ID as a string.

bool BillState::getPaidStatus ()

Get the payment status of the bill state.

Returns

true if the bill is paid, false otherwise.

int BillState::getTableID ()

Get the table ID associated with the bill state.

Returns

The table ID.

void BillState::loadFromFile (string *filename*)

Load the state from a file.

Parameters

filename	The name of the file from which to load the state.

void BillState::saveToFile (string *filename*)

Save the state to a file.

Parameters

filename	The name of the file to which to save the state.

void BillState::setCopyOrder (Order * *order*)

Set a copy of the order associated with the bill state.

Parameters

order	A pointer to the order to be copied and associated with the bill state.

void BillState::setCost (float *newCost*)

Set the cost of the bill state.

Parameters

newCost	The cost of the bill state to be set.

void BillState::setCustomerID (string *custID*)

Set the customer ID associated with the bill state.

Parameters

custID	The customer ID to be set.

void BillState::setPaid (bool *pay*)

Set the payment status of the bill state.

Parameters

pay	true if the bill is paid, false otherwise.

void BillState::setTableID (int *tabID*)

Set the table ID associated with the bill state.

Parameters

tabID	The table ID to be set.

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

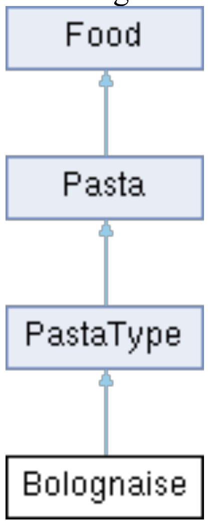
BillState.h

Bologniase Class Reference

The **Bolognaise** class represents **Bolognaise** pasta, which is a specific type of **PastaType**.

```
#include <Bolognaise.h>
```

Inheritance diagram for Bolognaise:



Public Member Functions

- **Bolognaise** ()
  *Constructor for **Bolognaise** to set its name and cost.*

- **~Bolognaise** ()
  *Destructor for **Bolognaise**.*

Public Member Functions inherited from PastaType

- **PastaType** ()
  *Constructor for **PastaType** to set cost to 0.0.*

- virtual double **total** ()
  Virtual method to get the total cost of the pasta.

- virtual void **decorate** (**Pasta** *pastaType)
  Virtual method to decorate the pasta.

Public Member Functions inherited from Pasta

- **Pasta** ()
  *Constructor for **Pasta** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pasta.

- void **setCost** (double cost)
  Set the cost of the pasta.

- virtual **~Pasta** ()
  *Virtual destructor for **Pasta**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pizza** *)

Additional Inherited Members

Protected Member Functions inherited from PastaType

- **~PastaType** ()
  *Destructor for **PastaType**.*

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **Bolognaise** class represents **Bolognaise** pasta, which is a specific type of **PastaType**.

Constructor & Destructor Documentation

Bolognaise::Bolognaise ()

Constructor for **Bolognaise** to set its name and cost.

Bolognaise::~Bolognaise ()

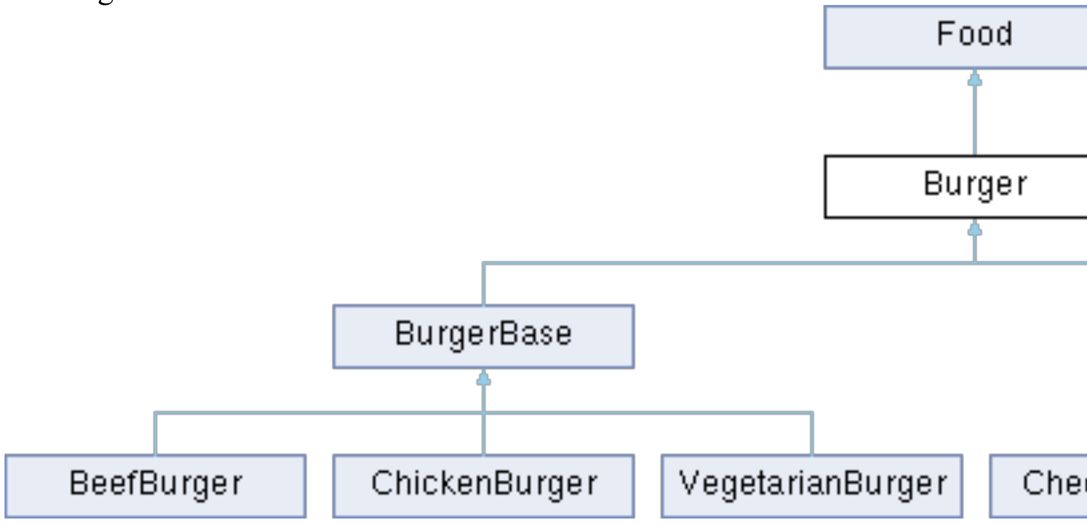
Destructor for **Bolognaise**.

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

Bolognaise.h

Burger Class Reference

The **Burger** class represents a generic burger.
`#include <Burger.h>`
Inheritance diagram for Burger:



Public Member Functions

- **Burger** ()
  *Constructor for **Burger** to set its cost to 0.0.*

- virtual void **decorate** (**Burger** *)=0
  Virtual method to decorate the burger.

- virtual double **total** ()=0
  Virtual method to get the total cost of the burger.

- double **getCost** ()
  Get the cost of the burger.

- void **setCost** (double cost)
  Set the cost of the burger.

- virtual ~**Burger** ()
  *Virtual destructor for **Burger**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()

- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **Burger** class represents a generic burger.

Constructor & Destructor Documentation

Burger::Burger ()

Constructor for **Burger** to set its cost to 0.0.

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

Virtual destructor for **Burger**.

Member Function Documentation

virtual void Burger::decorate (Burger *) `[pure virtual]`

Virtual method to decorate the burger.

Parameters

burger	A pointer to the **Burger** to be decorated.

Reimplemented from **Food** (*p.78*).

Implemented in **BurgerBase** (*p.36*), and **BurgerTopping** (*p.39*).

double Burger::getCost ()

Get the cost of the burger.

Returns

The cost of the burger.

void Burger::setCost (double *cost*)

Set the cost of the burger.

Parameters

cost	The cost of the burger.

virtual double Burger::total ()`[pure virtual]`

Virtual method to get the total cost of the burger.

Returns

The total cost of the burger.

Implements **Food** (*p.78*).

Implemented in **BurgerBase** (*p.37*), and **BurgerTopping** (*p.40*).

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

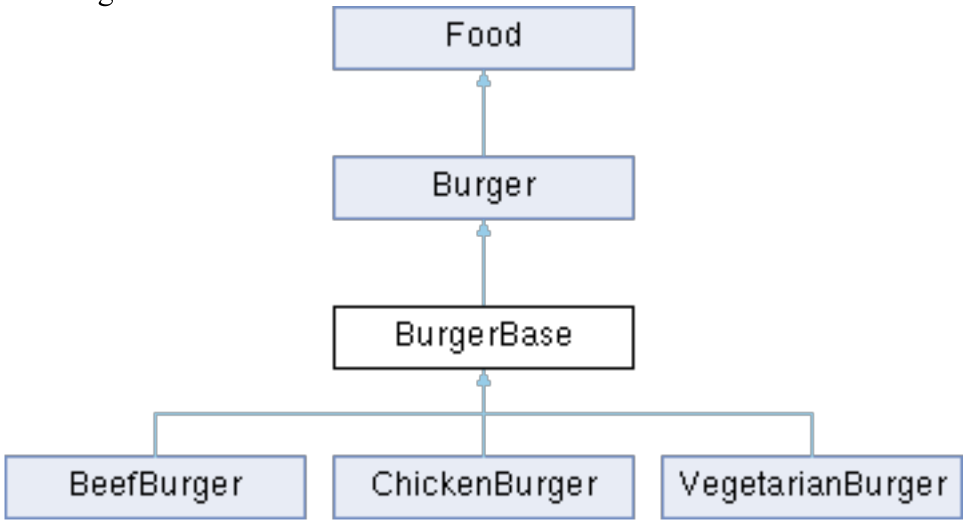
Burger.h

BurgerBase Class Reference

The **BurgerBase** class represents the base of a burger, which is a specific type of **Burger**.

`#include <BurgerBase.h>`

Inheritance diagram for BurgerBase:



Public Member Functions

- **BurgerBase** ()
  *Constructor for **BurgerBase**.*

- virtual double **total** ()
  Virtual method to get the total cost of the burger.

- virtual void **decorate** (**Burger** *)
  Virtual method to decorate the burger.

- **~BurgerBase** ()
  *Destructor for **BurgerBase**.*

Public Member Functions inherited from Burger

- **Burger** ()
  *Constructor for **Burger** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the burger.

- void **setCost** (double cost)
  Set the cost of the burger.

- virtual ~**Burger** ()
  *Virtual destructor for **Burger**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **BurgerBase** class represents the base of a burger, which is a specific type of **Burger**.

Constructor & Destructor Documentation

BurgerBase::BurgerBase ()

Constructor for **BurgerBase**.

BurgerBase::~BurgerBase ()

Destructor for **BurgerBase**.

Member Function Documentation

virtual void BurgerBase::decorate (Burger *) `[virtual]`

Virtual method to decorate the burger.

Parameters

burger	A pointer to the **Burger** to be decorated.

Implements **Burger** (*p.33*).

virtual double BurgerBase::total () `[virtual]`

Virtual method to get the total cost of the burger.

Returns

The total cost of the burger.

Implements **Burger** (*p.34*).

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

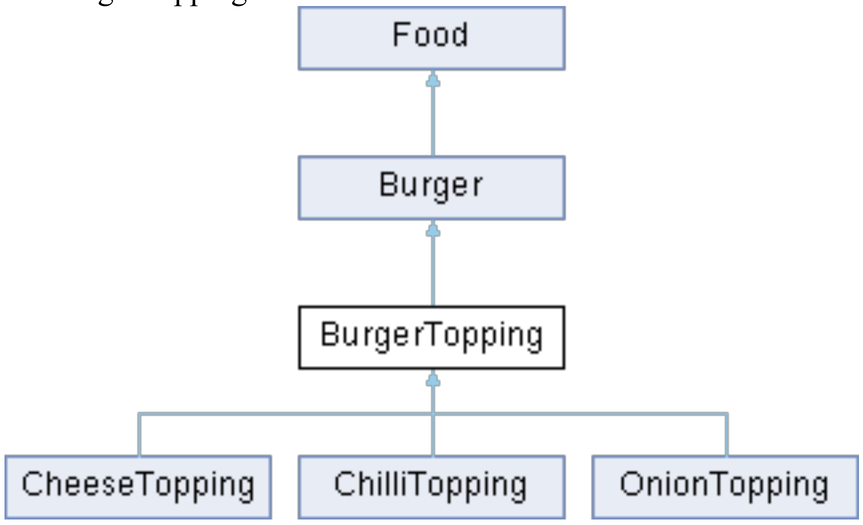
BurgerBase.h

BurgerTopping Class Reference

The **BurgerTopping** class represents a topping for a burger, which is a specific type of **Burger**.

```
#include <BurgerTopping.h>
```

Inheritance diagram for BurgerTopping:



Public Member Functions

- **BurgerTopping** ()
  *Constructor for **BurgerTopping** to set cost to 0.0.*
- virtual double **total** ()
  Virtual method to get the total cost of the burger.
- virtual void **decorate** (**Burger** *burgerTopping)
  Virtual method to decorate the burger.

Public Member Functions inherited from Burger

- **Burger** ()
  *Constructor for **Burger** to set its cost to 0.0.*
- double **getCost** ()
  Get the cost of the burger.
- void **setCost** (double cost)
  Set the cost of the burger.
- virtual ~**Burger** ()
  *Virtual destructor for **Burger**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Protected Member Functions

- **~BurgerTopping** ()
  *Destructor for **BurgerTopping**.*

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **BurgerTopping** class represents a topping for a burger, which is a specific type of **Burger**.

Constructor & Destructor Documentation

BurgerTopping::BurgerTopping ()

Constructor for **BurgerTopping** to set cost to 0.0.

BurgerTopping::~BurgerTopping () `[protected]`

Destructor for **BurgerTopping**.

Member Function Documentation

virtual void BurgerTopping::decorate (Burger * *burgerTopping*) `[virtual]`

Virtual method to decorate the burger.

Parameters

burgerTopping	A pointer to the **Burger** to be decorated.

Implements **Burger** (*p.33*).

virtual double BurgerTopping::total () `[virtual]`

Virtual method to get the total cost of the burger.

Returns

The total cost of the burger.

Implements **Burger** (*p.34*).

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

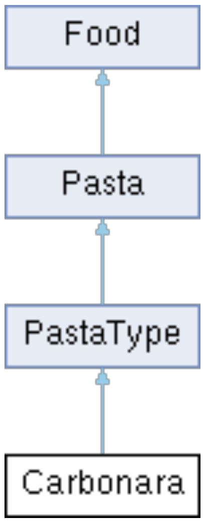
BurgerTopping.h

Carbonara Class Reference

The **Carbonara** class represents **Carbonara** pasta, which is a specific type of **PastaType**.

`#include <Carbonara.h>`

Inheritance diagram for Carbonara:



Public Member Functions

- **Carbonara** ()
  *Constructor for **Carbonara** to set its name and cost.*

- **~Carbonara** ()
  *Destructor for **Carbonara**.*

Public Member Functions inherited from PastaType

- **PastaType** ()
  *Constructor for **PastaType** to set cost to 0.0.*

- virtual double **total** ()
  Virtual method to get the total cost of the pasta.

- virtual void **decorate** (**Pasta** *pastaType)
  Virtual method to decorate the pasta.

Public Member Functions inherited from Pasta

- **Pasta** ()
  *Constructor for **Pasta** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pasta.

- void **setCost** (double cost)
  Set the cost of the pasta.

- virtual **~Pasta** ()
  *Virtual destructor for **Pasta**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pizza** *)

Additional Inherited Members

Protected Member Functions inherited from PastaType

- **~PastaType** ()
  *Destructor for **PastaType**.*

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **Carbonara** class represents **Carbonara** pasta, which is a specific type of **PastaType**.

Constructor & Destructor Documentation

Carbonara::Carbonara ()

Constructor for **Carbonara** to set its name and cost.

Carbonara::~Carbonara ()

Destructor for **Carbonara**.

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

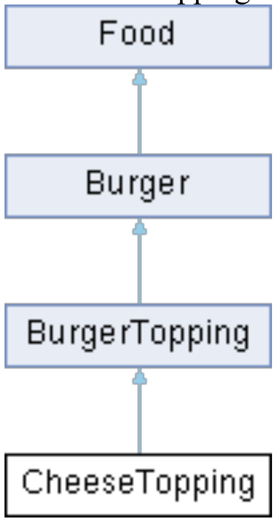
Carbonara.h

CheeseTopping Class Reference

The **CheeseTopping** class represents a cheese topping for a burger, which is a specific type of **BurgerTopping**.

```
#include <CheeseTopping.h>
```

Inheritance diagram for CheeseTopping:



Public Member Functions

- **CheeseTopping** ()
  *Constructor for **CheeseTopping** to set its name and cost.*

- **~CheeseTopping** ()
  *Destructor for **CheeseTopping**.*

Public Member Functions inherited from BurgerTopping

- **BurgerTopping** ()
  *Constructor for **BurgerTopping** to set cost to 0.0.*

- virtual double **total** ()
  Virtual method to get the total cost of the burger.

- virtual void **decorate** (**Burger** *burgerTopping)
  Virtual method to decorate the burger.

Public Member Functions inherited from Burger

- **Burger** ()
  *Constructor for **Burger** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the burger.

- void **setCost** (double cost)
  Set the cost of the burger.

- virtual **~Burger** ()
  *Virtual destructor for **Burger**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Member Functions inherited from BurgerTopping

- **~BurgerTopping** ()
  *Destructor for **BurgerTopping**.*

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **CheeseTopping** class represents a cheese topping for a burger, which is a specific type of **BurgerTopping**.

Constructor & Destructor Documentation

CheeseTopping::CheeseTopping ()

Constructor for **CheeseTopping** to set its name and cost.

CheeseTopping::~CheeseTopping ()

Destructor for **CheeseTopping**.

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

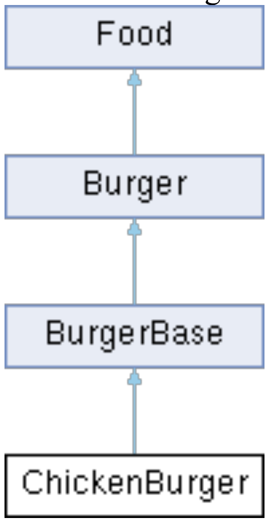
CheeseTopping.h

ChickenBurger Class Reference

The **ChickenBurger** class represents a chicken burger, which is a specific type of **BurgerBase**.

```
#include <ChickenBurger.h>
```

Inheritance diagram for ChickenBurger:



Public Member Functions

- **ChickenBurger** ()
  *Constructor for **ChickenBurger** to set its name and cost.*

Public Member Functions inherited from BurgerBase

- **BurgerBase** ()
  *Constructor for **BurgerBase**.*
- virtual double **total** ()
  Virtual method to get the total cost of the burger.
- virtual void **decorate** (**Burger** *)
  Virtual method to decorate the burger.
- **~BurgerBase** ()
  *Destructor for **BurgerBase**.*

Public Member Functions inherited from Burger

- **Burger** ()
  *Constructor for **Burger** to set its cost to 0.0.*
- double **getCost** ()
  Get the cost of the burger.

- void **setCost** (double cost)
  Set the cost of the burger.

- virtual **~Burger** ()
  *Virtual destructor for **Burger**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **ChickenBurger** class represents a chicken burger, which is a specific type of **BurgerBase**.

Constructor & Destructor Documentation

ChickenBurger::ChickenBurger ()

Constructor for **ChickenBurger** to set its name and cost.

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

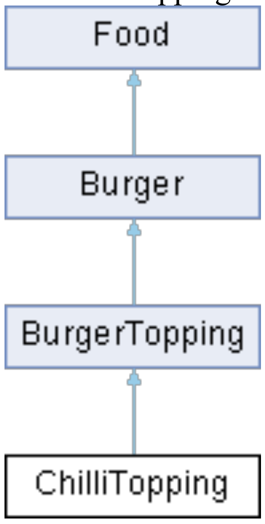
ChickenBurger.h

ChilliTopping Class Reference

The **ChilliTopping** class represents a chili topping for a burger, which is a specific type of **BurgerTopping**.

`#include <ChilliTopping.h>`

Inheritance diagram for ChilliTopping:



Public Member Functions

- **ChilliTopping** ()
  *Constructor for **ChilliTopping** to set its name and cost.*

- **~ChilliTopping** ()
  *Destructor for **ChilliTopping**.*

Public Member Functions inherited from BurgerTopping

- **BurgerTopping** ()
  *Constructor for **BurgerTopping** to set cost to 0.0.*

- virtual double **total** ()
  Virtual method to get the total cost of the burger.

- virtual void **decorate** (**Burger** *burgerTopping)
  Virtual method to decorate the burger.

Public Member Functions inherited from Burger

- **Burger** ()
  *Constructor for **Burger** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the burger.

- void **setCost** (double cost)
  Set the cost of the burger.

- virtual **~Burger** ()
  *Virtual destructor for **Burger**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Member Functions inherited from BurgerTopping

- **~BurgerTopping** ()
  *Destructor for **BurgerTopping**.*

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **ChilliTopping** class represents a chili topping for a burger, which is a specific type of **BurgerTopping**.

Constructor & Destructor Documentation

ChilliTopping::ChilliTopping ()

Constructor for **ChilliTopping** to set its name and cost.

ChilliTopping::~ChilliTopping ()

Destructor for **ChilliTopping**.

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

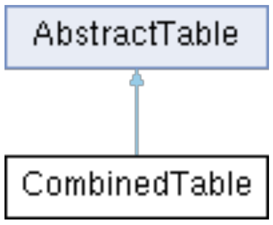
ChilliTopping.h

CombinedTable Class Reference

Represents a combined table that can group multiple **AbstractTable** instances.

`#include <CombinedTable.h>`

Inheritance diagram for CombinedTable:



Public Member Functions

- **CombinedTable** ()
  *Constructor for the **CombinedTable** class.*

- **~CombinedTable** ()
  *Destructor for the **CombinedTable** class.*

- bool **AddTable** (**AbstractTable** *table)
  *Add an **AbstractTable** to the combined table.*

- **AbstractTable** * **SeparateTable** ()
  *Separate an **AbstractTable** from the combined table.*

- vector< **Order** * > **PlaceOrder** ()
  Place orders for all tables in the combined table.

Public Member Functions inherited from AbstractTable

- int **getTableID** ()
  *Get the **Table** I D object.*

- void **setTableID** (int ID)
  *Set the **Table** ID.*

- void **setOccupied** (bool o)
  Set the Occupied object.

- bool **getOccupied** ()
  Get the Occupied object.

- int **getMaxPeople** ()
  Get the MaxPeople allowed in on the table.

- bool **visitTable** ()
  Set the Max People object.

- void **setMaxPeople** (int **maxPeople**)
- **TableState * getTableState** ()
- void **setTableState** (**TableState** *state)
- **CustomerGroup * getCustomerGroup** ()
- void **setCustomerGroup** (**CustomerGroup *customerGroup**)
- int **getCurrentPeople** ()
- void **setCurrentPeople** (int **currentPeople**)
- void **ReceiveOrder** (vector< **Order** * > orders)
- int **getRandomState** ()
- void **setRandomState** (int **RandomState**)
- string **EnquireState** ()
- **AbstractTable** ()
- virtual **~AbstractTable** ()
- bool **payBill** ()
- vector< **Review** > **ReviewFood** ()
- vector< **Review** > **ReviewService** ()

Additional Inherited Members

Protected Attributes inherited from AbstractTable

- int **maxPeople**
- **TableState * tableState**
- **CustomerGroup * customerGroup**
- int **currentPeople**
- int **RandomState**
- int **tableID**
- bool **occupied** =false

Static Protected Attributes inherited from AbstractTable

- static int **counter**

Detailed Description

Represents a combined table that can group multiple **AbstractTable** instances.

Constructor & Destructor Documentation

CombinedTable::CombinedTable ()

Constructor for the **CombinedTable** class.

CombinedTable::~CombinedTable ()

Destructor for the **CombinedTable** class.

Member Function Documentation

bool CombinedTable::AddTable (AbstractTable * *table*) `[virtual]`

Add an **AbstractTable** to the combined table.

Parameters

table	A pointer to the **AbstractTable** to be added.

Returns

true if the addition was successful, false otherwise.

Implements **AbstractTable** (*p.9*).

vector< Order * > CombinedTable::PlaceOrder () `[virtual]`

Place orders for all tables in the combined table.

Returns

A vector of **Order** pointers representing the placed orders.

Reimplemented from **AbstractTable** (*p.10*).

AbstractTable * CombinedTable::SeparateTable () `[virtual]`

Separate an **AbstractTable** from the combined table.

Returns

A pointer to the separated **AbstractTable**, or NULL if the combined table is empty.

Implements **AbstractTable** (*p.10*).

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

CombinedTable.h

Customer Class Reference

The **Customer** class.
`#include <Customer.h>`

Public Member Functions

- string **getID** ()
  Get the ID of the customer.
- void **setID** (string ID)
  Set the ID of the customer.
- void **setState** (**CustomerState** *state)
  Set the state of the customer.
- **CustomerState** * **getState** ()
  Get the state of the customer.
- string **GiveComment_Food** ()
  Give a comment about the food.
- string **GiveComment_Service** ()
  Give a comment about the service.
- int **GiveRating_Food** ()
  Give a rating for the food.
- int **GiveRating_Service** ()
  Give a rating for the service.
- **Customer** (string name)
  Construct a new ***Customer*** *object.*
- **Customer** ()
  Construct a new ***Customer*** *object.*
- void **receiveOrder** (**Order** *order)
  Receive an order.
- **Order** * **PlaceOrder** ()
  Place an order.

Static Public Attributes

- static int **SeedValue**

Detailed Description

The **Customer** class.

This class represents a customer in a restaurant.

Constructor & Destructor Documentation

Customer::Customer (string *name*) `[inline]`

Construct a new **Customer** object.

Parameters

name	The name of the customer.

Customer::Customer ()

Construct a new **Customer** object.

Member Function Documentation

string Customer::getID () `[inline]`

Get the ID of the customer.

Returns

The ID of the customer.

CustomerState * Customer::getState () `[inline]`

Get the state of the customer.

Returns

The state of the customer.

string Customer::GiveComment_Food ()

Give a comment about the food.

Returns

The comment about the food.

string Customer::GiveComment_Service ()

Give a comment about the service.

Returns

The comment about the service.

int Customer::GiveRating_Food ()

Give a rating for the food.

Returns

The rating for the food.

int Customer::GiveRating_Service ()

Give a rating for the service.

Returns

The rating for the service.

Order * Customer::PlaceOrder ()

Place an order.

Returns

The order placed.

void Customer::receiveOrder (Order * *order*)

Receive an order.

Parameters

order	The order to receive.

void Customer::setID (string *ID*)`[inline]`

Set the ID of the customer.

Parameters

ID	The ID to set.

void Customer::setState (CustomerState * *state*)`[inline]`

Set the state of the customer.

Parameters

state	The state to set.

Member Data Documentation

int Customer::SeedValue `[static]`

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

Customer.h

CustomerGroup Class Reference

Represents a group of customers in a restaurant.
`#include <CustomerGroup.h>`

Public Member Functions

- vector< **Customer** > **getCustomers** ()
  Get the customers in the group.
- void **setCustomers** (vector< **Customer** > customer)
  Set the customers in the group.
- int **getRandomState** ()
  Get the random state of the group.
- void **decrementRandomState** ()
- void **setRandomState** (int **RandomState**)
  Set the random state of the group.
- int **NumOfCustomer** ()
  Get the number of customers in the group.
- **Customer CustomerAt** (int index)
  Get a specific customer in the group.
- vector< **Bill** * > **mergeBill** ()
  Merge the bills of the customer group into a vector of bills.
- bool **addCustomer** (**Customer** customer)
  Add a customer to the customer group.
- **CustomerGroup** ()
  *Default constructor for the **CustomerGroup** class.*
- void **receiveOrder** (vector< **Order** * > **orders**)
  Receive orders for the group and update customer states based on food quality.
- bool **PayBill** ()
  Pay the bills of the customer group.
- vector< **Review** > **ReviewFood** ()
  Generate food reviews for the customer group.
- vector< **Review** > **ReviewService** ()
  Generate service reviews for the customer group.

- vector< **Order** * > **PlaceOrder** ()
  Place orders for each customer in the group.

- void **print** ()
  Print the IDs of the customers in the group.

Protected Attributes

- vector< **Customer** > **customers**
- int **RandomState**
- vector< **Order** * > **orders**

Detailed Description

Represents a group of customers in a restaurant.

Constructor & Destructor Documentation

CustomerGroup::CustomerGroup ()

Default constructor for the **CustomerGroup** class.

Member Function Documentation

bool CustomerGroup::addCustomer (Customer *customer*)

Add a customer to the customer group.

Parameters

customer	The **Customer** object to be added.

Returns

true if the addition was successful, false otherwise.

Customer CustomerGroup::CustomerAt (int *index*)

Get a specific customer in the group.

Parameters

index	The index of the customer to retrieve.

Returns

The **Customer** object at the specified index.

void CustomerGroup::decrementRandomState () `[inline]`

vector< Customer > CustomerGroup::getCustomers ()

Get the customers in the group.

Returns

A vector of **Customer** objects in the group.

int CustomerGroup::getRandomState ()

Get the random state of the group.

Returns

The random state as an integer.

vector< Bill * > CustomerGroup::mergeBill ()

Merge the bills of the customer group into a vector of bills.

Returns

A vector of **Bill** pointers representing merged bills.

int CustomerGroup::NumOfCustomer ()

Get the number of customers in the group.

Returns

The number of customers in the group.

bool CustomerGroup::PayBill ()

Pay the bills of the customer group.

Returns

true if the bills were paid successfully, false otherwise.

vector< Order * > CustomerGroup::PlaceOrder ()

Place orders for each customer in the group.

Returns

A vector of **Order** pointers representing the placed orders.

void CustomerGroup::print ()

Print the IDs of the customers in the group.

void CustomerGroup::receiveOrder (vector< Order * > *orders*)

Receive orders for the group and update customer states based on food quality.

Parameters

orders	A vector of **Order** pointers to be received.

vector< Review > CustomerGroup::ReviewFood ()

Generate food reviews for the customer group.

Returns

A vector of **Review** objects representing food reviews.

vector< Review > CustomerGroup::ReviewService ()

Generate service reviews for the customer group.

Returns

A vector of **Review** objects representing service reviews.

void CustomerGroup::setCustomers (vector< Customer > *customer*)

Set the customers in the group.

Parameters

customer	A vector of **Customer** objects to set as the group.

void CustomerGroup::setRandomState (int *RandomState*)

Set the random state of the group.

Parameters

RandomState	The random state to be set.

Member Data Documentation

vector<Customer> CustomerGroup::customers `[protected]`

vector<Order*> CustomerGroup::orders `[protected]`

int CustomerGroup::RandomState `[protected]`

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

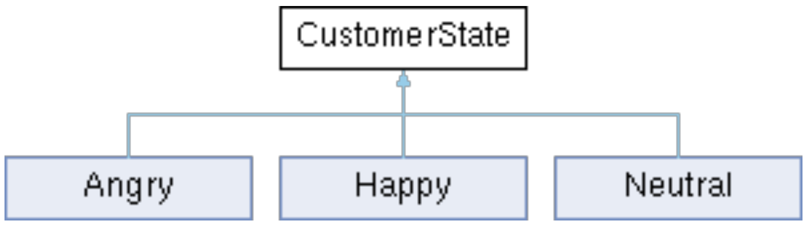
CustomerGroup.h

CustomerState Class Reference

Represents the state of a customer in a restaurant.

`#include <CustomerState.h>`

Inheritance diagram for CustomerState:



Public Member Functions

- virtual string **getStatus** ()=0
  Get the status of the customer state.

- virtual void **action** ()=0
  Perform an action associated with the customer state.

Detailed Description

Represents the state of a customer in a restaurant.

Member Function Documentation

virtual void CustomerState::action () `[pure virtual]`

Perform an action associated with the customer state.

Implemented in **Angry** (*p.15*), **Happy** (*p.81*), and **Neutral** (*p.100*).

virtual string CustomerState::getStatus () `[pure virtual]`

Get the status of the customer state.

Returns

The status as a string.

Implemented in **Angry** (*p.15*), **Happy** (*p.81*), and **Neutral** (*p.100*).

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

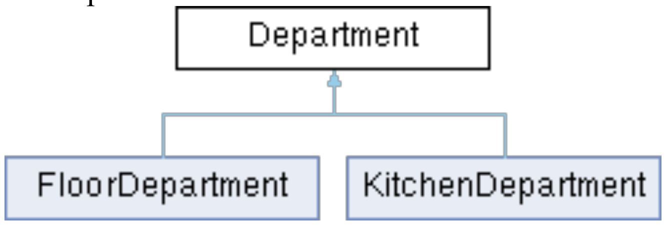
CustomerState.h

Department Class Reference

The **Department** class.

```
#include <Department.h>
```

Inheritance diagram for Department:



Public Member Functions

- virtual void **TakeReview** (const **Review** &review)=0
  Take a review.
- virtual void **DisplayReviews** ()=0
  Display the reviews.
- virtual double **CalculateAverageRating** () const =0
  Calculate the average rating.
- virtual void **DeleteReview** (const **Review** &review)=0
  Delete a review.

Protected Attributes

- std::vector< **Review** > **reviews**

Detailed Description

The **Department** class.

This class represents a department in a restaurant.

Member Function Documentation

virtual double Department::CalculateAverageRating () const `[pure virtual]`

Calculate the average rating.

Returns

The average rating.

Implemented in **FloorDepartment** (*p.76*), and **KitchenDepartment** (*p.86*).

virtual void Department::DeleteReview (const Review & *review*) `[pure virtual]`

Delete a review.

Parameters

review	The review to delete.

Implemented in **FloorDepartment** (*p.76*), and **KitchenDepartment** (*p.86*).

virtual void Department::DisplayReviews () `[pure virtual]`

Display the reviews.

Implemented in **FloorDepartment** (*p.76*), and **KitchenDepartment** (*p.86*).

virtual void Department::TakeReview (const Review & *review*) `[pure virtual]`

Take a review.

Parameters

review	The review to take.

Implemented in **FloorDepartment** (*p.76*), and **KitchenDepartment** (*p.86*).

Member Data Documentation

std::vector<Review> Department::reviews `[protected]`

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

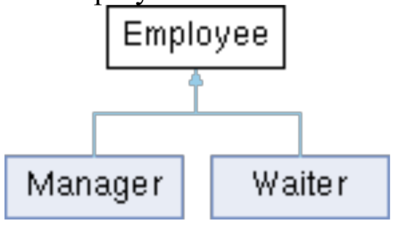
Department.h

Employee Class Reference

The **Employee** class.

`#include <Employee.h>`

Inheritance diagram for Employee:



Public Member Functions

- **Employee** (int id)
  *Construct a new **Employee** object.*

- virtual void **assignTables** (std::vector< **Table** * > **&tables**)
  Assign tables to the employee.

- virtual void **iterateTables** ()
  Iterate through the tables.

- void **moveToNextTable** ()
  Move to the next table.

- **Department** * **getDepartment** ()
  Get the department of the employee.

- **Table** * **getCurrentTable** ()
  Get the current table.

- void **setDepartment** (**Department** *dep)
  Set the department of the employee.

- void **setCurrTable** (**Table** *currTab)
  Set the current table.

- void **GetReview** (const std::vector< **Review** * > &reviewList)

Get the reviews.

- void **TakeOrder** (**Table** *table)
  Take an order.

- int **getEmployeeId** ()
  Get the ID of the employee.

- **~Employee** ()
  *Destroy the **Employee** object.*

Protected Attributes

- **Department * department**
- **Table * tables**
- **Table * currTable**
- **TableIterator * tableIterator**
- int **employeeId**

Detailed Description

The **Employee** class.

This class represents an employee in a restaurant.

Constructor & Destructor Documentation

Employee::Employee (int *id*)

Construct a new **Employee** object.

Parameters

id	The ID of the employee.

Employee::~Employee ()

Destroy the **Employee** object.

Member Function Documentation

virtual void Employee::assignTables (std::vector< Table * > & *tables*)`[virtual]`

Assign tables to the employee.

Parameters

tables	The tables to assign.

Table * Employee::getCurrentTable ()

Get the current table.

Returns

The current table.

Department * Employee::getDepartment ()

Get the department of the employee.

Returns

The department of the employee.

int Employee::getEmployeeId ()

Get the ID of the employee.

Returns

The ID of the employee.

void Employee::GetReview (const std::vector< Review * > & *reviewList*)

Get the reviews.

Parameters

reviewList	The list of reviews.

virtual void Employee::iterateTables () `[virtual]`

Iterate through the tables.

Reimplemented in **Waiter** (*p.161*).

void Employee::moveToNextTable ()

Move to the next table.

void Employee::setCurrTable (Table * *currTab*)

Set the current table.

Parameters

currTab	The current table to set.

void Employee::setDepartment (Department * *dep*)

Set the department of the employee.

Parameters

dep	The department to set.

void Employee::TakeOrder (Table * *table*)

Take an order.

Parameters

table	The table to take the order from.

Member Data Documentation

Table* Employee::currTable `[protected]`

Department* Employee::department `[protected]`

int Employee::employeeId `[protected]`

TableIterator* Employee::tableIterator `[protected]`

Table* Employee::tables `[protected]`

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

Employee.h

Floor Class Reference

This is the interface for floor.
`#include <Floor.h>`

Public Member Functions

- **Floor** (int)
  *Construct a new **Floor** object. Passes in the number of tables in the floor.*
- **Employee** * **createWaiter** ()
  *Create a **Waiter** object, and adds it to the list of waiters. Number of waiters cannot exceed number of tables returns null if waiters reached capacity.*
- **Employee** * **createManager** ()
  *Create a **Manager** object. If manager already exists, then current manager is returned.*
- bool **hasAvailableWaiter** ()
- bool **addCustomerGroup** (**CustomerGroup** *)
  Adds customer group to tables and assigns the group to a waiter.
- void **waiterIterateTables** ()
  performs one cycle of the waiter iteration
- void **reorderMaxTablesForWaiters** ()
- void **printTablesAndWaiters** ()

Protected Attributes

- std::vector< **Table** * > **tables**
- std::vector< **Employee** * > **waiters**
- **Manager** * **manager**
- int **capacity**
- int **numOccupiedTables**
- int **numAvailableWaiters**

Detailed Description

This is the interface for floor.

Constructor & Destructor Documentation

Floor::Floor (int)

Construct a new **Floor** object. Passes in the number of tables in the floor.

Member Function Documentation

bool Floor::addCustomerGroup (CustomerGroup *)

Adds customer group to tables and assigns the group to a waiter.

Returns

true if customer group is added

false if restaurant is full

Employee * Floor::createManager ()

Create a **Manager** object. If manager already exists, then current manager is returned.

Returns

Employee*

Employee * Floor::createWaiter ()

Create a **Waiter** object, and adds it to the list of waiters. Number of waiters cannot exceed number of tables returns null if waiters reached capacity.

Returns

Employee*

bool Floor::hasAvailableWaiter ()

void Floor::printTablesAndWaiters () `[inline]`

void Floor::reorderMaxTablesForWaiters ()

void Floor::waiterIterateTables ()

performs one cycle of the waiter iteration

Member Data Documentation

int Floor::capacity `[protected]`

Manager* Floor::manager `[protected]`

int Floor::numAvailableWaiters `[protected]`

int Floor::numOccupiedTables `[protected]`

std::vector<Table*> Floor::tables `[protected]`

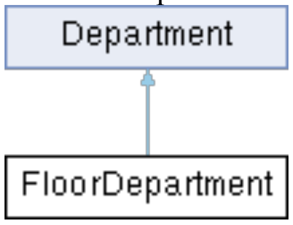
std::vector<Employee*> Floor::waiters `[protected]`

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

Floor.h

FloorDepartment Class Reference

The **FloorDepartment** class.
`#include <FloorDepartment.h>`
Inheritance diagram for FloorDepartment:



Public Member Functions

- void **TakeReview** (const **Review** &review) override
  Take a review.

- void **DisplayReviews** () override
  Display the reviews.

- double **CalculateAverageRating** () const override
  Calculate the average rating.

- void **DeleteReview** (const **Review** &review) override
  Delete a review.

Additional Inherited Members

Protected Attributes inherited from Department

- std::vector< **Review** > **reviews**

Detailed Description

The **FloorDepartment** class.

This class represents the floor department of a restaurant.

Member Function Documentation

double FloorDepartment::CalculateAverageRating () const`[override]`, `[virtual]`

Calculate the average rating.

Returns

The average rating.

Implements **Department** (*p.67*).

void FloorDepartment::DeleteReview (const Review & *review*)`[override]`, `[virtual]`

Delete a review.

Parameters

review	The review to delete.

Implements **Department** (*p.67*).

void FloorDepartment::DisplayReviews ()`[override]`, `[virtual]`

Display the reviews.

Implements **Department** (*p.67*).

void FloorDepartment::TakeReview (const Review & *review*)`[override]`, `[virtual]`

Take a review.

Parameters

review	The review to take.

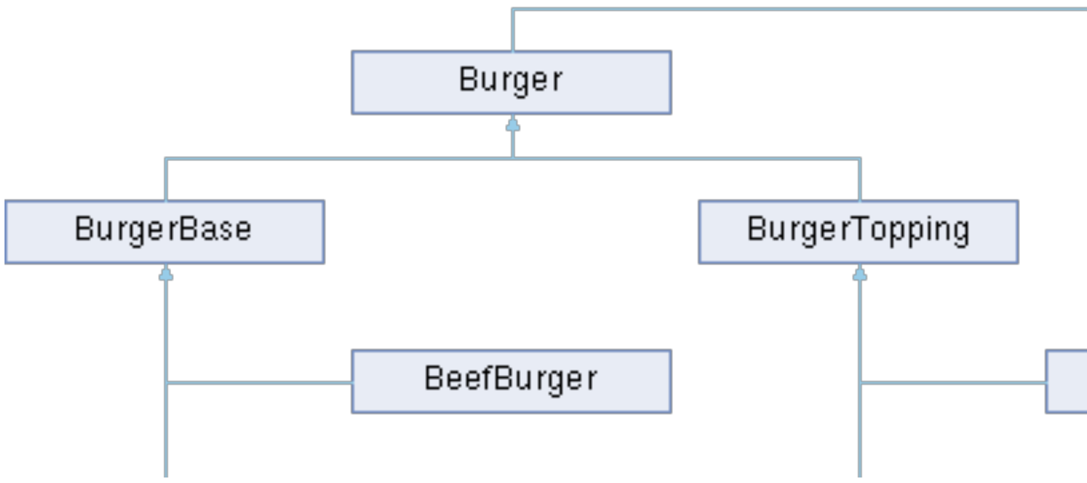
Implements **Department** (*p.67*).

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

FloorDepartment.h

Food Class Reference

The **Food** class.
`#include <Food.h>`
Inheritance diagram for Food:



Public Member Functions

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- virtual double **total** ()=0
  Calculate the total cost of the food.

- double **getCost** ()
- void **setCost** (double **cost**)
- virtual ~**Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Protected Attributes

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **Food** class.

This class represents a food item.

Constructor & Destructor Documentation

Food::Food ()

Construct a new **Food** object.

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

Member Function Documentation

void Food::addIngredient (string *ingredient*)

virtual void Food::decorate (Burger *) `[virtual]`

Reimplemented in **BurgerBase** (*p.36*), **Burger** (*p.33*), and **BurgerTopping** (*p.39*).

virtual void Food::decorate (Pasta *) `[virtual]`

Reimplemented in **pastaBase** (*p.116*), **Pasta** (*p.113*), and **PastaType** (*p.119*).

virtual void Food::decorate (Pizza *) `[virtual]`

Reimplemented in **PizzaBase** (*p.130*), **Pizza** (*p.127*), and **PizzaType** (*p.133*).

double Food::getCost ()

int Food::getFoodQuality ()

string Food::getName ()

void Food::setCost (double *cost*)

void Food::setFoodQuality (int)

void Food::setName (string *name*)

virtual double Food::total () `[pure virtual]`

Calculate the total cost of the food.

Returns

The total cost of the food.

Implemented in **BurgerBase** (*p.37*), **BurgerTopping** (*p.40*), **pastaBase** (*p.117*), **PastaType** (*p.120*), **PizzaBase** (*p.131*), **PizzaType** (*p.134*), **Burger** (*p.34*), **Pasta** (*p.114*), and **Pizza** (*p.128*).

Member Data Documentation

double Food::cost `[protected]`

vector<string> Food::ingredients `[protected]`

string Food::name `[protected]`

int Food::RandomFoodQuality `[protected]`

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

Food.h

FoodItem Struct Reference

Represents a food item with name, price, preparation method, and type.
`#include <Menu.h>`

Public Member Functions

- **FoodItem** (string, int, string, string)
- **~FoodItem** ()

Public Attributes

- string **name**
- int **price**
- string **method**
- string **type**

Detailed Description

Represents a food item with name, price, preparation method, and type.

Constructor & Destructor Documentation

FoodItem::FoodItem (string , int , string , string)

FoodItem::~FoodItem ()

Member Data Documentation

string FoodItem::method

The preparation method of the food item.

string FoodItem::name

The name of the food item.

int FoodItem::price

The price of the food item.

string FoodItem::type

The type of food item (e.g., **Burger**, **Pasta**, etc.).

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

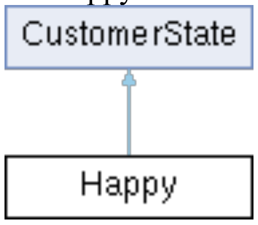
Menu.h

Happy Class Reference

The **Happy** class.

```
#include <Happy.h>
```

Inheritance diagram for Happy:



Public Member Functions

- string **getStatus** ()
  Get the status of the customer.

- void **action** ()
  *Perform the action for the **Happy** state.*

Detailed Description

The **Happy** class.

This class represents the **Happy** state of a customer.

Member Function Documentation

void Happy::action () `[inline]`, `[virtual]`

Perform the action for the **Happy** state.

Implements **CustomerState** (*p.64*).

string Happy::getStatus () `[inline]`, `[virtual]`

Get the status of the customer.

Returns

The status of the customer.

Implements **CustomerState** (*p.64*).

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

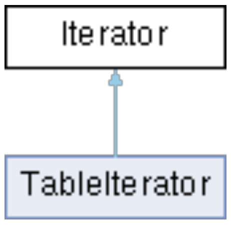
Happy.h

Iterator Class Reference

The **Iterator** class.

```
#include <Iterator.h>
```

Inheritance diagram for Iterator:



Public Member Functions

- virtual **Table** * **first** ()=0
  Get the first table.
- virtual **Table** * **next** ()=0
  Get the next table.
- virtual bool **hasNext** ()=0
  Check if there is a next table.
- virtual **Table** * **current** ()=0
  Get the current table.

Detailed Description

The **Iterator** class.

This class represents an iterator for a collection of tables.

Member Function Documentation

virtual Table * Iterator::current () `[pure virtual]`

Get the current table.

Returns

The current table.

Implemented in **TableIterator** (*p.149*).

virtual Table * Iterator::first () `[pure virtual]`

Get the first table.

Returns

The first table.

Implemented in **TableIterator** (*p.149*).

virtual bool Iterator::hasNext () `[pure virtual]`

Check if there is a next table.

Returns

True if there is a next table, false otherwise.

Implemented in **TableIterator** (*p.149*).

virtual Table * Iterator::next () `[pure virtual]`

Get the next table.

Returns

The next table.

Implemented in **TableIterator** (*p.149*).

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

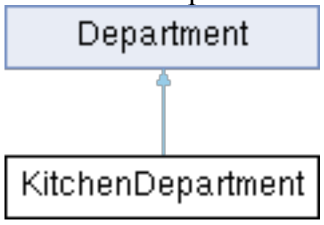
Iterator.h

KitchenDepartment Class Reference

The **KitchenDepartment** class.

`#include <KitchenDepartment.h>`

Inheritance diagram for KitchenDepartment:



Public Member Functions

- void **TakeReview** (const **Review** &review) override
  Take a review.

- void **DisplayReviews** () override
  Display the reviews.

- double **CalculateAverageRating** () const override
  Calculate the average rating.

- void **DeleteReview** (const **Review** &review) override
  Delete a review.

Additional Inherited Members

Protected Attributes inherited from Department

- std::vector< **Review** > **reviews**

Detailed Description

The **KitchenDepartment** class.

This class represents the kitchen department of a restaurant.

Member Function Documentation

double KitchenDepartment::CalculateAverageRating () const`[override]`, `[virtual]`

Calculate the average rating.

Returns

The average rating.

Implements **Department** (*p.67*).

void KitchenDepartment::DeleteReview (const Review & *review*)`[override]`, `[virtual]`

Delete a review.

Parameters

review	The review to delete.

Implements **Department** (*p.67*).

void KitchenDepartment::DisplayReviews ()`[override]`, `[virtual]`

Display the reviews.

Implements **Department** (*p.67*).

void KitchenDepartment::TakeReview (const Review & *review*)`[override]`, `[virtual]`

Take a review.

Parameters

review	The review to take.

Implements **Department** (*p.67*).

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

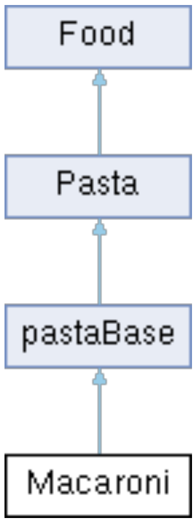
KitchenDepartment.h

Macaroni Class Reference

The **Macaroni** class represents macaroni pasta, which is a specific type of PastaBase.

```
#include <Macaroni.h>
```

Inheritance diagram for Macaroni:



Public Member Functions

- **Macaroni** ()
  *Constructor for **Macaroni** to set its name and cost.*

Public Member Functions inherited from pastaBase

- **pastaBase** ()
  *Constructor for **pastaBase**.*

- virtual double **total** ()
  Returns the cost of the pasta.

- virtual void **decorate** (**Pasta** *)
  Virtual method to decorate the pasta.

- **~pastaBase** ()
  *Destructor for **pastaBase**.*

Public Member Functions inherited from Pasta

- **Pasta** ()
  *Constructor for **Pasta** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pasta.

- void **setCost** (double cost)
  Set the cost of the pasta.

- virtual **~Pasta** ()
  *Virtual destructor for **Pasta**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pizza** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **Macaroni** class represents macaroni pasta, which is a specific type of PastaBase.

Constructor & Destructor Documentation

Macaroni::Macaroni ()

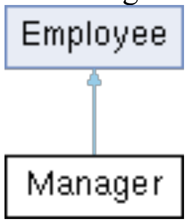
Constructor for **Macaroni** to set its name and cost.

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

Macaroni.h

Manager Class Reference

The **Manager** class.
`#include <Manager.h>`
Inheritance diagram for Manager:



Public Member Functions

- **Manager** (int id)
  *Construct a new **Manager** object.*

- void **getReviewsForFloorDepartment** ()
  Get the reviews for the floor department.

Public Member Functions inherited from Employee

- **Employee** (int id)
  *Construct a new **Employee** object.*

- virtual void **assignTables** (std::vector< **Table** * > **&tables**)
  Assign tables to the employee.

- virtual void **iterateTables** ()
  Iterate through the tables.

- void **moveToNextTable** ()
  Move to the next table.

- **Department** * **getDepartment** ()
  Get the department of the employee.

- **Table** * **getCurrentTable** ()
  Get the current table.

- void **setDepartment** (**Department** *dep)
  Set the department of the employee.

- void **setCurrTable** (**Table** *currTab)
  Set the current table.

- void **GetReview** (const std::vector< **Review** * > &reviewList)
  Get the reviews.

- void **TakeOrder** (**Table** *table)
  Take an order.

- int **getEmployeeId** ()
  Get the ID of the employee.

- **~Employee** ()
  *Destroy the **Employee** object.*

Additional Inherited Members

Protected Attributes inherited from Employee

- **Department * department**
- **Table * tables**
- **Table * currTable**
- **TableIterator * tableIterator**
- int **employeeId**

Detailed Description

The **Manager** class.

This class represents a manager in a restaurant.

Constructor & Destructor Documentation

Manager::Manager (int *id*)

Construct a new **Manager** object.

Parameters

id	The ID of the manager.

Member Function Documentation

void Manager::getReviewsForFloorDepartment ()

Get the reviews for the floor department.

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

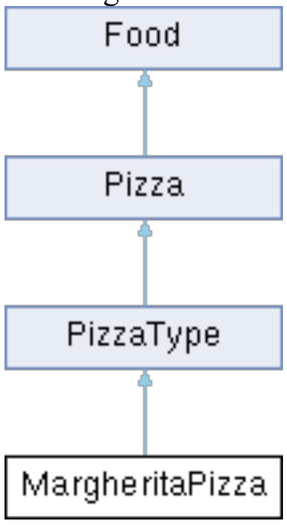
Manager.h

MargheritaPizza Class Reference

The **MargheritaPizza** class represents a Margherita pizza type, which is a specific type of **Pizza**.

```
#include <MargheritaPizza.h>
```

Inheritance diagram for MargheritaPizza:



Public Member Functions

- **MargheritaPizza** ()
  *Constructor for **MargheritaPizza** to set its name and cost.*

- **~MargheritaPizza** ()
  *Destructor for **MargheritaPizza**.*

Public Member Functions inherited from PizzaType

- **PizzaType** ()
  *Constructor for **PizzaType**.*

- virtual double **total** ()
  Returns the total cost of the pizza.

- virtual void **decorate** (**Pizza** *pizzaType)
  Decorates the pizza.

Public Member Functions inherited from Pizza

- **Pizza** ()
  *Constructor for **Pizza** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pizza.

- void **setCost** (double cost)
  Set the cost of the pizza.

- virtual **~Pizza** ()
  *Virtual destructor for **Pizza**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Member Functions inherited from PizzaType

- **~PizzaType** ()
  *Destructor for **PizzaType**.*

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **MargheritaPizza** class represents a Margherita pizza type, which is a specific type of **Pizza**.

Constructor & Destructor Documentation

MargheritaPizza::MargheritaPizza ()

Constructor for **MargheritaPizza** to set its name and cost.

MargheritaPizza::~MargheritaPizza ()

Destructor for **MargheritaPizza**.

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

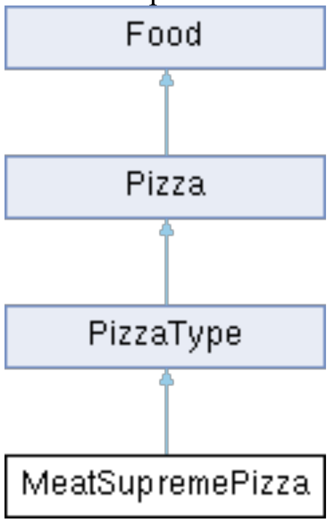
MargheritaPizza.h

MeatSupremePizza Class Reference

The **MeatSupremePizza** class represents a Meat Supreme pizza type, which is a specific type of **Pizza**.

```
#include <MeatSupremePizza.h>
```

Inheritance diagram for MeatSupremePizza:



Public Member Functions

- **MeatSupremePizza** ()
  Constructor for ***MeatSupremePizza*** *to set its name and cost.*

- **~MeatSupremePizza** ()
  Destructor for ***MeatSupremePizza****.*

Public Member Functions inherited from PizzaType

- **PizzaType** ()
  Constructor for ***PizzaType****.*

- virtual double **total** ()
  Returns the total cost of the pizza.

- virtual void **decorate** (**Pizza** *pizzaType)
  Decorates the pizza.

Public Member Functions inherited from Pizza

- **Pizza** ()
  Constructor for ***Pizza*** *to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pizza.

- void **setCost** (double cost)
  Set the cost of the pizza.

- virtual **~Pizza** ()
  *Virtual destructor for **Pizza**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Member Functions inherited from PizzaType

- **~PizzaType** ()
  *Destructor for **PizzaType**.*

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **MeatSupremePizza** class represents a Meat Supreme pizza type, which is a specific type of **Pizza**.

Constructor & Destructor Documentation

MeatSupremePizza::MeatSupremePizza ()

Constructor for **MeatSupremePizza** to set its name and cost.

MeatSupremePizza::~MeatSupremePizza ()

Destructor for **MeatSupremePizza**.

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

MeatSupremePizza.h

Menu Class Reference

Represents a menu for a restaurant.
`#include <Menu.h>`

Public Member Functions

- string **printMenu** ()
  Prints the menu.
- **~Menu** ()
  *Destroys the **Menu** instance.*
- **FoodItem** * **getFoodItem** ()
  returns a foodItem, for testing purposes

Static Public Member Functions

- static **Menu** * **getMenu** ()
  Gets the menu instance.

Public Attributes

- vector< **FoodItem** * > **menu**

Protected Member Functions

- **Menu** ()
  *Constructs a **Menu** instance.*

Detailed Description

Represents a menu for a restaurant.

Constructor & Destructor Documentation

Menu::~Menu ()

Destroys the **Menu** instance.

Menu::Menu () `[protected]`

Constructs a **Menu** instance.

Member Function Documentation

FoodItem * Menu::getFoodItem ()

returns a foodItem, for testing purposes

static Menu * Menu::getMenu () `[static]`

Gets the menu instance.

Returns

A pointer to the **Menu** instance.

string Menu::printMenu ()

Prints the menu.

Returns

A string containing the formatted menu.

Member Data Documentation

vector<FoodItem*> Menu::menu

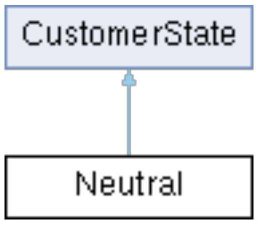
A vector to store food items in the menu.

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

Menu.h

Neutral Class Reference

The **Neutral** class.
`#include <Neutral.h>`
Inheritance diagram for Neutral:



Public Member Functions

- string **getStatus** ()
  Get the status of the customer.

- void **action** ()
  *Perform the action for the **Neutral** state.*

Detailed Description

The **Neutral** class.

This class represents the **Neutral** state of a customer.

Member Function Documentation

void Neutral::action () `[inline]`, `[virtual]`

Perform the action for the **Neutral** state.

Implements **CustomerState** (*p.64*).

string Neutral::getStatus () `[inline]`, `[virtual]`

Get the status of the customer.

Returns

The status of the customer.

Implements **CustomerState** (*p.64*).

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

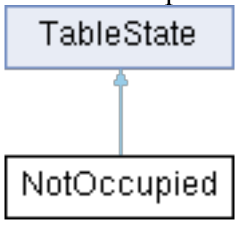
Neutral.h

NotOccupied Class Reference

The **NotOccupied** class.

```
#include <NotOccupied.h>
```

Inheritance diagram for NotOccupied:



Public Member Functions

- string **getStatus** ()
  Get the status of the table.

- bool **action** ()
  Perform the action for the ***NotOccupied*** *status.*

Public Member Functions inherited from TableState

- **TableState** ()
  Construct a new ***TableState*** *object.*

- void **setTable** (**AbstractTable *table**)
  Set the table.

Additional Inherited Members

Protected Attributes inherited from TableState

- **AbstractTable * table**

Detailed Description

The **NotOccupied** class.

This class represents the **NotOccupied** status of a table.

Member Function Documentation

bool NotOccupied::action () `[inline]`, `[virtual]`

Perform the action for the **NotOccupied** status.

Returns

False.

Implements **TableState** (*p.152*).

string NotOccupied::getStatus () `[inline]`, `[virtual]`

Get the status of the table.

Returns

The status of the table.

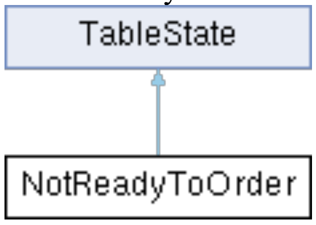
Implements **TableState** (*p.152*).

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

NotOccupied.h

NotReadyToOrder Class Reference

The **NotReadyToOrder** class.
`#include <NotReadyToOrder.h>`
Inheritance diagram for NotReadyToOrder:



Public Member Functions

- string **getStatus** ()
  Get the status of the table.

- bool **action** ()
  *Perform the action for the **NotReadyToOrder** status.*

Public Member Functions inherited from TableState

- **TableState** ()
  *Construct a new **TableState** object.*

- void **setTable** (**AbstractTable *table**)
  Set the table.

Additional Inherited Members

Protected Attributes inherited from TableState

- **AbstractTable * table**

Detailed Description

The **NotReadyToOrder** class.

This class represents the **NotReadyToOrder** status of a table.

Member Function Documentation

bool NotReadyToOrder::action () `[virtual]`

Perform the action for the **NotReadyToOrder** status.

Returns

True if the action was successful, false otherwise.

Implements **TableState** (*p.152*).

string NotReadyToOrder::getStatus () `[virtual]`

Get the status of the table.

Returns

The status of the table.

Implements **TableState** (*p.152*).

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

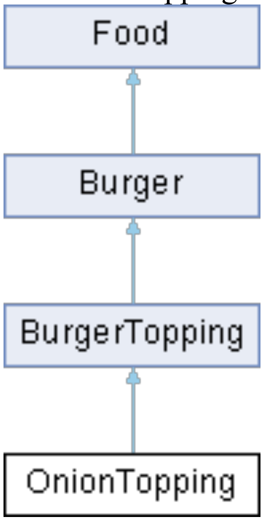
NotReadyToOrder.h

OnionTopping Class Reference

The **OnionTopping** class represents an onion topping for a burger, which is a specific type of **BurgerTopping**.

`#include <OnionTopping.h>`

Inheritance diagram for OnionTopping:



Public Member Functions

- **OnionTopping** ()
  *Constructor for **OnionTopping** to set its name and cost.*

- **~OnionTopping** ()
  *Destructor for **OnionTopping**.*

Public Member Functions inherited from BurgerTopping

- **BurgerTopping** ()
  *Constructor for **BurgerTopping** to set cost to 0.0.*

- virtual double **total** ()
  Virtual method to get the total cost of the burger.

- virtual void **decorate** (**Burger** *burgerTopping)
  Virtual method to decorate the burger.

Public Member Functions inherited from Burger

- **Burger** ()
  *Constructor for **Burger** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the burger.

- void **setCost** (double cost)
  Set the cost of the burger.

- virtual **~Burger** ()
  *Virtual destructor for **Burger**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Member Functions inherited from BurgerTopping

- **~BurgerTopping** ()
  *Destructor for **BurgerTopping**.*

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **OnionTopping** class represents an onion topping for a burger, which is a specific type of **BurgerTopping**.

Constructor & Destructor Documentation

OnionTopping::OnionTopping ()

Constructor for **OnionTopping** to set its name and cost.

OnionTopping::~OnionTopping ()

Destructor for **OnionTopping**.

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

OnionTopping.h

Order Class Reference

`#include <Order.h>`

Public Member Functions

- **Order** ()
- **~Order** ()
- std::vector< **FoodItem** * > **getItems** ()
- void **setItems** (std::vector< **FoodItem** * >)
- void **addFood** (**Food** *)
- vector< **Food** * > **getFood** ()
- **AbstractTable** * **getTable** ()
- void **setTable** (**AbstractTable** *)
- **Waiter** * **getWaiter** ()
- void **setWaiter** (**Waiter** *)
- void **setBill** (**Bill** *)
- **Bill** * **getBill** ()
- std::string **getOrderStatus** ()
- void **toReadyStatus** ()
- void **toReceivedStatus** ()
- void **toProcessingStatus** ()
- std::string **toString** ()
- void **print** ()

Constructor & Destructor Documentation

Order::Order ()

Order::~Order ()

Member Function Documentation

void Order::addFood (Food *)

Bill * Order::getBill ()

vector< Food * > Order::getFood ()

std::vector< FoodItem * > Order::getItems ()

std::string Order::getOrderStatus ()

AbstractTable * Order::getTable ()

Waiter * Order::getWaiter ()

void Order::print ()

void Order::setBill (Bill *)

void Order::setItems (std::vector< FoodItem * >)

void Order::setTable (AbstractTable *)

void Order::setWaiter (Waiter *)

void Order::toProcessingStatus ()

void Order::toReadyStatus ()

void Order::toReceivedStatus ()

std::string Order::toString ()

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

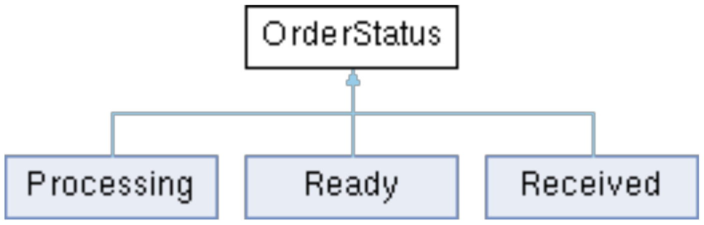
Order.h

OrderStatus Class Reference

The **OrderStatus** class.

`#include <OrderStatus.h>`

Inheritance diagram for OrderStatus:



Public Member Functions

- virtual std::string **getStatus** ()=0
  Get the status of the order.

Detailed Description

The **OrderStatus** class.

This class represents the status of an order.

Member Function Documentation

virtual std::string OrderStatus::getStatus () `[pure virtual]`

Get the status of the order.

Returns

The status of the order.

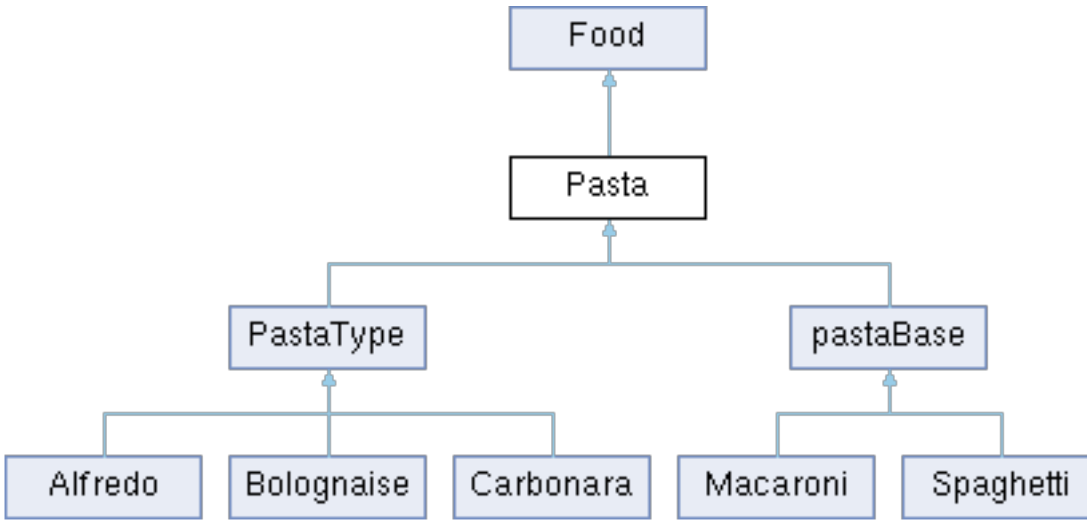
Implemented in **Processing** (*p.135*), **Ready** (*p.137*), and **Received** (*p.140*).

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

OrderStatus.h

Pasta Class Reference

The **Pasta** class represents a generic pasta dish.
`#include <Pasta.h>`
Inheritance diagram for Pasta:



Public Member Functions

- **Pasta** ()
  *Constructor for **Pasta** to set its cost to 0.0.*

- virtual void **decorate** (**Pasta** *)=0
  Virtual method to decorate the pasta.

- virtual double **total** ()=0
  Virtual method to get the total cost of the pasta.

- double **getCost** ()
  Get the cost of the pasta.

- void **setCost** (double cost)
  Set the cost of the pasta.

- virtual ~**Pasta** ()
  *Virtual destructor for **Pasta**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()

- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pizza** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **Pasta** class represents a generic pasta dish.

Constructor & Destructor Documentation

Pasta::Pasta ()

Constructor for **Pasta** to set its cost to 0.0.

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

Virtual destructor for **Pasta**.

Member Function Documentation

virtual void Pasta::decorate (Pasta *) `[pure virtual]`

Virtual method to decorate the pasta.

Parameters

pasta	A pointer to the **Pasta** to be decorated.

Reimplemented from **Food** (*p.78*).

Implemented in **pastaBase** (*p.116*), and **PastaType** (*p.119*).

double Pasta::getCost ()

Get the cost of the pasta.

Returns

The cost of the pasta.

void Pasta::setCost (double *cost*)

Set the cost of the pasta.

Parameters

cost	The cost of the pasta.

virtual double Pasta::total ()`[pure virtual]`

Virtual method to get the total cost of the pasta.

Returns

The total cost of the pasta.

Implements **Food** (*p.78*).

Implemented in **pastaBase** (*p.117*), and **PastaType** (*p.120*).

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

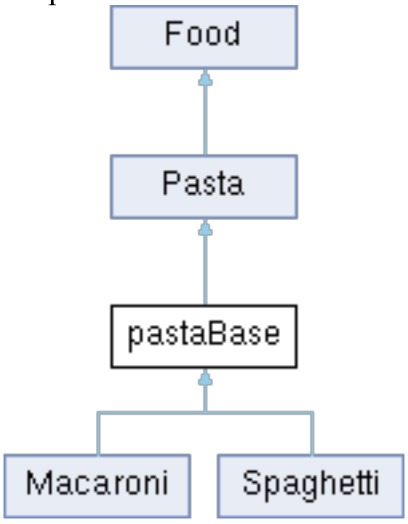
Pasta.h

pastaBase Class Reference

The **pastaBase** class represents the base of a pasta dish, which is a specific type of **Pasta**.

```
#include <pastaBase.h>
```

Inheritance diagram for pastaBase:



Public Member Functions

- **pastaBase** ()
  *Constructor for **pastaBase**.*

- virtual double **total** ()
  Returns the cost of the pasta.

- virtual void **decorate** (**Pasta** *)
  Virtual method to decorate the pasta.

- **~pastaBase** ()
  *Destructor for **pastaBase**.*

Public Member Functions inherited from Pasta

- **Pasta** ()
  *Constructor for **Pasta** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pasta.

- void **setCost** (double cost)
  Set the cost of the pasta.

- virtual ~**Pasta** ()
  *Virtual destructor for **Pasta**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pizza** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **pastaBase** class represents the base of a pasta dish, which is a specific type of **Pasta**.

Constructor & Destructor Documentation

pastaBase::pastaBase ()

Constructor for **pastaBase**.

pastaBase::~pastaBase ()

Destructor for **pastaBase**.

Member Function Documentation

virtual void pastaBase::decorate (Pasta *) `[virtual]`

Virtual method to decorate the pasta.

Parameters

pasta	A pointer to the **Pasta** to be decorated.

Implements **Pasta** (*p.113*).

virtual double pastaBase::total ()`[virtual]`

Returns the cost of the pasta.

Returns

The cost of the pasta.

Implements **Pasta** (*p.114*).

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

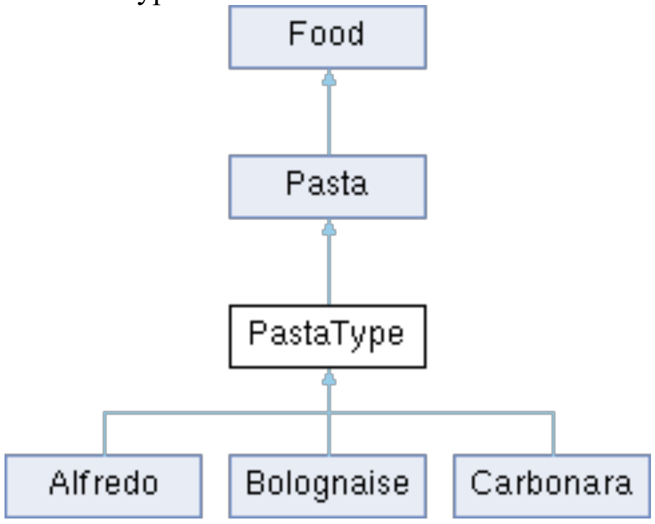
pastaBase.h

PastaType Class Reference

The **PastaType** class represents a specific type of pasta, which is a type of **Pasta**.

```
#include <PastaType.h>
```

Inheritance diagram for PastaType:



Public Member Functions

- **PastaType** ()
  *Constructor for **PastaType** to set cost to 0.0.*

- virtual double **total** ()
  Virtual method to get the total cost of the pasta.

- virtual void **decorate** (**Pasta** *pastaType)
  Virtual method to decorate the pasta.

Public Member Functions inherited from Pasta

- **Pasta** ()
  *Constructor for **Pasta** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pasta.

- void **setCost** (double cost)
  Set the cost of the pasta.

- virtual ~**Pasta** ()
  *Virtual destructor for **Pasta**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pizza** *)

Protected Member Functions

- **~PastaType** ()
  *Destructor for **PastaType**.*

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **PastaType** class represents a specific type of pasta, which is a type of **Pasta**.

Constructor & Destructor Documentation

PastaType::PastaType ()

Constructor for **PastaType** to set cost to 0.0.

PastaType::~PastaType () `[protected]`

Destructor for **PastaType**.

Member Function Documentation

virtual void PastaType::decorate (Pasta * *pastaType*) `[virtual]`

Virtual method to decorate the pasta.

Parameters

pastaType	A pointer to the **Pasta** to be decorated.

Implements **Pasta** (*p.113*).

virtual double PastaType::total ()`[virtual]`

Virtual method to get the total cost of the pasta.

Returns

The total cost of the pasta.

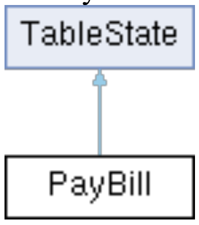
Implements **Pasta** (*p.114*).

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

PastaType.h

PayBill Class Reference

The **PayBill** class.
```
#include <PayBill.h>
```
Inheritance diagram for PayBill:



Public Member Functions

- string **getStatus** ()
  Get the status of the table.

- bool **action** ()
  *Perform the action for the **PayBill** status.*

Public Member Functions inherited from TableState

- **TableState** ()
  *Construct a new **TableState** object.*

- void **setTable** (**AbstractTable *table**)
  Set the table.

Additional Inherited Members

Protected Attributes inherited from TableState

- **AbstractTable * table**

Detailed Description

The **PayBill** class.

This class represents the **PayBill** status of a table.

Member Function Documentation

bool PayBill::action () `[virtual]`

Perform the action for the **PayBill** status.

Returns

True if the action was successful, false otherwise.

Implements **TableState** (*p.152*).

string PayBill::getStatus () `[virtual]`

Get the status of the table.

Returns

The status of the table.

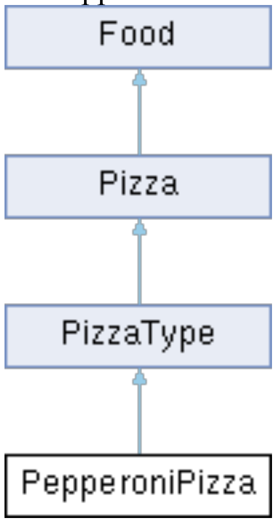
Implements **TableState** (*p.152*).

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

PayBill.h

PepperoniPizza Class Reference

The **PepperoniPizza** class represents a Pepperoni pizza type, which is a specific type of **Pizza**.
`#include <PepperoniPizza.h>`
Inheritance diagram for PepperoniPizza:



Public Member Functions

- **PepperoniPizza** ()
  *Constructor for **PepperoniPizza** to set its name and cost.*

- **~PepperoniPizza** ()
  *Destructor for **PepperoniPizza**.*

Public Member Functions inherited from PizzaType

- **PizzaType** ()
  *Constructor for **PizzaType**.*

- virtual double **total** ()
  Returns the total cost of the pizza.

- virtual void **decorate** (**Pizza** *pizzaType)
  Decorates the pizza.

Public Member Functions inherited from Pizza

- **Pizza** ()
  *Constructor for **Pizza** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pizza.

- void **setCost** (double cost)
  Set the cost of the pizza.

- virtual **~Pizza** ()
  *Virtual destructor for **Pizza**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Member Functions inherited from PizzaType

- **~PizzaType** ()
  *Destructor for **PizzaType**.*

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **PepperoniPizza** class represents a Pepperoni pizza type, which is a specific type of **Pizza**.

Constructor & Destructor Documentation

PepperoniPizza::PepperoniPizza ()

Constructor for **PepperoniPizza** to set its name and cost.

PepperoniPizza::~PepperoniPizza ()

Destructor for **PepperoniPizza**.

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

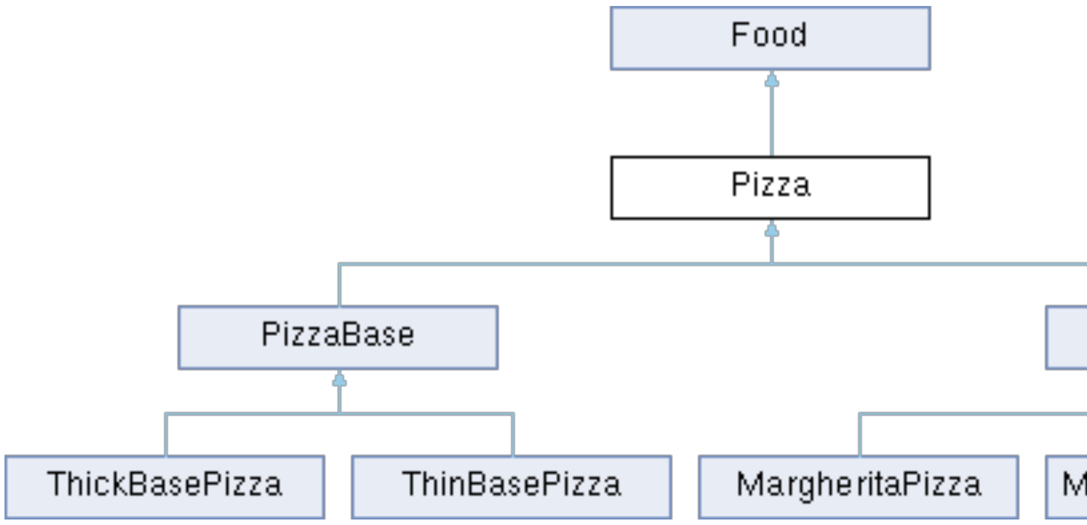
PepperoniPizza.h

Pizza Class Reference

The **Pizza** class represents a generic pizza.

`#include <Pizza.h>`

Inheritance diagram for Pizza:



Public Member Functions

- **Pizza** ()
  Constructor for ***Pizza*** *to set its cost to 0.0.*

- virtual void **decorate** (**Pizza** *)=0
  Virtual method to decorate the pizza.

- virtual double **total** ()=0
  Virtual method to get the total cost of the pizza.

- double **getCost** ()
  Get the cost of the pizza.

- void **setCost** (double cost)
  Set the cost of the pizza.

- virtual ~**Pizza** ()
  Virtual destructor for ***Pizza****.*

Public Member Functions inherited from Food

- **Food** ()
  Construct a new ***Food*** *object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()

- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **Pizza** class represents a generic pizza.

Constructor & Destructor Documentation

Pizza::Pizza ()

Constructor for **Pizza** to set its cost to 0.0.

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

Virtual destructor for **Pizza**.

Member Function Documentation

virtual void Pizza::decorate (Pizza *) `[pure virtual]`

Virtual method to decorate the pizza.

Parameters

pizza	A pointer to the **Pizza** to be decorated.

Reimplemented from **Food** (*p.78*).

Implemented in **PizzaBase** (*p.130*), and **PizzaType** (*p.133*).

double Pizza::getCost ()

Get the cost of the pizza.

Returns

The cost of the pizza.

void Pizza::setCost (double *cost*)

Set the cost of the pizza.

Parameters

cost	The cost of the pizza.

virtual double Pizza::total ()`[pure virtual]`

Virtual method to get the total cost of the pizza.

Returns

The total cost of the pizza.

Implements **Food** (*p.78*).

Implemented in **PizzaBase** (*p.131*), and **PizzaType** (*p.134*).

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

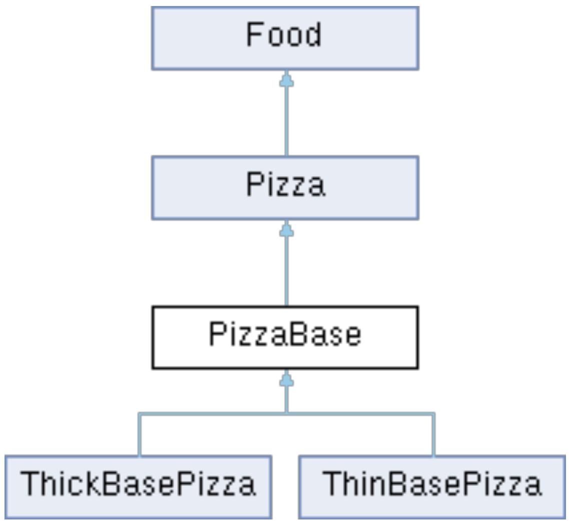
Pizza.h

PizzaBase Class Reference

The **PizzaBase** class represents the base of a pizza, which is a specific type of **Pizza**.

```
#include <PizzaBase.h>
```

Inheritance diagram for PizzaBase:



Public Member Functions

- **PizzaBase** ()
  *Constructor for **PizzaBase**.*
- virtual double **total** ()
  Returns the total cost of the pizza.
- virtual void **decorate** (**Pizza** *)
  Decorates the pizza.
- **~PizzaBase** ()
  *Destructor for **PizzaBase**.*

Public Member Functions inherited from Pizza

- **Pizza** ()
  *Constructor for **Pizza** to set its cost to 0.0.*
- double **getCost** ()
  Get the cost of the pizza.
- void **setCost** (double cost)
  Set the cost of the pizza.
- virtual ~**Pizza** ()
  *Virtual destructor for **Pizza**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **PizzaBase** class represents the base of a pizza, which is a specific type of **Pizza**.

Constructor & Destructor Documentation

PizzaBase::PizzaBase ()

Constructor for **PizzaBase**.

PizzaBase::~PizzaBase ()

Destructor for **PizzaBase**.

Member Function Documentation

virtual void PizzaBase::decorate (Pizza *) `[virtual]`

Decorates the pizza.

Parameters

pizza	A pointer to the **Pizza** to be decorated.

Implements **Pizza** (*p.127*).

virtual double PizzaBase::total () `[virtual]`

Returns the total cost of the pizza.

Returns

The total cost of the pizza.

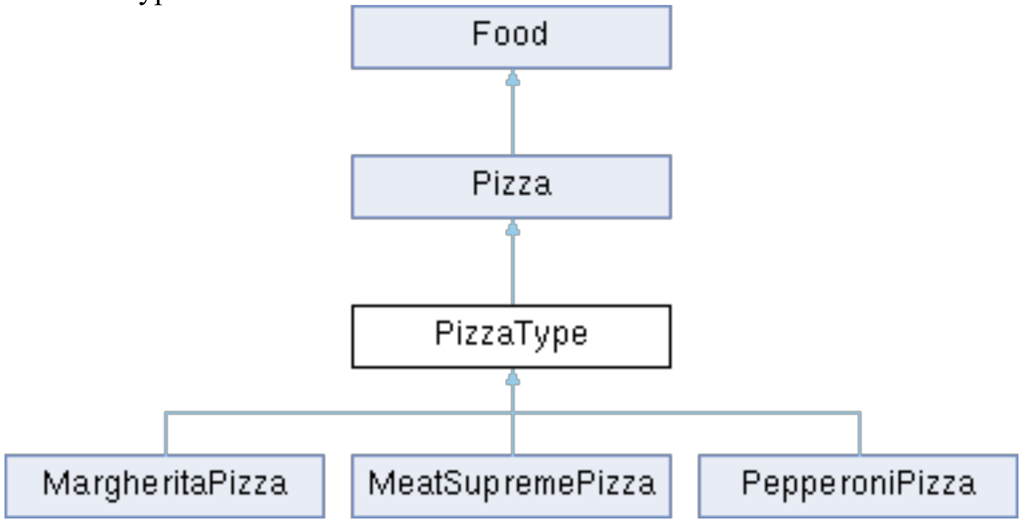
Implements **Pizza** (*p.128*).

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

PizzaBase.h

PizzaType Class Reference

The **PizzaType** class represents a specific type of pizza, which is a type of **Pizza**.
`#include <PizzaType.h>`
Inheritance diagram for PizzaType:



Public Member Functions

- **PizzaType** ()
  *Constructor for **PizzaType**.*

- virtual double **total** ()
  Returns the total cost of the pizza.

- virtual void **decorate** (**Pizza** *pizzaType)
  Decorates the pizza.

Public Member Functions inherited from Pizza

- **Pizza** ()
  *Constructor for **Pizza** to set its cost to 0.0.*

- double **getCost** ()
  Get the cost of the pizza.

- void **setCost** (double cost)
  Set the cost of the pizza.

- virtual ~**Pizza** ()
  *Virtual destructor for **Pizza**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pasta** *)

Protected Member Functions

- **~PizzaType** ()
  *Destructor for **PizzaType**.*

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **PizzaType** class represents a specific type of pizza, which is a type of **Pizza**.

Constructor & Destructor Documentation

PizzaType::PizzaType ()

Constructor for **PizzaType**.

PizzaType::~PizzaType () `[protected]`

Destructor for **PizzaType**.

Member Function Documentation

virtual void PizzaType::decorate (Pizza * *pizzaType*) `[virtual]`

Decorates the pizza.

Parameters

pizzaType	A pointer to the **Pizza** to be decorated.

Implements **Pizza** (*p.127*).

virtual double PizzaType::total ()`[virtual]`

Returns the total cost of the pizza.

Returns

The total cost of the pizza.

Implements **Pizza** (*p.128*).

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

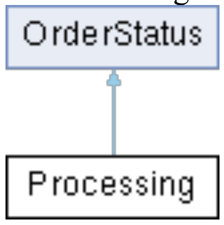
PizzaType.h

Processing Class Reference

The **Processing** class.

`#include <Processing.h>`

Inheritance diagram for Processing:



Public Member Functions

- virtual string **getStatus** ()
  Get the status of the order.

Detailed Description

The **Processing** class.

This class represents the **Processing** status of an order.

Member Function Documentation

virtual string Processing::getStatus () `[inline]`, `[virtual]`

Get the status of the order.

Returns

The status of the order.

Implements **OrderStatus** (*p.111*).

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

Processing.h

RandomString Class Reference

The **RandomString** class.
`#include <RandomString.h>`

Static Public Attributes

- static string **PositiveComment** [10]
  An array of positive comments.

- static string **NegativeComment** [10]
  An array of negative comments.

Detailed Description

The **RandomString** class.

This class represents a collection of positive and negative comments about a restaurant.

Member Data Documentation

string RandomString::NegativeComment[10] `[static]`

An array of negative comments.

string RandomString::PositiveComment[10] `[static]`

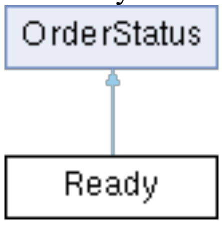
An array of positive comments.

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

RandomString.h

Ready Class Reference

The **Ready** class.
`#include <Ready.h>`
Inheritance diagram for Ready:



Public Member Functions

- virtual string **getStatus** ()
  Get the status of the order.

Detailed Description

The **Ready** class.

This class represents the **Ready** status of an order.

Member Function Documentation

virtual string Ready::getStatus () `[inline]`, `[virtual]`

Get the status of the order.

Returns

The status of the order.

Implements **OrderStatus** (*p.111*).

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

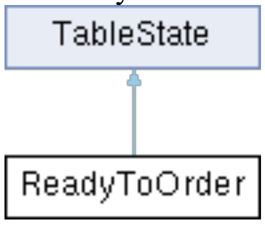
Ready.h

ReadyToOrder Class Reference

A class that represents the state of a table when it is ready to order.

`#include <ReadyToOrder.h>`

Inheritance diagram for ReadyToOrder:



Public Member Functions

- string **getStatus** ()
  Returns the status of the table as a string.

- bool **action** ()
  Performs the action of taking the order from the table.

Public Member Functions inherited from TableState

- **TableState** ()
  *Construct a new **TableState** object.*

- void **setTable** (**AbstractTable *table**)
  Set the table.

Additional Inherited Members

Protected Attributes inherited from TableState

- **AbstractTable * table**

Detailed Description

A class that represents the state of a table when it is ready to order.

This class inherits from the **TableState** abstract class and implements the getStatus and action methods. It is used to indicate that the customers at the table are ready to place their order and the waiter can take it.

Member Function Documentation

bool ReadyToOrder::action () `[virtual]`

Performs the action of taking the order from the table.

This method overrides the action method of the **TableState** class and simulates the process of taking the order from the customers. It may also change the state of the table to another state, such as WaitingForFood or Eating, depending on the outcome of the action.

Returns

A boolean value that indicates whether the action was successful or not.

Implements **TableState** (*p.152*).

string ReadyToOrder::getStatus () `[virtual]`

Returns the status of the table as a string.

This method overrides the getStatus method of the **TableState** class and returns "Ready to order" as the status.

Returns

A string that represents the status of the table.

Implements **TableState** (*p.152*).

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

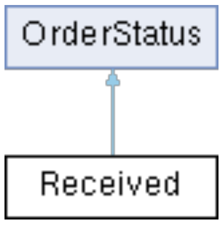
ReadyToOrder.h

Received Class Reference

The **Received** class.

`#include <Received.h>`

Inheritance diagram for Received:



Public Member Functions

- virtual std::string **getStatus** ()
  Get the status of the order.

Detailed Description

The **Received** class.

This class represents the **Received** status of an order.

Member Function Documentation

virtual std::string Received::getStatus () `[inline]`, `[virtual]`

Get the status of the order.

Returns

The status of the order.

Implements **OrderStatus** (*p.111*).

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

Received.h

Review Class Reference

The **Review** class.
`#include <Review.h>`

Public Member Functions

- **Review** ()
- **Review** (const std::string &comment, int rating)
- std::string **getReviewComment** () const
  Get the review comment.
- int **getRating** () const
  Get the rating.
- bool **operator==** (const **Review** &other) const
  Check if two reviews are equal.
- void **setRating** (int r)
  Set the rating.
- void **setReviewComment** (string c)
  Set the review comment.

Detailed Description

The **Review** class.

This class represents a review of a restaurant.

Constructor & Destructor Documentation

Review::Review ()

Review::Review (const std::string & *comment*, int *rating*)

Member Function Documentation

int Review::getRating () const

Get the rating.

Returns

The rating.

std::string Review::getReviewComment () const

Get the review comment.

Returns

The review comment.

bool Review::operator== (const Review & *other*) const

Check if two reviews are equal.

Parameters

other	The other review to compare.

Returns

True if the reviews are equal, false otherwise.

void Review::setRating (int *r*)`[inline]`

Set the rating.

Parameters

r	The rating to set.

void Review::setReviewComment (string *c*)`[inline]`

Set the review comment.

Parameters

c	The review comment to set.

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

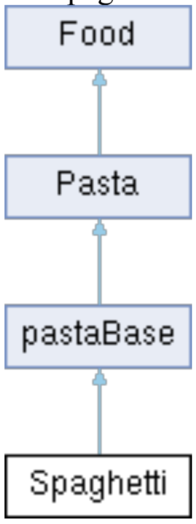
Review.h

Spaghetti Class Reference

The **Spaghetti** class represents spaghetti pasta, which is a specific type of PastaBase.

```
#include <Spaghetti.h>
```

Inheritance diagram for Spaghetti:



Public Member Functions

- **Spaghetti** ()
  *Constructor for **Spaghetti** to set its name and cost.*

Public Member Functions inherited from pastaBase

- **pastaBase** ()
  *Constructor for **pastaBase**.*
- virtual double **total** ()
  Returns the cost of the pasta.
- virtual void **decorate** (**Pasta** *)
  Virtual method to decorate the pasta.
- **~pastaBase** ()
  *Destructor for **pastaBase**.*

Public Member Functions inherited from Pasta

- **Pasta** ()
  *Constructor for **Pasta** to set its cost to 0.0.*
- double **getCost** ()
  Get the cost of the pasta.

- void **setCost** (double cost)
  Set the cost of the pasta.

- virtual **~Pasta** ()
  *Virtual destructor for **Pasta**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pizza** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **Spaghetti** class represents spaghetti pasta, which is a specific type of PastaBase.

Constructor & Destructor Documentation

Spaghetti::Spaghetti ()

Constructor for **Spaghetti** to set its name and cost.

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

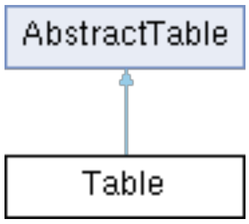
Spaghetti.h

Table Class Reference

The **Table** class.

```
#include <Table.h>
```

Inheritance diagram for Table:



Public Member Functions

- **Table** ()
- **~Table** ()
- bool **AddTable** (**AbstractTable** *table)
- **AbstractTable** * **SeparateTable** ()
- int **getTableNumber** ()
  Get the table number.

Public Member Functions inherited from AbstractTable

- int **getTableID** ()
  *Get the **Table** I D object.*
- void **setTableID** (int ID)
  *Set the **Table** ID.*
- void **setOccupied** (bool o)
  Set the Occupied object.
- bool **getOccupied** ()
  Get the Occupied object.
- int **getMaxPeople** ()
  Get the MaxPeople allowed in on the table.
- bool **visitTable** ()
  Set the Max People object.

- void **setMaxPeople** (int **maxPeople**)
- **TableState** * **getTableState** ()
- void **setTableState** (**TableState** *state)
- **CustomerGroup** * **getCustomerGroup** ()
- void **setCustomerGroup** (**CustomerGroup** ***customerGroup**)
- int **getCurrentPeople** ()
- void **setCurrentPeople** (int **currentPeople**)
- virtual vector< **Order** * > **PlaceOrder** ()
- void **ReceiveOrder** (vector< **Order** * > orders)
- int **getRandomState** ()
- void **setRandomState** (int **RandomState**)
- string **EnquireState** ()
- **AbstractTable** ()
- virtual **~AbstractTable** ()
- bool **payBill** ()
- vector< **Review** > **ReviewFood** ()
- vector< **Review** > **ReviewService** ()

Additional Inherited Members

Protected Attributes inherited from AbstractTable

- int **maxPeople**
- **TableState** * **tableState**
- **CustomerGroup** * **customerGroup**
- int **currentPeople**
- int **RandomState**
- int **tableID**
- bool **occupied** =false

Static Protected Attributes inherited from AbstractTable

- static int **counter**

Detailed Description

The **Table** class.

This class represents a table in a restaurant.

Constructor & Destructor Documentation

Table::Table () `[inline]`

Table::~Table () `[inline]`

Member Function Documentation

bool Table::AddTable (AbstractTable * *table*) `[inline]`, `[virtual]`

Implements **AbstractTable** (*p.9*).

int Table::getTableNumber ()

Get the table number.

Returns

The table number.

AbstractTable * Table::SeparateTable () `[inline]`, `[virtual]`

Implements **AbstractTable** (*p.10*).

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

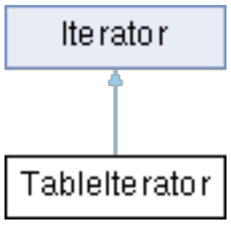
Table.h

TableIterator Class Reference

The **TableIterator** class.

```
#include <TableIterator.h>
```

Inheritance diagram for TableIterator:



Public Member Functions

- **TableIterator** (const std::vector< **Table** * > &tables)
  *Construct a new **TableIterator** object.*
- **~TableIterator** ()
  *Destroy the **TableIterator** object.*
- **Table** * **first** () override
  Get the first table.
- **Table** * **next** () override
  Get the next table.
- bool **hasNext** () override
  Check if there is a next table.
- **Table** * **current** () override
  Get the current table.

Detailed Description

The **TableIterator** class.

This class represents an iterator for a collection of tables.

Constructor & Destructor Documentation

TableIterator::TableIterator (const std::vector< Table * > & *tables*)

Construct a new **TableIterator** object.

Parameters

tables	The tables to iterate over.

TableIterator::~TableIterator ()

Destroy the **TableIterator** object.

Member Function Documentation

Table * TableIterator::current () `[override]`, `[virtual]`

Get the current table.

Returns

The current table.

Implements **Iterator** (*p.83*).

Table * TableIterator::first () `[override]`, `[virtual]`

Get the first table.

Returns

The first table.

Implements **Iterator** (*p.84*).

bool TableIterator::hasNext () `[override]`, `[virtual]`

Check if there is a next table.

Returns

True if there is a next table, false otherwise.

Implements **Iterator** (*p.84*).

Table * TableIterator::next () `[override]`, `[virtual]`

Get the next table.

Returns

The next table.

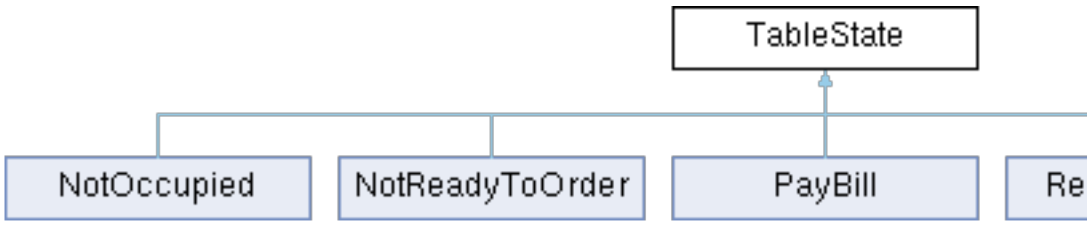
Implements **Iterator** (*p.84*).

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

TableIterator.h

TableState Class Reference

The **TableState** class.
`#include <TableState.h>`
Inheritance diagram for TableState:



Public Member Functions

- **TableState** ()
  *Construct a new **TableState** object.*

- virtual string **getStatus** ()=0
  Get the status of the table.

- virtual bool **action** ()=0
  Perform the action for the table state.

- void **setTable** (**AbstractTable** ***table**)
  Set the table.

Protected Attributes

- **AbstractTable * table**

Detailed Description

The **TableState** class.

This class represents the state of a table.

Constructor & Destructor Documentation

TableState::TableState () `[inline]`

Construct a new **TableState** object.

Member Function Documentation

virtual bool TableState::action () `[pure virtual]`

Perform the action for the table state.

Returns

True if the action was successful, false otherwise.

Implemented in **NotOccupied** (*p.103*), **NotReadyToOrder** (*p.105*), **PayBill** (*p.122*), **ReadyToOrder** (*p.139*), and **Waiting** (*p.164*).

virtual string TableState::getStatus () `[pure virtual]`

Get the status of the table.

Returns

The status of the table.

Implemented in **NotOccupied** (*p.103*), **NotReadyToOrder** (*p.105*), **PayBill** (*p.122*), **ReadyToOrder** (*p.139*), and **Waiting** (*p.164*).

void TableState::setTable (AbstractTable * *table*) `[inline]`

Set the table.

Parameters

table	The table to set.

Member Data Documentation

AbstractTable* TableState::table `[protected]`

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

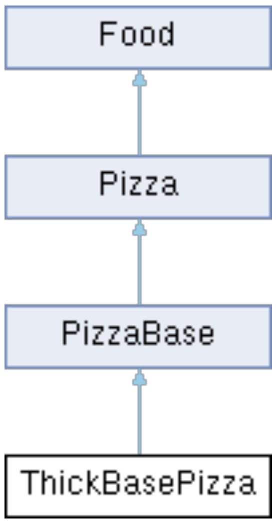
TableState.h

ThickBasePizza Class Reference

The **ThickBasePizza** class represents a pizza with a thick crust base, which is a specific type of **PizzaBase**.

```
#include <ThickBasePizza.h>
```

Inheritance diagram for ThickBasePizza:



Public Member Functions

- **ThickBasePizza** ()
  *Constructor for **ThickBasePizza** to set its name and cost.*

Public Member Functions inherited from PizzaBase

- **PizzaBase** ()
  *Constructor for **PizzaBase**.*
- virtual double **total** ()
  Returns the total cost of the pizza.
- virtual void **decorate** (**Pizza** *)
  Decorates the pizza.
- **~PizzaBase** ()
  *Destructor for **PizzaBase**.*

Public Member Functions inherited from Pizza

- **Pizza** ()
  *Constructor for **Pizza** to set its cost to 0.0.*
- double **getCost** ()
  Get the cost of the pizza.

- void **setCost** (double cost)
  Set the cost of the pizza.

- virtual **~Pizza** ()
  *Virtual destructor for **Pizza**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **ThickBasePizza** class represents a pizza with a thick crust base, which is a specific type of **PizzaBase**.

Constructor & Destructor Documentation

ThickBasePizza::ThickBasePizza ()

Constructor for **ThickBasePizza** to set its name and cost.

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

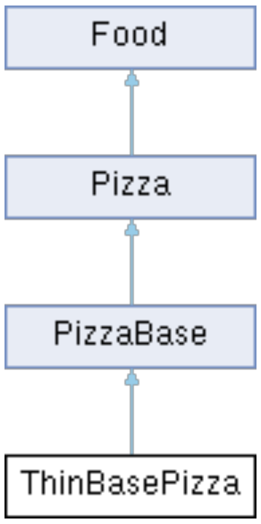
ThickBasePizza.h

ThinBasePizza Class Reference

The **ThinBasePizza** class represents a pizza with a thin crust base, which is a specific type of **PizzaBase**.

`#include <ThinBasePizza.h>`

Inheritance diagram for ThinBasePizza:



Public Member Functions

- **ThinBasePizza** ()
  *Constructor for **ThinBasePizza** to set its name and cost.*

Public Member Functions inherited from PizzaBase

- **PizzaBase** ()
  *Constructor for **PizzaBase**.*
- virtual double **total** ()
  Returns the total cost of the pizza.
- virtual void **decorate** (**Pizza** *)
  Decorates the pizza.
- **~PizzaBase** ()
  *Destructor for **PizzaBase**.*

Public Member Functions inherited from Pizza

- **Pizza** ()
  *Constructor for **Pizza** to set its cost to 0.0.*
- double **getCost** ()
  Get the cost of the pizza.

- void **setCost** (double cost)
  Set the cost of the pizza.

- virtual **~Pizza** ()
  *Virtual destructor for **Pizza**.*

Public Member Functions inherited from Food

- **Food** ()
  *Construct a new **Food** object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Burger** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **ThinBasePizza** class represents a pizza with a thin crust base, which is a specific type of **PizzaBase**.

Constructor & Destructor Documentation

ThinBasePizza::ThinBasePizza ()

Constructor for **ThinBasePizza** to set its name and cost.

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

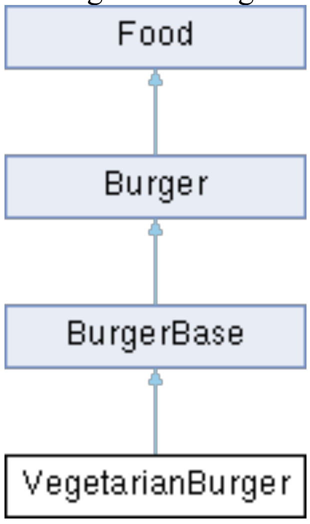
ThinBasePizza.h

VegetarianBurger Class Reference

The **VegetarianBurger** class represents a vegetarian burger, which is a specific type of **BurgerBase**.

```
#include <VegetarianBurger.h>
```

Inheritance diagram for VegetarianBurger:



Public Member Functions

- **VegetarianBurger** ()
  *Constructor for **VegetarianBurger** to set its name and cost.*

Public Member Functions inherited from BurgerBase

- **BurgerBase** ()
  *Constructor for **BurgerBase**.*
- virtual double **total** ()
  Virtual method to get the total cost of the burger.
- virtual void **decorate** (**Burger** *)
  Virtual method to decorate the burger.
- **~BurgerBase** ()
  *Destructor for **BurgerBase**.*

Public Member Functions inherited from Burger

- **Burger** ()
  *Constructor for **Burger** to set its cost to 0.0.*
- double **getCost** ()
  Get the cost of the burger.

- void **setCost** (double cost)
  Set the cost of the burger.

- virtual **~Burger** ()
  Virtual destructor for ***Burger****.*

Public Member Functions inherited from Food

- **Food** ()
  Construct a new ***Food*** *object.*

- void **setFoodQuality** (int)
- int **getFoodQuality** ()
- string **getName** ()
- void **setName** (string **name**)
- void **addIngredient** (string ingredient)
- double **getCost** ()
- void **setCost** (double **cost**)
- virtual **~Food** ()
- virtual void **decorate** (**Pizza** *)
- virtual void **decorate** (**Pasta** *)

Additional Inherited Members

Protected Attributes inherited from Food

- string **name**
- vector< string > **ingredients**
- int **RandomFoodQuality**
- double **cost**

Detailed Description

The **VegetarianBurger** class represents a vegetarian burger, which is a specific type of **BurgerBase**.

Constructor & Destructor Documentation

VegetarianBurger::VegetarianBurger ()

Constructor for **VegetarianBurger** to set its name and cost.

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

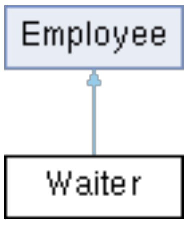
VegetarianBurger.h

Waiter Class Reference

Represents a waiter who takes and delivers orders in a restaurant.

`#include <Waiter.h>`

Inheritance diagram for Waiter:



Public Member Functions

- **Waiter** (int id)
  *Constructs a **Waiter** with the given ID.*
- **~Waiter** ()
- void **getReviewsForKitchenDepartment** ()
  Retrieves reviews from the kitchen department.
- void **CollectOrderFromKitchen** (**Order** *)
  Collects an order from the kitchen and updates the order.
- void **TakeOrder** (**Table** *)
  Takes an order from a table.
- void **TakeOrder** (**Order** *)
- void **deliverOrders** ()
  Delivers orders to the respective tables.
- void **assignTables** (vector< **Table** * > &)
- bool **isFullyOccupied** ()
- void **setMaxTables** (int)
- void **iterateTables** ()
  Iterate through the tables.

Public Member Functions inherited from Employee

- **Employee** (int id)

*Construct a new **Employee** object.*

- virtual void **assignTables** (std::vector< **Table** * > &**tables**)
Assign tables to the employee.

- void **moveToNextTable** ()
Move to the next table.

- **Department** * **getDepartment** ()
Get the department of the employee.

- **Table** * **getCurrentTable** ()
Get the current table.

- void **setDepartment** (**Department** *dep)
Set the department of the employee.

- void **setCurrTable** (**Table** *currTab)
Set the current table.

- void **GetReview** (const std::vector< **Review** * > &reviewList)
Get the reviews.

- void **TakeOrder** (**Table** *table)
Take an order.

- int **getEmployeeId** ()
Get the ID of the employee.

- **~Employee** ()
*Destroy the **Employee** object.*

Additional Inherited Members

Protected Attributes inherited from Employee

- **Department** * **department**
- **Table** * **tables**
- **Table** * **currTable**
- **TableIterator** * **tableIterator**
- int **employeeId**

Detailed Description

Represents a waiter who takes and delivers orders in a restaurant.

Constructor & Destructor Documentation

Waiter::Waiter (int *id*)

Constructs a **Waiter** with the given ID.

Parameters

id	The ID of the waiter.

Waiter::~Waiter ()

Member Function Documentation

void Waiter::assignTables (vector< Table * > &)

void Waiter::CollectOrderFromKitchen (Order *)

Collects an order from the kitchen and updates the order.

Parameters

order	The order to be updated.

void Waiter::deliverOrders ()

Delivers orders to the respective tables.

void Waiter::getReviewsForKitchenDepartment ()

Retrieves reviews from the kitchen department.

bool Waiter::isFullyOccupied ()

void Waiter::iterateTables () `[virtual]`

Iterate through the tables.

Reimplemented from **Employee** (*p.70*).

void Waiter::setMaxTables (int)

void Waiter::TakeOrder (Order *)

void Waiter::TakeOrder (Table *)

Takes an order from a table.

Parameters

table	The table from which the order is taken.

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

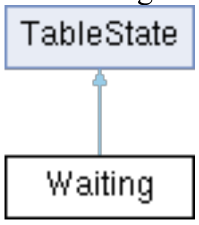
Waiter.h

Waiting Class Reference

The **Waiting** class.

```
#include <Waiting.h>
```

Inheritance diagram for Waiting:



Public Member Functions

- string **getStatus** ()
  Get the status of the table.
- bool **action** ()
  Perform the action for the waiting state.

Public Member Functions inherited from TableState

- **TableState** ()
  Construct a new ***TableState*** *object.*
- void **setTable** (**AbstractTable *table**)
  Set the table.

Additional Inherited Members

Protected Attributes inherited from TableState

- **AbstractTable * table**

Detailed Description

The **Waiting** class.

This class represents the waiting state of a table.

Member Function Documentation

bool Waiting::action () `[virtual]`

Perform the action for the waiting state.

Returns

False.

Implements **TableState** (*p.152*).

string Waiting::getStatus () `[virtual]`

Get the status of the table.

Returns

The status of the table.

Implements **TableState** (*p.152*).

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

Waiting.h

File Documentation

AbstractTable.h File Reference

```
#include <iostream>
#include "TableState.h"
#include "CustomerGroup.h"
#include "Order.h"
#include "Review.h"
#include <vector>
```

Classes

class **AbstractTable**

AbstractTable.h

Go to the documentation of this file.

```cpp
#ifndef ABSTRACT_TABLE_H
#define ABSTRACT_TABLE_H
#include <iostream>

#include "TableState.h"
#include "CustomerGroup.h"
#include "Order.h"
#include "Review.h"
#include <vector>

using namespace std;


class AbstractTable
{
    protected:
        int maxPeople;
        TableState* tableState;
        CustomerGroup* customerGroup;
        int currentPeople;
        int RandomState;
        static int counter;
        int tableID;
        bool occupied=false;

    public:
        int getTableID(){return tableID;}
        void setTableID(int ID){this->tableID=ID;}
        void setOccupied(bool o){this->occupied=o;}//t
        bool getOccupied(){return occupied;}//t
        int getMaxPeople(){return this->maxPeople;}//t
        bool visitTable();
        void setMaxPeople(int maxPeople){this->maxPeople=maxPeople;}//t
        TableState* getTableState(){return tableState;}//t
        void setTableState(TableState* state){this->tableState=state;}//t
        CustomerGroup* getCustomerGroup(){return customerGroup;}//t
        void setCustomerGroup(CustomerGroup* customerGroup){this->customerGroup=customerGroup;}//t
        virtual bool AddTable(AbstractTable* table)=0;
        virtual AbstractTable* SeparateTable()=0;
        int getCurrentPeople(){return currentPeople;}//t
        void setCurrentPeople(int currentPeople){this->currentPeople=currentPeople;}//t
        virtual vector<Order*> PlaceOrder();//t
        void ReceiveOrder(vector<Order*> orders);//t
        int getRandomState();//t
        void setRandomState(int RandomState){this->RandomState=RandomState;}//t
        string EnquireState();//t
        AbstractTable();//t
        virtual ~AbstractTable();//t
        bool payBill();//t
        vector<Review> ReviewFood(){return customerGroup->ReviewFood();}//t
        vector<Review> ReviewService(){return customerGroup->ReviewService();}//t


};

//#include "AbstractTable.cpp"
#endif
```

Alfredo.h File Reference

```
#include "PastaType.h"
```

Classes

class **Alfredo***The **Alfredo** class represents **Alfredo** pasta, which is a specific type of **PastaType**.*

Alfredo.h

Go to the documentation of this file.

```
#ifndef ALFREDO_H
#define ALFREDO_H

#include "PastaType.h"

class Alfredo : public PastaType {
public:
    Alfredo();

    ~Alfredo();
};

#endif
```

Angry.h File Reference

Declaration of the **Angry** class, a subclass of **CustomerState**.

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

Classes

class **Angry***Represents the "Angry" state of a customer.*

Detailed Description

Declaration of the **Angry** class, a subclass of **CustomerState**.

Angry.h

Go to the documentation of this file.

```
#ifndef ANGRY_H
#define ANGRY_H

#include "CustomerState.h"
#include <iostream>
using namespace std;

class Angry : public CustomerState
{
public:
    string getStatus() { return "ANGRY"; }

    void action() { cout << "Customer: THE FOOD HERE SUCKS"; }
};

#endif
```

BeefBurger.h File Reference

```
#include "BurgerBase.h"
```

Classes

class **BeefBurger***The **BeefBurger** class represents a beef burger, which is a specific type of **BurgerBase**.*

BeefBurger.h

Go to the documentation of this file.

```
#ifndef BEEFBURGER_H
#define BEEFBURGER_H

#include "BurgerBase.h"

class BeefBurger : public BurgerBase {
public:
    BeefBurger();
};

#endif
```

Bill.h File Reference

Declaration of the **Bill** class.

```
#include "BillState.h"
#include <string>
#include <vector>
#include <map>
#include "BillMemento.h"
#include "Order.h"
```

Classes

class **Bill***Represents a bill associated with a customer's order.*

Detailed Description

Declaration of the **Bill** class.

Bill.h

Go to the documentation of this file.

```
#ifndef BILL_H
#define BILL_H

#include "BillState.h"
#include <string>
#include <vector>
#include <map>
using namespace std;
#include "BillMemento.h"
#include "Order.h"

class Bill
{
private:
    float cost;
    bool paid;
    int tableID;
    Order* CopyOrders;
    string customerID;

public:
    Bill();

    Order* getCopyOrder();

    void setCopyOrder(Order* order);

    float getCost();

    void setCost(float orderCost);

    BillState* getBillState();

    bool getBillStatus();

    void setBillStatus(bool BillStatus);

    void setTableID(int ID);

    void setCustomerID(string ID);

    std::string getCustomerID();

     int getTableID();

     void recoverBill(BillMemento* mem);

     BillMemento* saveState();

     void print();
};

#endif
```

BillCaretaker.h File Reference

Declaration of the **BillCaretaker** class.

```
#include <string>
#include <vector>
#include <map>
#include "BillMemento.h"
```

Classes

class **BillCaretaker***Manages the storage and retrieval of bill Memento objects.*

Detailed Description

Declaration of the **BillCaretaker** class.

BillCaretaker.h

Go to the documentation of this file.

```
#ifndef BILLCARETAKER_H
#define BILLCARETAKER_H

#include <string>
#include <vector>
#include <map>
#include "BillMemento.h"

class BillCaretaker {
private:
    vector<BillMemento*> bills;

public:
    BillCaretaker();

    void storeMemento(BillMemento* mem);

    BillMemento* retrieveMemento(string customerID);
};

#endif
```

BillMemento.h File Reference

Declaration of the **BillMemento** class.

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

Classes

class **BillMemento***Represents a Memento for storing the state of a bill.*

Detailed Description

Declaration of the **BillMemento** class.

BillMemento.h

Go to the documentation of this file.

```
#ifndef BILLMEMENTO_H
#define BILLMEMENTO_H

#include <string>
#include "BillState.h"

class BillMemento
{
private:
    BillState* state;

public:
    BillMemento() { state = nullptr; }

    BillState* getState();

    void setState(BillState* bs);
};

#endif
```

BillState.h File Reference

Declaration of the **BillState** class.

```
#include <vector>
#include <string>
#include "Order.h"
```

Classes

class **BillState***Represents the state of a bill associated with an order.*

Detailed Description

Declaration of the **BillState** class.

BillState.h

Go to the documentation of this file.

```
#ifndef BILLSTATE_H
#define BILLSTATE_H

#include <vector>
#include <string>
#include "Order.h"

class BillState {
private:
    Order* CopyOrder;
    float cost;
    bool paid;
    string customerID;
    int tableID;

public:
    void loadFromFile(string filename);

    void saveToFile(string filename);

    Order* getCopyOrder();

    float getCost();

    bool getPaidStatus();

    std::string getCustomerID();

    int getTableID();

    void setCopyOrder(Order* order);

    void setCost(float newCost);

    void setPaid(bool pay);

    void setCustomerID(string custID);

    void setTableID(int tabID);
};

#endif
```

Bologniase.h File Reference

`#include "PastaType.h"`

Classes

class **Bolognaise***The **Bolognaise** class represents **Bolognaise** pasta, which is a specific type of **PastaType**.*

Bolognaise.h

Go to the documentation of this file.

```
#ifndef BOLOGNAISE_H
#define BOLOGNAISE_H

#include "PastaType.h"

class Bolognaise : public PastaType {
public:
    Bolognaise();

    ~Bolognaise();
};

#endif
```

Burger.h File Reference

```
#include "../Food.h"
#include <string>
#include <iostream>
```

Classes

class **Burger***The **Burger** class represents a generic burger.*

Burger.h

Go to the documentation of this file.

```
#ifndef BURGER_H
#define BURGER_H

#include "../Food.h"
#include <string>
#include <iostream>

class Burger : public Food {
private:
    double cost;

public:
    Burger();

    virtual void decorate(Burger*) = 0;

    virtual double total() = 0;

    double getCost();

    void setCost(double cost);

    virtual ~Burger();
};

#endif
```

BurgerBase.h File Reference

```
#include "Burger.h"
```

Classes

class **BurgerBase***The **BurgerBase** class represents the base of a burger, which is a specific type of **Burger**.*

BurgerBase.h

Go to the documentation of this file.

```
#ifndef BURGERBASE_H
#define BURGERBASE_H

#include "Burger.h"

class BurgerBase : public Burger {
public:
    BurgerBase();

    virtual double total();

    virtual void decorate(Burger*);

    ~BurgerBase();
};

#endif

```

BurgerTopping.h File Reference

```
#include "Burger.h"
```

Classes

class **BurgerTopping***The* ***BurgerTopping*** *class represents a topping for a burger, which is a specific type of* ***Burger****.*

BurgerTopping.h

Go to the documentation of this file.

```
#ifndef BURGERTOPPING_H
#define BURGERTOPPING_H

#include "Burger.h"

class BurgerTopping : public Burger {
private:
    Burger* topping;

public:
    BurgerTopping();

    virtual double total();

    virtual void decorate(Burger* burgerTopping);

protected:
    ~BurgerTopping();
};

#endif
```

Carbonara.h File Reference

```
#include "PastaType.h"
```

Classes

class **Carbonara***The **Carbonara** class represents **Carbonara** pasta, which is a specific type of **PastaType**.*

Carbonara.h

Go to the documentation of this file.

```
#ifndef CARBONARA_H
#define CARBONARA_H

#include "PastaType.h"

class Carbonara : public PastaType {
public:
    Carbonara();

    ~Carbonara();
};

#endif
```

CheeseTopping.h File Reference

```
#include "BurgerTopping.h"
```

Classes

class **CheeseTopping***The* ***CheeseTopping*** *class represents a cheese topping for a burger, which is a specific type of* ***BurgerTopping****.*

CheeseTopping.h

Go to the documentation of this file.

```
#ifndef CHEESETOPPING_H
#define CHEESETOPPING_H

#include "BurgerTopping.h"

class CheeseTopping : public BurgerTopping {
public:
    CheeseTopping();

    ~CheeseTopping();
};

#endif
```

ChickenBurger.h File Reference

```
#include "BurgerBase.h"
```

Classes

class **ChickenBurger***The **ChickenBurger** class represents a chicken burger, which is a specific type of **BurgerBase**.*

ChickenBurger.h

Go to the documentation of this file.

```
#ifndef CHICKENBURGER_H
#define CHICKENBURGER_H

#include "BurgerBase.h"

class ChickenBurger : public BurgerBase {
public:
    ChickenBurger();
};

#endif
```

ChilliTopping.h File Reference

```
#include "BurgerTopping.h"
```

Classes

class **ChilliTopping***The **ChilliTopping** class represents a chili topping for a burger, which is a specific type of **BurgerTopping**.*

ChilliTopping.h

Go to the documentation of this file.

```
#ifndef CHILLITOPPING_H
#define CHILLITOPPING_H

#include "BurgerTopping.h"

class ChilliTopping : public BurgerTopping {
public:
    ChilliTopping();

    ~ChilliTopping();
};

#endif
```

CombinedTable.h File Reference

Declaration of the **CombinedTable** class.

```
#include "AbstractTable.h"
#include <vector>
#include "Order.h"
```

Classes

class **CombinedTable***Represents a combined table that can group multiple **AbstractTable** instances.*

Detailed Description

Declaration of the **CombinedTable** class.

CombinedTable.h

Go to the documentation of this file.

```
#ifndef COMBINEDTABLE_H
#define COMBINEDTABLE_H

#include "AbstractTable.h"
#include <vector>
#include "Order.h"

class CombinedTable : public AbstractTable
{
private:
    vector<AbstractTable*> table;

public:
    CombinedTable();

    ~CombinedTable();

    bool AddTable(AbstractTable* table);

    AbstractTable* SeparateTable();

    vector<Order*> PlaceOrder();
};

#endif
```

Customer.h File Reference

Header file for the **Customer** class.
```
#include "Customer.h"
#include <iostream>
#include <string>
#include "CustomerState.h"
#include "Order.h"
```

Classes

class **Customer***The **Customer** class.*

Detailed Description

Header file for the **Customer** class.

This file contains the declaration of the **Customer** class.

Customer.h

Go to the documentation of this file.

```
#ifndef Customer_H
#define Customer_H

#include "Customer.h"
#include <iostream>
#include <string>
using namespace std;
#include "CustomerState.h"
#include"Order.h"

class Customer
{
    //srand((unsigned) time(NULL));
    private:
        string ID;
        CustomerState* state;
    public:
        static int SeedValue;

    public:
        string getID(){return ID;};//t
        void setID(string ID){this->ID=ID;};//t
        void setState(CustomerState* state){this->state=state;};//t
        CustomerState* getState(){return state;};//t
        string GiveComment_Food();//t
        string GiveComment_Service();//t
        int GiveRating_Food();//t
        int GiveRating_Service();//t
        Customer(string name){ID=name;};//t
        Customer();//t
        void receiveOrder(Order* order);//check
         Order* PlaceOrder();//t
};

//#include "Customer.cpp"

#endif
```

CustomerGroup.h File Reference

Declaration of the **CustomerGroup** class.
```
#include <vector>
#include "Customer.h"
#include "Order.h"
#include "Review.h"
```

Classes

class **CustomerGroup***Represents a group of customers in a restaurant.*

Detailed Description

Declaration of the **CustomerGroup** class.

CustomerGroup.h

Go to the documentation of this file.

```cpp
#ifndef CUSTOMER_GROUP_H
#define CUSTOMER_GROUP_H

#include <vector>
#include "Customer.h"
#include "Order.h"
#include "Review.h"

class CustomerGroup
{
protected:
    vector<Customer> customers;
    int RandomState;
    vector<Order*> orders;

public:
    vector<Customer> getCustomers();

    void setCustomers(vector<Customer> customer);

    int getRandomState();
    void decrementRandomState(){RandomState--;}

    void setRandomState(int RandomState);

    int NumOfCustomer();

    Customer CustomerAt(int index);

    vector<Bill*> mergeBill();

    bool addCustomer(Customer customer);

    CustomerGroup();

    void receiveOrder(vector<Order*> orders);

    bool PayBill();

    vector<Review> ReviewFood();

    vector<Review> ReviewService();

    vector<Order*> PlaceOrder();

    void print();
};

#endif
```

CustomerState.h File Reference

Declaration of the **CustomerState** class.

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

Classes

class **CustomerState***Represents the state of a customer in a restaurant.*

Detailed Description

Declaration of the **CustomerState** class.

CustomerState.h

Go to the documentation of this file.

```
#ifndef CUSTOMER_STATE
#define CUSTOMER_STATE

#include <iostream>
#include <string>

class CustomerState
{
public:
    virtual string getStatus() = 0;

    virtual void action() = 0;
};

#endif
```

Department.h File Reference

Header file for the **Department** class.
```
#include "Review.h"
#include <vector>
```

Classes

class **Department***The* ***Department*** *class.*

Detailed Description

Header file for the **Department** class.

This file contains the declaration of the **Department** class.

Department.h

Go to the documentation of this file.

```

#ifndef DEPARTMENT_H
#define DEPARTMENT_H

#include "Review.h"
#include <vector>

class Department {
public:
    virtual void TakeReview(const Review& review) = 0;
    virtual void DisplayReviews() = 0;
    virtual double CalculateAverageRating() const = 0;
    virtual void DeleteReview(const Review& review) = 0;
protected:
    std::vector<Review> reviews;
};

#endif // DEPARTMENT_H
```

Employee.h File Reference

Header file for the **Employee** class.

```
#include <string>
#include "Department.h"
#include "TableIterator.h"
```

Classes

class **Employee***The **Employee** class.*

Detailed Description

Header file for the **Employee** class.

This file contains the declaration of the **Employee** class.

Employee.h

Go to the documentation of this file.

```

#ifndef EMPLOYEE_H
#define EMPLOYEE_H

#include <string>
using namespace std;
#include "Department.h"

#include "TableIterator.h"

class Employee {
public:
    Employee(int id);
    virtual void assignTables(std::vector<Table*>& tables);
    virtual void iterateTables();
    void moveToNextTable();
    Department* getDepartment();
    Table* getCurrentTable();
    void setDepartment(Department* dep);
    void setCurrTable(Table* currTab);
    void GetReview(const std::vector<Review*>& reviewList);
    void TakeOrder(Table* table);
    int getEmployeeId();
    ~Employee();
protected:
    Department* department;
    Table* tables;
    Table* currTable;
    TableIterator* tableIterator;
    int employeeId;
};
#endif
```

Floor.h File Reference

```
#include "Waiter.h"
#include "Employee.h"
#include "Manager.h"
#include "Iterator.h"
#include "Table.h"
#include "CombinedTable.h"
#include <iomanip>
```

Classes

class **Floor***This is the interface for floor.*

Floor.h

Go to the documentation of this file.

```
#ifndef ABSTRACT_FLOOR_H
#define ABSTRACT_FLOOR_H

#include "Waiter.h"
#include "Employee.h"
#include "Manager.h"
#include "Iterator.h"
#include "Table.h"
#include "CombinedTable.h"
#include <iomanip>
using namespace std;
class Table;
class CustomerGroup;

class Floor{
    protected:
        std::vector<Table*> tables;
        std::vector<Employee*> waiters;
        Manager* manager;
        int capacity;
        int numOccupiedTables;
        int numAvailableWaiters;


    public:
        Floor(int);
        Employee* createWaiter();
        Employee* createManager();
        bool hasAvailableWaiter();
        bool addCustomerGroup(CustomerGroup*);
        void waiterIterateTables();
        void reorderMaxTablesForWaiters();
        void printTablesAndWaiters(){
            cout << "Printing floors and waiters" << endl;

            for(Table* table: tables){
                cout << left << setw(20) << table->EnquireState() << "|";
            }
            cout << endl << "Number of tables: " << tables.size() <<
             "\nNumber of occupied tables: " << this->numOccupiedTables << endl;
        }

};

#endif
```

FloorDepartment.h File Reference

Header file for the **FloorDepartment** class.
```
#include "Department.h"
```

Classes

class **FloorDepartment***The **FloorDepartment** class.*

Detailed Description

Header file for the **FloorDepartment** class.

This file contains the declaration of the **FloorDepartment** class.

FloorDepartment.h

Go to the documentation of this file.

```

#ifndef FLOORDEPARTMENT_H
#define FLOORDEPARTMENT_H

#include "Department.h"

class FloorDepartment : public Department {
public:
    void TakeReview(const Review& review) override;
    void DisplayReviews() override;
    double CalculateAverageRating() const override;
    void DeleteReview(const Review& review) override;
};

#endif // FLOORDEPARTMENT_H
```

Food.h File Reference

Header file for the **Food** class.
```
#include <string>
#include <vector>
#include <iostream>
```

Classes

class **Food***The **Food** class.*

Detailed Description

Header file for the **Food** class.

This file contains the declaration of the **Food** class.

Food.h

Go to the documentation of this file.

```

#ifndef FOOD_H
#define FOOD_H

#include <string>
#include <vector>
using namespace std;
#include <iostream>
class Burger;
class Pizza;
class Pasta;

class Food {
protected:
    string name;
    vector<string> ingredients;
    int RandomFoodQuality;
    double cost;
public:
    Food();
    void setFoodQuality(int);
    int getFoodQuality();
    string getName();
    void setName(string name);
    void addIngredient(string ingredient);//for extras
    virtual double total() = 0;
    double getCost();
    void setCost(double cost);
    virtual ~Food();
    virtual void decorate(Burger*) ;
    virtual void decorate(Pizza*) ;
    virtual void decorate(Pasta*) ;
    };

#endif
```

Happy.h File Reference

Header file for the **Happy** class.

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

Classes

class **Happy***The* ***Happy*** *class.*

Detailed Description

Header file for the **Happy** class.

This file contains the declaration of the **Happy** class.

Happy.h

Go to the documentation of this file.

```
#ifndef HAPPY_H
#define HAPPY_H

#include "CustomerState.h"
#include <iostream>
using namespace std;

class Happy: public CustomerState
{
    public:
        string getStatus(){return "HAPPY";}
        void action(){cout<<"Customer: THE FOOD HERE IS SO AMAZING";}

};
#endif
```

Iterator.h File Reference

Header file for the **Iterator** class.

Classes

class **Iterator***The **Iterator** class.*

Detailed Description

Header file for the **Iterator** class.

This file contains the declaration of the **Iterator** class.

Iterator.h

Go to the documentation of this file.

```

#ifndef ITERATOR_H
#define ITERATOR_H

class Table;

class Iterator {
public:
    virtual Table* first() = 0;
    virtual Table* next() = 0;
    virtual bool hasNext() = 0;
    virtual Table* current() = 0;
};

#endif // ITERATOR_H
```

KitchenDepartment.h File Reference

Header file for the **KitchenDepartment** class.
`#include "Department.h"`

Classes

class **KitchenDepartment***The* ***KitchenDepartment*** *class.*

Detailed Description

Header file for the **KitchenDepartment** class.

This file contains the declaration of the **KitchenDepartment** class.

KitchenDepartment.h

Go to the documentation of this file.

```

#ifndef KITCHENDEPARTMENT_H
#define KITCHENDEPARTMENT_H

#include "Department.h"

class KitchenDepartment : public Department {
public:
    void TakeReview(const Review& review) override;
    void DisplayReviews() override;
    double CalculateAverageRating() const override;
    void DeleteReview(const Review& review) override;
};

#endif // KITCHENDEPARTMENT_H
```

Macaroni.h File Reference

`#include "pastaBase.h"`

Classes

class **Macaroni***The **Macaroni** class represents macaroni pasta, which is a specific type of PastaBase.*

Macaroni.h

Go to the documentation of this file.

```
#ifndef MACARONI_H
#define MACARONI_H

#include "pastaBase.h"

class Macaroni : public pastaBase  {
public:
    Macaroni();
};

#endif
```

Manager.h File Reference

Header file for the **Manager** class.
```
#include "Employee.h"
```

Classes

class **Manager***The* ***Manager*** *class.*

Detailed Description

Header file for the **Manager** class.

This file contains the declaration of the **Manager** class.

Manager.h

Go to the documentation of this file.

```
#ifndef MANAGER_H
#define MANAGER_H

#include "Employee.h"

class Manager : public Employee {
public:
    Manager(int id);
    void getReviewsForFloorDepartment();
};

#endif // MANAGER_H
```

MargheritaPizza.h File Reference

```
#include "PizzaType.h"
```

Classes

class **MargheritaPizza***The **MargheritaPizza** class represents a Margherita pizza type, which is a specific type of **Pizza**.*

MargheritaPizza.h

Go to the documentation of this file.

```
#ifndef MARGHERITAPIZZA_H
#define MARGHERITAPIZZA_H

#include "PizzaType.h"

class MargheritaPizza : public PizzaType {
public:
    MargheritaPizza();

    ~MargheritaPizza();
};

#endif
```

MeatSupremePizza.h File Reference

```
#include "PizzaType.h"
```

Classes

class **MeatSupremePizza***The **MeatSupremePizza** class represents a Meat Supreme pizza type, which is a specific type of **Pizza**.*

MeatSupremePizza.h

Go to the documentation of this file.

```
#ifndef MEATSUPREMEPIZZA_H
#define MEATSUPREMEPIZZA_H

#include "PizzaType.h"

class MeatSupremePizza : public PizzaType {
public:
    MeatSupremePizza();

    ~MeatSupremePizza();
};

#endif
```

Menu.h File Reference

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

Classes

struct **FoodItem***Represents a food item with name, price, preparation method, and type.*

class **Menu***Represents a menu for a restaurant.*

Menu.h

Go to the documentation of this file.

```
#ifndef MENU
#define MENU

#include <string>
#include <iostream>
#include <vector>

using namespace std;

struct FoodItem{
    string name;
    int price;
    string method;
    string type;
    FoodItem(string, int, string, string);
    ~FoodItem();
};

class Menu{
public:
    static Menu* getMenu();

    string printMenu();

    ~Menu();

    FoodItem* getFoodItem();
    vector<FoodItem*> menu;
protected:
    Menu();



private:

    static Menu* Menu_instance;
};

#endif
```

Neutral.h File Reference

Header file for the **Neutral** class.
```
#include "CustomerState.h"
#include <iostream>
```

Classes

class **Neutral***The* ***Neutral*** *class.*

Detailed Description

Header file for the **Neutral** class.

This file contains the declaration of the **Neutral** class.

Neutral.h

Go to the documentation of this file.

```

#ifndef NEUTRAL_H
#define NEUTRAL_H

#include "CustomerState.h"
#include <iostream>
using namespace std;

class Neutral: public CustomerState
{
    public:
        string getStatus(){return "NEUTRAL";};
        void action(){cout<<"Customer:  Am so hungry";};

};
#endif
```

NotOccupied.h File Reference

Header file for the **NotOccupied** class.
```
#include <string>
#include <iostream>
#include "TableState.h"
```

Classes

class **NotOccupied***The **NotOccupied** class.*

Detailed Description

Header file for the **NotOccupied** class.

This file contains the declaration of the **NotOccupied** class.

NotOccupied.h

Go to the documentation of this file.

```

#ifndef NOTOCCUPIED_H
#define NOTOCCUPIED_H

#include <string>
#include <iostream>
using namespace std;
#include "TableState.h"

class AbstractTable;

class NotOccupied : public TableState
{
    public:
        string getStatus(){return "NotOccupied";};
        bool action(){return false;};
};

#endif
```

NotReadyToOrder.h File Reference

Header file for the **NotReadyToOrder** class.
```
#include <string>
#include <iostream>
#include "TableState.h"
```

Classes

class **NotReadyToOrder***The **NotReadyToOrder** class.*

Detailed Description

Header file for the **NotReadyToOrder** class.

This file contains the declaration of the **NotReadyToOrder** class.

NotReadyToOrder.h

Go to the documentation of this file.

```

#ifndef NOTREADYTOORDER_H
#define NOTREADYTOORDER_H

#include <string>
#include <iostream>
using namespace std;
#include "TableState.h"

class AbstractTable;

class NotReadyToOrder : public TableState
{
    public:
        string getStatus();
        bool action();
};

#endif
```

OnionTopping.h File Reference

```
#include "BurgerTopping.h"
```

Classes

class **OnionTopping***The **OnionTopping** class represents an onion topping for a burger, which is a specific type of **BurgerTopping**.*

OnionTopping.h

Go to the documentation of this file.

```
#ifndef ONIONTOPPING_H
#define ONIONTOPPING_H

#include "BurgerTopping.h"

class OnionTopping : public BurgerTopping {
public:
    OnionTopping();

    ~OnionTopping();
};

#endif
```

Order.h File Reference

```
#include <string>
#include <vector>
#include "OrderStatus.h"
#include "Ready.h"
#include "Processing.h"
#include "Received.h"
#include "Menu.h"
#include "Food.h"
#include "Employee.h"
```

Classes

class **Order**

Order.h

Go to the documentation of this file.

```
#ifndef ORDER_H
#define ORDER_H
using namespace std;
#include <string>
#include <vector>

#include "OrderStatus.h"
#include "Ready.h"
#include "Processing.h"
#include "Received.h"
#include "Menu.h"
#include "Food.h"
class Bill;
class Table;
#include "Employee.h"
class Waiter;
class AbstractTable;

class Order{

    private:
        OrderStatus* orderStatus;
        std::vector<FoodItem*> items;
        std::vector<Food*> food;
        Bill* bill;
        AbstractTable* table;//I have changed this to abstract  Table
        Waiter* waiter;//change it back to employee
        void deallocateStatus(){
            if(orderStatus != nullptr) delete orderStatus;
        }

    public:
        Order(); //
        ~Order();
        std::vector<FoodItem*> getItems();
        void setItems(std::vector<FoodItem*>);
        void addFood(Food* );
        vector<Food*> getFood();
        AbstractTable* getTable();
        void setTable(AbstractTable*);
        Waiter* getWaiter();     //change it back to employee
        void setWaiter(Waiter*);//change it back to Employee
        void setBill(Bill*);
        Bill* getBill();
        std::string getOrderStatus();
        void toReadyStatus();
        void toReceivedStatus();
        void toProcessingStatus();
        std::string toString();
        void print();

};

#endif
```

OrderStatus.h File Reference

Header file for the **OrderStatus** class.
`#include <string>`

Classes

class **OrderStatus***The* ***OrderStatus*** *class.*

Detailed Description

Header file for the **OrderStatus** class.

This file contains the declaration of the **OrderStatus** class.

OrderStatus.h

Go to the documentation of this file.

```
#ifndef ORDER_STATUS_H
#define ORDER_STATUS_H

#include <string>

class OrderStatus{
    public:
        virtual std::string getStatus() = 0;


};

#endif
```

Pasta.h File Reference

```
#include "../Food.h"
#include <string>
#include <iostream>
```

Classes

class **Pasta***The **Pasta** class represents a generic pasta dish.*

Pasta.h

Go to the documentation of this file.

```
#ifndef PASTA_H
#define PASTA_H

#include "../Food.h"
#include <string>
#include <iostream>

class Pasta : public Food {
private:
    double cost;

public:
    Pasta();

    virtual void decorate(Pasta*) = 0;

    virtual double total() = 0;

    double getCost();

    void setCost(double cost);

    virtual ~Pasta();
};

#endif

```

pastaBase.h File Reference

```
#include "Pasta.h"
```

Classes

class **pastaBase***The **pastaBase** class represents the base of a pasta dish, which is a specific type of **Pasta**.*

pastaBase.h

Go to the documentation of this file.

```
#ifndef PASTABASE_H
#define PASTABASE_H

#include "Pasta.h"

class pastaBase : public Pasta {
public:
    pastaBase();

    virtual double total();

    virtual void decorate(Pasta*);

    ~pastaBase();
};

#endif
```

PastaType.h File Reference

`#include "Pasta.h"`

Classes

class **PastaType***The **PastaType** class represents a specific type of pasta, which is a type of **Pasta**.*

PastaType.h

Go to the documentation of this file.

```
#ifndef PASTATYPE_H
#define PASTATYPE_H

#include "Pasta.h"

class PastaType : public Pasta {
private:
    Pasta* type;

public:
    PastaType();

    virtual double total();

    virtual void decorate(Pasta* pastaType);

protected:
    ~PastaType();
};

#endif
```

PayBill.h File Reference

Header file for the **PayBill** class.
```
#include <string>
#include <iostream>
#include "TableState.h"
```

Classes

class **PayBill***The **PayBill** class.*

Detailed Description

Header file for the **PayBill** class.

This file contains the declaration of the **PayBill** class.

PayBill.h

Go to the documentation of this file.

```
#ifndef PAYBILL_H
#define PAYBILL_H

#include <string>
#include <iostream>
#include "TableState.h"

class AbstractTable;

using namespace std;

class PayBill: public TableState
{
    public:
        string getStatus();
        bool action();
};

#endif
```

PepperoniPizza.h File Reference

`#include "PizzaType.h"`

Classes

class **PepperoniPizza***The **PepperoniPizza** class represents a Pepperoni pizza type, which is a specific type of **Pizza**.*

PepperoniPizza.h

Go to the documentation of this file.

```
#ifndef PEPPERONIPIZZA_H
#define PEPPERONIPIZZA_H

#include "PizzaType.h"

class PepperoniPizza : public PizzaType {
public:
    PepperoniPizza();

    ~PepperoniPizza();
};

#endif
```

Pizza.h File Reference

```
#include "../Food.h"
#include <string>
#include <iostream>
```

Classes

class **Pizza***The **Pizza** class represents a generic pizza.*

Pizza.h

Go to the documentation of this file.

```
#ifndef PIZZA_H
#define PIZZA_H

#include "../Food.h"
#include <string>
#include <iostream>

class Pizza : public Food {
private:
    double cost;

public:
    Pizza();

    virtual void decorate(Pizza*) = 0;

    virtual double total() = 0;

    double getCost();

    void setCost(double cost);

    virtual ~Pizza();
};

#endif

```

PizzaBase.h File Reference

```
#include "Pizza.h"
```

Classes

class **PizzaBase***The **PizzaBase** class represents the base of a pizza, which is a specific type of **Pizza**.*

PizzaBase.h

Go to the documentation of this file.

```
#ifndef PIZZABASE_H
#define PIZZABASE_H

#include "Pizza.h"

class PizzaBase : public Pizza {
public:
    PizzaBase();

    virtual double total();

    virtual void decorate(Pizza*);

    ~PizzaBase();
};

#endif
```

PizzaType.h File Reference

```
#include "Pizza.h"
```

Classes

class **PizzaType***The **PizzaType** class represents a specific type of pizza, which is a type of **Pizza**.*

PizzaType.h

Go to the documentation of this file.

```
#ifndef PIZZATYPE_H
#define PIZZATYPE_H

#include "Pizza.h"

class PizzaType : public Pizza {
private:
    Pizza* type;

public:
    PizzaType();

    virtual double total();

    virtual void decorate(Pizza* pizzaType);

protected:
    ~PizzaType();
};

#endif
```

Processing.h File Reference

Header file for the **Processing** class.

```
#include "OrderStatus.h"
```

Classes

class **Processing***The **Processing** class.*

Detailed Description

Header file for the **Processing** class.

This file contains the declaration of the **Processing** class.

Processing.h

Go to the documentation of this file.

```

#ifndef PROCESSING_H
#define PROCESSING_H

#include "OrderStatus.h"
using namespace std;

class Processing: public OrderStatus{

    public:
        virtual string getStatus(){
            return "Processing";
        }

};

#endif
```

RandomString.h File Reference

Header file for the **RandomString** class.
```
#include <string>
```

Classes

class **RandomString***The **RandomString** class.*

Detailed Description

Header file for the **RandomString** class.

This file contains the declaration of the **RandomString** class.

RandomString.h

Go to the documentation of this file.

```

#ifndef RANDOM_STRING_H
#define RANDOM_STRING_H

#include <string>
using namespace std;

class RandomString
{
    public:
        static string PositiveComment[10];
        static string NegativeComment[10];
};

#endif
```

Ready.h File Reference

Header file for the **Ready** class.
```
#include "OrderStatus.h"
```

Classes

class **Ready***The **Ready** class.*

Detailed Description

Header file for the **Ready** class.

This file contains the declaration of the **Ready** class.

Ready.h

Go to the documentation of this file.

```

#ifndef READY_H
#define READY_H

#include "OrderStatus.h"

class Ready: public OrderStatus{
    public:
        virtual string getStatus(){
            return "Ready";
        }

};

#endif
```

ReadyToOrder.h File Reference

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

Classes

class **ReadyToOrder***A class that represents the state of a table when it is ready to order.*

ReadyToOrder.h

Go to the documentation of this file.

```
#ifndef READYTOORDER_H
#define READYTOORDER_H
#include <string>
#include <iostream>
using namespace std;
#include "TableState.h"

class AbstractTable;

class ReadyToOrder : public TableState

{

    public:
        string getStatus();

        bool action();

};

#endif
```

Received.h File Reference

Header file for the **Received** class.
```
#include <string>
#include "OrderStatus.h"
```

Classes

class **Received***The **Received** class.*

Detailed Description

Header file for the **Received** class.

This file contains the declaration of the **Received** class.

Received.h

Go to the documentation of this file.

```

#ifndef RECEIVED_H
#define RECEIVED_H

#include <string>
#include "OrderStatus.h"
using namespace std;

class Received: public OrderStatus{
    public:
        virtual std::string getStatus(){
            return "Received";
        }

};

#endif
```

Review.h File Reference

Header file for the **Review** class.
```
#include <string>
```

Classes

class **Review***The* ***Review*** *class.*

Detailed Description

Header file for the **Review** class.

This file contains the declaration of the **Review** class.

Review.h

Go to the documentation of this file.

```

#ifndef REVIEW_H
#define REVIEW_H

using namespace std;
#include <string>

class Review {
public:
    Review();
    Review(const std::string& comment, int rating);
    std::string getReviewComment() const;
    int getRating() const;
    bool operator==(const Review& other) const;
    void setRating(int r){this->Rating=r;}
    void setReviewComment(string c){this->ReviewComment=c;}
private:
    std::string ReviewComment;
    int Rating;
};
//#include "Review.cpp"
#endif // REVIEW_H
```

Spaghetti.h File Reference

`#include "pastaBase.h"`

Classes

class **Spaghetti***The **Spaghetti** class represents spaghetti pasta, which is a specific type of PastaBase.*

Spaghetti.h

Go to the documentation of this file.

```
#ifndef SPAGHETTI_H
#define SPAGHETTI_H

#include "pastaBase.h"

class Spaghetti : public pastaBase  {
public:
    Spaghetti();
};

#endif
```

Table.h File Reference

Header file for the **Table** class.

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

Classes

class **Table***The **Table** class.*

Detailed Description

Header file for the **Table** class.

This file contains the declaration of the **Table** class.

Table.h

Go to the documentation of this file.

```

#ifndef TABLE_H
#define TABLE_H

#include <iostream>
using namespace std;
#include "AbstractTable.h"

class Table : public AbstractTable
{
    private:
        /* data */
    public:
        Table(){};
        ~Table(){};
        bool AddTable(AbstractTable* table){return false;}
        AbstractTable* SeparateTable(){return NULL;}
        int getTableNumber();


};



#endif
```

TableIterator.h File Reference

Header file for the **TableIterator** class.
```
#include "Iterator.h"
#include <vector>
```

Classes

class **TableIterator***The* ***TableIterator*** *class.*

Detailed Description

Header file for the **TableIterator** class.

This file contains the declaration of the **TableIterator** class.

TableIterator.h

Go to the documentation of this file.

```

#ifndef TABLEITERATOR_H
#define TABLEITERATOR_H

#include "Iterator.h"
#include <vector>

class Table;

class TableIterator : public Iterator {
public:
    TableIterator(const std::vector<Table*>& tables);
    ~TableIterator();
    Table* first() override;
    Table* next() override;
    bool hasNext() override;
    Table* current() override;
private:
    std::vector<Table*> tables;
    int currentPos;
};

#endif // TABLEITERATOR_H
```

TableState.h File Reference

Header file for the **TableState** class.
```
#include <string>
#include <iostream>
```

Classes

class **TableState***The **TableState** class.*

Detailed Description

Header file for the **TableState** class.

This file contains the declaration of the **TableState** class.

TableState.h

Go to the documentation of this file.

```

#ifndef TABLE_STATE
#define TABLE_STATE

#include <string>
#include <iostream>
using namespace std;


class AbstractTable;

class TableState
{
    protected:
        AbstractTable * table;

    public:
        TableState(){
            this->table = nullptr;
        }
        virtual string getStatus() = 0;
        virtual bool action() = 0;
        void setTable(AbstractTable* table){
            this->table = table;
        }

};
#endif
```

ThickBasePizza.h File Reference

```
#include "PizzaBase.h"
```

Classes

class **ThickBasePizza***The **ThickBasePizza** class represents a pizza with a thick crust base, which is a specific type of **PizzaBase**.*

ThickBasePizza.h

Go to the documentation of this file.

```
#ifndef THICKBASEPIZZA_H
#define THICKBASEPIZZA_H

#include "PizzaBase.h"

class ThickBasePizza : public PizzaBase {
public:
    ThickBasePizza();
};

#endif
```

ThinBasePizza.h File Reference

```
#include "PizzaBase.h"
```

Classes

class **ThinBasePizza***The* ***ThinBasePizza*** *class represents a pizza with a thin crust base, which is a specific type of* ***PizzaBase****.*

ThinBasePizza.h

Go to the documentation of this file.

```
#ifndef THINBASEPIZZA_H
#define THINBASEPIZZA_H

#include "PizzaBase.h"

class ThinBasePizza : public PizzaBase {
public:
    ThinBasePizza();
};

#endif
```

VegetarianBurger.h File Reference

`#include "BurgerBase.h"`

Classes

class **VegetarianBurger***The* ***VegetarianBurger*** *class represents a vegetarian burger, which is a specific type of* ***BurgerBase****.*

VegetarianBurger.h

Go to the documentation of this file.

```
#ifndef VEGETARIANBURGER_H
#define VEGETARIANBURGER_H

#include "BurgerBase.h"

class VegetarianBurger : public BurgerBase {
public:
    VegetarianBurger();
};

#endif

```

Waiter.h File Reference

```
#include "TableIterator.h"
#include "Employee.h"
#include "Order.h"
#include <vector>
```

Classes

class **Waiter***Represents a waiter who takes and delivers orders in a restaurant.*

Waiter.h

Go to the documentation of this file.

```
#ifndef WAITER_H
#define WAITER_H

#include "TableIterator.h"
#include "Employee.h"
#include "Order.h"
#include <vector>

class Waiter :public Employee{
public:
    Waiter(int id);
    ~Waiter();
    void getReviewsForKitchenDepartment();

    void CollectOrderFromKitchen(Order*);

    void TakeOrder(Table*);//change it back to table
    void TakeOrder(Order*);
    void deliverOrders();
    void assignTables(vector<Table*>&);
    bool isFullyOccupied();
    void setMaxTables(int);
    void iterateTables();


private:
    vector<Order*> customerOrder;
    int employeeId, maxTables;
    vector<Table*> tables;
    TableIterator* tableIterator;

};

#endif // WAITER_H

```

Waiting.h File Reference

Header file for the **Waiting** class.
```
#include <string>
#include <iostream>
#include "TableState.h"
```

Classes

class **Waiting***The **Waiting** class.*

Detailed Description

Header file for the **Waiting** class.

This file contains the declaration of the **Waiting** class.

Waiting.h

Go to the documentation of this file.

```

#ifndef WAITING_H
#define WAITING_H

#include <string>
#include <iostream>
using namespace std;
#include "TableState.h"

class AbstractTable;

class Waiting : public TableState
{
    public:
        string getStatus();
        bool action();

};

#endif
```

Index

INDEX