

My Project

Generated by Doxygen 1.9.1

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 Class Documentation	7
3.1 __pthread_cleanup_frame Struct Reference	7
3.2 __pthread_unwind_buf_t Struct Reference	7
3.3 _pthread_cleanup_buffer Struct Reference	8
3.4 AbstractBill Class Reference	8
3.5 AbstractTable Class Reference	9
3.5.1 Detailed Description	11
3.5.2 Member Function Documentation	11
3.5.2.1 getBill()	11
3.5.2.2 getCustomers()	12
3.5.2.3 getnumberOfSeats()	12
3.5.2.4 getState()	12
3.5.2.5 getTableID()	12
3.5.2.6 getWaiter()	13
3.5.2.7 setCustomers()	13
3.5.2.8 setState()	13
3.5.2.9 setWaiter()	13
3.6 App Class Reference	14
3.7 BBQRibs Class Reference	14
3.8 BBQSteak Class Reference	15
3.9 Beverage Class Reference	16
3.9.1 Detailed Description	17
3.10 Bill Class Reference	17
3.11 BillDecorator Class Reference	19
3.12 BillItem Class Reference	20
3.13 BubblegumMilkshake Class Reference	21
3.14 BuffaloWings Class Reference	22
3.15 Card Class Reference	23
3.16 Cash Class Reference	24
3.17 CheeseBurger Class Reference	25
3.18 CheeseSteak Class Reference	26
3.19 Chef Class Reference	27
3.19.1 Detailed Description	29
3.19.2 Member Function Documentation	29
3.19.2.1 GetRole()	29
3.19.2.2 SetNextChef()	29
3.19.2.3 VisitTable()	29

3.20 ChefNotifier Class Reference	30
3.21 ChickenNuggets Class Reference	30
3.22 ChickenTenders Class Reference	31
3.23 ChickenWings Class Reference	32
3.24 Chips Class Reference	33
3.25 ChocolateBrownies Class Reference	34
3.26 Coke Class Reference	35
3.27 CokeZero Class Reference	36
3.28 Coleslaw Class Reference	37
3.28.1 Constructor & Destructor Documentation	38
3.28.1.1 Coleslaw()	38
3.29 commisChef Class Reference	39
3.29.1 Detailed Description	40
3.29.2 Constructor & Destructor Documentation	40
3.29.2.1 commisChef()	40
3.29.3 Member Function Documentation	40
3.29.3.1 PrepareDish()	40
3.30 ConcreteTableIterator Class Reference	41
3.31 Customer Class Reference	43
3.31.1 Detailed Description	44
3.31.2 Constructor & Destructor Documentation	44
3.31.2.1 Customer()	44
3.31.3 Member Function Documentation	45
3.31.3.1 accept()	45
3.31.3.2 assignCustomerTable()	45
3.31.3.3 checkOrder()	45
3.31.3.4 complimentWaiter()	46
3.31.3.5 createTab()	46
3.31.3.6 getBill()	46
3.31.3.7 getMood()	46
3.31.3.8 getName()	47
3.31.3.9 getTableNum()	47
3.31.3.10 leaveRestaurant()	47
3.31.3.11 pay()	47
3.31.3.12 setMood()	48
3.31.3.13 setReadyToLeaveStatus()	48
3.31.3.14 setReadyToOrderStatus()	48
3.31.3.15 setReadyToPayStatus()	48
3.31.3.16 setTab()	49
3.31.3.17 setTableNum()	49
3.32 CustomTipDecorator Class Reference	50
3.33 Dessert Class Reference	51

3.33.1 Detailed Description	51
3.34 Dish Class Reference	52
3.34.1 Detailed Description	53
3.34.2 Constructor & Destructor Documentation	53
3.34.2.1 Dish()	53
3.34.3 Member Function Documentation	54
3.34.3.1 getCustomerName()	54
3.34.3.2 getCustomerTable()	54
3.34.3.3 getDishStatus()	54
3.34.3.4 setDishStatus()	54
3.35 DishStatus Class Reference	55
3.35.1 Detailed Description	56
3.36 Donuts Class Reference	57
3.37 Engine Class Reference	58
3.38 Facade Class Reference	58
3.38.1 Detailed Description	59
3.38.2 Constructor & Destructor Documentation	59
3.38.2.1 Facade()	59
3.38.3 Member Function Documentation	59
3.38.3.1 GetInstance()	60
3.39 Floor Class Reference	60
3.39.1 Detailed Description	61
3.39.2 Member Function Documentation	61
3.39.2.1 splitTables()	61
3.40 generalWaiter Class Reference	61
3.40.1 Detailed Description	62
3.40.2 Member Function Documentation	63
3.40.2.1 addToTab()	63
3.40.2.2 getTab()	63
3.40.2.3 payTab()	63
3.41 HeadChef Class Reference	64
3.41.1 Detailed Description	65
3.41.2 Member Function Documentation	65
3.41.2.1 AddDish()	65
3.41.2.2 Attach()	66
3.41.2.3 Notify()	66
3.41.2.4 PrepareDish()	66
3.42 HotDog Class Reference	67
3.43 Kitchen Class Reference	68
3.44 MacAndCheese Class Reference	68
3.45 MainDish Class Reference	69
3.45.1 Detailed Description	70

3.46 MaitreD Class Reference	70
3.46.1 Detailed Description	71
3.46.2 Member Function Documentation	71
3.46.2.1 allocateTable()	71
3.46.2.2 mergeTables()	72
3.46.2.3 splitTables()	72
3.47 Manager Class Reference	72
3.47.1 Member Function Documentation	73
3.47.1.1 getComplaints()	73
3.47.1.2 handleComplaint()	73
3.47.1.3 visitTable()	73
3.48 Mediator Class Reference	74
3.49 Menu Class Reference	74
3.49.1 Detailed Description	76
3.50 MenuDecorator Class Reference	76
3.50.1 Detailed Description	77
3.51 MushroomSauce Class Reference	78
3.51.1 Constructor & Destructor Documentation	79
3.51.1.1 MushroomSauce()	79
3.52 Observer Class Reference	79
3.53 Occupied Class Reference	80
3.53.1 Member Function Documentation	81
3.53.1.1 getState()	81
3.53.1.2 handleState()	81
3.54 OnionRings Class Reference	82
3.55 Pancakes Class Reference	83
3.56 PaymentStrategy Class Reference	84
3.57 Preparing Class Reference	85
3.57.1 Detailed Description	86
3.57.2 Member Function Documentation	86
3.57.2.1 getStatus()	86
3.58 ReadyForPickUp Class Reference	87
3.58.1 Detailed Description	88
3.58.2 Member Function Documentation	88
3.58.2.1 getStatus()	88
3.59 Salad Class Reference	89
3.59.1 Constructor & Destructor Documentation	90
3.59.1.1 Salad()	90
3.60 Sprite Class Reference	90
3.61 Starter Class Reference	91
3.61.1 Detailed Description	92
3.62 StillQueued Class Reference	92

3.62.1 Detailed Description	93
3.62.2 Member Function Documentation	93
3.62.2.1 getStatus()	94
3.63 StrawberryMilkshake Class Reference	94
3.64 SubBill Class Reference	95
3.65 Tab Class Reference	96
3.65.1 Detailed Description	97
3.65.2 Member Function Documentation	97
3.65.2.1 addToTab()	97
3.65.2.2 getName()	98
3.65.2.3 getTab()	98
3.65.2.4 subtractFromTab()	98
3.66 Table Class Reference	99
3.66.1 Member Function Documentation	100
3.66.1.1 acceptVisitor()	100
3.66.1.2 clone()	100
3.66.1.3 getBill()	100
3.66.1.4 getCustomers()	101
3.66.1.5 getState()	101
3.66.1.6 getWaiter()	101
3.66.1.7 operator+() [1/2]	101
3.66.1.8 operator+() [2/2]	102
3.66.1.9 setState()	102
3.66.1.10 setWaiter()	102
3.67 TableGroup Class Reference	103
3.67.1 Member Function Documentation	105
3.67.1.1 acceptVisitor()	105
3.67.1.2 addTable()	105
3.67.1.3 clone()	105
3.67.1.4 getTables()	106
3.67.1.5 operator+() [1/2]	106
3.67.1.6 operator+() [2/2]	106
3.68 TableIterator Class Reference	107
3.69 TableState Class Reference	109
3.70 Unoccupied Class Reference	109
3.70.1 Member Function Documentation	110
3.70.1.1 getState()	110
3.70.1.2 handleState()	110
3.71 Visitor Class Reference	111
3.72 Waffles Class Reference	111
3.73 Waiter Class Reference	112
3.73.1 Detailed Description	113

3.73.2 Member Function Documentation	114
3.73.2.1 deliverOrder()	114
3.73.2.2 sendOrder()	114
Index	115

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

__pthread_cleanup_frame	7
__pthread_unwind_buf_t	7
__pthread_cleanup_buffer	8
AbstractBill	8
Bill	17
BillItem	20
SubBill	95
BillDecorator	19
CustomTipDecorator	50
AbstractTable	9
Table	99
TableGroup	103
App	14
Chef	27
HeadChef	64
commisChef	39
ChefNotifier	30
Customer	43
Dish	52
DishStatus	55
Preparing	85
ReadyForPickUp	87
StillQueued	92
Facade	58
Floor	60
Kitchen	68
Manager	72
Mediator	74
Engine	58
Menu	74
Beverage	16
BubblegumMilkshake	21
Coke	35

CokeZero	36
Sprite	90
StrawberryMilkshake	94
Dessert	51
ChocolateBrownies	34
Donuts	57
Pancakes	83
Waffles	111
MainDish	69
BBQRibs	14
BBQSteak	15
BuffaloWings	22
CheeseBurger	25
CheeseSteak	26
ChickenTenders	31
HotDog	67
MacAndCheese	68
MenuDecorator	76
Coleslaw	37
MushroomSauce	78
Salad	89
Starter	91
ChickenNuggets	30
ChickenWings	32
Chips	33
OnionRings	82
Observer	79
PaymentStrategy	84
Card	23
Cash	24
Tab	96
TableIterator	107
ConcreteTableIterator	41
TableState	109
Occupied	80
Unoccupied	109
Visitor	111
Waiter	112
MaitreD	70
generalWaiter	61

Chapter 2

Class Index

2.1 Class List

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

__pthread_cleanup_frame	7
__pthread_unwind_buf_t	7
_pthread_cleanup_buffer	8
AbstractBill	8
AbstractTable	
Customers are seated and served at tables	9
App	14
BBQRibs	14
BBQSteak	15
Beverage	
Beverages base class Beverages are coke zero, coke, sprite, strawberry milkshakes and bubblegum milkshakes Beverages will all take 1 second to prepare	16
Bill	17
BillDecorator	19
BillItem	20
BubblegumMilkshake	21
BuffaloWings	22
Card	23
Cash	24
CheeseBurger	25
CheeseSteak	26
Chef	
Chef base class Chef classes are commisChef and HeadChef	27
ChefNotifier	30
ChickenNuggets	30
ChickenTenders	31
ChickenWings	32
Chips	33
ChocolateBrownies	34
Coke	35
CokeZero	36
Coleslaw	37
commisChef	
CommisChef class commisChef is responsible for making dishes given to them by their HeadChef once a given a dish a commisChef	39

ConcreteTableIterator	41
Customer	
Represents a restaurant customer	43
CustomTipDecorator	50
Dessert	
Desserts base class Desserts are waffle, chocolate brownies, donuts and pancakes Desserts will all take 3 seconds to prepare	51
Dish	
Dish class definition each Dish consists of multiple menu objects as well as the customer's name, their table and the state of the dish once it is being prepared	52
DishStatus	
DishStatus base class, inherits from Dish There are 3 different DishStatus classes: Preparing, ReadyForPickUp and StillQueued each DishStatus updates a dish's status to their corresponding class	55
Donuts	57
Engine	58
Facade	
Acts as main interface for system/interface	58
Floor	
Floor is responsible for managing the tables and waiters	60
generalWaiter	
GeneralWaiter is responsible for serving customers	61
HeadChef	
HeadChef class HeadChef delegates making of dishes to commisChef class HeadChef can also add and remove commisChefs as necessary commisChefs given dishes are moved from a free↔ Chef to a busyChefs queue once a dish is complete commisChef is moved back to freeChefs	64
HotDog	67
Kitchen	68
MacAndCheese	68
MainDish	
Main dishes base class Main dishes are cheeseburgers, hot dogs, mac and cheese, bbq ribs, chicken tenders, cheesesteak, bbq steak and buffalo wings Main dishes will all take 10 seconds to prepare	69
MaitreD	
MaitreD is responsible for allocating tables to customers	70
Manager	72
Mediator	74
Menu	
Menu base class	74
MenuDecorator	
MenuDecorator class	76
MushroomSauce	78
Observer	79
Occupied	80
OnionRings	82
Pancakes	83
PaymentStrategy	84
Preparing	
DishStatus Preparing class used to indicate when a dish is still being prepared	85
ReadyForPickUp	
DishStatus ReadyForPickUp class used to indicate when a dish is ready to be picked up	87
Salad	89
Sprite	90
Starter	
Starters base class Starters are onion rings, chicken wings and chicken nuggets Starters will all take 5 seconds to prepare	91
StillQueued	
DishStatus StillQueued class used to indicate when a dish has yet to begin preparation	92

StrawberryMilkshake	94
SubBill	95
Tab	
Holds the running tab of a restaurant customer	96
Table	99
TableGroup	103
TableIterator	107
TableState	109
Unoccupied	109
Visitor	111
Waffles	111
Waiter	
Waiters are responsible for serving customers	112

Chapter 3

Class Documentation

3.1 `__pthread_cleanup_frame` Struct Reference

Public Attributes

- `void(* __cancel_routine)(void *)`
- `void * __cancel_arg`
- `int __do_it`
- `int __cancel_type`

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

- `pthread.h`

3.2 `__pthread_unwind_buf_t` Struct Reference

Public Attributes

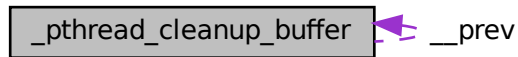
- struct {
 `__jmp_buf __cancel_jmp_buf`
 `int __mask_was_saved`
} `__cancel_jmp_buf [1]`
- `void * __pad [4]`

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

- `pthread.h`

3.3 `_pthread_cleanup_buffer` Struct Reference

Collaboration diagram for `_pthread_cleanup_buffer`:



Public Attributes

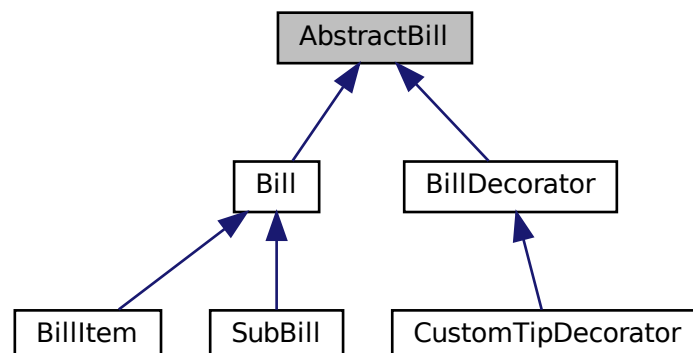
- `void(* __routine)(void *)`
- `void * __arg`
- `int __canceltype`
- `struct _pthread_cleanup_buffer * __prev`

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

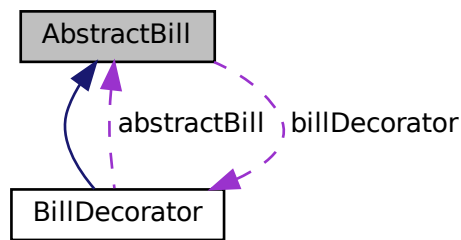
- `pthread.h`

3.4 AbstractBill Class Reference

Inheritance diagram for AbstractBill:



Collaboration diagram for AbstractBill:



Public Member Functions

- virtual double **getTotalCost** ()=0

Public Attributes

- [BillDecorator](#) * **billDecorator** {}

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

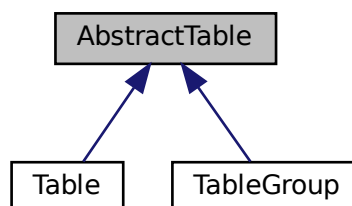
- Bill.h

3.5 AbstractTable Class Reference

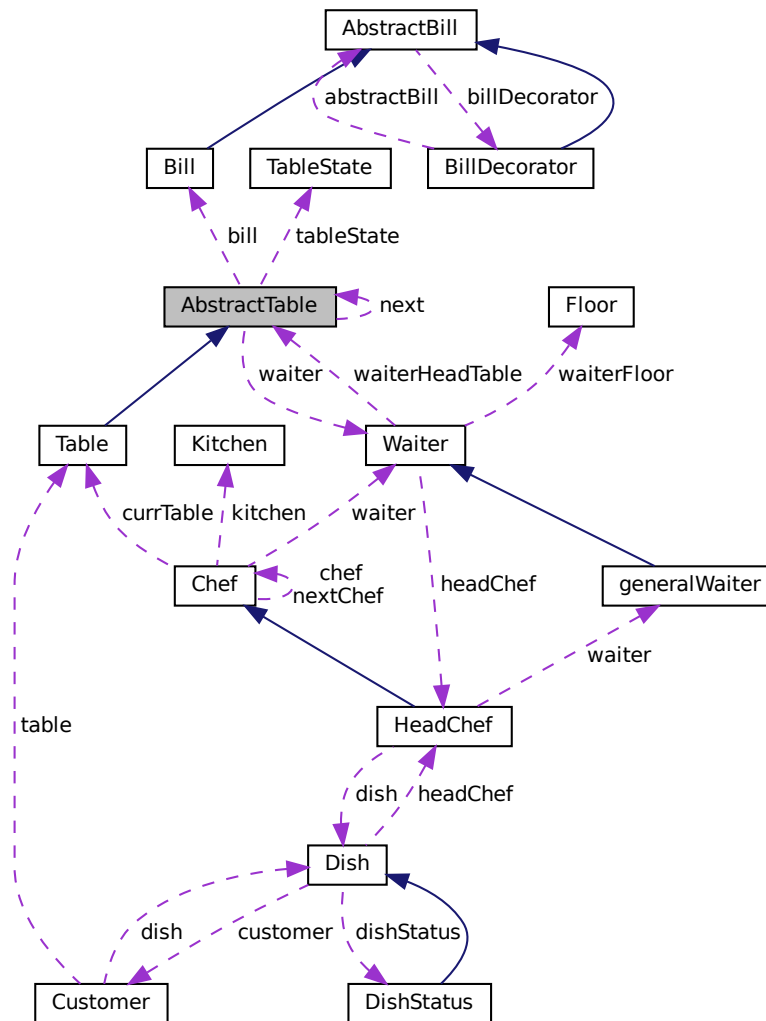
Customers are seated and served at tables.

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

Inheritance diagram for AbstractTable:



Collaboration diagram for AbstractTable:



Public Member Functions

- **AbstractTable** (int numberOfSeats)
- virtual void **acceptVisitor** ([Visitor](#) *visitor)=0
- virtual [AbstractTable](#) * **operator+** ([TableGroup](#) *tableGroup)=0
- virtual [AbstractTable](#) * **operator+** ([Table](#) *table)=0
- virtual [AbstractTable](#) * **clone** ()=0
- int [getnumberOfSeats](#) ()
Returns table's number of seats.
- int [getTableID](#) ()
Returns table ID.
- std::vector< [Customer](#) * > [getCustomers](#) ()
Returns customers currently at table.
- void [setState](#) ([TableState](#) *tableState)

- Sets the state of the table.*
- `TableState * getState ()`
Returns table's state.
- `void handleState ()`
State handler for table class.
- `Bill * getBill (Customer *customer)`
Returns customer bill.
- `void setWaiter (Waiter *waiter)`
Sets waiter to the table.
- `Waiter * getWaiter ()`
Gets the table's assigned waiter.
- `void getOrders ()`
- `void setCustomers (std::vector< Customer * > newCustomers)`
Adds customers to the table.

Public Attributes

- `AbstractTable * next`

Protected Attributes

- `TableState * tableState`
- `std::vector< Customer * > customers`
- `Bill * bill`
- `Waiter * waiter`
- `int numberOfSeats`
- `int tableID`

3.5.1 Detailed Description

Customers are seated and served at tables.

3.5.2 Member Function Documentation

3.5.2.1 getBill()

```
Bill * AbstractTable::getBill (
    Customer * customer )
```

Returns customer bill.

Parameters

<i>customer</i>	
-----------------	--

Returns

Bill*

3.5.2.2 getCustomers()

```
std::vector< Customer * > AbstractTable::getCustomers ( )
```

Returns customers currently at table.

Returns

std::vector<Customer *>

3.5.2.3 getnumberOfSeats()

```
int AbstractTable::getnumberOfSeats ( )
```

Returns table's number of seats.

Returns

int

3.5.2.4 getState()

```
TableState * AbstractTable::getState ( )
```

Returns table's state.

Returns

TableState*

3.5.2.5 getTableID()

```
int AbstractTable::getTableID ( )
```

Returns table ID.

Returns

int

3.5.2.6 getWaiter()

```
Waiter * AbstractTable::getWaiter ( )
```

Gets the table's assigned waiter.

Returns

Waiter*

3.5.2.7 setCustomers()

```
void AbstractTable::setCustomers (
    std::vector< Customer * > newCustomers )
```

Adds customers to the table.

Parameters

<i>newCustomers</i>	
---------------------	--

3.5.2.8 setState()

```
void AbstractTable::setState (
    TableState * tableState )
```

Sets the state of the table.

Parameters

<i>tableState</i>	
-------------------	--

3.5.2.9 setWaiter()

```
void AbstractTable::setWaiter (
    Waiter * waiter )
```

Sets waiter to the table.

Parameters

<i>waiter</i>	
---------------	--

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

- Table.h
- Table.cpp

3.6 App Class Reference

Public Member Functions

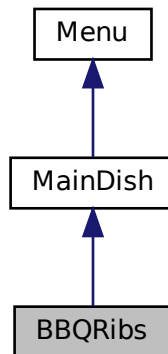
- **App** (int timerSeconds)
- void **startSimulation** ()
- void **setFacade** ([Facade](#) *f)

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

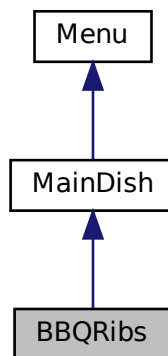
- App.cpp

3.7 BBQRibs Class Reference

Inheritance diagram for BBQRibs:



Collaboration diagram for BBQRibs:



Public Member Functions

- [BBQRibs](#) ()
Constrcutor for BBQ ribs main dish.

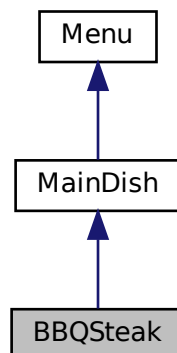
Additional Inherited Members

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

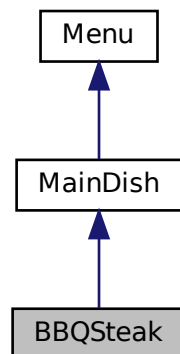
- Menu.h
- Menu.cpp

3.8 BBQSteak Class Reference

Inheritance diagram for BBQSteak:



Collaboration diagram for BBQSteak:



Public Member Functions

- [BBQSteak\(\)](#)
Constrcutor for BBQ steak main dish.

Additional Inherited Members

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

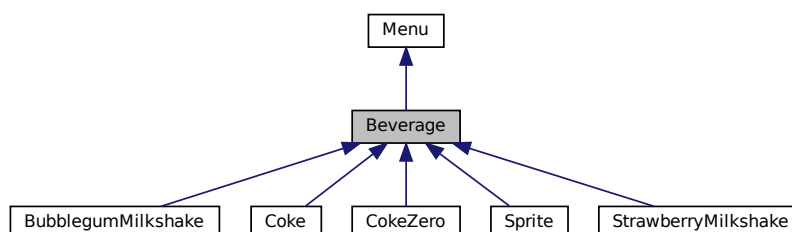
- Menu.h
- Menu.cpp

3.9 Beverage Class Reference

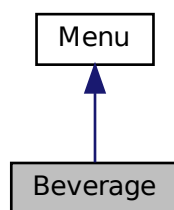
Beverages base class Beverages are coke zero, coke, sprite, strawberry milkshakes and bubblegum milkshakes
Beverages will all take 1 second to prepare.

```
#include <Menu.h>
```

Inheritance diagram for Beverage:



Collaboration diagram for Beverage:



Additional Inherited Members

3.9.1 Detailed Description

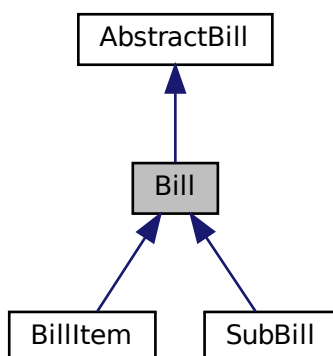
Beverages base class Beverages are coke zero, coke, sprite, strawberry milkshakes and bubblegum milkshakes
Beverages will all take 1 second to prepare.

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

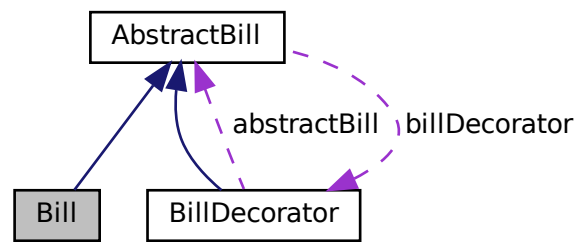
- Menu.h

3.10 Bill Class Reference

Inheritance diagram for Bill:



Collaboration diagram for Bill:



Public Member Functions

- virtual void **paymentMethod** ()
- void **handleTip** ()
- void **getBill** ()
- void **addTip** (float tip)
- **Bill** (double price)
- double **calculateBill** ()
- void **setPaymentMethod** ([PaymentStrategy](#) *method)
- double **getTotalCost** ()
- virtual void **getSubBill** (std::string customerName)

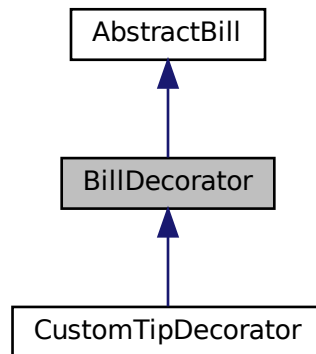
Additional Inherited Members

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

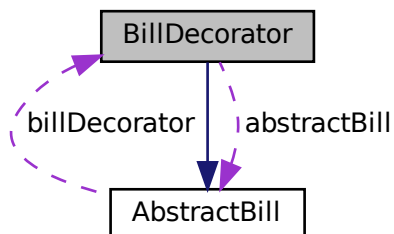
- Bill.h
- Bill.cpp

3.11 BillDecorator Class Reference

Inheritance diagram for BillDecorator:



Collaboration diagram for BillDecorator:



Public Member Functions

- **BillDecorator** (**Bill** *bill)
- double **getTotalCost** ()

Public Attributes

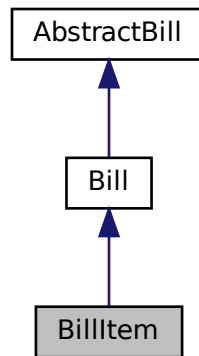
- **AbstractBill** * **abstractBill**

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

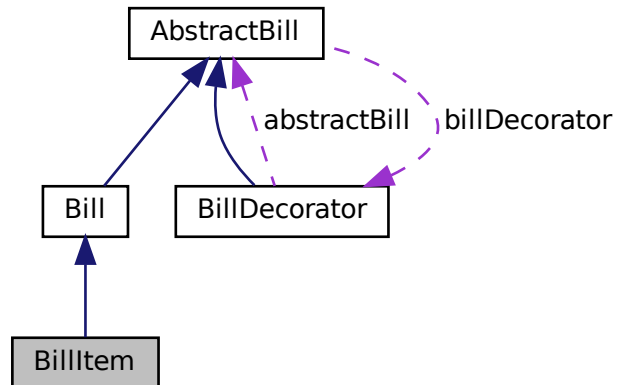
- **Bill.h**
- **Bill.cpp**

3.12 BillItem Class Reference

Inheritance diagram for BillItem:



Collaboration diagram for BillItem:



Public Member Functions

- **BillItem** (std::string item, double cost)
- double **getTotalCost** ()

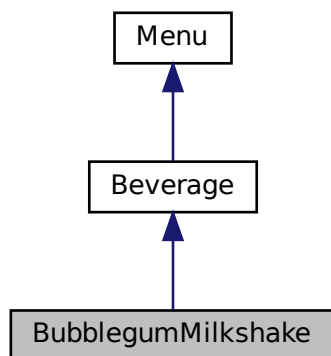
Additional Inherited Members

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

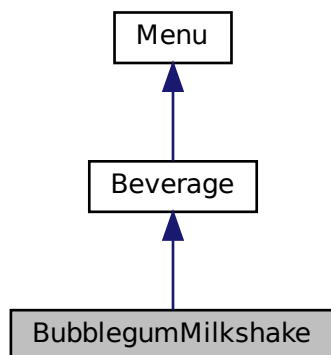
- Bill.h
- Bill.cpp

3.13 BubblegumMilkshake Class Reference

Inheritance diagram for BubblegumMilkshake:



Collaboration diagram for BubblegumMilkshake:



Public Member Functions

- [BubblegumMilkshake](#) ()
Constrcutor for bubblegum milkshake beverage.

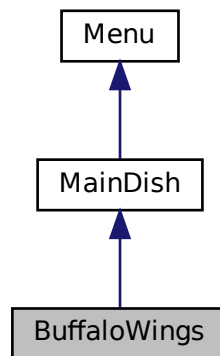
Additional Inherited Members

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

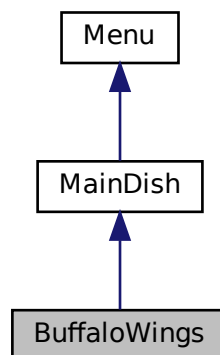
- Menu.h
- Menu.cpp

3.14 BuffaloWings Class Reference

Inheritance diagram for BuffaloWings:



Collaboration diagram for BuffaloWings:



Public Member Functions

- [BuffaloWings](#) ()
Constrcutor for buffalo wings main dish.

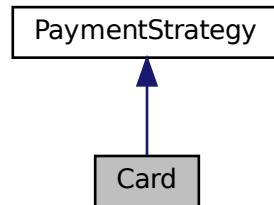
Additional Inherited Members

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

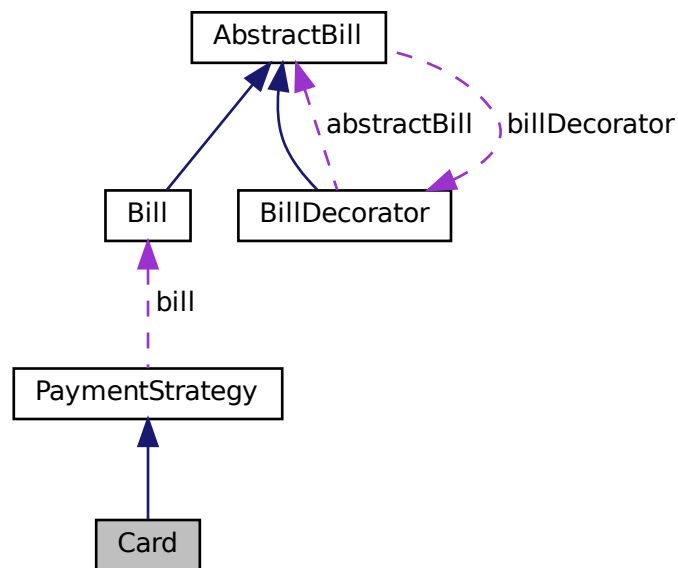
- Menu.h
- Menu.cpp

3.15 Card Class Reference

Inheritance diagram for Card:



Collaboration diagram for Card:



Public Member Functions

- void **paymentMethod** ()
- void **getPaymentMethod** ()

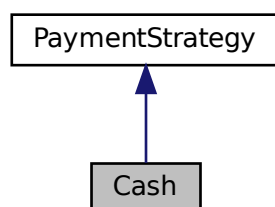
Additional Inherited Members

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

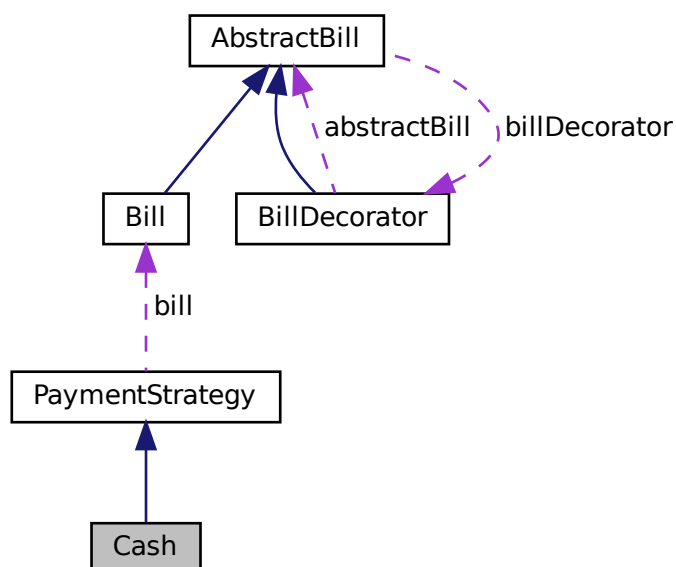
- Bill.h
- Bill.cpp

3.16 Cash Class Reference

Inheritance diagram for Cash:



Collaboration diagram for Cash:



Public Member Functions

- void **paymentMethod** ()
- void **getPaymentMethod** ()

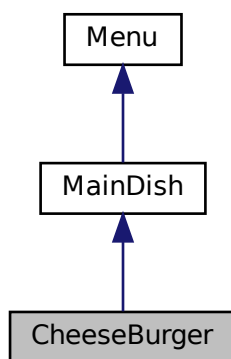
Additional Inherited Members

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

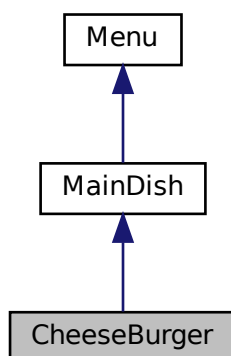
- Bill.h
- Bill.cpp

3.17 CheeseBurger Class Reference

Inheritance diagram for CheeseBurger:



Collaboration diagram for CheeseBurger:



Public Member Functions

- [CheeseBurger](#) ()
Constrcutor for cheese burger main dish.

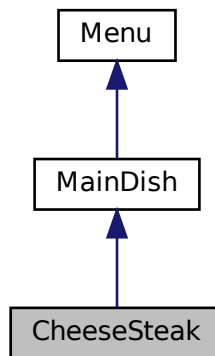
Additional Inherited Members

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

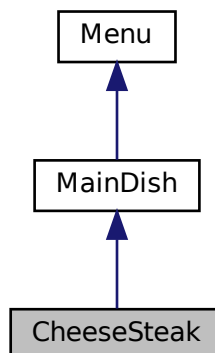
- Menu.h
- Menu.cpp

3.18 CheeseSteak Class Reference

Inheritance diagram for CheeseSteak:



Collaboration diagram for CheeseSteak:



Public Member Functions

- [CheeseSteak](#) ()

Constrcutor for cheesesteak main dish.

Additional Inherited Members

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

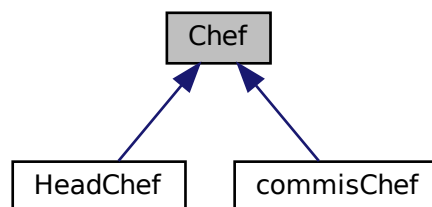
- Menu.h
- Menu.cpp

3.19 Chef Class Reference

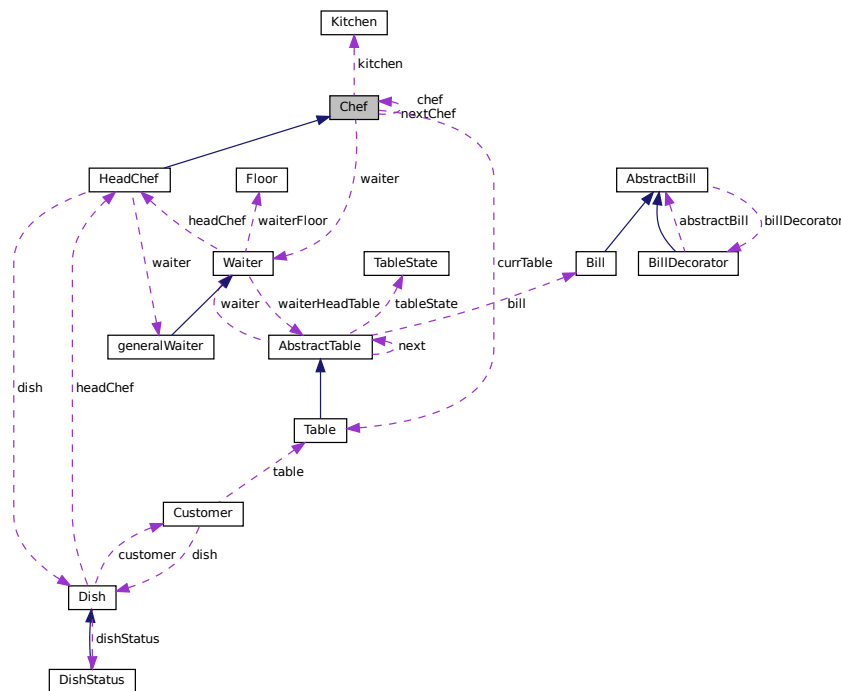
[Chef](#) base class [Chef](#) classes are [commisChef](#) and [HeadChef](#).

```
#include <Chef.h>
```

Inheritance diagram for Chef:



Collaboration diagram for Chef:



Public Member Functions

- void **VisitTable** (**Table** *table)
assigns a table to a chef when they visit it
- std::string **GetRole** ()
returns what type of chef a specific chef object is, commis or Head
- virtual void **PrepareDish** (**Dish** *dish)=0
- void **SetNextChef** (**Chef** *chef)
sets the next chef after the current chef who will carry on the flow of the system

Public Attributes

- **Kitchen** * kitchen
- **Chef** * chef
- **Waiter** * waiter

Protected Attributes

- std::string role
- **Chef** * nextChef
- **Table** * currTable

3.19.1 Detailed Description

[Chef](#) base class [Chef](#) classes are [commisChef](#) and [HeadChef](#).

3.19.2 Member Function Documentation

3.19.2.1 GetRole()

```
std::string Chef::GetRole ( )
```

returns what type of chef a specific chef object is, commis or Head

Returns

std::string

3.19.2.2 SetNextChef()

```
void Chef::SetNextChef (
    Chef * chef )
```

sets the next chef after the current chef who will carry on the flow of the system

Parameters

<i>chef</i>	
-------------	--

3.19.2.3 VisitTable()

```
void Chef::VisitTable (
    Table * table )
```

assigns a table to a chef when they visit it

Parameters

<i>table</i>	
--------------	--

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

- Chef.h
- Chef.cpp

3.20 ChefNotifier Class Reference

Public Member Functions

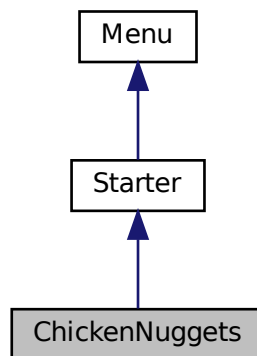
- virtual void **Attach** ([Observer](#) *observer)=0
- virtual void **Detach** ([Observer](#) *observer)=0
- virtual void **Notify** ()=0

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

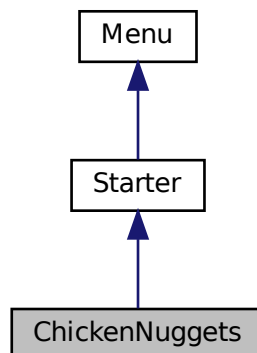
- Chef.h

3.21 ChickenNuggets Class Reference

Inheritance diagram for ChickenNuggets:



Collaboration diagram for ChickenNuggets:



Public Member Functions

- [ChickenNuggets\(\)](#)
Constrcutor for chicken nuggets starter.

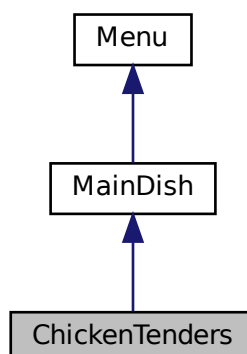
Additional Inherited Members

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

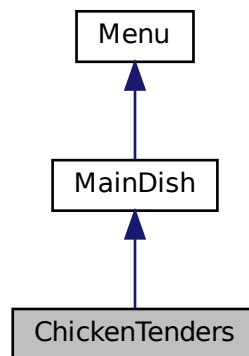
- Menu.h
- Menu.cpp

3.22 ChickenTenders Class Reference

Inheritance diagram for ChickenTenders:



Collaboration diagram for ChickenTenders:



Public Member Functions

- [ChickenTenders](#) ()
Constrcutor for chicken tenders main dish.

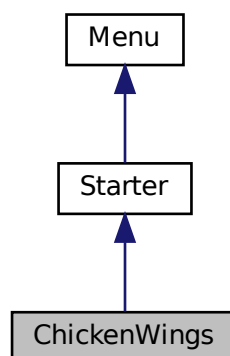
Additional Inherited Members

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

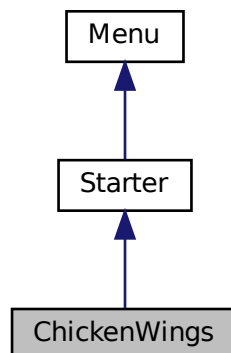
- Menu.h
- Menu.cpp

3.23 ChickenWings Class Reference

Inheritance diagram for ChickenWings:



Collaboration diagram for ChickenWings:



Public Member Functions

- [ChickenWings](#) ()
Constrcutor for chicken wings starter.

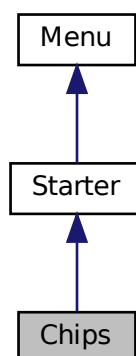
Additional Inherited Members

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

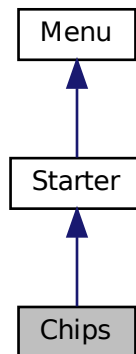
- Menu.h
- Menu.cpp

3.24 Chips Class Reference

Inheritance diagram for Chips:



Collaboration diagram for Chips:



Public Member Functions

- [Chips](#) ()
Constrcutor for chips starter.

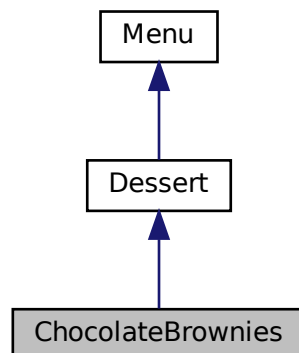
Additional Inherited Members

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

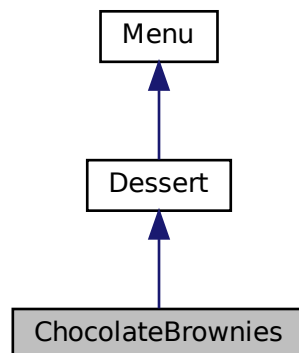
- Menu.h
- Menu.cpp

3.25 ChocolateBrownies Class Reference

Inheritance diagram for ChocolateBrownies:



Collaboration diagram for ChocolateBrownies:



Public Member Functions

- [ChocolateBrownies\(\)](#)
Constrcutor for chocolate brownies dessert.

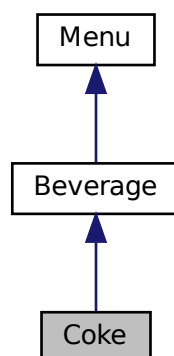
Additional Inherited Members

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

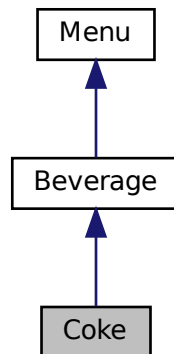
- Menu.h
- Menu.cpp

3.26 Coke Class Reference

Inheritance diagram for Coke:



Collaboration diagram for Coke:



Public Member Functions

- [Coke](#) ()
Constrcutor for [Coke](#) beverage.

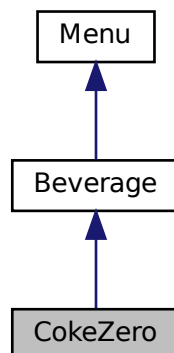
Additional Inherited Members

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

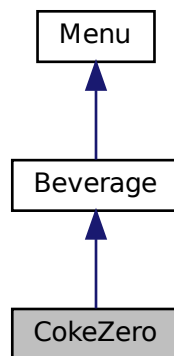
- Menu.h
- Menu.cpp

3.27 CokeZero Class Reference

Inheritance diagram for CokeZero:



Collaboration diagram for CokeZero:



Public Member Functions

- [CokeZero](#) ()
Constrcutor for [CokeZero](#) beverage.

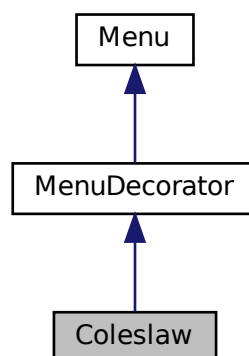
Additional Inherited Members

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

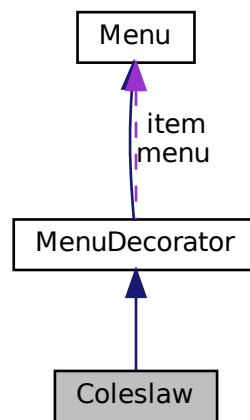
- Menu.h
- Menu.cpp

3.28 Coleslaw Class Reference

Inheritance diagram for Coleslaw:



Collaboration diagram for Coleslaw:



Public Member Functions

- `std::string` **getDescription** ()
 - `double` **getPrice** ()
 - **Coleslaw** (`Menu` *baseItem, `std::string` description, `double` price, `int` timeToprepare)
- Constrcutor for coleslaw custom addition.*
- `int` **getTimeToPrepare** ()

Additional Inherited Members

3.28.1 Constructor & Destructor Documentation

3.28.1.1 Coleslaw()

```

Coleslaw::Coleslaw (
    Menu * baseItem,
    std::string description,
    double price,
    int timeToprepare )

```

Constrcutor for coleslaw custom addition.

Parameters

<i>baseItem</i>	
<i>description</i>	
<i>price</i>	
<i>timeToprepare</i>	

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

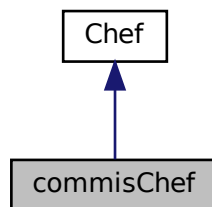
- Menu.h
- Menu.cpp

3.29 commisChef Class Reference

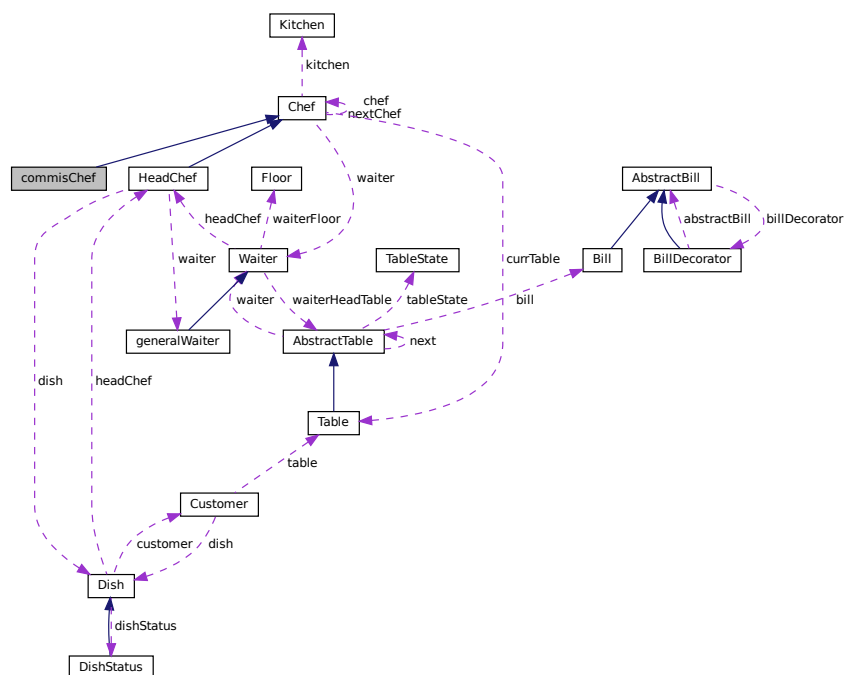
`commisChef` class `commisChef` is responsible for making dishes given to them by their `HeadChef` once a given a dish a `commisChef`

```
#include <Chef.h>
```

Inheritance diagram for `commisChef`:



Collaboration diagram for `commisChef`:



Public Member Functions

- [commisChef](#) ([HeadChef](#) *headChef)

Construct a new [commisChef](#) object takes a [HeadChef](#) object as a parameter to set the headChef of the [commisChef](#) as the passed in [HeadChef](#).

- void [PrepareDish](#) ([Dish](#) *dish)

Have a [commisChef](#) start preparing a dish assign a dish to [commisChef](#) who will be the one preparing it.

- void [Notify](#) ()

Notify the [HeadChef](#) that the dish is complete and ready for delivery Calls [HeadChef](#)'s [Notify](#) function.

Additional Inherited Members

3.29.1 Detailed Description

[commisChef](#) class [commisChef](#) is responsible for making dishes given to them by their [HeadChef](#) once a given a dish a [commisChef](#)

3.29.2 Constructor & Destructor Documentation

3.29.2.1 [commisChef](#)()

```
commisChef::commisChef (
    HeadChef * headChef )
```

Construct a new [commisChef](#) object takes a [HeadChef](#) object as a parameter to set the headChef of the [commisChef](#) as the passed in [HeadChef](#).

Parameters

headChef	
--------------------------	--

3.29.3 Member Function Documentation

3.29.3.1 [PrepareDish](#)()

```
void commisChef::PrepareDish (
    Dish * dish ) [virtual]
```

Have a [commisChef](#) start preparing a dish assign a dish to [commisChef](#) who will be the one preparing it.

Parameters

<i>dish</i>	
-------------	--

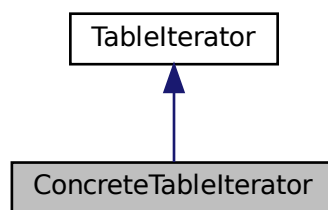
Implements [Chef](#).

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

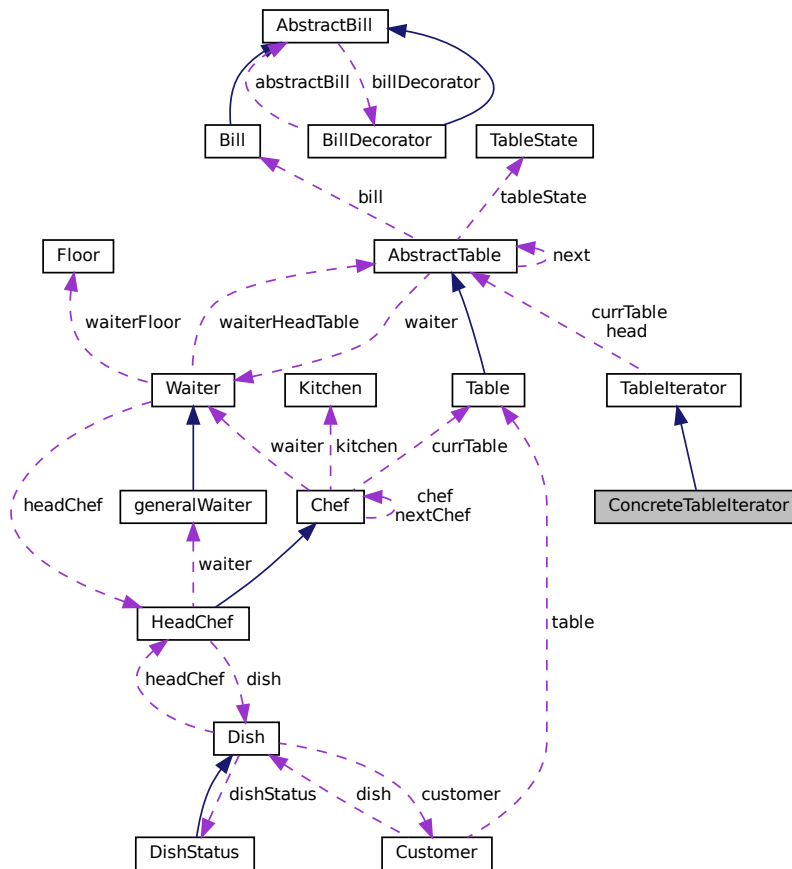
- Chef.h
- Chef.cpp

3.30 ConcreteTableIterator Class Reference

Inheritance diagram for ConcreteTableIterator:



Collaboration diagram for ConcreteTableIterator:



Public Member Functions

- [AbstractTable](#) * **next** ()
- bool **hasNext** ()
- **ConcreteTableIterator** ([AbstractTable](#) *aTable)

Additional Inherited Members

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

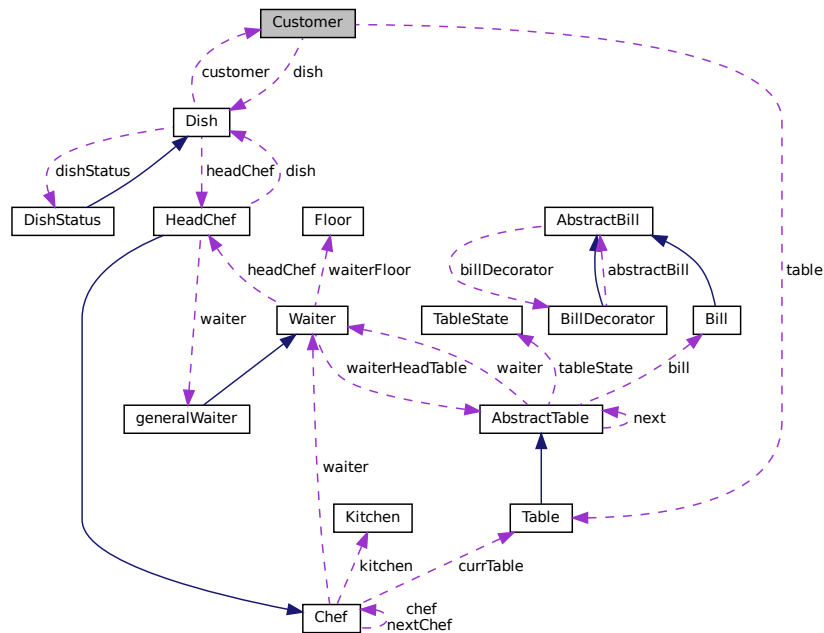
- TableIterator.h
- TableIterator.cpp

3.31 Customer Class Reference

Represents a restaurant customer.

```
#include <Customer.h>
```

Collaboration diagram for Customer:



Public Member Functions

- **Customer** (std::string customerName, int tableNum, **Table** *table)
Construct a new **Customer**:: **Customer** object.
- **~Customer** ()
Destroy the **Customer**:: **Customer** object.
- void **pay** (**PaymentStrategy** *aMethodOfPayment)
Method gets called when customer is ready to pay need to choose method of payment (strategy design pattern)
- void **tip** ()
allows customer to add a tip to the **Bill** customer chooses how much to tip
- void **customer** ()
- void **checkOrder** (**Dish** *order)
sets customer mood according to how long it took for their dish to be prepared
- void **setMood** (std::string cstmrMood)
Sets the customer mood.
- std::string **getMood** ()
Returns the customers mood.
- void **leaveRestaurant** (**Bill** *bill)
Method is called when the customer is ready to leave the restaurant acts as the constructor (conditional)
- void **makeComplaint** (**Manager** *manager)

*gets called when customer is not happy with her waiting period state of the food sends the complaint to the manager
customer can still make a complaint even when the manager is not visiting the table*

- **Bill** * **getBill** ()
Returns bill associated with customer.
- **Tab** * **createTab** ()
Creates a tab for a customer.
- void **setTab** (**Tab** *table)
Sets a tab for a customer.
- **Dish** * **placeOrder** ()
Places an order.
- void **setTableNum** (int table)
Sets table number of customer.
- int **getTableNum** ()
Returns table number in which customer is seated.
- std::string **getName** ()
Returns customer name.
- void **accept** (**Visitor** *visitor)
Accepts visitor to table.
- void **assignCustomerTable** (**AbstractTable** *customerTable)
Assigns customer to table.
- std::string **complimentWaiter** ()
User input to complement waiter.
- void **setReadyToOrderStatus** (bool readyToOrderStatus)
Sets the customer status.
- void **setReadyToLeaveStatus** (bool readyToLeaveStatus)
Sets the customer status.
- void **setReadyToPayStatus** (bool readyToPayStatus)
Sets the customer status.

Public Attributes

- **Dish** * **dish**
- **Table** * **table**

3.31.1 Detailed Description

Represents a restaurant customer.

3.31.2 Constructor & Destructor Documentation

3.31.2.1 Customer()

```
Customer::Customer (
    std::string customerName,
    int tableNum,
    Table * table )
```

Construct a new **Customer:: Customer** object.

Parameters

<i>customerName</i>	
<i>tableNum</i>	
<i>table</i>	

3.31.3 Member Function Documentation

3.31.3.1 accept()

```
void Customer::accept (
    Visitor * visitor )
```

Accepts visitor to table.

Parameters

<i>visitor</i>	
----------------	--

3.31.3.2 assignCustomerTable()

```
void Customer::assignCustomerTable (
    AbstractTable * customerTable )
```

Assigns customer to table.

Parameters

<i>customerTable</i>	
----------------------	--

3.31.3.3 checkOrder()

```
void Customer::checkOrder (
    Dish * order )
```

sets customer mood according to how long it took for their dish to be prepared

Parameters

<i>order</i>	
--------------	--

3.31.3.4 complimentWaiter()

```
std::string Customer::complimentWaiter ( )
```

User input to complement waiter.

Returns

std::string

3.31.3.5 createTab()

```
Tab * Customer::createTab ( )
```

Creates a tab for a customer.

Returns

Tab*

3.31.3.6 getBill()

```
Bill * Customer::getBill ( )
```

Returns bill associated with customer.

Returns

Bill*

3.31.3.7 getMood()

```
std::string Customer::getMood ( )
```

Returns the customers mood.

Returns

std::string

3.31.3.8 getName()

```
std::string Customer::getName ( )
```

Returns customer name.

Returns

std::string

3.31.3.9 getTableNum()

```
int Customer::getTableNum ( )
```

Returns table number in which customer is seated.

Returns

int

3.31.3.10 leaveRestaurant()

```
void Customer::leaveRestaurant (
    Bill * bill )
```

Method is called when the customer is ready to leave the restaurant acts as the constructor (conditional)

Parameters

<i>bill</i>	
-------------	--

3.31.3.11 pay()

```
void Customer::pay (
    PaymentStrategy * aMethodOfPayment )
```

Method gets called when customer is ready to pay need to choose method of payment (strategy design pattern)

Parameters

<i>aMethodOfPayment</i>	
-------------------------	--

3.31.3.12 setMood()

```
void Customer::setMood (
    std::string cstmrMood )
```

Sets the customer mood.

Parameters

<i>cstmrMood</i>	
------------------	--

3.31.3.13 setReadyToLeaveStatus()

```
void Customer::setReadyToLeaveStatus (
    bool readyToLeaveStatus )
```

Sets the customer status.

Parameters

<i>readyToPayStatus</i>	
-------------------------	--

3.31.3.14 setReadyToOrderStatus()

```
void Customer::setReadyToOrderStatus (
    bool readyToOrderStatus )
```

Sets the customer status.

Parameters

<i>readyToPayStatus</i>	
-------------------------	--

3.31.3.15 setReadyToPayStatus()

```
void Customer::setReadyToPayStatus (
    bool readyToPayStatus )
```

Sets the customer status.

Parameters

<i>readyToPayStatus</i>	
-------------------------	--

3.31.3.16 setTab()

```
void Customer::setTab (  
    Tab * tab )
```

Sets a tab for a customer.

Parameters

<i>tab</i>	
------------	--

3.31.3.17 setTableNum()

```
void Customer::setTableNum (  
    int table )
```

Sets table number of customer.

Parameters

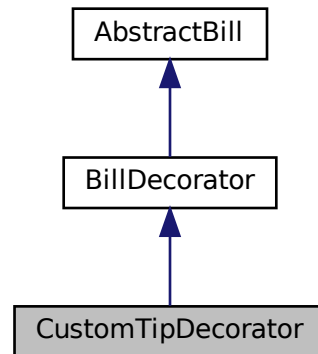
<i>table</i>	
--------------	--

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

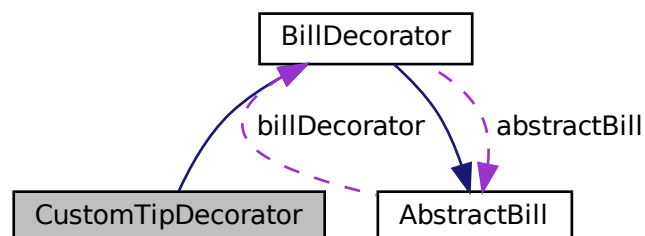
- Customer.h
- Customer.cpp

3.32 CustomTipDecorator Class Reference

Inheritance diagram for CustomTipDecorator:



Collaboration diagram for CustomTipDecorator:



Public Member Functions

- `CustomTipDecorator` (`Bill *bill`, double tip)
- double `getTotalCost` ()

Additional Inherited Members

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

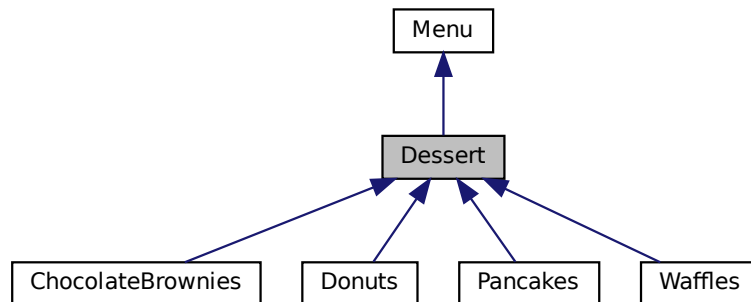
- `Bill.h`
- `Bill.cpp`

3.33 Dessert Class Reference

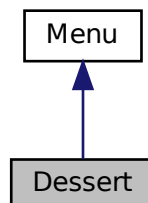
Desserts base class Desserts are waffle, chocolate brownies, donuts and pancakes Desserts will all take 3 seconds to prepare.

```
#include <Menu.h>
```

Inheritance diagram for Dessert:



Collaboration diagram for Dessert:



Additional Inherited Members

3.33.1 Detailed Description

Desserts base class Desserts are waffle, chocolate brownies, donuts and pancakes Desserts will all take 3 seconds to prepare.

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

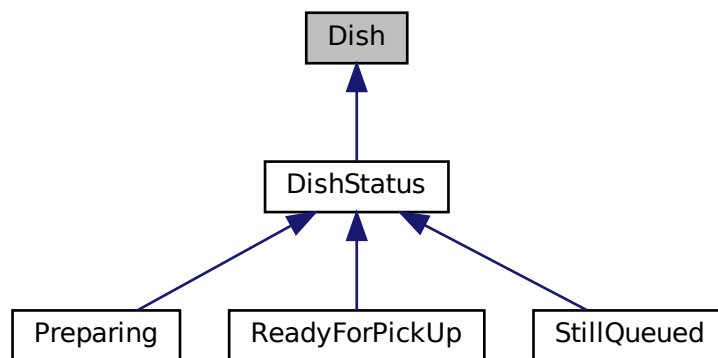
- Menu.h

3.34 Dish Class Reference

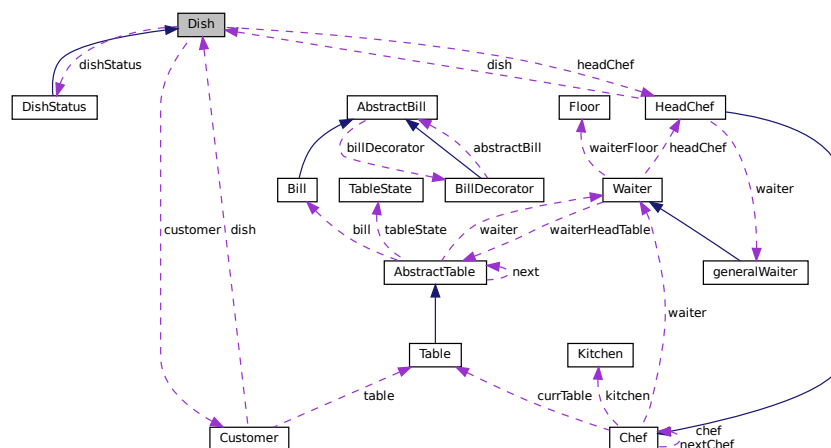
[Dish](#) class definition each [Dish](#) consists of multiple menu objects as well as the customer's name, their table and the state of the dish once it is being prepared.

```
#include <Dish.h>
```

Inheritance diagram for Dish:



Collaboration diagram for Dish:



Public Member Functions

- [Dish](#) ()
Default Constructor for [Dish](#) Class All attributes set to null, empty or 0.
- [Dish](#) (std::string customerName, int customerTable)

Constructor for [Dish](#) class that takes in arguments Corresponding attributes are initialized to passed in arguments remaining attributes are initialized by createDish function call.

- void [createDish](#) ()
Initializes [Dish](#) class's [Menu](#) objects randomly initializes the [Menu](#) attributes by using randomly generated numbers not all [Menu](#) objects may be initialized.
- void [dishState](#) ()
- [DishStatus](#) * [getDishStatus](#) ()
Returns the current state of the dish.
- void [change](#) ()
- void [setDishStatus](#) ([DishStatus](#) *state)
set the status of the dish sets the current status of the dish to the passed in parameter
- std::string [getCustomerName](#) ()
Returns the dish's customer name Returns the name of the customer who ordered the dish.
- int [getCustomerTable](#) ()
Returns the customer's table number Returns the table number of the customer who ordered the dish.

Public Attributes

- [Customer](#) * [customer](#)
- [DishStatus](#) * [dishStatus](#)
- [HeadChef](#) * [headChef](#)

3.34.1 Detailed Description

[Dish](#) class definition each [Dish](#) consists of multiple menu objects as well as the customer's name, their table and the state of the dish once it is being prepared.

3.34.2 Constructor & Destructor Documentation

3.34.2.1 Dish()

```
Dish::Dish (
    std::string customerName,
    int customerTable )
```

Constructor for [Dish](#) class that takes in arguments Corresponding attributes are initialized to passed in arguments remaining attributes are initialized by createDish function call.

Parameters

<i>customerName</i>	
<i>customerTable</i>	

3.34.3 Member Function Documentation

3.34.3.1 `getCustomerName()`

```
std::string Dish::getCustomerName ( )
```

Returns the dish's customer name Returns the name of the customer who ordered the dish.

Returns

std::string

3.34.3.2 `getCustomerTable()`

```
int Dish::getCustomerTable ( )
```

Returns the customer's table number Returns the table number of the customer who ordered the dish.

Returns

int

3.34.3.3 `getDishStatus()`

```
DishStatus * Dish::getDishStatus ( )
```

Returns the current state of the dish.

Returns

DishStatus*

3.34.3.4 `setDishStatus()`

```
void Dish::setDishStatus (
    DishStatus * state )
```

set the status of the dish sets the current status of the dish to the passed in parameter

Parameters

<i>state</i>	
--------------	--

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

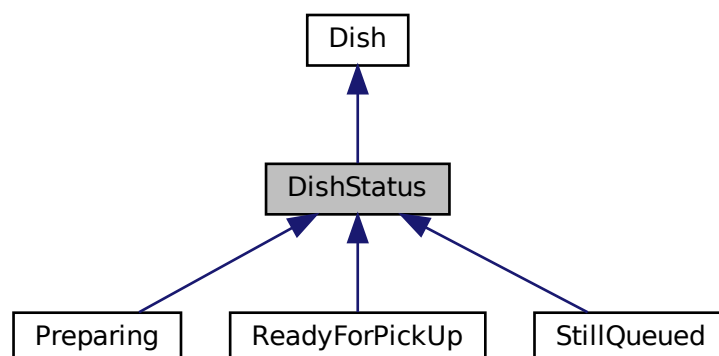
- Dish.h
- Dish.cpp

3.35 DishStatus Class Reference

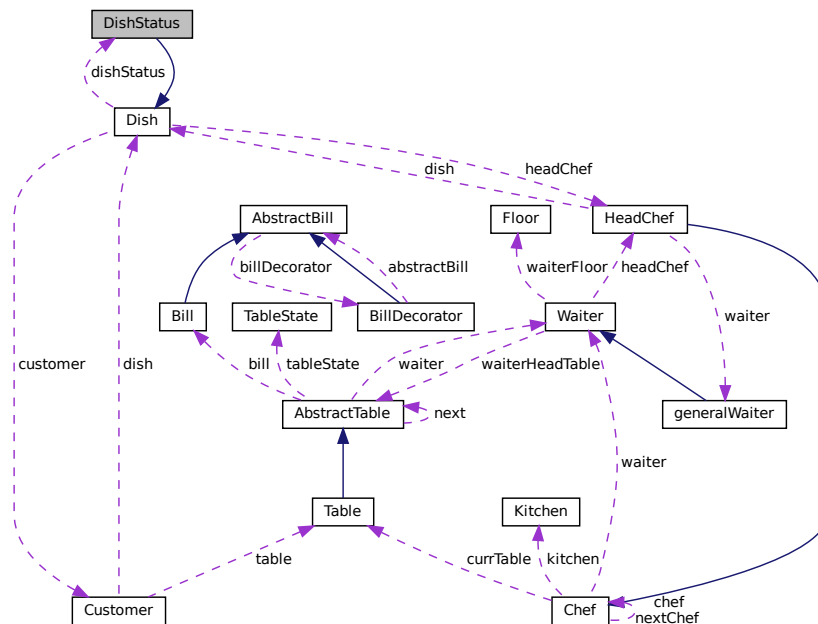
[DishStatus](#) base class, inherits from [Dish](#) There are 3 different [DishStatus](#) classes: [Preparing](#), [ReadyForPickUp](#) and [StillQueued](#) each [DishStatus](#) updates a dish's status to their corresponding class.

```
#include <Dish.h>
```

Inheritance diagram for DishStatus:



Collaboration diagram for DishStatus:



Public Member Functions

- [DishStatus](#) ()
Default constructor for [DishStatus](#) base class.
- virtual void **updateDishStatus** ()=0
- void **DishState** ()
- virtual std::string **getStatus** ()=0

Protected Attributes

- std::string **status**

Additional Inherited Members

3.35.1 Detailed Description

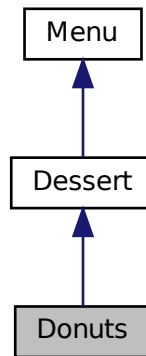
[DishStatus](#) base class, inherits from [Dish](#) There are 3 different [DishStatus](#) classes: [Preparing](#), [ReadyForPickUp](#) and [StillQueued](#) each [DishStatus](#) updates a dish's status to their corresponding class.

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

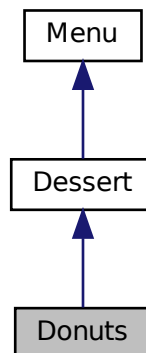
- Dish.h
- Dish.cpp

3.36 Donuts Class Reference

Inheritance diagram for Donuts:



Collaboration diagram for Donuts:



Public Member Functions

- [Donuts](#) ()
Constrcutor for donuts dessert.

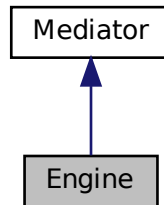
Additional Inherited Members

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

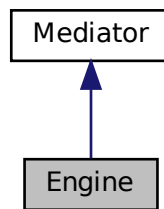
- Menu.h
- Menu.cpp

3.37 Engine Class Reference

Inheritance diagram for Engine:



Collaboration diagram for Engine:



Public Member Functions

- void **update** ()

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

- Engine.h
- Engine.cpp

3.38 Facade Class Reference

acts as main interface for system/interface

```
#include <Facade.h>
```

Public Member Functions

- [Facade](#) ([Floor](#) *f, [Engine](#) *e)
Construct a new [Facade](#) object.
- void [Decrement](#) ()
notify all related timers to decrement
- void [displayMenu](#) ()
show interface for users
- void [createWaiter](#) ()
Create a [Waiter](#) object (template code)
- [Facade](#) * [GetInstance](#) ([Floor](#) *f, [Engine](#) *e)
Get the Instance object or create if it doesn't exist.
- [Facade](#) ()
Construct a new [Facade](#) object.
- void [operation](#) ()
(template code)
- [Facade](#) (std::string)
Construct a new [Facade](#) object (template code)

3.38.1 Detailed Description

acts as main interface for system/interface

3.38.2 Constructor & Destructor Documentation

3.38.2.1 Facade()

```
Facade::Facade (
    Floor * f,
    Engine * e ) [inline]
```

Construct a new [Facade](#) object.

Parameters

<i>f</i>	
<i>e</i>	

3.38.3 Member Function Documentation

3.38.3.1 GetInstance()

```
Facade* Facade::GetInstance (
    Floor * f,
    Engine * e ) [inline]
```

Get the Instance object or create if it doesn't exist.

Parameters

<i>f</i>	
<i>e</i>	

Returns

Facade*

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

- Facade.h
- Facade.cpp

3.39 Floor Class Reference

[Floor](#) is responsible for managing the tables and waiters.

```
#include <Floor.h>
```

Public Member Functions

- [AbstractTable](#) * [constructTable](#) ()
Construct a new [Floor::Floor](#) object.
- [AbstractTable](#) * [destructTable](#) ()
Function to destruct a table.
- void [Decrement](#) ()
Function to decrement the timer.
- [AbstractTable](#) * [getHeadTable](#) ()
- void [constructWaiter](#) (std::string, [HeadChef](#) *hc)
Function to construct a waiter.
- void [printWaiters](#) ()
Function to print the waiters.
- [Tab](#) * [getTab](#) (std::string aName)
Function to get the tab.
- [Manager](#) * [getManager](#) ()
Function to get the manager.
- void [setManager](#) ([Manager](#) *aManager)
Function to set the manager.
- void [getManagerComplaints](#) ()
- void [mergeTables](#) (int table1, int table2)
Function to merge tables.
- void [splitTables](#) ([TableGroup](#) *table)
Function to split tables This function will only split tablegroup and not normal tables, It takes a tablegroup and splits it into its individual tables.

3.39.1 Detailed Description

[Floor](#) is responsible for managing the tables and waiters.

3.39.2 Member Function Documentation

3.39.2.1 splitTables()

```
void Floor::splitTables (
    TableGroup * table )
```

Function to split tables This function will only split tablegroup and not normal tables, It takes a tablegroup and splits it into its individual tables.

Parameters

<i>table</i>	
--------------	--

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

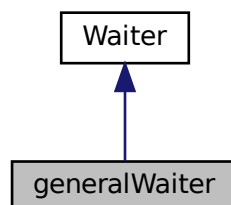
- Floor.h
- Floor.cpp

3.40 generalWaiter Class Reference

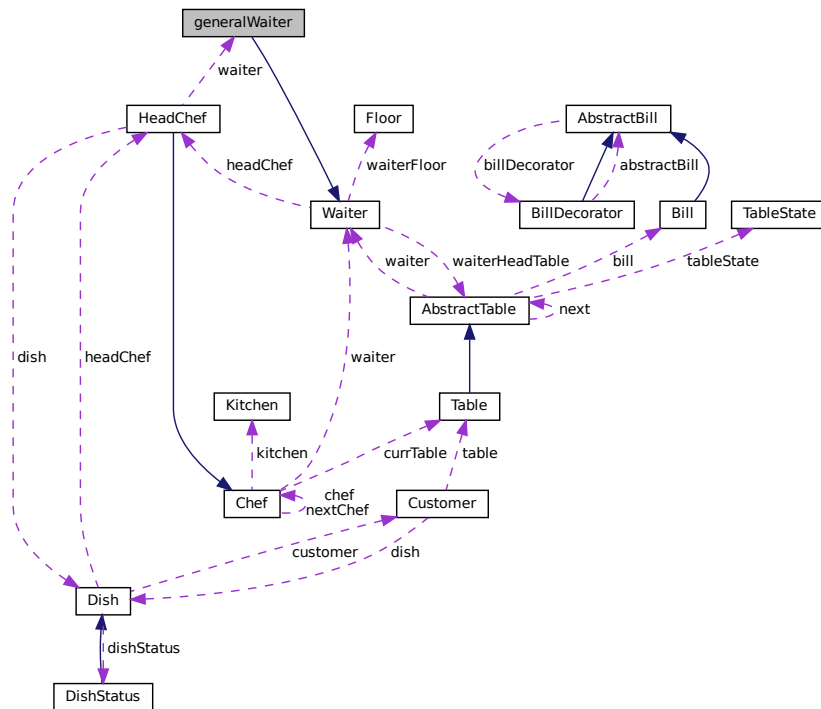
GeneralWaiter is responsible for serving customers.

```
#include <Waiter.h>
```

Inheritance diagram for generalWaiter:



Collaboration diagram for generalWaiter:



Public Member Functions

- **generalWaiter** (std::string basicString, [HeadChef](#) *hc, [Floor](#) *pFloor)
- void **getAllocatedTable** ([Table](#) *table)
- void **performTask** ()
- virtual void **visitTable** ([AbstractTable](#) *table)
- void **addToTab** (std::string name, double amount)
Adds the total of the dish to the customers tab.
- void **payTab** (std::string name, double amount)
Deduct from the total of a customers tab.
- [Tab](#) * **getTab** (std::string name)
Returns a reference to a customers tab.
- void **decrementTimer** ()
- void **receiveCompliment** (const std::string &compliment)
Constructor for the [MaitreD](#) class.

Additional Inherited Members

3.40.1 Detailed Description

GeneralWaiter is responsible for serving customers.

3.40.2 Member Function Documentation

3.40.2.1 addToTab()

```
void generalWaiter::addToTab (
    std::string customerName,
    double amount )
```

Adds the total of the dish to the customers tab.

Parameters

<i>customerName</i>	
<i>amount</i>	

3.40.2.2 getTab()

```
Tab * generalWaiter::getTab (
    std::string customerName )
```

Returns a reference to a customers tab.

Parameters

<i>customerName</i>	
---------------------	--

Returns

Tab*

3.40.2.3 payTab()

```
void generalWaiter::payTab (
    std::string customerName,
    double amount )
```

Deduct from the total of a customers tab.

Parameters

<i>customerName</i>	
<i>amount</i>	

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

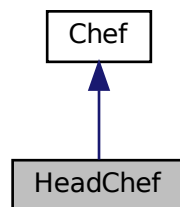
- Waiter.h
- Waiter.cpp

3.41 HeadChef Class Reference

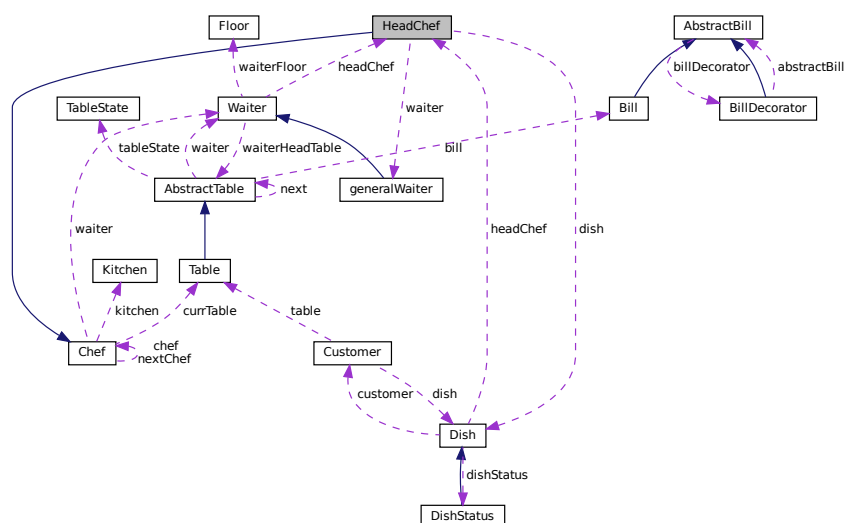
[HeadChef](#) class [HeadChef](#) delegates making of dishes to [commisChef](#) class [HeadChef](#) can also add and remove commisChefs as necessary commisChefs given dishes are moved from a freeChef to a busyChefs queue once a dish is complete [commisChef](#) is moved back to freeChefs.

```
#include <Chef.h>
```

Inheritance diagram for HeadChef:



Collaboration diagram for HeadChef:



Public Member Functions

- [HeadChef](#) ()
Construct a new [HeadChef](#) object Set role to "HeadChef" nextChef and currTable set to NULL.
- void [AddDish](#) ([Dish](#) *d)
add a dish to the dishQueue enqueues a dish object to [HeadChef](#)'s dishQueue
- void [PrepareDish](#) ([Dish](#) *dish)
gives a dish to a [commisChef](#) to prepare dequeues a dish object from [HeadChef](#)'s dishQueue and assigns a [commisChef](#) to start preparing it
- void [addChef](#) ()
add a chef to the kitchen adds a [commisChef](#) object to [HeadChef](#)'s freeChefs [commisChef](#) queue
- void [removeChef](#) ()
remove a chef from the kitchen remove a [commisChef](#) object from [HeadChef](#)'s freeChefs [commisChef](#) queue
- void [Notify](#) ([generalWaiter](#) *waiter)
notify a waiter an order is ready to be delivered call the passed in [generalWaiter](#) object's deliverOrder object to deliver a finished order to the appropriate table
- void [Attach](#) ([generalWaiter](#) *waiter)
add a new [generalWaiter](#) attach a new [generalWaiter](#) object to the [HeadChef](#)'s [generalWaiter](#) queue
- void [Detach](#) ()
remove a [generalWaiter](#) remove a [generalWaiter](#) object from [HeadChef](#)'s [generalWaiter](#) queue

Public Attributes

- [generalWaiter](#) * waiter
- [Dish](#) * dish

Additional Inherited Members

3.41.1 Detailed Description

[HeadChef](#) class [HeadChef](#) delegates making of dishes to [commisChef](#) class [HeadChef](#) can also add and remove [commisChefs](#) as necessary [commisChefs](#) given dishes are moved from a freeChef to a busyChefs queue once a dish is complete [commisChef](#) is moved back to freeChefs.

3.41.2 Member Function Documentation

3.41.2.1 AddDish()

```
void HeadChef::AddDish (
    Dish * d )
```

add a dish to the dishQueue enqueues a dish object to [HeadChef](#)'s dishQueue

Parameters

d	
-------------------	--

3.41.2.2 Attach()

```
void HeadChef::Attach (
    generalWaiter * waiter )
```

add a new [generalWaiter](#) attach a new [generalWaiter](#) object to the [HeadChef](#)'s [generalWaiter](#) queue

Parameters

<i>waiter</i>	
---------------	--

3.41.2.3 Notify()

```
void HeadChef::Notify (
    generalWaiter * waiter )
```

notify a waiter an order is ready to be delivered call the passed in [generalWaiter](#) object's deliverOrder object to deliver a finished order to the appropriate table

Parameters

<i>waiter</i>	
---------------	--

3.41.2.4 PrepareDish()

```
void HeadChef::PrepareDish (
    Dish * dish ) [virtual]
```

gives a dish to a [commisChef](#) to prepare dequeues a dish object from [HeadChef](#)'s dishQueue and assigns a [commisChef](#) to start preparing it

Parameters

<i>dish</i>	
-------------	--

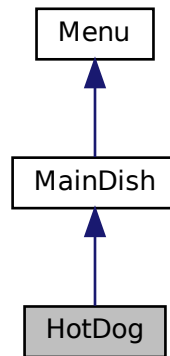
Implements [Chef](#).

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

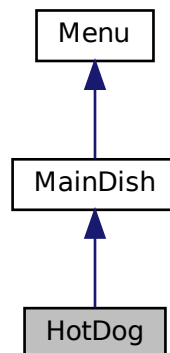
- Chef.h
- Chef.cpp

3.42 HotDog Class Reference

Inheritance diagram for HotDog:



Collaboration diagram for HotDog:



Public Member Functions

- [HotDog](#) ()
Constrcutor for hot dog main dish.

Additional Inherited Members

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

- Menu.h
- Menu.cpp

3.43 Kitchen Class Reference

Public Member Functions

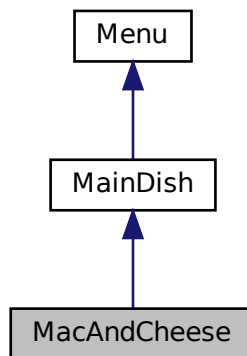
- void **receiveOrder** ([Waiter](#) *aWaiter)

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

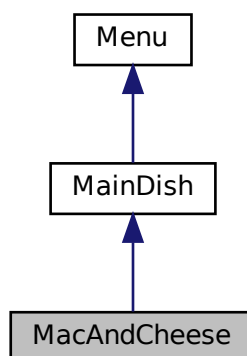
- Kitchen.h
- Kitchen.cpp

3.44 MacAndCheese Class Reference

Inheritance diagram for MacAndCheese:



Collaboration diagram for MacAndCheese:



Public Member Functions

- [MacAndCheese](#) ()
Constrcutor for mac and cheese main dish.

Additional Inherited Members

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

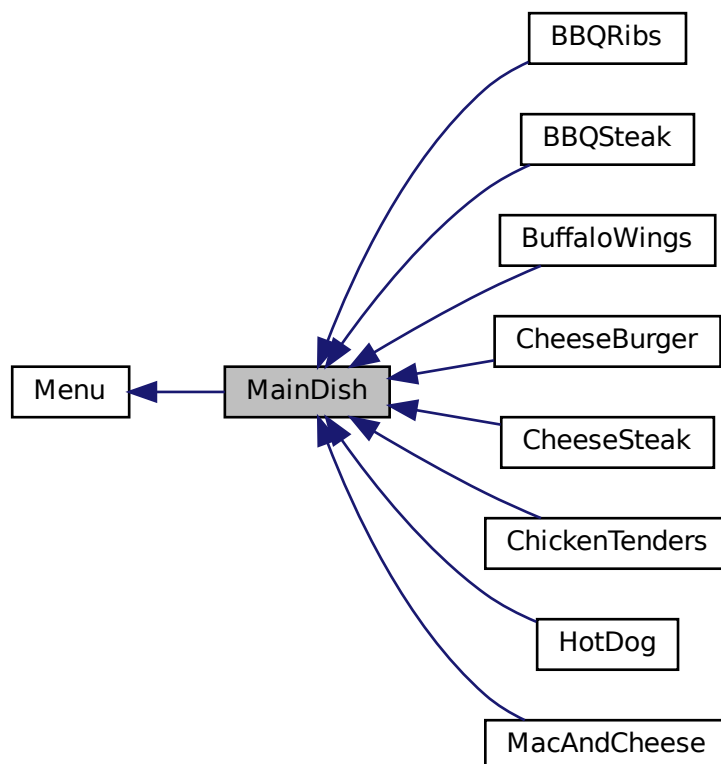
- Menu.h
- Menu.cpp

3.45 MainDish Class Reference

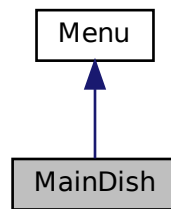
Main dishes base class Main dishes are cheeseburgers, hot dogs, mac and cheese, bbq ribs, chicken tenders, cheesesteak, bbq steak and buffalo wings Main dishes will all take 10 seconds to prepare.

```
#include <Menu.h>
```

Inheritance diagram for MainDish:



Collaboration diagram for MainDish:



Additional Inherited Members

3.45.1 Detailed Description

Main dishes base class Main dishes are cheeseburgers, hot dogs, mac and cheese, bbq ribs, chicken tenders, cheesesteak, bbq steak and buffalo wings Main dishes will all take 10 seconds to prepare.

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

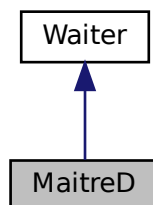
- Menu.h

3.46 MaitreD Class Reference

[MaitreD](#) is responsible for allocating tables to customers.

```
#include <Waiter.h>
```

Inheritance diagram for MaitreD:



```
classDiagram
    class AbstractBill {
        +abstractBill()
        +billDecorator()
    }
    class BillDecorator {
    }
    class Bill {
    }
    class TableState {
    }
    class AbstractTable {
    }
    class Table {
    }
    class Waiter {
    }
    class Floor {
    }
    class MaitreD {
    }
    class generalWaiter {
    }
    class Kitchen {
    }
    class Chef {
    }
    class HeadChef {
    }
    class Dish {
    }
    class DishStatus {
    }
    class Customer {
    }

    AbstractBill <|-- BillDecorator
    AbstractBill <|-- Bill
    AbstractTable <|-- Table
    Waiter <|-- MaitreD
    Waiter <|-- generalWaiter
    Chef <|-- HeadChef
    Dish <|-- DishStatus

    AbstractBill ..> BillDecorator : abstractBill / billDecorator
    AbstractTable ..> Table : table
    AbstractTable ..> Waiter : waiter
    AbstractTable ..> AbstractTable : next
    AbstractTable ..> TableState : tableState
    AbstractTable ..> Chef : currTable
    AbstractTable ..> Customer : customer
    Waiter ..> Floor : waiterFloor
    Waiter ..> MaitreD : MaitreD
    Waiter ..> generalWaiter : generalWaiter
    Waiter ..> Kitchen : kitchen
    Waiter ..> HeadChef : headChef
    Waiter ..> Chef : chef
    Waiter ..> Waiter : waiterHeadTable
    Chef ..> HeadChef : headChef
    Chef ..> Chef : nextChef
    Chef ..> Kitchen : kitchen
    Chef ..> Dish : dish
    HeadChef ..> Dish : dish
    HeadChef ..> HeadChef : headChef
    Dish ..> DishStatus : dishStatus
    Dish ..> Dish : dish
    Dish ..> Customer : customer
```

- void `performTask()`
Constructor for the `MaitreD` class.
- void `allocateTable(std::vector< Customer * >)`
Allocate an available table to 1 or more customers.
- void `mergeTables(int table1, int table2)`
Merge 2 tables together.
- void `splitTables(TableGroup *tableGroup)`
Split a table into 2.

3.46.1 Detailed Description

3.46.2 Member Function Documentation

Allocate an available table to 1 or more customers.

Parameters

<i>customers</i>	
------------------	--

3.46.2.2 mergeTables()

```
void MaitreD::mergeTables (
    int table1,
    int table2 )
```

Merge 2 tables together.

Parameters

<i>table1</i>	
<i>table2</i>	

3.46.2.3 splitTables()

```
void MaitreD::splitTables (
    TableGroup * table )
```

Split a table into 2.

Parameters

<i>table</i>	
--------------	--

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

- Waiter.h
- Waiter.cpp

3.47 Manager Class Reference

Public Member Functions

- virtual void [visitTable](#) ([Table](#) *table)
Manager visits the table and checks the customer's mood.
- std::vector< std::string > [getComplaints](#) ()
Returns the complaints made by customers.
- void [handleComplaint](#) (const std::string &complaint)
Handles the complaint made by the customer.

3.47.1 Member Function Documentation

3.47.1.1 getComplaints()

```
std::vector< std::string > Manager::getComplaints ( )
```

Returns the complaints made by customers.

Returns

`std::vector<std::string>`

3.47.1.2 handleComplaint()

```
void Manager::handleComplaint (
    const std::string & complaint )
```

Handles the complaint made by the customer.

Parameters

<i>complaint</i>	
------------------	--

3.47.1.3 visitTable()

```
void Manager::visitTable (
    Table * table ) [virtual]
```

[Manager](#) visits the table and checks the customer's mood.

Parameters

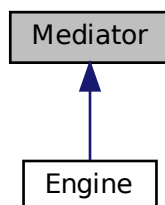
<i>table</i>	
--------------	--

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

- Manager.h
- Manager.cpp

3.48 Mediator Class Reference

Inheritance diagram for Mediator:



Public Member Functions

- void **setFloor** ([Floor](#) *f)
- void **notify** ()

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

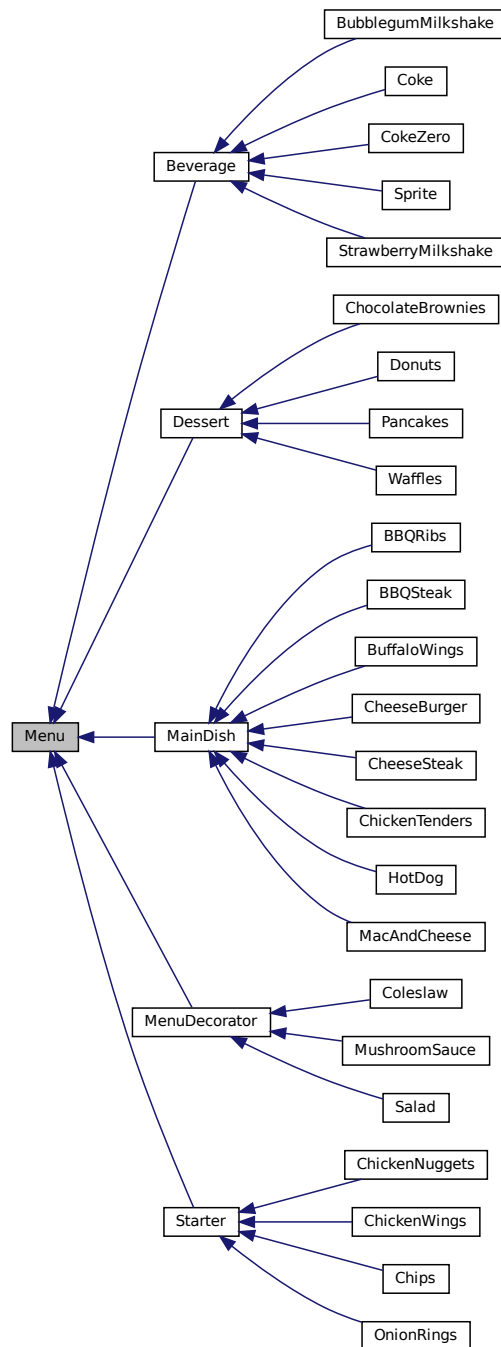
- Engine.h
- Engine.cpp

3.49 Menu Class Reference

[Menu](#) base class.

```
#include <Menu.h>
```

Inheritance diagram for Menu:



Public Member Functions

- `std::string getDescription ()`
Construct a new `Menu::Menu` object.
- `double getPrice ()`
Construct a new `Menu::Menu` object.
- `int getTimeToPrepare ()`
Returns preparation time for the menu object.

Protected Attributes

- `std::string` **description**
- `int` **timeToPrepare**
- `double` **price**

3.49.1 Detailed Description

[Menu](#) base class.

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

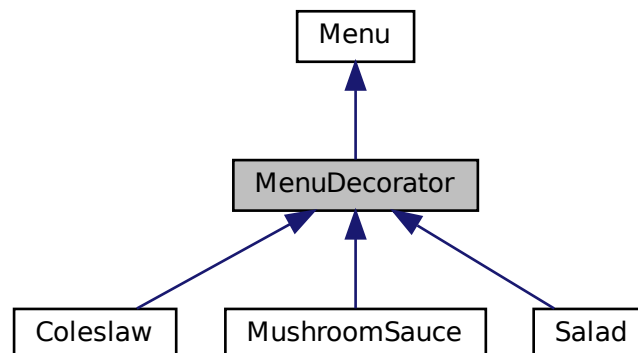
- `Menu.h`
- `Menu.cpp`

3.50 MenuDecorator Class Reference

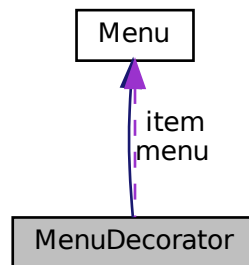
[MenuDecorator](#) class.

```
#include <Menu.h>
```

Inheritance diagram for MenuDecorator:



Collaboration diagram for MenuDecorator:



Public Member Functions

- `std::string` **getDescription** ()
- `double` **getPrice** ()
- [MenuDecorator](#) ([Menu](#) *baseItem)
Sets the base menu item that the customer wants to add custom additions to.
- `int` **getTimeToPrepare** ()

Public Attributes

- [Menu](#) * menu

Protected Attributes

- [Menu](#) * item

3.50.1 Detailed Description

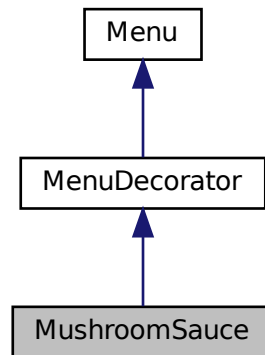
[MenuDecorator](#) class.

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

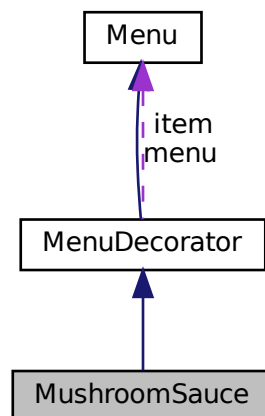
- Menu.h
- Menu.cpp

3.51 MushroomSauce Class Reference

Inheritance diagram for MushroomSauce:



Collaboration diagram for MushroomSauce:



Public Member Functions

- `std::string` **getDescription** ()
- `double` **getPrice** ()
- [MushroomSauce](#) ([Menu](#) *baseItem, `std::string` description, `double` price, `int` timeToprepare)
Constrcutor for mushroom sauce custom addition.
- `int` **getTimeToPrepare** ()

Additional Inherited Members

3.51.1 Constructor & Destructor Documentation

3.51.1.1 MushroomSauce()

```
MushroomSauce::MushroomSauce (
    Menu * baseItem,
    std::string description,
    double price,
    int timeToprepare )
```

Constructor for mushroom sauce custom addition.

Parameters

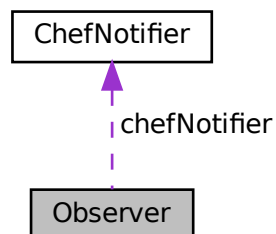
<i>baseItem</i>	
<i>description</i>	
<i>price</i>	
<i>timeToprepare</i>	

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

- Menu.h
- Menu.cpp

3.52 Observer Class Reference

Collaboration diagram for Observer:



Public Member Functions

- virtual void **DeliverOrder** ()=0

Public Attributes

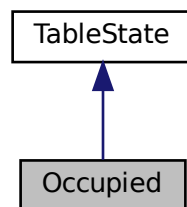
- [ChefNotifier](#) * **chefNotifier**

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

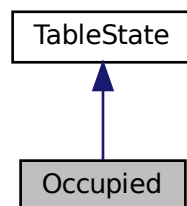
- Chef.h

3.53 Occupied Class Reference

Inheritance diagram for Occupied:



Collaboration diagram for Occupied:



Public Member Functions

- void [handleState](#) ([AbstractTable](#) *table)
[Occupied](#) state handle, if there are no customers change from occupied to unoccupied.
- std::string [getState](#) ()
Returns string which specifies table state.

3.53.1 Member Function Documentation

3.53.1.1 `getState()`

```
std::string Occupied::getState ( ) [virtual]
```

Returns string which specifies table state.

Returns

std::string

Implements [TableState](#).

3.53.1.2 `handleState()`

```
void Occupied::handleState (
    AbstractTable * table ) [virtual]
```

[Occupied](#) state handle, if there are no customers change from occupied to unoccupied.

Parameters

<i>table</i>	
--------------	--

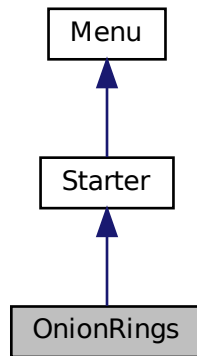
Implements [TableState](#).

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

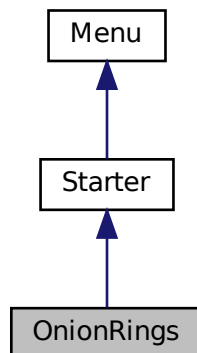
- Table.h
- Table.cpp

3.54 OnionRings Class Reference

Inheritance diagram for OnionRings:



Collaboration diagram for OnionRings:



Public Member Functions

- [OnionRings](#) ()
Constrcutor for onion rings starter.

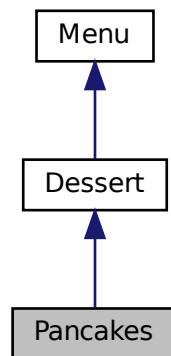
Additional Inherited Members

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

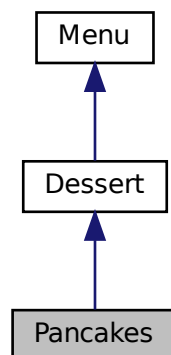
- Menu.h
- Menu.cpp

3.55 Pancakes Class Reference

Inheritance diagram for Pancakes:



Collaboration diagram for Pancakes:



Public Member Functions

- [Pancakes](#) ()
Constrcutor for pancakes dessert.

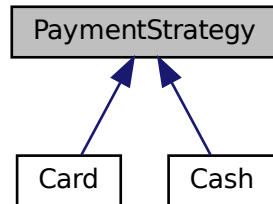
Additional Inherited Members

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

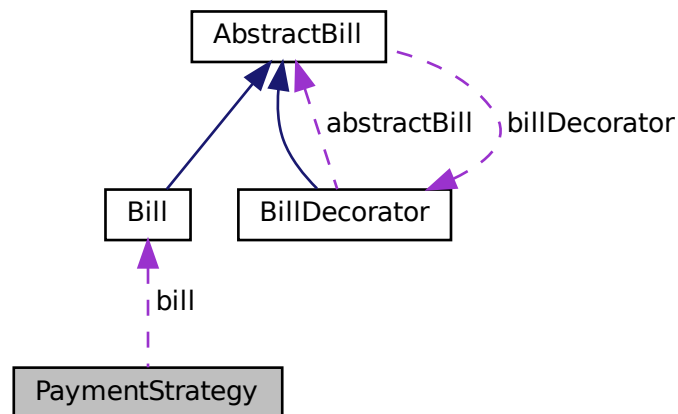
- Menu.h
- Menu.cpp

3.56 PaymentStrategy Class Reference

Inheritance diagram for PaymentStrategy:



Collaboration diagram for PaymentStrategy:



Public Member Functions

- void `paymentMethod` ()

Public Attributes

- `Bill` * `bill`

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

- `Bill.h`
- `Bill.cpp`

Public Member Functions

- [Preparing](#) ()
Default construct for [DishStatus Preparing](#) class sets the status attribute accordingly.
- void [updateDishStatus](#) ()
Update the dish's current status Set the dish's status to the current [DishStatus](#) class.
- std::string [getStatus](#) ()
Returns the current status of the dish.

Additional Inherited Members

3.57.1 Detailed Description

[DishStatus Preparing](#) class used to indicate when a dish is still being prepared.

3.57.2 Member Function Documentation

3.57.2.1 [getStatus\(\)](#)

```
std::string Preparing::getStatus ( ) [virtual]
```

Returns the current status of the dish.

Returns

std::string

Implements [DishStatus](#).

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

- Dish.h
- Dish.cpp

Public Member Functions

- [ReadyForPickUp](#) ()
Default construct for [DishStatus ReadyForPickUp](#) class sets the status attribute accordingly.
- void [updateDishStatus](#) ()
Update the dish's current status Set the dish's status to the current [DishStatus](#) class.
- std::string [getStatus](#) ()
Returns the current status of the dish.

Additional Inherited Members

3.58.1 Detailed Description

[DishStatus ReadyForPickUp](#) class used to indicate when a dish is ready to be picked up.

3.58.2 Member Function Documentation

3.58.2.1 [getStatus\(\)](#)

```
std::string ReadyForPickUp::getStatus ( ) [virtual]
```

Returns the current status of the dish.

Returns

std::string

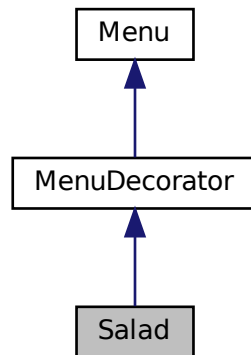
Implements [DishStatus](#).

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

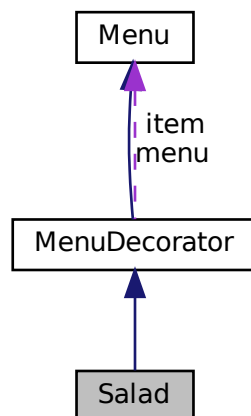
- Dish.h
- Dish.cpp

3.59 Salad Class Reference

Inheritance diagram for Salad:



Collaboration diagram for Salad:



Public Member Functions

- `std::string` **getDescription** ()
- `double` **getPrice** ()
- [Salad](#) ([Menu](#) *baseItem, `std::string` description, `double` price, `int` timeToprepare)
Constrcutor for salad custom addition.
- `int` **getTimeToPrepare** ()

Additional Inherited Members

3.59.1 Constructor & Destructor Documentation

3.59.1.1 Salad()

```
Salad::Salad (
    Menu * baseItem,
    std::string description,
    double price,
    int timeToprepare )
```

Constructor for salad custom addition.

Parameters

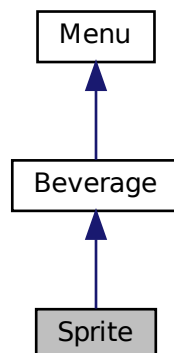
<i>baseItem</i>	
<i>description</i>	
<i>price</i>	
<i>timeToprepare</i>	

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

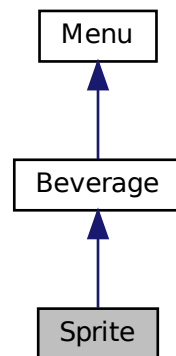
- Menu.h
- Menu.cpp

3.60 Sprite Class Reference

Inheritance diagram for Sprite:



Collaboration diagram for Sprite:



Public Member Functions

- [Sprite](#) ()
Constrcutor for [Sprite](#) beverage.

Additional Inherited Members

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

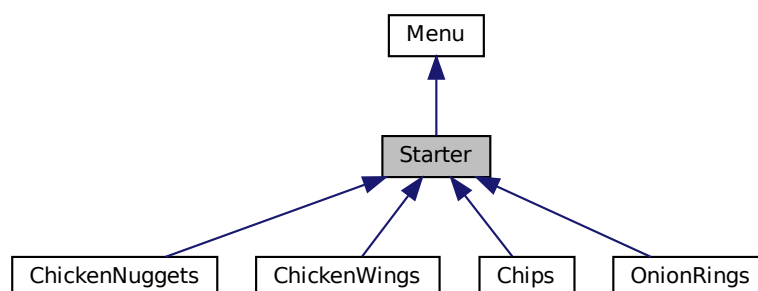
- Menu.h
- Menu.cpp

3.61 Starter Class Reference

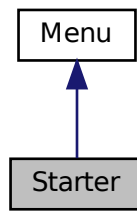
Starters base class Starters are onion rings, chicken wings and chicken nuggets Starters will all take 5 seconds to prepare.

```
#include <Menu.h>
```

Inheritance diagram for Starter:



Collaboration diagram for Starter:



Additional Inherited Members

3.61.1 Detailed Description

Starters base class Starters are onion rings, chicken wings and chicken nuggets Starters will all take 5 seconds to prepare.

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

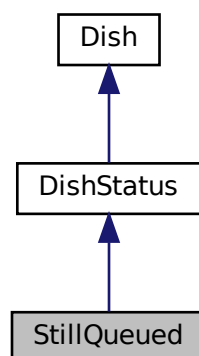
- Menu.h

3.62 StillQueued Class Reference

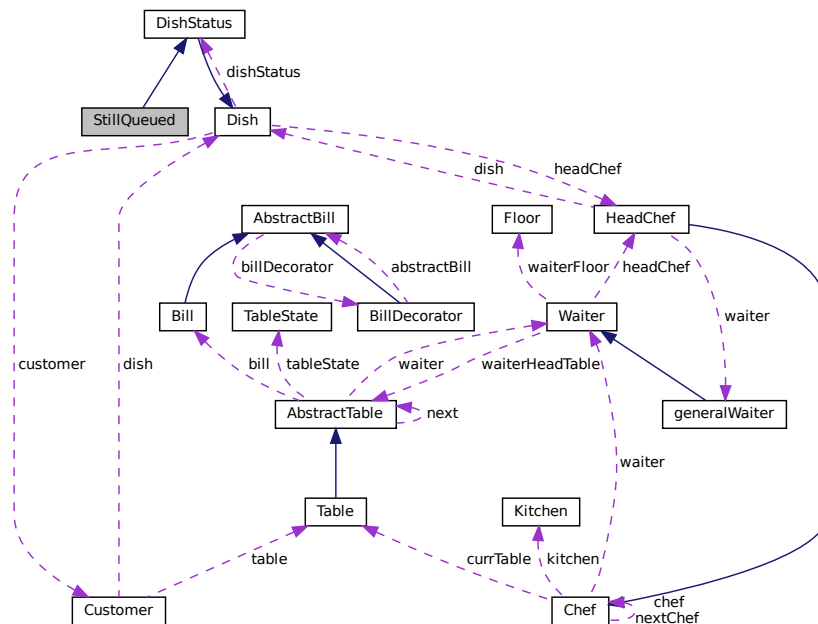
[DishStatus](#) [StillQueued](#) class used to indicate when a dish has yet to begin preparation.

```
#include <Dish.h>
```

Inheritance diagram for StillQueued:



Collaboration diagram for StillQueued:



Public Member Functions

- [StillQueued](#) ()
Default construct for [DishStatus StillQueued](#) class sets the status attribute accordingly.
- void [updateDishStatus](#) ()
Update the dish's current status Set the dish's status to the current [DishStatus](#) class.
- std::string [getStatus](#) ()
Returns the current status of the dish.

Additional Inherited Members

3.62.1 Detailed Description

[DishStatus StillQueued](#) class used to indicate when a dish has yet to begin preparation.

3.62.2 Member Function Documentation

3.62.2.1 getStatus()

```
std::string StillQueued::getStatus ( ) [virtual]
```

Returns the current status of the dish.

Returns

std::string

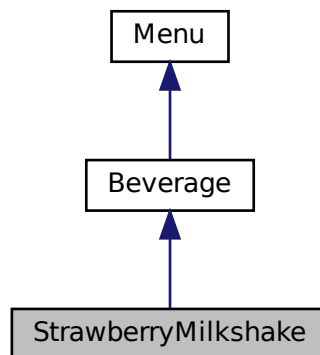
Implements [DishStatus](#).

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

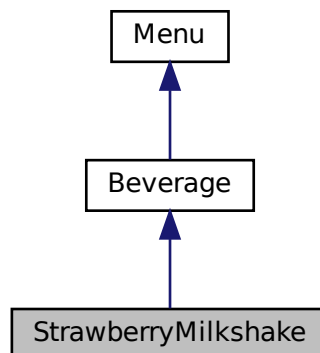
- Dish.h
- Dish.cpp

3.63 StrawberryMilkshake Class Reference

Inheritance diagram for StrawberryMilkshake:



Collaboration diagram for StrawberryMilkshake:



Public Member Functions

- [StrawberryMilkshake](#) ()
Constrcutor for Strawberry milkshake beverage.

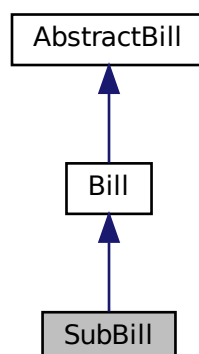
Additional Inherited Members

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

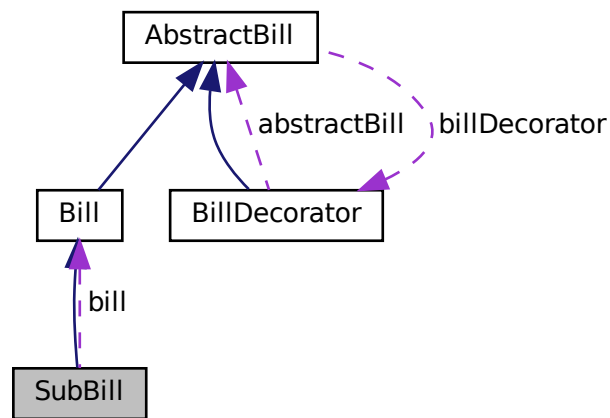
- Menu.h
- Menu.cpp

3.64 SubBill Class Reference

Inheritance diagram for SubBill:



Collaboration diagram for SubBill:



Public Member Functions

- void **paymentMethod** ()
- double **getTotalCost** ()
- void **addItem** (SubBill item)
- void **getSubBill** (std::string customerName)

Public Attributes

- **Bill * bill**

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

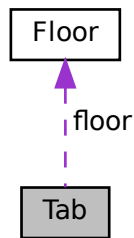
- Bill.h
- Bill.cpp

3.65 Tab Class Reference

The **Tab** class holds the running tab of a restaurant customer.

```
#include <Tab.h>
```


Collaboration diagram for Tab:



Public Member Functions

- **Tab** (std::string customerName)
- std::string [getName](#) ()
Returns the customer name associated with this tab.
- double [getTab](#) ()
Returns the running total for this tab.
- void [addToTab](#) (double)
Adds valueToAdd to the current tab.
- void [subtractFromTab](#) (double)
Deducts valueToSubtract from current tab.

Public Attributes

- [Floor](#) * **floor**

3.65.1 Detailed Description

The [Tab](#) class holds the running tab of a restaurant customer.

3.65.2 Member Function Documentation

3.65.2.1 addToTab()

```
void Tab::addToTab (  
    double valueToAdd )
```

Adds valueToAdd to the current tab.

Parameters

<i>valueToAdd</i>	
-------------------	--

3.65.2.2 getName()

```
std::string Tab::getName ( )
```

Returns the customer name associated with this tab.

Returns

std::string

3.65.2.3 getTab()

```
double Tab::getTab ( )
```

Returns the running total for this tab.

Returns

double

3.65.2.4 subtractFromTab()

```
void Tab::subtractFromTab (
    double valueToSubtract )
```

Deducts *valueToSubtract* from current tab.

Parameters

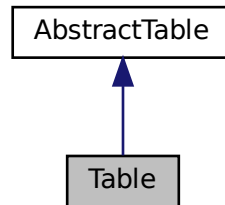
<i>valueToSubtract</i>	
------------------------	--

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

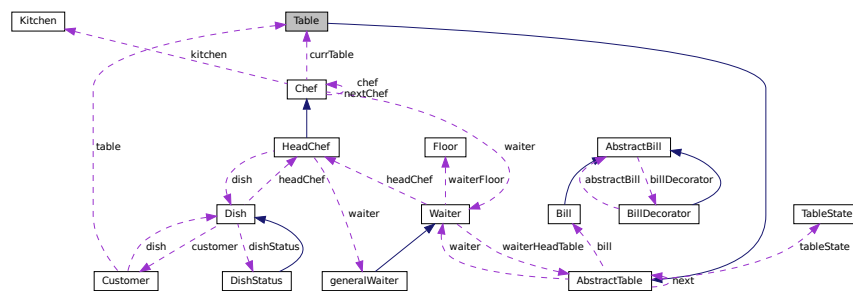
- Tab.h
- Tab.cpp

3.66 Table Class Reference

Inheritance diagram for Table:



Collaboration diagram for Table:



Public Member Functions

- **Table** (int numberOfSeats)
- **AbstractTable** * **operator+** (**Table** *table)
 - operator overload to add 2 tables and return a table group
- **AbstractTable** * **operator+** (**TableGroup** *tableGroup)
 - operator overload to add a table to a table group
- void **acceptVisitor** (**Visitor** *visitor)
 - Function used in visitor design pattern to accept a waiter to the table.
- **AbstractTable** * **clone** ()
 - Returns a copy of the table.
- std::vector< **Customer** * > **getCustomers** ()
 - Returns a vector of customers seated at table.
- void **setState** (**TableState** *tableState)
 - Sets state for current table.
- **TableState** * **getState** ()
 - gets state
- void **handleState** ()

- State handler for table class.*
 - [Bill](#) * [getBill](#) ([Customer](#) *customer)
Returns bill for the table.
 - void [setWaiter](#) ([Waiter](#) *waiter)
Sets waiter for current table.
 - [Waiter](#) * [getWaiter](#) ()
Gets waiter for current table.
 - void [getOrders](#) ()

Additional Inherited Members

3.66.1 Member Function Documentation

3.66.1.1 [acceptVisitor\(\)](#)

```
void Table::acceptVisitor (  
    Visitor * visitor ) [virtual]
```

Function used in visitor design pattern to accept a waiter to the table.

Parameters

visitor	
-------------------------	--

Implements [AbstractTable](#).

3.66.1.2 [clone\(\)](#)

```
AbstractTable * Table::clone ( ) [virtual]
```

Returns a copy of the table.

Returns

[AbstractTable](#)*

Implements [AbstractTable](#).

3.66.1.3 [getBill\(\)](#)

```
Bill * Table::getBill (  
    Customer * customer )
```

Returns bill for the table.

Parameters

<i>customer</i>	
-----------------	--

Returns

Bill*

3.66.1.4 getCustomers()

```
std::vector< Customer * > Table::getCustomers ( )
```

Returns a vector of customers seated at table.

Returns

std::vector<Customer *>

3.66.1.5 getState()

```
TableState * Table::getState ( )
```

gets state

Returns

TableState*

3.66.1.6 getWaiter()

```
Waiter * Table::getWaiter ( )
```

Gets waiter for current table.

Returns

Waiter*

3.66.1.7 operator+() [1/2]

```
AbstractTable * Table::operator+ (
    Table * table ) [virtual]
```

- operator overload to add 2 tables and return a table group

Parameters

<i>table</i>	
--------------	--

Returns

AbstractTable*

Implements [AbstractTable](#).

3.66.1.8 operator+() [2/2]

```
AbstractTable * Table::operator+ (
    TableGroup * tableGroup ) [virtual]
```

operator overload to add a table to a table group

Parameters

<i>tableGroup</i>	
-------------------	--

Returns

AbstractTable*

Implements [AbstractTable](#).

3.66.1.9 setState()

```
void Table::setState (
    TableState * state )
```

Sets state for current table.

Parameters

<i>state</i>	
--------------	--

3.66.1.10 setWaiter()

```
void Table::setWaiter (
    Waiter * waiter )
```

Sets waiter for current table.

Parameters

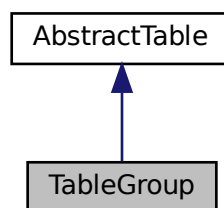
<i>waiter</i>	
---------------	--

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

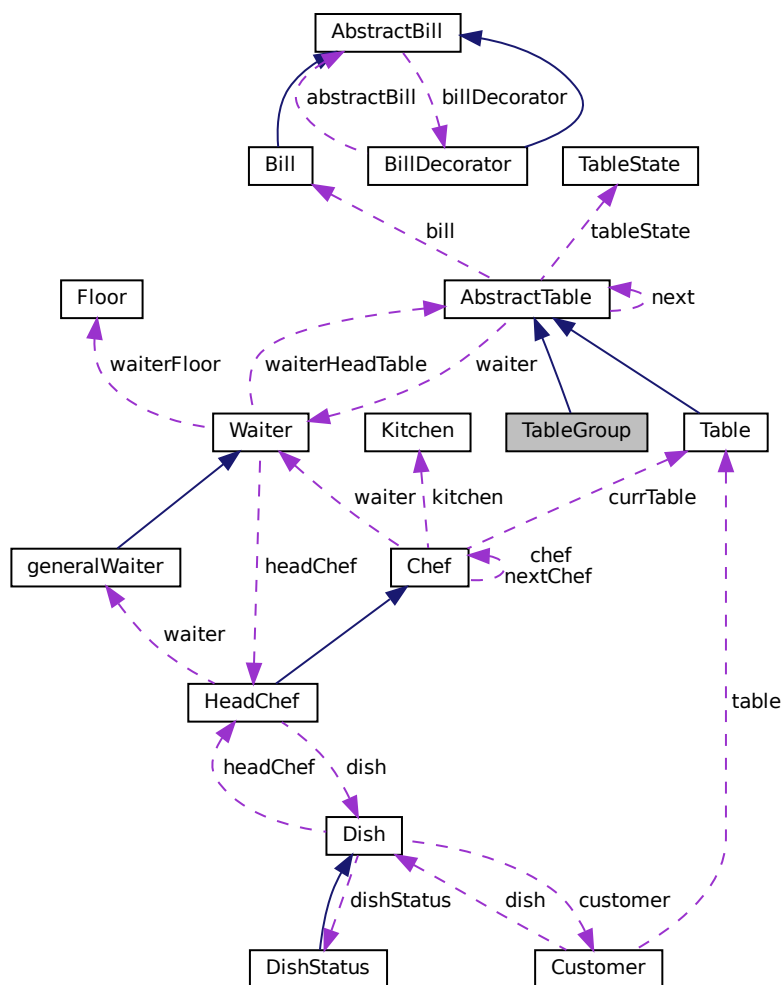
- Table.h
- Table.cpp

3.67 TableGroup Class Reference

Inheritance diagram for TableGroup:



Collaboration diagram for TableGroup:



Public Member Functions

- **TableGroup** (int numberOfSeats=0)
- void **addTable** (**AbstractTable** *aTable)
Adds a table to tables vector.
- void **acceptVisitor** (**Visitor** *visitor)
Function to accept a table visitor.
- **AbstractTable** * **operator+** (**TableGroup** *tableGroup)
operator overload to add 2 table groups and return a table group
- **AbstractTable** * **operator+** (**Table** *table)
– operator overload to add a table to the table group
- **AbstractTable** * **clone** ()
Returns reference to current table.
- std::vector< **AbstractTable** * > **getTables** ()
Returns tables vector.

Additional Inherited Members

3.67.1 Member Function Documentation

3.67.1.1 acceptVisitor()

```
void TableGroup::acceptVisitor (
    Visitor * visitor ) [virtual]
```

Function to accept a table visitor.

Parameters

<i>visitor</i>	
----------------	--

Implements [AbstractTable](#).

3.67.1.2 addTable()

```
void TableGroup::addTable (
    AbstractTable * aTable )
```

Adds a table to tables vector.

Parameters

<i>aTable</i>	
---------------	--

3.67.1.3 clone()

```
AbstractTable * TableGroup::clone ( ) [virtual]
```

Returns reference to current table.

Returns

AbstractTable*

Implements [AbstractTable](#).

3.67.1.4 `getTables()`

```
std::vector< AbstractTable * > TableGroup::getTables ( )
```

Returns tables vector.

Returns

`std::vector<AbstractTable *>`

3.67.1.5 `operator+()` [1/2]

```
AbstractTable * TableGroup::operator+ (
    Table * table ) [virtual]
```

- operator overload to add a table to the table group

Parameters

<i>table</i>	
--------------	--

Returns

`AbstractTable*`

Implements [AbstractTable](#).

3.67.1.6 `operator+()` [2/2]

```
AbstractTable * TableGroup::operator+ (
    TableGroup * tableGroup ) [virtual]
```

operator overload to add 2 table groups and return a table group

Parameters

<i>tableGroup</i>	
-------------------	--

Returns

`AbstractTable*`

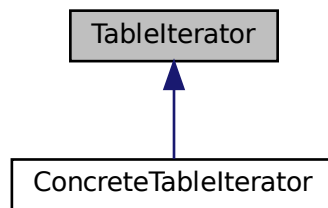
Implements [AbstractTable](#).

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

- Table.h
- Table.cpp

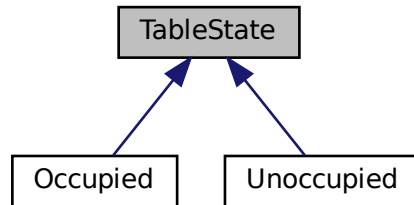
3.68 TableIterator Class Reference

Inheritance diagram for TableIterator:



3.69 TableState Class Reference

Inheritance diagram for TableState:



Public Member Functions

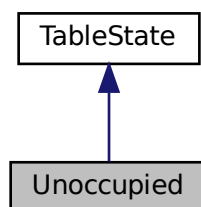
- virtual std::string **getState** ()=0
- virtual void **handleState** ([AbstractTable](#) *table)=0

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

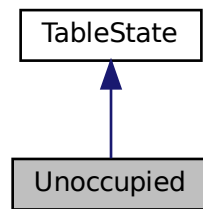
- Table.h

3.70 Unoccupied Class Reference

Inheritance diagram for Unoccupied:



Collaboration diagram for Unoccupied:



Public Member Functions

- void `handleState` (`AbstractTable` *table)
State handler for table state.
- std::string `getState` ()
Returns string which specifies table state.

3.70.1 Member Function Documentation

3.70.1.1 `getState()`

```
std::string Unoccupied::getState ( ) [virtual]
```

Returns string which specifies table state.

Returns

std::string

Implements `TableState`.

3.70.1.2 `handleState()`

```
void Unoccupied::handleState (  
    AbstractTable * table ) [virtual]
```

State handler for table state.

Parameters

<i>table</i>	
--------------	--

Implements [TableState](#).

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

- Table.h
- Table.cpp

3.71 Visitor Class Reference

Public Member Functions

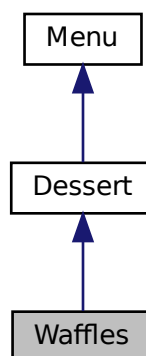
- virtual void **visitTable** ([AbstractTable](#) *)

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

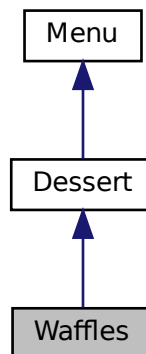
- Visitor.h
- Visitor.cpp

3.72 Waffles Class Reference

Inheritance diagram for Waffles:



Collaboration diagram for Waffles:



Public Member Functions

- [Waffles](#) ()
Constrcutor for waffles dessert.

Additional Inherited Members

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

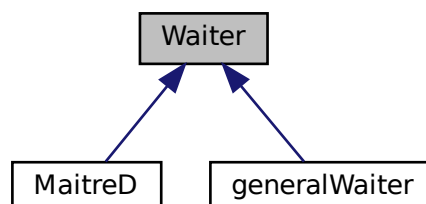
- Menu.h
- Menu.cpp

3.73 Waiter Class Reference

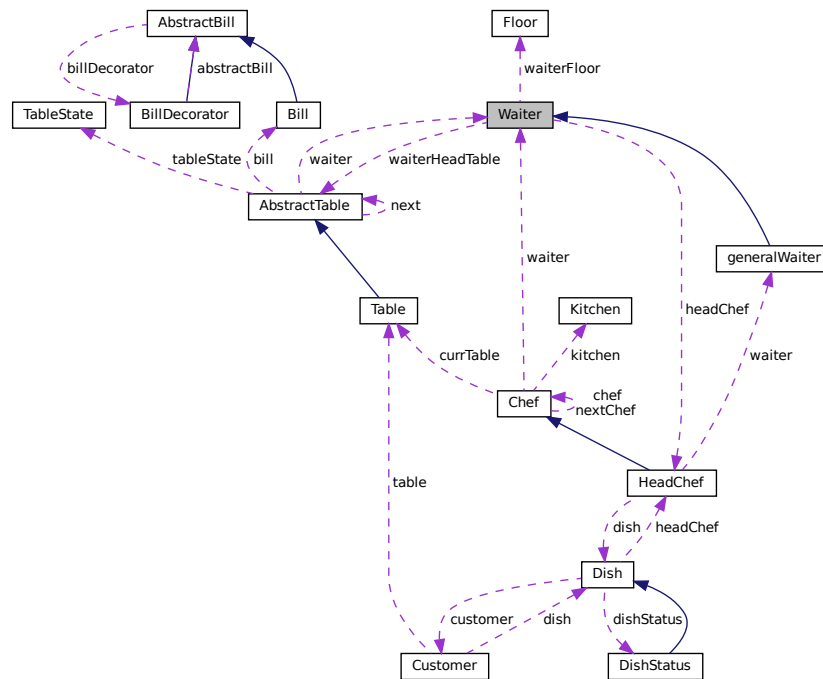
Waiters are responsible for serving customers.

```
#include <Waiter.h>
```

Inheritance diagram for Waiter:



Collaboration diagram for Waiter:



Public Member Functions

- **Waiter** (std::string WaiterName, [HeadChef](#) *hc, [Floor](#) *floor)
- virtual void **performTask** ()=0
- void **deliverOrder** ([Dish](#) *dish)
Delivers an order from the head chef to the relevant customer.
- void **getOrder** ([Dish](#) *dish)
Gets the orders of all the customers and sends each to the head chef use place order function of all customers at table.
- void **sendOrder** ([Dish](#) *)
Send order function for the waiter class This function sends all the customer orders to the head chef and changes the state of the dishes sent to "preparing".

Public Attributes

- [Floor](#) * **waiterFloor**
- int **waiterWaitTime**
- [HeadChef](#) * **headChef**
- std::string **waiterName**
- [AbstractTable](#) * **waiterHeadTable**

3.73.1 Detailed Description

Waiters are responsible for serving customers.

3.73.2 Member Function Documentation

3.73.2.1 deliverOrder()

```
void Waiter::deliverOrder (
    Dish * order )
```

Delivers an order from the head chef to the relevant customer.

Parameters

<i>order</i>	
--------------	--

3.73.2.2 sendOrder()

```
void Waiter::sendOrder (
    Dish * order )
```

Send order function for the waiter class This function sends all the customer orders to the head chef and changes the state of the dishes sent to "preparing".

Parameters

<i>order</i>	
--------------	--

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

- Waiter.h
- Waiter.cpp

Index

[__pthread_cleanup_frame](#), 7
[__pthread_unwind_buf_t](#), 7
[_pthread_cleanup_buffer](#), 8

[AbstractBill](#), 8
[AbstractTable](#), 9
 [getBill](#), 11
 [getCustomers](#), 12
 [getNumberOfSeats](#), 12
 [getState](#), 12
 [getTableID](#), 12
 [getWaiter](#), 12
 [setCustomers](#), 13
 [setState](#), 13
 [setWaiter](#), 13

[accept](#)
 [Customer](#), 45
[acceptVisitor](#)
 [Table](#), 100
 [TableGroup](#), 105

[AddDish](#)
 [HeadChef](#), 65
[addTable](#)
 [TableGroup](#), 105
[addToTab](#)
 [generalWaiter](#), 63
 [Tab](#), 97

[allocateTable](#)
 [MaitreD](#), 71

[App](#), 14
[assignCustomerTable](#)
 [Customer](#), 45

[Attach](#)
 [HeadChef](#), 66

[BBQRibs](#), 14
[BBQSteak](#), 15
[Beverage](#), 16
[Bill](#), 17
[BillDecorator](#), 19
[BillItem](#), 20
[BubblegumMilkshake](#), 21
[BuffaloWings](#), 22

[Card](#), 23
[Cash](#), 24
[checkOrder](#)
 [Customer](#), 45
[CheeseBurger](#), 25
[CheeseSteak](#), 26

[Chef](#), 27
 [GetRole](#), 29
 [SetNextChef](#), 29
 [VisitTable](#), 29
[ChefNotifier](#), 30
[ChickenNuggets](#), 30
[ChickenTenders](#), 31
[ChickenWings](#), 32
[Chips](#), 33
[ChocolateBrownies](#), 34

[clone](#)
 [Table](#), 100
 [TableGroup](#), 105

[Coke](#), 35
[CokeZero](#), 36
[Coleslaw](#), 37

[Coleslaw](#), 38
[commisChef](#), 39
 [commisChef](#), 40
 [PrepareDish](#), 40

[complimentWaiter](#)
 [Customer](#), 46

[ConcreteTableIterator](#), 41

[createTab](#)
 [Customer](#), 46

[Customer](#), 43
 [accept](#), 45
 [assignCustomerTable](#), 45
 [checkOrder](#), 45
 [complimentWaiter](#), 46
 [createTab](#), 46
 [Customer](#), 44
 [getBill](#), 46
 [getMood](#), 46
 [getName](#), 46
 [getTableNum](#), 47
 [leaveRestaurant](#), 47
 [pay](#), 47
 [setMood](#), 48
 [setReadyToLeaveStatus](#), 48
 [setReadyToOrderStatus](#), 48
 [setReadyToPayStatus](#), 48
 [setTab](#), 49
 [setTableNum](#), 49

[CustomTipDecorator](#), 50

[deliverOrder](#)
 [Waiter](#), 114

[Dessert](#), 51
[Dish](#), 52

- Dish, 53
- getCustomerName, 54
- getCustomerTable, 54
- getDishStatus, 54
- setDishStatus, 54
- DishStatus, 55
- Donuts, 57
- Engine, 58
- Facade, 58
 - Facade, 59
 - GetInstance, 59
- Floor, 60
 - splitTables, 61
- generalWaiter, 61
 - addToTab, 63
 - getTab, 63
 - payTab, 63
- getBill
 - AbstractTable, 11
 - Customer, 46
 - Table, 100
- getComplaints
 - Manager, 73
- getCustomerName
 - Dish, 54
- getCustomers
 - AbstractTable, 12
 - Table, 101
- getCustomerTable
 - Dish, 54
- getDishStatus
 - Dish, 54
- GetInstance
 - Facade, 59
- getMood
 - Customer, 46
- getName
 - Customer, 46
 - Tab, 98
- getnumberOfSeats
 - AbstractTable, 12
- GetRole
 - Chef, 29
- getState
 - AbstractTable, 12
 - Occupied, 81
 - Table, 101
 - Unoccupied, 110
- getStatus
 - Preparing, 86
 - ReadyForPickUp, 88
 - StillQueued, 93
- getTab
 - generalWaiter, 63
 - Tab, 98
- getTableID
 - AbstractTable, 12
- getTableNum
 - Customer, 47
- getTables
 - TableGroup, 105
- getWaiter
 - AbstractTable, 12
 - Table, 101
- handleComplaint
 - Manager, 73
- handleState
 - Occupied, 81
 - Unoccupied, 110
- HeadChef, 64
 - AddDish, 65
 - Attach, 66
 - Notify, 66
 - PrepareDish, 66
- HotDog, 67
- Kitchen, 68
- leaveRestaurant
 - Customer, 47
- MacAndCheese, 68
- MainDish, 69
- MaitreD, 70
 - allocateTable, 71
 - mergeTables, 72
 - splitTables, 72
- Manager, 72
 - getComplaints, 73
 - handleComplaint, 73
 - visitTable, 73
- Mediator, 74
- Menu, 74
- MenuDecorator, 76
- mergeTables
 - MaitreD, 72
- MushroomSauce, 78
 - MushroomSauce, 79
- Notify
 - HeadChef, 66
- Observer, 79
- Occupied, 80
 - getState, 81
 - handleState, 81
- OnionRings, 82
- operator+
 - Table, 101, 102
 - TableGroup, 106
- Pancakes, 83
- pay
 - Customer, 47
- PaymentStrategy, 84

payTab
 generalWaiter, 63
PrepareDish
 commisChef, 40
 HeadChef, 66
Preparing, 85
 getStatus, 86

ReadyForPickUp, 87
 getStatus, 88

Salad, 89
 Salad, 90
sendOrder
 Waiter, 114
setCustomers
 AbstractTable, 13
setDishStatus
 Dish, 54
setMood
 Customer, 48
SetNextChef
 Chef, 29
setReadyToLeaveStatus
 Customer, 48
setReadyToOrderStatus
 Customer, 48
setReadyToPayStatus
 Customer, 48
setState
 AbstractTable, 13
 Table, 102
setTab
 Customer, 49
setTableNum
 Customer, 49
setWaiter
 AbstractTable, 13
 Table, 102
splitTables
 Floor, 61
 MaitreD, 72
Sprite, 90
Starter, 91
StillQueued, 92
 getStatus, 93
StrawberryMilkshake, 94
SubBill, 95
subtractFromTab
 Tab, 98

Tab, 96
 addToTab, 97
 getName, 98
 getTab, 98
 subtractFromTab, 98
Table, 99
 acceptVisitor, 100
 clone, 100
 getBill, 100
 getCustomers, 101
 getState, 101
 getWaiter, 101
 operator+, 101, 102
 setState, 102
 setWaiter, 102
TableGroup, 103
 acceptVisitor, 105
 addTable, 105
 clone, 105
 getTables, 105
 operator+, 106
TableIterator, 107
TableState, 109

Unoccupied, 109
 getState, 110
 handleState, 110

Visitor, 111
VisitTable
 Chef, 29
visitTable
 Manager, 73

Waffles, 111
Waiter, 112
 deliverOrder, 114
 sendOrder, 114