

## COS214 Project

Generated by Doxygen 1.9.1



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>9</b>
4.1 Beer Class Reference	9
4.1.1 Constructor & Destructor Documentation	10
4.1.1.1 Beer()	11
4.1.1.2 ~Beer()	11
4.2 BeerBuilder Class Reference	11
4.3 Burger Class Reference	13
4.3.1 Constructor & Destructor Documentation	14
4.3.1.1 Burger()	15
4.3.1.2 ~Burger()	15
4.4 BurgerBuilder Class Reference	15
4.5 Chef Class Reference	17
4.5.1 Constructor & Destructor Documentation	18
4.5.1.1 Chef()	18
4.5.1.2 ~Chef()	19
4.5.2 Member Function Documentation	19
4.5.2.1 request()	19
4.6 Chips Class Reference	20
4.6.1 Constructor & Destructor Documentation	21
4.6.1.1 Chips()	21
4.6.1.2 ~Chips()	21
4.7 ChipsBuilder Class Reference	22
4.8 ComplexOrder Class Reference	23
4.8.1 Constructor & Destructor Documentation	24
4.8.1.1 ComplexOrder()	24
4.8.1.2 ~ComplexOrder()	25
4.8.2 Member Function Documentation	25
4.8.2.1 addToOrder()	25
4.8.2.2 appendToOrder()	25
4.8.2.3 calculatePrice()	26
4.9 Customer Class Reference	26
4.9.1 Constructor & Destructor Documentation	26
4.9.1.1 Customer()	26
4.9.1.2 ~Customer()	27

4.9.2 Member Function Documentation	27
4.9.2.1 acceptWaiter()	27
4.9.2.2 calculatePayment()	27
4.9.2.3 changeRating()	28
4.9.2.4 getOrder()	28
4.9.2.5 getOrderRequest()	29
4.9.2.6 getTable()	29
4.9.2.7 getTimestamp()	29
4.9.2.8 getWaiter()	30
4.9.2.9 receiveOrder()	30
4.10 Drink Class Reference	31
4.10.1 Constructor & Destructor Documentation	32
4.10.1.1 Drink()	32
4.11 DrinkBuilder Class Reference	33
4.11.1 Member Function Documentation	34
4.11.1.1 getItem()	34
4.12 DrinkChef Class Reference	35
4.12.1 Constructor & Destructor Documentation	36
4.12.1.1 DrinkChef()	36
4.12.1.2 ~DrinkChef()	36
4.12.2 Member Function Documentation	36
4.12.2.1 preparePart()	36
4.13 Fish Class Reference	37
4.13.1 Constructor & Destructor Documentation	38
4.13.1.1 Fish()	39
4.13.1.2 ~Fish()	39
4.14 FishBuilder Class Reference	39
4.15 Floor Class Reference	41
4.15.1 Constructor & Destructor Documentation	41
4.15.1.1 Floor()	41
4.15.1.2 ~Floor()	42
4.15.2 Member Function Documentation	42
4.15.2.1 getNumTables()	42
4.15.2.2 getTable()	42
4.15.2.3 seatCustomer()	43
4.15.2.4 setRestaurant()	43
4.16 HeadChef Class Reference	44
4.16.1 Constructor & Destructor Documentation	45
4.16.1.1 HeadChef()	45
4.16.1.2 ~HeadChef()	45
4.17 Interface Class Reference	45
4.17.1 Constructor & Destructor Documentation	46

4.17.1.1 Interface()	46
4.17.1.2 ~Interface()	46
4.17.2 Member Function Documentation	46
4.17.2.1 generateNumberOfCustomers()	46
4.17.2.2 generateOrderString()	47
4.17.2.3 getCurrentUnixTime()	47
4.17.2.4 runCustomer()	47
4.18 Item Class Reference	48
4.18.1 Constructor & Destructor Documentation	49
4.18.1.1 Item()	49
4.18.2 Member Function Documentation	49
4.18.2.1 calculatePrice()	49
4.19 ItemBuilder Class Reference	50
4.20 Kitchen Class Reference	51
4.20.1 Constructor & Destructor Documentation	51
4.20.1.1 Kitchen()	51
4.20.2 Member Function Documentation	51
4.20.2.1 makeNextOrder()	52
4.20.2.2 receiveOrder()	52
4.20.2.3 setRestaurant()	52
4.21 MainBuilder Class Reference	53
4.21.1 Member Function Documentation	54
4.21.1.1 getItem()	54
4.22 MainChef Class Reference	55
4.22.1 Constructor & Destructor Documentation	56
4.22.1.1 MainChef()	56
4.22.1.2 ~MainChef()	56
4.22.2 Member Function Documentation	56
4.22.2.1 preparePart()	56
4.23 MainMeal Class Reference	57
4.23.1 Constructor & Destructor Documentation	58
4.23.1.1 MainMeal()	58
4.24 Neutral Class Reference	59
4.24.1 Member Function Documentation	60
4.24.1.1 calculateTip()	60
4.24.1.2 changeState()	60
4.24.1.3 getRating()	60
4.25 Order Class Reference	61
4.25.1 Constructor & Destructor Documentation	61
4.25.1.1 Order()	61
4.25.2 Member Function Documentation	62
4.25.2.1 addToOrder()	62

4.25.2.2 appendToOrder()	62
4.25.2.3 calculatePrice()	62
4.25.2.4 getWaiter()	63
4.26 OrderContainer Class Reference	63
4.26.1 Constructor & Destructor Documentation	63
4.26.1.1 OrderContainer()	63
4.26.2 Member Function Documentation	64
4.26.2.1 getOrder()	64
4.26.2.2 getRequestedOrder()	64
4.27 Rating Class Reference	64
4.28 Restaurant Class Reference	65
4.28.1 Constructor & Destructor Documentation	65
4.28.1.1 Restaurant()	65
4.28.1.2 ~Restaurant()	66
4.28.2 Member Function Documentation	66
4.28.2.1 cleanUp()	66
4.28.2.2 initialise()	66
4.28.2.3 placeOrder()	67
4.28.2.4 seatCustomer()	67
4.29 Salad Class Reference	68
4.29.1 Constructor & Destructor Documentation	69
4.29.1.1 Salad()	69
4.29.1.2 ~Salad()	70
4.30 SaladBuilder Class Reference	70
4.31 Satisfied Class Reference	72
4.31.1 Member Function Documentation	73
4.31.1.1 calculateTip()	73
4.31.1.2 changeState()	73
4.31.1.3 getRating()	73
4.32 Side Class Reference	74
4.32.1 Constructor & Destructor Documentation	75
4.32.1.1 Side()	75
4.33 SideBuilder Class Reference	76
4.33.1 Member Function Documentation	77
4.33.1.1 getItem()	78
4.34 SideChef Class Reference	78
4.34.1 Constructor & Destructor Documentation	79
4.34.1.1 SideChef()	80
4.34.1.2 ~SideChef()	80
4.34.2 Member Function Documentation	80
4.34.2.1 preparePart()	80
4.35 Soda Class Reference	81

4.35.1 Constructor & Destructor Documentation	82
4.35.1.1 Soda()	82
4.35.1.2 ~Soda()	82
4.36 SodaBuilder Class Reference	83
4.37 Steak Class Reference	84
4.37.1 Constructor & Destructor Documentation	85
4.37.1.1 Steak()	86
4.37.1.2 ~Steak()	86
4.38 SteakBuilder Class Reference	86
4.39 Table Class Reference	88
4.39.1 Constructor & Destructor Documentation	88
4.39.1.1 Table()	88
4.39.1.2 ~Table()	88
4.39.2 Member Function Documentation	89
4.39.2.1 addCustomer()	89
4.39.2.2 cleanUp()	89
4.39.2.3 getCustomer()	89
4.40 Unhappy Class Reference	90
4.40.1 Member Function Documentation	90
4.40.1.1 calculateTip()	91
4.40.1.2 changeState()	91
4.40.1.3 getRating()	91
4.41 Waiter Class Reference	92
4.41.1 Constructor & Destructor Documentation	92
4.41.1.1 Waiter()	92
4.41.1.2 ~Waiter()	92
4.41.2 Member Function Documentation	93
4.41.2.1 cleanUp()	93
4.41.2.2 getCustomer()	93
4.41.2.3 getRestaurant()	93
4.41.2.4 serveCustomer()	93
4.41.2.5 takeOrder()	94
4.42 Water Class Reference	95
4.42.1 Constructor & Destructor Documentation	96
4.42.1.1 Water()	96
4.42.1.2 ~Water()	96
4.43 WaterBuilder Class Reference	97
<b>5 File Documentation</b>	<b>99</b>
5.1 Beer.cpp File Reference	99
5.1.1 Detailed Description	100
5.2 Beer.h File Reference	101

---

5.2.1 Detailed Description . . . . .	102
5.3 BeerBuilder.cpp File Reference . . . . .	102
5.3.1 Detailed Description . . . . .	103
5.4 BeerBuilder.h File Reference . . . . .	104
5.4.1 Detailed Description . . . . .	105
5.5 Burger.cpp File Reference . . . . .	105
5.5.1 Detailed Description . . . . .	106
5.6 Burger.h File Reference . . . . .	107
5.6.1 Detailed Description . . . . .	108
5.7 BurgerBuilder.cpp File Reference . . . . .	108
5.7.1 Detailed Description . . . . .	109
5.8 BurgerBuilder.h File Reference . . . . .	110
5.8.1 Detailed Description . . . . .	111
5.9 Chef.cpp File Reference . . . . .	111
5.9.1 Detailed Description . . . . .	112
5.10 Chef.h File Reference . . . . .	112
5.10.1 Detailed Description . . . . .	114
5.11 Chips.cpp File Reference . . . . .	114
5.11.1 Detailed Description . . . . .	115
5.12 Chips.h File Reference . . . . .	116
5.12.1 Detailed Description . . . . .	117
5.13 ChipsBuilder.cpp File Reference . . . . .	117
5.13.1 Detailed Description . . . . .	118
5.14 ChipsBuilder.h File Reference . . . . .	119
5.14.1 Detailed Description . . . . .	120
5.15 ComplexOrder.cpp File Reference . . . . .	120
5.15.1 Detailed Description . . . . .	121
5.16 ComplexOrder.h File Reference . . . . .	121
5.16.1 Detailed Description . . . . .	122
5.17 Customer.cpp File Reference . . . . .	123
5.17.1 Detailed Description . . . . .	123
5.18 Customer.h File Reference . . . . .	124
5.18.1 Detailed Description . . . . .	124
5.19 Drink.cpp File Reference . . . . .	125
5.19.1 Detailed Description . . . . .	125
5.20 Drink.h File Reference . . . . .	126
5.20.1 Detailed Description . . . . .	127
5.21 DrinkBuilder.cpp File Reference . . . . .	127
5.21.1 Detailed Description . . . . .	128
5.22 DrinkBuilder.h File Reference . . . . .	129
5.22.1 Detailed Description . . . . .	130
5.23 DrinkChef.cpp File Reference . . . . .	130



---

5.23.1 Detailed Description . . . . .	131
5.24 DrinkChef.h File Reference . . . . .	131
5.24.1 Detailed Description . . . . .	133
5.25 Fish.cpp File Reference . . . . .	133
5.25.1 Detailed Description . . . . .	134
5.26 Fish.h File Reference . . . . .	135
5.26.1 Detailed Description . . . . .	136
5.27 FishBuilder.cpp File Reference . . . . .	136
5.27.1 Detailed Description . . . . .	137
5.28 FishBuilder.h File Reference . . . . .	137
5.28.1 Detailed Description . . . . .	139
5.29 Floor.cpp File Reference . . . . .	139
5.29.1 Detailed Description . . . . .	140
5.30 Floor.h File Reference . . . . .	140
5.30.1 Detailed Description . . . . .	142
5.31 HeadChef.cpp File Reference . . . . .	142
5.31.1 Detailed Description . . . . .	143
5.32 HeadChef.h File Reference . . . . .	144
5.32.1 Detailed Description . . . . .	145
5.33 Interface.cpp File Reference . . . . .	145
5.33.1 Detailed Description . . . . .	146
5.34 Interface.h File Reference . . . . .	146
5.34.1 Detailed Description . . . . .	148
5.35 Item.cpp File Reference . . . . .	148
5.35.1 Detailed Description . . . . .	149
5.36 Item.h File Reference . . . . .	149
5.36.1 Detailed Description . . . . .	150
5.37 ItemBuilder.h File Reference . . . . .	150
5.37.1 Detailed Description . . . . .	151
5.38 Kitchen.cpp File Reference . . . . .	151
5.38.1 Detailed Description . . . . .	152
5.39 Kitchen.h File Reference . . . . .	152
5.39.1 Detailed Description . . . . .	154
5.40 main.cpp File Reference . . . . .	154
5.40.1 Detailed Description . . . . .	155
5.40.2 Function Documentation . . . . .	155
5.40.2.1 main() . . . . .	155
5.41 MainBuilder.cpp File Reference . . . . .	155
5.41.1 Detailed Description . . . . .	156
5.42 MainBuilder.h File Reference . . . . .	157
5.42.1 Detailed Description . . . . .	158
5.43 MainChef.cpp File Reference . . . . .	158

5.43.1 Detailed Description . . . . .	159
5.44 MainChef.h File Reference . . . . .	159
5.44.1 Detailed Description . . . . .	161
5.45 MainMeal.cpp File Reference . . . . .	161
5.45.1 Detailed Description . . . . .	162
5.46 MainMeal.h File Reference . . . . .	162
5.46.1 Detailed Description . . . . .	164
5.47 Neutral.cpp File Reference . . . . .	164
5.47.1 Detailed Description . . . . .	164
5.48 Neutral.h File Reference . . . . .	165
5.48.1 Detailed Description . . . . .	165
5.49 Order.cpp File Reference . . . . .	166
5.49.1 Detailed Description . . . . .	166
5.50 Order.h File Reference . . . . .	166
5.50.1 Detailed Description . . . . .	167
5.51 OrderContainer.cpp File Reference . . . . .	168
5.51.1 Detailed Description . . . . .	168
5.52 OrderContainer.h File Reference . . . . .	168
5.52.1 Detailed Description . . . . .	169
5.53 Rating.cpp File Reference . . . . .	170
5.53.1 Detailed Description . . . . .	170
5.54 Rating.h File Reference . . . . .	170
5.54.1 Detailed Description . . . . .	171
5.55 Restaurant.cpp File Reference . . . . .	171
5.55.1 Detailed Description . . . . .	172
5.56 Restaurant.h File Reference . . . . .	172
5.56.1 Detailed Description . . . . .	172
5.57 Salad.cpp File Reference . . . . .	173
5.57.1 Detailed Description . . . . .	173
5.58 Salad.h File Reference . . . . .	174
5.58.1 Detailed Description . . . . .	175
5.59 SaladBuilder.cpp File Reference . . . . .	175
5.59.1 Detailed Description . . . . .	176
5.60 SaladBuilder.h File Reference . . . . .	177
5.60.1 Detailed Description . . . . .	178
5.61 Satisfied.cpp File Reference . . . . .	178
5.61.1 Detailed Description . . . . .	179
5.62 Satisfied.h File Reference . . . . .	179
5.62.1 Detailed Description . . . . .	180
5.63 Side.cpp File Reference . . . . .	181
5.63.1 Detailed Description . . . . .	181
5.64 Side.h File Reference . . . . .	182

5.64.1 Detailed Description . . . . .	183
5.65 SideBuilder.cpp File Reference . . . . .	183
5.65.1 Detailed Description . . . . .	184
5.66 SideBuilder.h File Reference . . . . .	185
5.66.1 Detailed Description . . . . .	186
5.67 SideChef.cpp File Reference . . . . .	186
5.67.1 Detailed Description . . . . .	187
5.68 SideChef.h File Reference . . . . .	187
5.68.1 Detailed Description . . . . .	189
5.69 Soda.cpp File Reference . . . . .	189
5.69.1 Detailed Description . . . . .	190
5.70 Soda.h File Reference . . . . .	191
5.70.1 Detailed Description . . . . .	192
5.71 SodaBuilder.cpp File Reference . . . . .	192
5.71.1 Detailed Description . . . . .	193
5.72 SodaBuilder.h File Reference . . . . .	193
5.72.1 Detailed Description . . . . .	195
5.73 Steak.cpp File Reference . . . . .	195
5.73.1 Detailed Description . . . . .	196
5.74 Steak.h File Reference . . . . .	197
5.74.1 Detailed Description . . . . .	198
5.75 SteakBuilder.cpp File Reference . . . . .	198
5.75.1 Detailed Description . . . . .	199
5.76 SteakBuilder.h File Reference . . . . .	199
5.76.1 Detailed Description . . . . .	201
5.77 Table.cpp File Reference . . . . .	201
5.77.1 Detailed Description . . . . .	202
5.78 Table.h File Reference . . . . .	202
5.78.1 Detailed Description . . . . .	203
5.79 Unhappy.cpp File Reference . . . . .	203
5.79.1 Detailed Description . . . . .	204
5.80 Unhappy.h File Reference . . . . .	204
5.80.1 Detailed Description . . . . .	205
5.81 Waiter.cpp File Reference . . . . .	205
5.81.1 Detailed Description . . . . .	206
5.82 Waiter.h File Reference . . . . .	206
5.82.1 Detailed Description . . . . .	207
5.83 Water.cpp File Reference . . . . .	207
5.83.1 Detailed Description . . . . .	208
5.84 Water.h File Reference . . . . .	209
5.84.1 Detailed Description . . . . .	210
5.85 WaterBuilder.cpp File Reference . . . . .	210

5.85.1 Detailed Description . . . . .	211
5.86 WaterBuilder.h File Reference . . . . .	211
5.86.1 Detailed Description . . . . .	213
<b>Index</b>	<b>215</b>

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Chef . . . . .	17
DrinkChef . . . . .	35
HeadChef . . . . .	44
MainChef . . . . .	55
SideChef . . . . .	78
Customer . . . . .	26
Floor . . . . .	41
Interface . . . . .	45
ItemBuilder . . . . .	50
DrinkBuilder . . . . .	33
BeerBuilder . . . . .	11
SodaBuilder . . . . .	83
WaterBuilder . . . . .	97
MainBuilder . . . . .	53
BurgerBuilder . . . . .	15
FishBuilder . . . . .	39
SteakBuilder . . . . .	86
SideBuilder . . . . .	76
ChipsBuilder . . . . .	22
SaladBuilder . . . . .	70
Kitchen . . . . .	51
Order . . . . .	61
ComplexOrder . . . . .	23
Item . . . . .	48
Drink . . . . .	31
Beer . . . . .	9
Soda . . . . .	81
Water . . . . .	95
MainMeal . . . . .	57
Burger . . . . .	13
Fish . . . . .	37
Steak . . . . .	84
Side . . . . .	74
Chips . . . . .	20

Salad . . . . .	68
OrderContainer . . . . .	63
Rating . . . . .	64
Neutral . . . . .	59
Satisfied . . . . .	72
Unhappy . . . . .	90
Restaurant . . . . .	65
Table . . . . .	88
Waiter . . . . .	92

## Chapter 2

# Class Index

### 2.1 Class List

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

Beer	9
BeerBuilder	11
Burger	13
BurgerBuilder	15
Chef	17
Chips	20
ChipsBuilder	22
ComplexOrder	23
Customer	26
Drink	31
DrinkBuilder	33
DrinkChef	35
Fish	37
FishBuilder	39
Floor	41
HeadChef	44
Interface	45
Item	48
ItemBuilder	50
Kitchen	51
MainBuilder	53
MainChef	55
MainMeal	57
Neutral	59
Order	61
OrderContainer	63
Rating	64
Restaurant	65
Salad	68
SaladBuilder	70
Satisfied	72
Side	74
SideBuilder	76
SideChef	78
Soda	81

<a href="#">SodaBuilder</a>	83
<a href="#">Steak</a>	84
<a href="#">SteakBuilder</a>	86
<a href="#">Table</a>	88
<a href="#">Unhappy</a>	90
<a href="#">Waiter</a>	92
<a href="#">Water</a>	95
<a href="#">WaterBuilder</a>	97



## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">Beer.cpp</a>	Contains implementation for the <a href="#">Beer</a> class . . . . .	99
<a href="#">Beer.h</a>	Contains the declaration for the <a href="#">Beer</a> class . . . . .	101
<a href="#">BeerBuilder.cpp</a>	Implementation of the <a href="#">BeerBuilder</a> class . . . . .	102
<a href="#">BeerBuilder.h</a>	Contains declaration for the <a href="#">BeerBuilder</a> class . . . . .	104
<a href="#">Burger.cpp</a>	Contains implementation for the <a href="#">Burger</a> class . . . . .	105
<a href="#">Burger.h</a>	Contains declaration for the <a href="#">Burger</a> class . . . . .	107
<a href="#">BurgerBuilder.cpp</a>	Contains the implementation for the <a href="#">BurgerBuilder</a> class . . . . .	108
<a href="#">BurgerBuilder.h</a>	Contains declaration for the <a href="#">BurgerBuilder</a> class . . . . .	110
<a href="#">Chef.cpp</a>	Contains implementation for the <a href="#">Chef</a> class . . . . .	111
<a href="#">Chef.h</a>	Contains declaration for the <a href="#">Chef</a> class . . . . .	112
<a href="#">Chips.cpp</a>	Contains implementation for the <a href="#">Chips</a> class . . . . .	114
<a href="#">Chips.h</a>	Contains declaration for the <a href="#">Chips</a> class . . . . .	116
<a href="#">ChipsBuilder.cpp</a>	Implementation of the <a href="#">ChipsBuilder</a> class . . . . .	117
<a href="#">ChipsBuilder.h</a>	Contains declaration for the <a href="#">ChipsBuilder</a> class . . . . .	119
<a href="#">ComplexOrder.cpp</a>	Contains the implementation for the <a href="#">ComplexOrder</a> class . . . . .	120
<a href="#">ComplexOrder.h</a>	Contains declaration for the <a href="#">ComplexOrder</a> class . . . . .	121
<a href="#">Customer.cpp</a>	Contains implementation for the <a href="#">Customer</a> class . . . . .	123
<a href="#">Customer.h</a>	Contains declaration for the <a href="#">Customer</a> class . . . . .	124

<a href="#">Drink.cpp</a>	Contains implementation for the <a href="#">Drink</a> class . . . . .	125
<a href="#">Drink.h</a>	Contains declaration for the <a href="#">Drink</a> class . . . . .	126
<a href="#">DrinkBuilder.cpp</a>	Implementation of the <a href="#">DrinkBuilder</a> class . . . . .	127
<a href="#">DrinkBuilder.h</a>	Contains declaration for the <a href="#">DrinkBuilder</a> class . . . . .	129
<a href="#">DrinkChef.cpp</a>	Contains implementation for the <a href="#">DrinkChef</a> class . . . . .	130
<a href="#">DrinkChef.h</a>	Contains declaration for the <a href="#">DrinkChef</a> class . . . . .	131
<a href="#">Fish.cpp</a>	Contains implementation for the <a href="#">Fish</a> class . . . . .	133
<a href="#">Fish.h</a>	Contains declaration for the <a href="#">Fish</a> class . . . . .	135
<a href="#">FishBuilder.cpp</a>	Contains the implementation for the <a href="#">FishBuilder</a> class . . . . .	136
<a href="#">FishBuilder.h</a>	Contains declaration for the <a href="#">FishBuilder</a> class . . . . .	137
<a href="#">Floor.cpp</a>	Contains implementation for the <a href="#">Floor</a> class . . . . .	139
<a href="#">Floor.h</a>	Contains declaration for the <a href="#">Floor</a> class . . . . .	140
<a href="#">HeadChef.cpp</a>	Contains implementation for the <a href="#">HeadChef</a> class . . . . .	142
<a href="#">HeadChef.h</a>	Contains declaration for the <a href="#">HeadChef</a> class . . . . .	144
<a href="#">Interface.cpp</a>	Contains implementation for the <a href="#">Interface</a> class . . . . .	145
<a href="#">Interface.h</a>	Contains declaration for the <a href="#">Interface</a> class . . . . .	146
<a href="#">Item.cpp</a>	Contains implementation for the <a href="#">Item</a> class . . . . .	148
<a href="#">Item.h</a>	Contains declaration for the <a href="#">Item</a> class . . . . .	149
<a href="#">ItemBuilder.h</a>	Contains declaration for the <a href="#">ItemBuilder</a> class . . . . .	150
<a href="#">Kitchen.cpp</a>	Contains implementation for the <a href="#">Kitchen</a> class . . . . .	151
<a href="#">Kitchen.h</a>	Contains declaration for the <a href="#">Kitchen</a> class . . . . .	152
<a href="#">main.cpp</a>	This is the file that the user will interact with . . . . .	154
<a href="#">MainBuilder.cpp</a>	Contains the implementation for the <a href="#">MainBuilder</a> class . . . . .	155
<a href="#">MainBuilder.h</a>	Contains declaration for the <a href="#">MainBuilder</a> class . . . . .	157
<a href="#">MainChef.cpp</a>	Contains implementation for the <a href="#">MainChef</a> class . . . . .	158
<a href="#">MainChef.h</a>	Contains declaration for the <a href="#">MainChef</a> class . . . . .	159
<a href="#">MainMeal.cpp</a>	Contains implementation for the <a href="#">MainMeal</a> class . . . . .	161
<a href="#">MainMeal.h</a>	Contains declaration for the <a href="#">MainMeal</a> class . . . . .	162
<a href="#">Neutral.cpp</a>	Contains implementation for the <a href="#">Neutral</a> class . . . . .	164

<a href="#">Neutral.h</a>	Contains declaration for the <a href="#">Neutral</a> class . . . . .	165
<a href="#">Order.cpp</a>	Contains implementation for the <a href="#">Order</a> class . . . . .	166
<a href="#">Order.h</a>	Contains declaration for the <a href="#">Order</a> class . . . . .	166
<a href="#">OrderContainer.cpp</a>	Contains implementation for the <a href="#">OrderContainer</a> class . . . . .	168
<a href="#">OrderContainer.h</a>	The <a href="#">OrderContainer</a> class represents a container for an <a href="#">Order</a> object and its corresponding requested order string . . . . .	168
<a href="#">Rating.cpp</a>	Contains implementation for the <a href="#">Rating</a> class . . . . .	170
<a href="#">Rating.h</a>	Contains declaration for the <a href="#">Rating</a> class . . . . .	170
<a href="#">Restaurant.cpp</a>	Contains implementation for the <a href="#">Restaurant</a> class . . . . .	171
<a href="#">Restaurant.h</a>	Contains declaration for the <a href="#">Restaurant</a> class . . . . .	172
<a href="#">Salad.cpp</a>	Contains implementation for the <a href="#">Salad</a> class . . . . .	173
<a href="#">Salad.h</a>	Contains declaration for the <a href="#">Salad</a> class . . . . .	174
<a href="#">SaladBuilder.cpp</a>	Contains implementation for the <a href="#">SaladBuilder</a> class . . . . .	175
<a href="#">SaladBuilder.h</a>	Contains declaration for the <a href="#">SaladBuilder</a> class . . . . .	177
<a href="#">Satisfied.cpp</a>	Contains implementation for the <a href="#">Satisfied</a> class . . . . .	178
<a href="#">Satisfied.h</a>	Contains declaration for the <a href="#">Satisfied</a> class . . . . .	179
<a href="#">Side.cpp</a>	Contains implementation for the <a href="#">Side</a> class . . . . .	181
<a href="#">Side.h</a>	Contains declaration for the <a href="#">Side</a> class . . . . .	182
<a href="#">SideBuilder.cpp</a>	Contains implementation for the <a href="#">SideBuilder</a> class . . . . .	183
<a href="#">SideBuilder.h</a>	Contains declaration for the <a href="#">SideBuilder</a> class . . . . .	185
<a href="#">SideChef.cpp</a>	Contains implementation for the <a href="#">SideChef</a> class . . . . .	186
<a href="#">SideChef.h</a>	Contains declaration for the <a href="#">SideChef</a> class . . . . .	187
<a href="#">Soda.cpp</a>	Contains implementation for the <a href="#">Soda</a> class . . . . .	189
<a href="#">Soda.h</a>	Contains declaration for the <a href="#">Soda</a> class . . . . .	191
<a href="#">SodaBuilder.cpp</a>	Contains implementation for the <a href="#">SodaBuilder</a> class . . . . .	192
<a href="#">SodaBuilder.h</a>	Contains declaration for the <a href="#">SodaBuilder</a> class . . . . .	193
<a href="#">Steak.cpp</a>	Contains implementation for the <a href="#">Steak</a> class . . . . .	195
<a href="#">Steak.h</a>	Contains declaration for the <a href="#">Steak</a> class . . . . .	197
<a href="#">SteakBuilder.cpp</a>	Contains the implementation for the <a href="#">SteakBuilder</a> class . . . . .	198

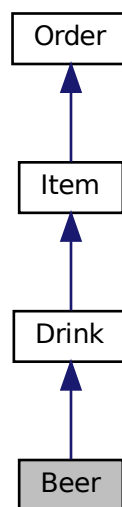
<a href="#">SteakBuilder.h</a>	
Contains declaration for the <a href="#">SteakBuilder</a> class . . . . .	199
<a href="#">Table.cpp</a>	
Contains implementation for the <a href="#">Table</a> class . . . . .	201
<a href="#">Table.h</a>	
Contains declaration for the <a href="#">Table</a> class . . . . .	202
<a href="#">Unhappy.cpp</a>	
Contains implementation for the <a href="#">Unhappy</a> class . . . . .	203
<a href="#">Unhappy.h</a>	
Contains declaration for the <a href="#">Unhappy</a> class . . . . .	204
<b>UnitTesting.hpp</b> . . . . .	<b>??</b>
<a href="#">Waiter.cpp</a>	
Contains implementation for the <a href="#">Waiter</a> class . . . . .	205
<a href="#">Waiter.h</a>	
Contains declaration for the <a href="#">Waiter</a> class . . . . .	206
<a href="#">Water.cpp</a>	
Contains implementation for the <a href="#">Water</a> class . . . . .	207
<a href="#">Water.h</a>	
Contains declaration for the <a href="#">Water</a> class . . . . .	209
<a href="#">WaterBuilder.cpp</a>	
Contains implementation for the <a href="#">WaterBuilder</a> class . . . . .	210
<a href="#">WaterBuilder.h</a>	
Contains declaration for the <a href="#">WaterBuilder</a> class . . . . .	211

## Chapter 4

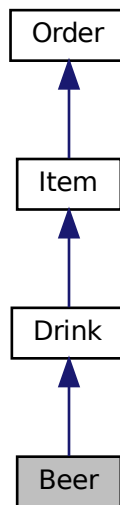
# Class Documentation

### 4.1 Beer Class Reference

Inheritance diagram for Beer:



Collaboration diagram for Beer:



## Public Member Functions

- [Beer](#) ()  
*Beer Constructor.*
- [~Beer](#) ()  
*Beer Destructor.*

## Public Attributes

- bool [gotBeerGlass](#) = false  
*Whether a beer glass has been obtained.*
- bool [pouredBeer](#) = false  
*Whether beer has been poured into the glass.*
- bool [assembledBeer](#) = false  
*Whether the beer has been assembled.*

## Additional Inherited Members

### 4.1.1 Constructor & Destructor Documentation

#### 4.1.1.1 Beer()

```
Beer::Beer ( )
```

[Beer](#) Constructor.

##### Authors

Aidan Chapman (u22738917)

#### 4.1.1.2 ~Beer()

```
Beer::~~Beer ( )
```

[Beer](#) Destructor.

##### Authors

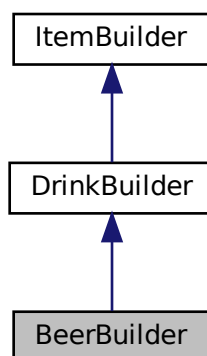
Aidan Chapman (u22738917)

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

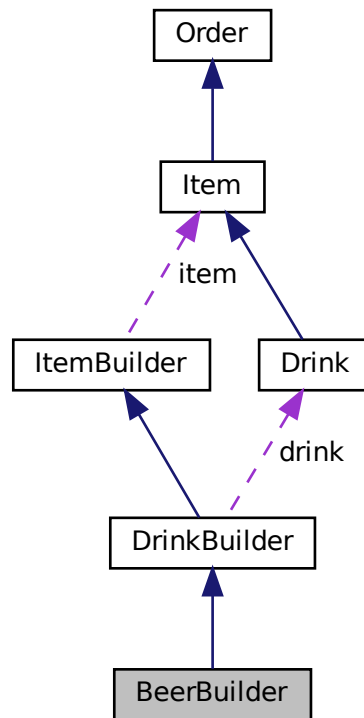
- [Beer.h](#)
- [Beer.cpp](#)

## 4.2 BeerBuilder Class Reference

Inheritance diagram for BeerBuilder:



Collaboration diagram for BeerBuilder:



## Public Member Functions

- **BeerBuilder** ()  
*Construct a new Beer Builder:: Beer Builder object.*
- **~BeerBuilder** ()  
*Destroy the Beer Builder:: Beer Builder object.*
- void **getGlass** ()  
*prepare the glass*
- void **pourDrink** ()  
*Pour the Drink object.*
- void **assembleDrink** ()  
*Assemble the Drink object.*
- void **getBeerGlass** ()  
*Get the Beer Glass object.*
- void **pourBeer** ()  
*Pour the Beer object.*
- void **assembleBeer** ()  
*Assemble the Beer object.*



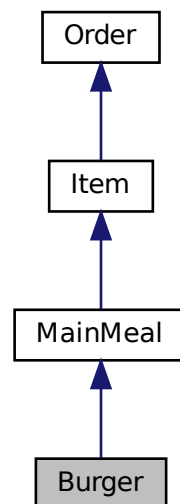
### Additional Inherited Members

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

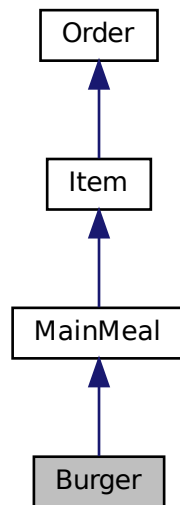
- [BeerBuilder.h](#)
- [BeerBuilder.cpp](#)

## 4.3 Burger Class Reference

Inheritance diagram for Burger:



Collaboration diagram for Burger:



## Public Member Functions

- [Burger \(\)](#)  
*Burger Constructor.*
- [~Burger \(\)](#)  
*Burger Destructor.*

## Public Attributes

- bool [cookedPatty](#) = false  
*Whether the patty has been cooked.*
- bool [assembledBurger](#) = false  
*Whether the burger has been assembled.*
- bool [batteredBun](#) = false  
*Whether the bun has been battered.*
- bool [preparedVegetables](#) = false  
*Whether the vegetables have been prepared.*

## Additional Inherited Members

### 4.3.1 Constructor & Destructor Documentation

#### 4.3.1.1 `Burger()`

```
Burger::Burger ( )
```

[Burger](#) Constructor.

##### Authors

Aidan Chapman (u22738917)

#### 4.3.1.2 `~Burger()`

```
Burger::~~Burger ( )
```

[Burger](#) Destructor.

##### Authors

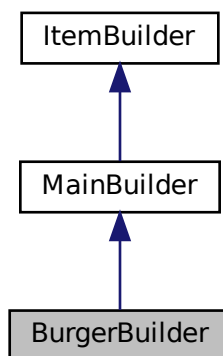
Aidan Chapman (u22738917)

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

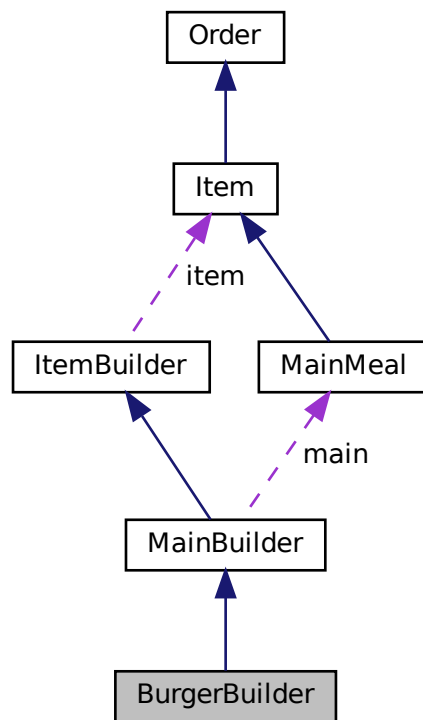
- [Burger.h](#)
- [Burger.cpp](#)

## 4.4 BurgerBuilder Class Reference

Inheritance diagram for BurgerBuilder:



Collaboration diagram for BurgerBuilder:



## Public Member Functions

- [BurgerBuilder](#) ()  
Constructs a new [BurgerBuilder](#) object.
- [~BurgerBuilder](#) ()  
Destructor for the [BurgerBuilder](#) class.
- void [prepareMeat](#) ()  
Prepares the meat for the burger.
- void [seasonMeat](#) ()  
Seasons the meat for the burger.
- void [cookMeat](#) ()  
Cooks the meat for the burger.
- void [plateMain](#) ()  
Plates the main item of the burger.
- void [butterBun](#) ()  
Butters the bun for the burger.
- void [prepareVegetables](#) ()  
Prepares the vegetables for the burger.
- void [cookPatty](#) ()  
Cooks the patty for the burger.
- void [applySauce](#) ()

- Applies sauce to the burger.*
  - void [assembleBurger](#) ()  
*Assembles the burger.*

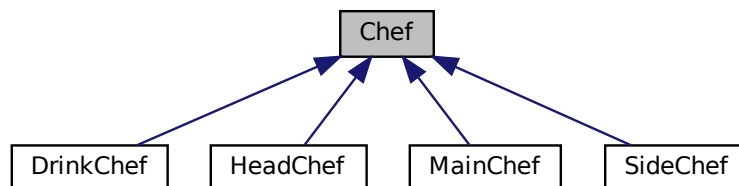
### Additional Inherited Members

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

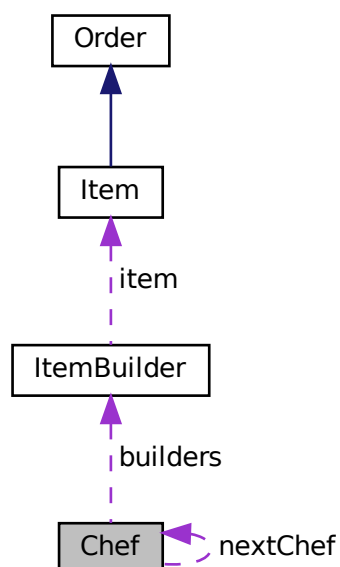
- [BurgerBuilder.h](#)
- [BurgerBuilder.cpp](#)

## 4.5 Chef Class Reference

Inheritance diagram for Chef:



Collaboration diagram for Chef:



## Public Types

- enum **itemBuilders** {  
    **steak** , **burger** , **fish** , **chips** ,  
    **salad** , **beer** , **water** , **soda** }

## Public Member Functions

- [Chef](#) ()  
    *Constructor of the [Chef](#) class.*
- virtual [~Chef](#) ()  
    *Destructor of the [Chef](#) class.*
- virtual void **preparePart** (string order, [Order](#) \*o)=0
- int [request](#) (string &order)  
    *member function of the [Chef](#) class, implementing Adapter functionality*

## Public Attributes

- [Chef](#) \* [nextChef](#)  
    *Pointer to next chef in chain of responsibility.*

## Protected Attributes

- [ItemBuilder](#) \* [builders](#) [8]  
    *Array of [ItemBuilder](#) pointers.*

## 4.5.1 Constructor & Destructor Documentation

### 4.5.1.1 Chef()

```
Chef::Chef ( )
```

Constructor of the [Chef](#) class.

#### Authors

Aidan Chapman (u22738917)

#### 4.5.1.2 ~Chef()

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

Destructor of the [Chef](#) class.

##### Author

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

### 4.5.2 Member Function Documentation

#### 4.5.2.1 request()

```
int Chef::request (
    string & order )
```

member function of the [Chef](#) class, implementing Adapter functionality

##### Parameters

<i>order</i>	: string&
--------------	-----------

##### Authors

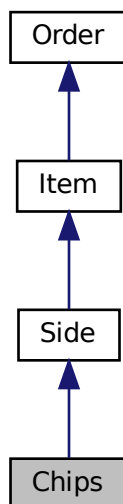
Aidan Chapman (u22738917)

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

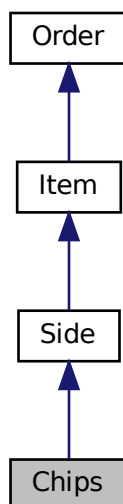
- [Chef.h](#)
- [Chef.cpp](#)

## 4.6 Chips Class Reference

Inheritance diagram for Chips:



Collaboration diagram for Chips:





## Public Member Functions

- [Chips](#) ()  
*Chips Constructor.*
- [~Chips](#) ()  
*Chips Destructor.*

## Public Attributes

- bool [washedPotatoes](#) = false  
*Whether the potatoes have been washed.*
- bool [cutPotatoes](#) = false  
*Whether the potatoes have been cut.*
- bool [friedPotatoes](#) = false  
*Whether the potatoes have been fried.*
- bool [seasonedChips](#) = false  
*Whether the chips have been seasoned.*

## Additional Inherited Members

### 4.6.1 Constructor & Destructor Documentation

#### 4.6.1.1 Chips()

`Chips::Chips ( )`

[Chips](#) Constructor.

##### Authors

Aidan Chapman (u22738917)

#### 4.6.1.2 ~Chips()

`Chips::~~Chips ( )`

[Chips](#) Destructor.

##### Authors

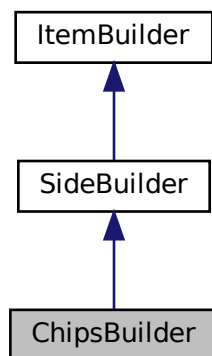
Aidan Chapman (u22738917)

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

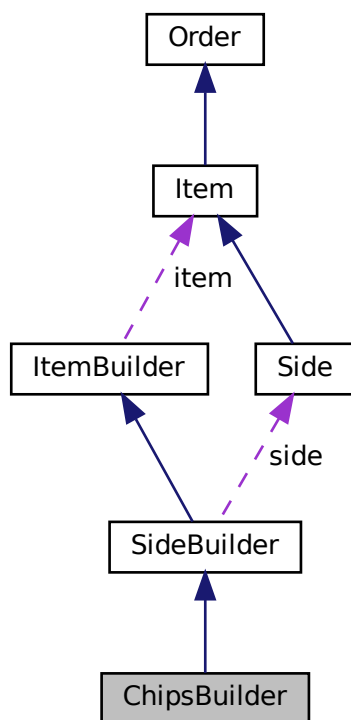
- [Chips.h](#)
- [Chips.cpp](#)

## 4.7 ChipsBuilder Class Reference

Inheritance diagram for ChipsBuilder:



Collaboration diagram for ChipsBuilder:



## Public Member Functions

- [ChipsBuilder](#) ()  
*Construct a new [Chips](#) Builder:: [Chips](#) Builder object.*
- [~ChipsBuilder](#) ()  
*Destroy the [Chips](#) Builder:: [Chips](#) Builder object.*
- void [washVegetables](#) ()  
*Washes the vegetables for the chips.*
- void [chopVegetables](#) ()  
*Chops the vegetables for the chips.*
- void [assembleSide](#) ()  
*Assembles the side dish.*
- void [plateSide](#) ()  
*Plates the side dish.*
- void [washPotato](#) ()  
*Washes the potatoes for the chips.*
- void [cutPotato](#) ()  
*Cuts the potatoes for the chips.*
- void [fryPotato](#) ()  
*Fries the potatoes for the chips.*
- void [seasonChips](#) ()  
*Seasons the chips.*

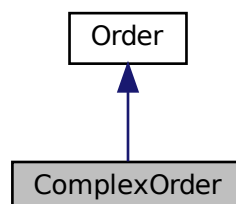
## Additional Inherited Members

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

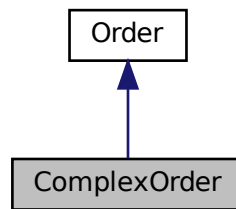
- [ChipsBuilder.h](#)
- [ChipsBuilder.cpp](#)

## 4.8 ComplexOrder Class Reference

Inheritance diagram for ComplexOrder:



Collaboration diagram for ComplexOrder:



## Public Member Functions

- `ComplexOrder (Waiter *waiter)`  
*Constructor for the `ComplexOrder` class.*
- `~ComplexOrder ()`  
*Destructor for the `ComplexOrder` class.*
- `void addToOrder (Order *item)`  
*Adds an `Order` object to the `ComplexOrder`.*
- `void appendToOrder (Order *order)`  
*Appends an `Order` object to the end of the `ComplexOrder`.*
- `float calculatePrice ()`  
*Calculates the total price of the `ComplexOrder`.*

## 4.8.1 Constructor & Destructor Documentation

### 4.8.1.1 ComplexOrder()

```
ComplexOrder::ComplexOrder (  
    Waiter * waiter )
```

Constructor for the `ComplexOrder` class.

This constructor takes in a `Waiter` pointer to initialise the `Order`'s `Waiter` member variable.

#### Parameters

<code>waiter</code>	A <code>Waiter</code> pointer representing the waiter who took the order.
---------------------	---

#### 4.8.1.2 ~ComplexOrder()

```
ComplexOrder::~~ComplexOrder ( )
```

Destructor for the [ComplexOrder](#) class.

This destructor frees up memory allocated to the [ComplexOrder](#) object.

### 4.8.2 Member Function Documentation

#### 4.8.2.1 addToOrder()

```
void ComplexOrder::addToOrder (
    Order * item ) [virtual]
```

Adds an [Order](#) object to the [ComplexOrder](#).

This method adds an [Order](#) object to the [ComplexOrder](#). If the [ComplexOrder](#) already contains an [Order](#) object, it creates a new [ComplexOrder](#) object and adds the existing [Order](#) object and the new [Order](#) object to it.

##### Parameters

<i>item</i>	An <a href="#">Order</a> pointer representing the <a href="#">Order</a> object to be added to the <a href="#">ComplexOrder</a> .
-------------	--

Reimplemented from [Order](#).

#### 4.8.2.2 appendToOrder()

```
void ComplexOrder::appendToOrder (
    Order * orderItem ) [virtual]
```

Appends an [Order](#) object to the end of the [ComplexOrder](#).

This method appends an [Order](#) object to the end of the [ComplexOrder](#). If the [ComplexOrder](#) already contains an [Order](#) object, it recursively calls itself on the nextOrderItem pointer until it reaches the end of the [ComplexOrder](#).

##### Parameters

<i>orderItem</i>	An <a href="#">Order</a> pointer representing the <a href="#">Order</a> object to be appended to the <a href="#">ComplexOrder</a> .
------------------	---

Reimplemented from [Order](#).

#### 4.8.2.3 calculatePrice()

```
float ComplexOrder::calculatePrice ( ) [virtual]
```

Calculates the total price of the [ComplexOrder](#).

This method calculates the total price of the [ComplexOrder](#) by recursively calling itself on the nextOrderItem pointer until it reaches the end of the [ComplexOrder](#). It then adds the price of the current [Order](#) object to the total price.

##### Returns

A float representing the total price of the [ComplexOrder](#).

Reimplemented from [Order](#).

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

- [ComplexOrder.h](#)
- [ComplexOrder.cpp](#)

## 4.9 Customer Class Reference

### Public Member Functions

- [Customer](#) (int timestamp)  
*The constructor for the [Customer](#) class.*
- [~Customer](#) ()  
*The destructor for the [Customer](#) class.*
- void [acceptWaiter](#) ([Waiter](#) \*waiter)  
*The waiter member variable setter for the [Customer](#) class.*
- [Order](#) \* [getOrder](#) ()  
*The order member variable getter for the [Customer](#) class.*
- string [getOrderRequest](#) ()  
*A member function that generates a random order using [Interface](#)'s generateOrderString() function.*
- void [changeRating](#) ([Rating](#) \*rating)  
*The satisfaction member variable setter for the [Customer](#) class, also deletes the previous rating if it exists.*
- void [receiveOrder](#) ([Order](#) \*order)  
*Sets the order member variable to the passed in value.*
- float [calculatePayment](#) ()  
*A function used to calculate what the customer pays for their meal, including the tip based on how happy they were with the service.*
- [Table](#) \* [getTable](#) ()  
*The table member variable getter for the [Customer](#) class.*
- [Waiter](#) \* [getWaiter](#) ()  
*The waiter member variable getter the [Customer](#) class.*
- int [getTimestamp](#) ()  
*The timestamp member variable getter for the [Customer](#) class.*

### 4.9.1 Constructor & Destructor Documentation

#### 4.9.1.1 Customer()

```
Customer::Customer (
    int timestamp )
```

The constructor for the [Customer](#) class.

## Parameters

<i>timestamp</i>	an int
------------------	--------

## Authors

Aidan Chapman (u22738917)

#### 4.9.1.2 ~Customer()

```
Customer::~~Customer ( )
```

The destructor for the [Customer](#) class.

## Authors

Aidan Chapman (u22738917), Douglas Porter (u21797545), Kabelo Chuene(u14046492)

### 4.9.2 Member Function Documentation

#### 4.9.2.1 acceptWaiter()

```
void Customer::acceptWaiter (
    Waiter * waiter )
```

The waiter member variable setter for the [Customer](#) class.

## Parameters

<i>waiter</i>	a <a href="#">Waiter</a> pointer
---------------	----------------------------------

## Authors

Aidan Chapman (u22738917), Douglas Porter (u21797545), Kabelo Chuene(u14046492)

#### 4.9.2.2 calculatePayment()

```
float Customer::calculatePayment ( )
```

A function used to calculate what the customer pays for their meal, including the tip based on how happy they were with the service.

**Returns**

a float

**Authors**

Aidan Chapman (u22738917)

**4.9.2.3 changeRating()**

```
void Customer::changeRating (
    Rating * rating )
```

The satisfaction member variable setter for the [Customer](#) class, also deletes the previous rating if it exists.

**Parameters**

<i>rating</i>	a <a href="#">Rating</a> pointer
---------------	----------------------------------

**Authors**

Aidan Chapman (u22738917), Douglas Porter (u21797545)

**4.9.2.4 getOrder()**

```
Order * Customer::getOrder ( )
```

The order member variable getter for the [Customer](#) class.

**Returns**

an [Order](#) pointer

**Authors**

Douglas Porter (u21797545)



#### 4.9.2.5 getOrderRequest()

```
string Customer::getOrderRequest ( )
```

A member function that generates a random order using [Interface](#)'s generateOrderString() function.

##### Returns

a string

##### Authors

Aidan Chapman (u22738917)

#### 4.9.2.6 getTable()

```
Table * Customer::getTable ( )
```

The table member variable getter for the [Customer](#) class.

##### Returns

a [Table](#) pointer

##### Authors

Douglas Porter (u21797545)

#### 4.9.2.7 getTimestamp()

```
int Customer::getTimestamp ( )
```

The timestamp member variable getter for the [Customer](#) class.

##### Returns

an int

##### Authors

Douglas Porter (u21797545)

#### 4.9.2.8 getWaiter()

```
Waiter * Customer::getWaiter ( )
```

The waiter member variable getter the [Customer](#) class.

##### Returns

a waiter pointer

##### Authors

Aidan Chapman (u22738917)

#### 4.9.2.9 receiveOrder()

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

Sets the order member variable to the passed in value.

##### Parameters

<i>order</i>	an <a href="#">Order</a> pointer
--------------	----------------------------------

##### Authors

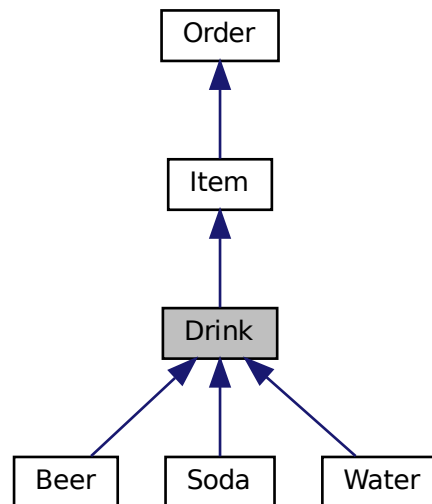
Aidan Chapman (u22738917), Douglas Porter (u21797545)

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

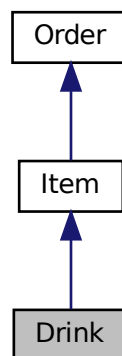
- [Customer.h](#)
- [Customer.cpp](#)

## 4.10 Drink Class Reference

Inheritance diagram for Drink:



Collaboration diagram for Drink:



### Public Member Functions

- `Drink` (float `price`)  
Construct a new `Drink::Drink` object.
- `~Drink` ()  
Destroy the `Drink::Drink` object.

## Public Attributes

- bool [gotGlass](#) = false  
*Whether the glass has been obtained.*
- bool [pouredDrink](#) = false  
*Whether the drink has been poured.*
- bool [assembledDrink](#) = false  
*Whether the drink has been assembled.*

## Additional Inherited Members

### 4.10.1 Constructor & Destructor Documentation

#### 4.10.1.1 Drink()

```
Drink::Drink (  
    float price )
```

Construct a new [Drink:: Drink](#) object.

##### Parameters

<i>price</i>	
--------------	--

##### Returns

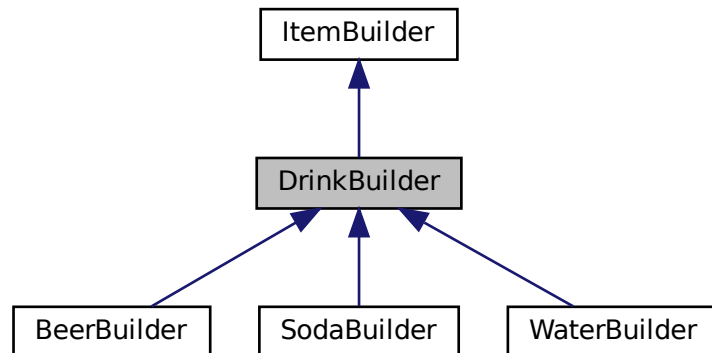
\* Constructor

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

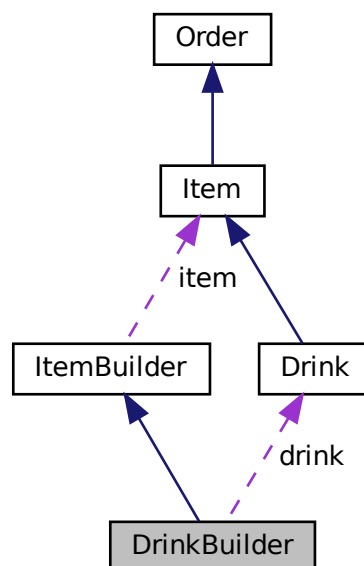
- [Drink.h](#)
- [Drink.cpp](#)

## 4.11 DrinkBuilder Class Reference

Inheritance diagram for DrinkBuilder:



Collaboration diagram for DrinkBuilder:



### Public Member Functions

- virtual void [prepareIngredients](#) ()

- Prepares the ingredients for the drink.*
  - virtual void [assembleItem](#) ()  
*Assembles the drink by pouring and assembling it.*
- virtual [Item](#) \* [getItem](#) ()  
*Returns the item that was built.*
- virtual void [getGlass](#) ()=0
- virtual void [pourDrink](#) ()=0
- virtual void [assembleDrink](#) ()=0

## Protected Attributes

- [Drink](#) \* [drink](#)  
*The drink that is being built.*

## 4.11.1 Member Function Documentation

### 4.11.1.1 getItem()

[Item](#) \* [DrinkBuilder::getItem](#) ( ) [virtual]

Returns the item that was built.

#### Returns

[Item](#)\* Pointer to the item that was built.

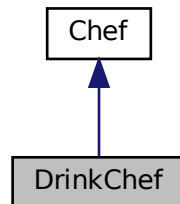
Implements [ItemBuilder](#).

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

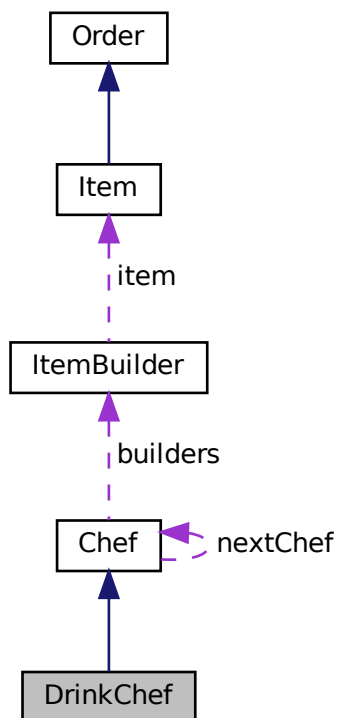
- [DrinkBuilder.h](#)
- [DrinkBuilder.cpp](#)

## 4.12 DrinkChef Class Reference

Inheritance diagram for DrinkChef:



Collaboration diagram for DrinkChef:



### Public Member Functions

- [DrinkChef](#) ()

Constructor of the [DrinkChef](#) class.

- [~DrinkChef](#) ()

Destructor of the [DrinkChef](#) class.

- void [preparePart](#) (string order, [Order](#) \*o)

Member function of the [DrinkChef](#) class, implementing Chain of Responsibility functionality.

## Additional Inherited Members

### 4.12.1 Constructor & Destructor Documentation

#### 4.12.1.1 DrinkChef()

```
DrinkChef::DrinkChef ( )
```

Constructor of the [DrinkChef](#) class.

##### Authors

Aidan Chapman (u22738917)

#### 4.12.1.2 ~DrinkChef()

```
DrinkChef::~~DrinkChef ( )
```

Destructor of the [DrinkChef](#) class.

##### Authors

Aidan Chapman (u22738917)

### 4.12.2 Member Function Documentation

#### 4.12.2.1 preparePart()

```
void DrinkChef::preparePart (
    string order,
    Order * o ) [virtual]
```

Member function of the [DrinkChef](#) class, implementing Chain of Responsibility functionality.



## Parameters

<i>order</i>	a string
<i>o</i>	an <a href="#">Order</a> pointer

## Authors

Aidan Chapman (u22738917)

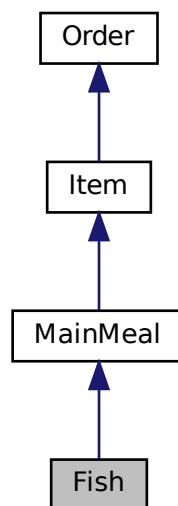
Implements [Chef](#).

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

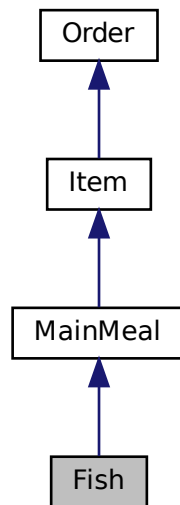
- [DrinkChef.h](#)
- [DrinkChef.cpp](#)

## 4.13 Fish Class Reference

Inheritance diagram for Fish:



Collaboration diagram for Fish:



## Public Member Functions

- [Fish \(\)](#)  
*Fish Constructor.*
- [~Fish \(\)](#)  
*Fish Destructor.*

## Public Attributes

- bool [descaledFish](#) = false  
*Whether the fish has been descaled.*
- bool [seasonedFish](#) = false  
*Whether the fish has been seasoned.*
- bool [cookedFish](#) = false  
*Whether the fish has been cooked.*
- bool [platedFish](#) = false  
*Whether the fish has been plated.*

## Additional Inherited Members

### 4.13.1 Constructor & Destructor Documentation

#### 4.13.1.1 Fish()

```
Fish::Fish ( )
```

[Fish](#) Constructor.

##### Authors

Aidan Chapman (u22738917)

#### 4.13.1.2 ~Fish()

```
Fish::~~Fish ( )
```

[Fish](#) Destructor.

##### Authors

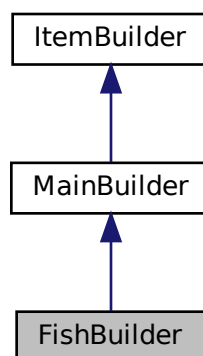
Aidan Chapman (u22738917)

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

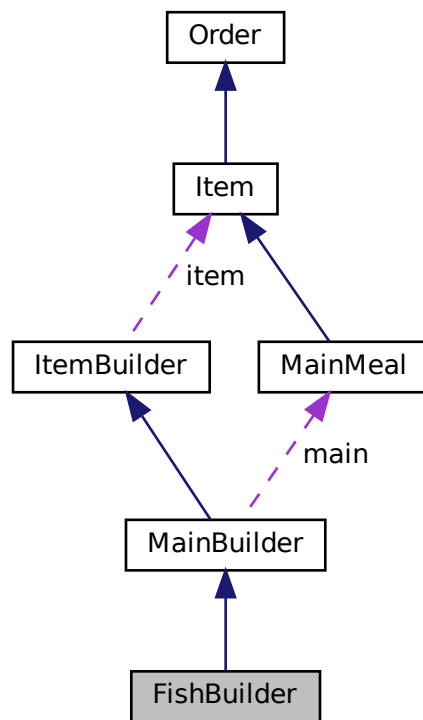
- [Fish.h](#)
- [Fish.cpp](#)

## 4.14 FishBuilder Class Reference

Inheritance diagram for FishBuilder:



Collaboration diagram for FishBuilder:



## Public Member Functions

- **FishBuilder** ()  
Constructs a new *FishBuilder* object.
- **~FishBuilder** ()  
Destroys the *FishBuilder* object.
- void **prepareMeat** ()  
Prepares the fish by descaling it and marking it as prepared.
- void **seasonMeat** ()  
Seasons the fish and marks it as seasoned.
- void **cookMeat** ()  
Cooks the fish and marks it as cooked.
- void **plateMain** ()  
Plates the fish and marks it as plated.
- void **descaleFish** ()  
Descals the fish and marks it as descaled.
- void **seasonFish** ()  
Seasons the fish and marks it as seasoned.
- void **cookFish** ()  
Cooks the fish and marks it as cooked.
- void **plateFish** ()  
Plates the fish and marks it as plated.

## Additional Inherited Members

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

- [FishBuilder.h](#)
- [FishBuilder.cpp](#)

## 4.15 Floor Class Reference

### Public Member Functions

- [Floor](#) (int numTables)  
*Constructor of the [Floor](#) class.*
- [~Floor](#) ()  
*Destructor of the [Floor](#) class.*
- void [seatCustomer](#) ([Customer](#) \*customer)  
*A member function of the [Floor](#) class, finding an available table for the customer to sit at.*
- void [setRestaurant](#) ([Restaurant](#) \*restaurant)  
*Member function of the [Floor](#) class, allowing for the setting of the [Restaurant](#) in the context of the Mediator pattern.*
- int [getNumTables](#) ()  
*A member function of the [Floor](#) class, returning the number of tables.*
- [Table](#) \* [getTable](#) ([Customer](#) \*customer)  
*A member function of the [Floor](#) class, returns the table that a customer is at. Returns nullptr if no customer is found at any table.*

### 4.15.1 Constructor & Destructor Documentation

#### 4.15.1.1 Floor()

```
Floor::Floor (  
    int numTables )
```

Constructor of the [Floor](#) class.

##### Parameters

<i>numTables</i>	an integer
------------------	------------

##### Authors

Aidan Chapman (u22738917)

#### 4.15.1.2 ~Floor()

```
Floor::~~Floor ( )
```

Destructor of the [Floor](#) class.

##### Authors

Aidan Chapman (u22738917)

### 4.15.2 Member Function Documentation

#### 4.15.2.1 getNumTables()

```
int Floor::getNumTables ( )
```

A member function of the [Floor](#) class, returning the number of tables.

##### Returns

an integer

##### Authors

Aidan Chapman (u22738917)

#### 4.15.2.2 getTable()

```
Table * Floor::getTable (
    Customer * customer )
```

A member function of the [Floor](#) class, returns the table that a customer is at. Returns nullptr if no customer is found at any table.

##### Parameters

<i>customer</i>	a <a href="#">Customer</a> pointer
-----------------	------------------------------------

##### Returns

a [Table](#) pointer

## Authors

Aidan Chapman (u22738917)

### 4.15.2.3 seatCustomer()

```
void Floor::seatCustomer (
    Customer * customer )
```

A member function of the [Floor](#) class, finding an available table for the customer to sit at.

## Parameters

<i>customer</i>	a <a href="#">Customer</a> pointer
-----------------	------------------------------------

## Authors

Aidan Chapman (u22738917)

### 4.15.2.4 setRestaurant()

```
void Floor::setRestaurant (
    Restaurant * restaurant )
```

Member function of the [Floor](#) class, allowing for the setting of the [Restaurant](#) in the context of the Mediator pattern.

## Parameters

<i>restaurant</i>	a <a href="#">Restaurant</a> pointer
-------------------	--------------------------------------

## Authors

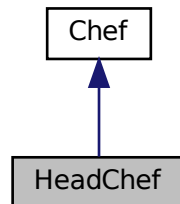
Aidan Chapman (u22738917)

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

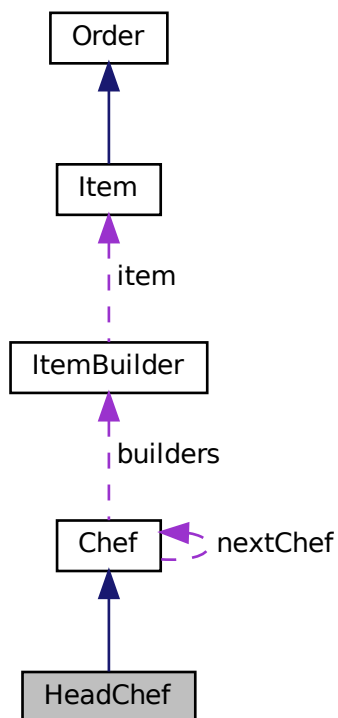
- [Floor.h](#)
- [Floor.cpp](#)

## 4.16 HeadChef Class Reference

Inheritance diagram for HeadChef:



Collaboration diagram for HeadChef:



### Public Member Functions

- [HeadChef](#) ()



- Constructor of the [HeadChef](#) class.*
  - [~HeadChef](#) ()
- Destructor of the [HeadChef](#) class.*
  - void **preparePart** (string order, [Order](#) \*o)

## Additional Inherited Members

### 4.16.1 Constructor & Destructor Documentation

#### 4.16.1.1 HeadChef()

`HeadChef::HeadChef ( )`

Constructor of the [HeadChef](#) class.

##### Authors

Aidan Chapman (u22738917)

#### 4.16.1.2 ~HeadChef()

`HeadChef::~~HeadChef ( )`

Destructor of the [HeadChef](#) class.

##### Authors

Aidan Chapman (u22738917)

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

- [HeadChef.h](#)
- [HeadChef.cpp](#)

## 4.17 Interface Class Reference

### Public Member Functions

- [Interface](#) ()
  - Constructor of the [Interface](#) class.*
- [~Interface](#) ()
  - Destructor of the [Interface](#) class.*
- int [generateNumberOfCustomers](#) ()
  - Member function of the [Interface](#) class, returns a random number of customers between 1 and 10.*
- float [runCustomer](#) ()
  - Member function of the [Interface](#) class, returns the amount that the [Customer](#) paid.*
- [Restaurant](#) \* **getRestaurant** ()

## Static Public Member Functions

- static int [getCurrentUnixTime](#) ()  
*Member function of the [Interface](#) class, returns the current unix time. Dependent on system clock.*
- static string [generateOrderString](#) ()  
*Member function of the [Interface](#) class, returns a randomOrder string to be adapted by the chef. The string is composed of at least 1 main meal. There are a maximum of 6 mains, sides and drinks per order.*

## 4.17.1 Constructor & Destructor Documentation

### 4.17.1.1 Interface()

```
Interface::Interface ( )
```

Constructor of the [Interface](#) class.

#### Authors

Aidan Chapman (u22738917), Douglas Porter (u21797545),Kabelo CHuene(14046492)

### 4.17.1.2 ~Interface()

```
Interface::~~Interface ( )
```

Destructor of the [Interface](#) class.

#### Authors

Aidan Chapman (u22738917), Douglas Porter (u21797545),Kabelo CHuene(14046492)

## 4.17.2 Member Function Documentation

### 4.17.2.1 generateNumberOfCustomers()

```
int Interface::generateNumberOfCustomers ( )
```

Member function of the [Interface](#) class, returns a random number of customers between 1 and 10.

#### Returns

an int

#### Authors

Aidan Chapman (u22738917)

#### 4.17.2.2 generateOrderString()

```
string Interface::generateOrderString ( ) [static]
```

Member function of the [Interface](#) class, returns a randomOrder string to be adapted by the chef. The string is composed of at least 1 main meal. There are a maximum of 6 mains, sides and drinks per order.

##### Returns

an string

##### Authors

Aidan Chapman (u22738917),Kabelo CHuene(14046492)

#### 4.17.2.3 getCurrentUnixTime()

```
int Interface::getCurrentUnixTime ( ) [static]
```

Member function of the [Interface](#) class, returns the current unix time. Dependent on system clock.

##### Returns

an int

##### Authors

Aidan Chapman (u22738917)

#### 4.17.2.4 runCustomer()

```
float Interface::runCustomer ( )
```

Member function of the [Interface](#) class, returns the amount that the [Customer](#) paid.

##### Returns

a float

##### Authors

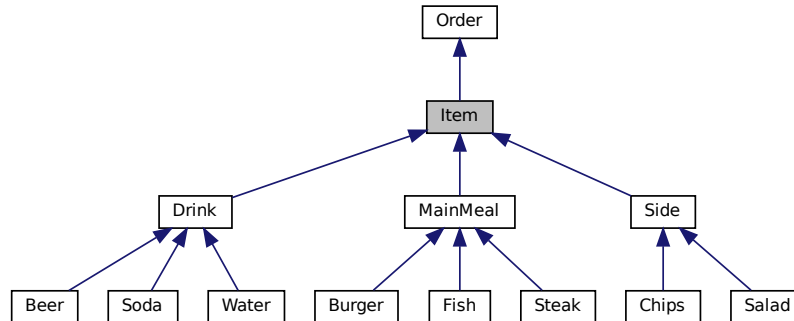
Aidan Chapman (u22738917), Douglas Porter (u21797545),Kabelo CHuene(14046492)

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

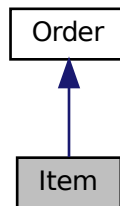
- [Interface.h](#)
- [Interface.cpp](#)

## 4.18 Item Class Reference

Inheritance diagram for Item:



Collaboration diagram for Item:



### Public Member Functions

- `Item` (float `price`)  
*Constructor of the `Item` class. Sets the float variable with the price input.*
- `~Item` ()  
*Destructor of the `Item` class. Does nothing.*
- float `calculatePrice` ()  
*Calculates the price of the item.*

### Protected Attributes

- float `price`  
*The price of the item.*

## 4.18.1 Constructor & Destructor Documentation

### 4.18.1.1 Item()

```
Item::Item (
    float price )
```

Constructor of the [Item](#) class. Sets the float variable with the price input.

#### Parameters

<i>price</i>	a float
--------------	---------

#### Authors

Aidan Chapman (u22738917)

## 4.18.2 Member Function Documentation

### 4.18.2.1 calculatePrice()

```
float Item::calculatePrice ( ) [virtual]
```

Calculates the price of the item.

#### Returns

float the price of the item

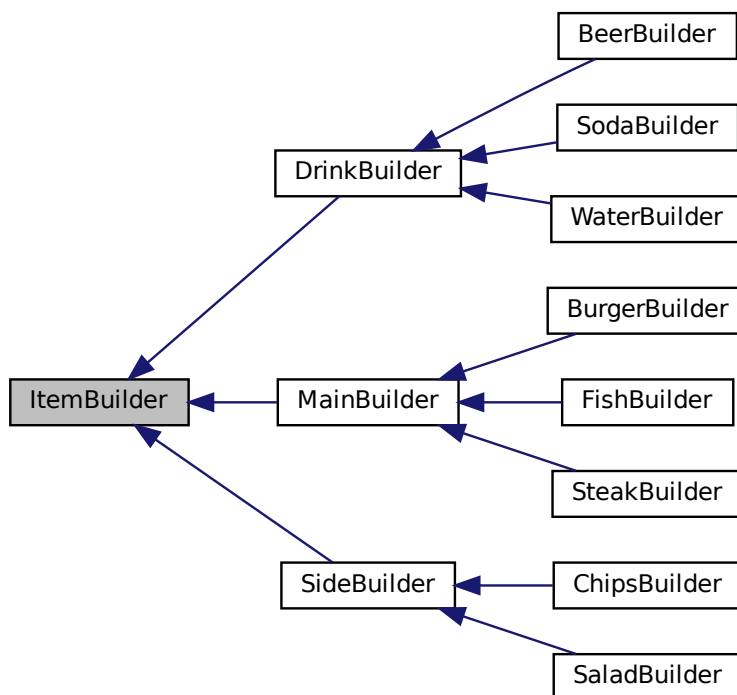
Reimplemented from [Order](#).

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

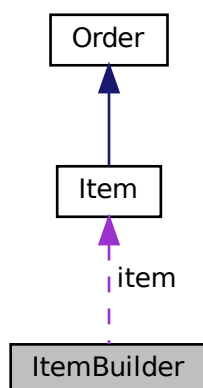
- [Item.h](#)
- [Item.cpp](#)

## 4.19 ItemBuilder Class Reference

Inheritance diagram for ItemBuilder:



Collaboration diagram for ItemBuilder:



## Public Member Functions

- virtual void **prepareIngredients** ()=0
- virtual void **assembleItem** ()=0
- virtual [Item](#) \* **getItem** ()=0

## Protected Attributes

- [Item](#) \* **item**  
*[Item](#) object.*

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

- [ItemBuilder.h](#)

## 4.20 Kitchen Class Reference

### Public Member Functions

- [Kitchen](#) ()  
*Constructor of the [Kitchen](#) class.*
- void [receiveOrder](#) ([OrderContainer](#) \*orderContainer)  
*A member function of the [Kitchen](#) class, adds order specified in parameter to the orderQueue member variable.*
- void [makeNextOrder](#) ()  
*Member function of the [Kitchen](#) class, pops the next order from the queue and sends it to the chefs for preparation.*
- void [setRestaurant](#) ([Restaurant](#) \*restaurant)  
*Member variable of the [Kitchen](#) class, setting the [Restaurant](#) member variable for use with the mediator pattern.*

### 4.20.1 Constructor & Destructor Documentation

#### 4.20.1.1 Kitchen()

```
Kitchen::Kitchen ( )
```

Constructor of the [Kitchen](#) class.

Destructor of the [Kitchen](#) class.

#### Authors

Aidan Chapman (u22738917)

### 4.20.2 Member Function Documentation

#### 4.20.2.1 makeNextOrder()

```
void Kitchen::makeNextOrder ( )
```

Member function of the [Kitchen](#) class, pops the next order from the queue and sends it to the chefs for preparation.

##### Authors

Aidan Chapman (u22738917)

#### 4.20.2.2 receiveOrder()

```
void Kitchen::receiveOrder (
    OrderContainer * orderContainer )
```

A member function of the [Kitchen](#) class, adds order specified in parameter to the orderQueue member variable.

##### Parameters

<i>orderContainer</i>	an <a href="#">OrderContainer</a> pointer
-----------------------	---

##### Authors

Aidan Chapman (u22738917)

#### 4.20.2.3 setRestaurant()

```
void Kitchen::setRestaurant (
    Restaurant * restaurant )
```

Member variable of the [Kitchen](#) class, setting the [Restaurant](#) member variable for use with the mediator pattern.

##### Parameters

<i>restaurant</i>	a <a href="#">Restaurant</a> pointer
-------------------	--------------------------------------

##### Authors

Aidan Chapman (u22738917)

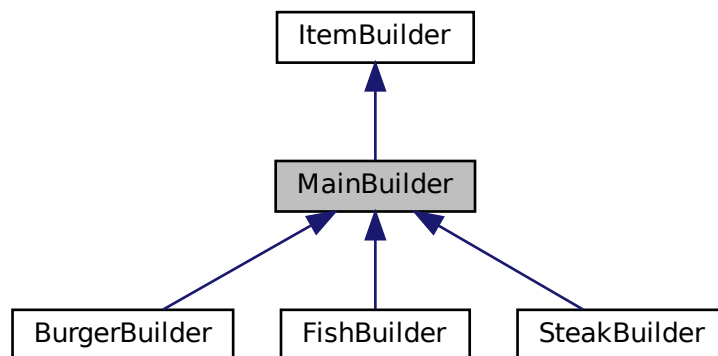
The documentation for this class was generated from the following files:

- [Kitchen.h](#)
- [Kitchen.cpp](#)

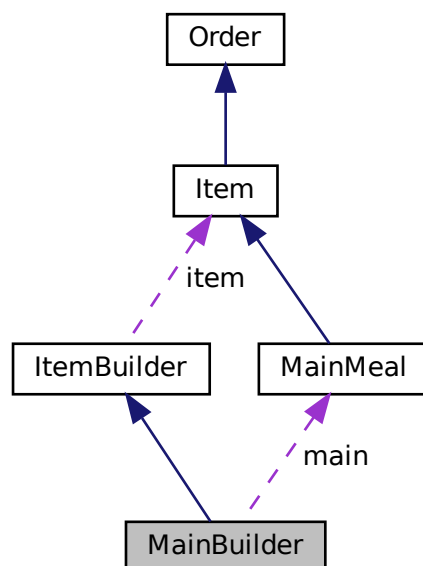


## 4.21 MainBuilder Class Reference

Inheritance diagram for MainBuilder:



Collaboration diagram for MainBuilder:



### Public Member Functions

- virtual void **prepareIngredients** ()

- virtual void [assembleItem](#) ()  
*Assembles the main dish.*
- virtual [Item](#) \* [getItem](#) ()  
*Returns the item that was built.*
- virtual void **prepareMeat** ()=0
- virtual void **seasonMeat** ()=0
- virtual void **cookMeat** ()=0
- virtual void **plateMain** ()=0

## Protected Attributes

- [MainMeal](#) \* [main](#)  
*[MainMeal](#) object.*

## 4.21.1 Member Function Documentation

### 4.21.1.1 [getItem](#)()

[Item](#) \* [MainBuilder::getItem](#) ( ) [virtual]

Returns the item that was built.

#### Returns

[Item](#)\* Pointer to the item that was built

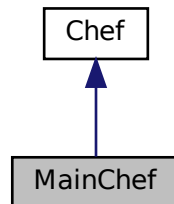
Implements [ItemBuilder](#).

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

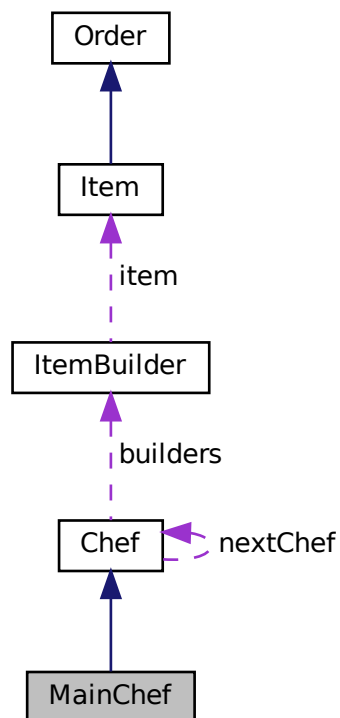
- [MainBuilder.h](#)
- [MainBuilder.cpp](#)

## 4.22 MainChef Class Reference

Inheritance diagram for MainChef:



Collaboration diagram for MainChef:



### Public Member Functions

- [MainChef](#) ()

Constructor of the [MainChef](#) class.

- [~MainChef](#) ( )

Destructor of the [MainChef](#) class.

- void [preparePart](#) (string order, [Order](#) \*o)

The Chain of responsibility [handle\(\)](#) method.

## Additional Inherited Members

### 4.22.1 Constructor & Destructor Documentation

#### 4.22.1.1 MainChef()

```
MainChef::MainChef ( )
```

Constructor of the [MainChef](#) class.

##### Authors

Aidan Chapman (u22738917)

#### 4.22.1.2 ~MainChef()

```
MainChef::~~MainChef ( )
```

Destructor of the [MainChef](#) class.

##### Authors

Aidan Chapman (u22738917)

### 4.22.2 Member Function Documentation

#### 4.22.2.1 preparePart()

```
void MainChef::preparePart (
    string order,
    Order * o ) [virtual]
```

The Chain of responsibility [handle\(\)](#) method.

## Parameters

<i>order</i>	a string
<i>o</i>	an <a href="#">Order</a> pointer

## Authors

Aidan Chapman (u22738917)

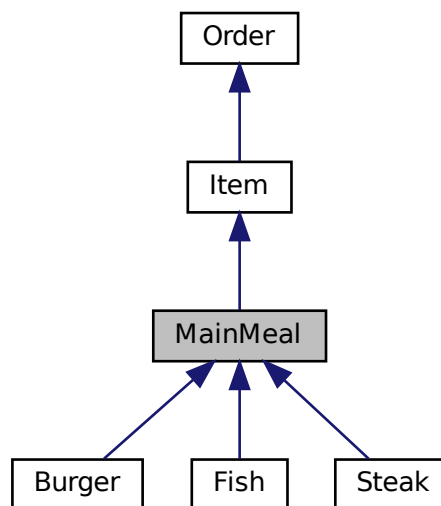
Implements [Chef](#).

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

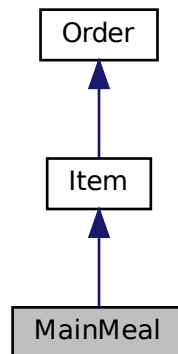
- [MainChef.h](#)
- [MainChef.cpp](#)

## 4.23 MainMeal Class Reference

Inheritance diagram for MainMeal:



Collaboration diagram for MainMeal:



## Public Member Functions

- [MainMeal](#) (float [price](#))  
*Constructor for the [MainMeal](#) class.*
- [~MainMeal](#) ()  
*Destructor for the [MainMeal](#) class.*

## Public Attributes

- bool [preparedMain](#) = false  
*Whether the main has been prepared.*
- bool [seasonedMain](#) = false  
*Whether the main has been seasoned.*
- bool [cookedMain](#) = false  
*Whether the main has been cooked.*
- bool [platedMain](#) = false  
*Whether the main has been plated.*

## Additional Inherited Members

### 4.23.1 Constructor & Destructor Documentation

#### 4.23.1.1 MainMeal()

```
MainMeal::MainMeal (  
    float price )
```

Constructor for the [MainMeal](#) class.

## Parameters

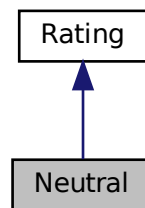
<i>price</i>	The price of the <a href="#">MainMeal</a>
--------------	---

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

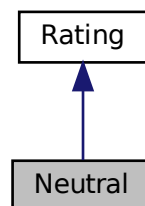
- [MainMeal.h](#)
- [MainMeal.cpp](#)

## 4.24 Neutral Class Reference

Inheritance diagram for Neutral:



Collaboration diagram for Neutral:



### Public Member Functions

- [Neutral](#) ()  
*Constructs a new [Neutral](#) object.*
- [~Neutral](#) ()  
*Destroys the [Neutral](#) object.*

- float [calculateTip](#) ()  
*Function for calculating the tip of the customer.*
- void [changeState](#) ([Customer](#) \*customer)  
*Function for changing the rating of the customer.*
- string [getRating](#) ()  
*Function for getting the rating of the customer.*

## 4.24.1 Member Function Documentation

### 4.24.1.1 [calculateTip\(\)](#)

```
float Neutral::calculateTip ( ) [virtual]
```

Function for calculating the tip of the customer.

#### Returns

float The tip of the customer.

Implements [Rating](#).

### 4.24.1.2 [changeState\(\)](#)

```
void Neutral::changeState (
    Customer * customer ) [virtual]
```

Function for changing the rating of the customer.

#### Parameters

<i>customer</i>	The customer whose rating will be changed.
-----------------	--

Implements [Rating](#).

### 4.24.1.3 [getRating\(\)](#)

```
string Neutral::getRating ( )
```

Function for getting the rating of the customer.



## Returns

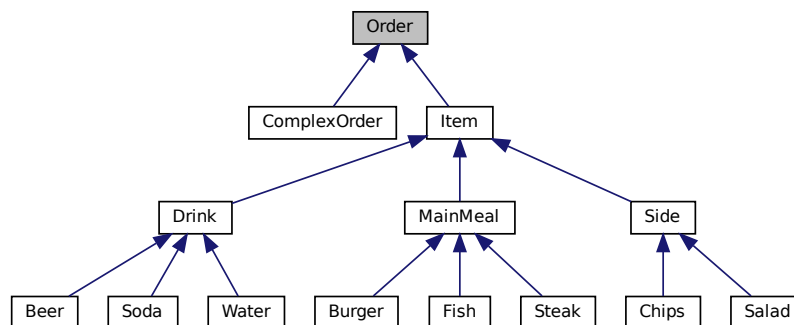
string The rating of the customer.

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

- [Neutral.h](#)
- [Neutral.cpp](#)

## 4.25 Order Class Reference

Inheritance diagram for Order:



### Public Member Functions

- [Order](#) ()  
*Default constructor for the [Order](#) class.*
- [Order](#) ([Waiter](#) \*waiter)  
*Constructor for the [Order](#) class.*
- virtual [~Order](#) ()  
*Destructor for the [Order](#) class.*
- virtual void [addToOrder](#) ([Order](#) \*item)  
*Adds an [Order](#) object to the order.*
- virtual void [appendToOrder](#) ([Order](#) \*order)  
*Appends an [Order](#) object to the order.*
- virtual float [calculatePrice](#) ()  
*Calculates the price of the order.*
- [Waiter](#) \* [getWaiter](#) ()  
*Gets the waiter that is attached to the order.*

### 4.25.1 Constructor & Destructor Documentation

#### 4.25.1.1 Order()

```
Order::Order (
    Waiter * waiter )
```

Constructor for the [Order](#) class.

## Parameters

<i>waiter</i>	The waiter that is attached to the order.
---------------	---

## 4.25.2 Member Function Documentation

### 4.25.2.1 addToOrder()

```
void Order::addToOrder (
    Order * item ) [virtual]
```

Adds an [Order](#) object to the order.

## Parameters

<i>item</i>	An <a href="#">Order</a> pointer representing the <a href="#">Order</a> object to be added to the order.
-------------	--

Reimplemented in [ComplexOrder](#).

### 4.25.2.2 appendToOrder()

```
void Order::appendToOrder (
    Order * order ) [virtual]
```

Appends an [Order](#) object to the order.

## Parameters

<i>order</i>	An <a href="#">Order</a> pointer representing the <a href="#">Order</a> object to be appended to the order.
--------------	---

Reimplemented in [ComplexOrder](#).

### 4.25.2.3 calculatePrice()

```
float Order::calculatePrice ( ) [virtual]
```

Calculates the price of the order.

## Returns

float The price of the order.

Reimplemented in [Item](#), and [ComplexOrder](#).

#### 4.25.2.4 getWaiter()

```
Waiter * Order::getWaiter ( )
```

Gets the waiter that is attached to the order.

##### Returns

Waiter\* A pointer to the waiter that is attached to the order.

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

- [Order.h](#)
- [Order.cpp](#)

## 4.26 OrderContainer Class Reference

### Public Member Functions

- [OrderContainer](#) (string o, [Order](#) \*order)  
*Constructs an [OrderContainer](#) object with the given string and [Order](#) pointer.*
- [~OrderContainer](#) ()  
*Destructor for the [OrderContainer](#) class.*
- [Order](#) \* [getOrder](#) ()  
*Returns the [Order](#) pointer stored in the container.*
- string [getRequestedOrder](#) ()  
*Returns the string stored in the container.*

### 4.26.1 Constructor & Destructor Documentation

#### 4.26.1.1 OrderContainer()

```
OrderContainer::OrderContainer (
    string o,
    Order * order )
```

Constructs an [OrderContainer](#) object with the given string and [Order](#) pointer.

##### Parameters

<i>o</i>	The string to be set as the order container's identifier.
<i>order</i>	The <a href="#">Order</a> pointer to be stored in the container.

## 4.26.2 Member Function Documentation

### 4.26.2.1 getOrder()

```
Order * OrderContainer::getOrder ( )
```

Returns the [Order](#) pointer stored in the container.

#### Returns

An [Order](#) pointer representing the [Order](#) object stored in the container.

### 4.26.2.2 getRequestedOrder()

```
string OrderContainer::getRequestedOrder ( )
```

Returns the string stored in the container.

#### Returns

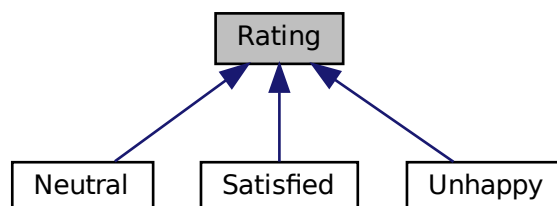
A string representing the string stored in the container.

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

- [OrderContainer.h](#)
- [OrderContainer.cpp](#)

## 4.27 Rating Class Reference

Inheritance diagram for Rating:



## Public Member Functions

- [Rating](#) ()  
*Default constructor for the [Rating](#) class.*
- virtual [~Rating](#) ()  
*Destructor for the [Rating](#) class.*
- virtual float **calculateTip** ()=0
- virtual void **changeState** ([Customer](#) \*customer)=0

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

- [Rating.h](#)
- [Rating.cpp](#)

## 4.28 Restaurant Class Reference

### Public Member Functions

- [Restaurant](#) (int numTables)  
*Constructor of the [Restaurant](#) class.*
- [~Restaurant](#) ()  
*Destructor of the [Restaurant](#) class.*
- void [seatCustomer](#) ([Customer](#) \*customer)  
*implements the functionality to seat a customer, link a [Waiter](#) observer to the customer and take the customer's order*
- void [placeOrder](#) ([OrderContainer](#) \*orderContainer)  
*A method used to send the order to the kitchen.*
- void **makeNextOrder** ()
- void [initialise](#) ()  
*A function to be called directly after the constructor for [Restaurant](#) has been called in order to link all member variables properly.*
- void [cleanUp](#) ([Customer](#) \*customer)  
*A function to be called when the customer leaves to properly delete/clean the customer and their related objects.*

### 4.28.1 Constructor & Destructor Documentation

#### 4.28.1.1 Restaurant()

```
Restaurant::Restaurant (
    int numTables )
```

Constructor of the [Restaurant](#) class.

#### Parameters

<i>numTables</i>	an int
------------------	--------

**Authors**

Aidan Chapman (u22738917)

**4.28.1.2 ~Restaurant()**

```
Restaurant::~~Restaurant ( )
```

Destructor of the [Restaurant](#) class.

**Authors**

Aidan Chapman (u22738917)

**4.28.2 Member Function Documentation****4.28.2.1 cleanUp()**

```
void Restaurant::cleanUp (
    Customer * customer )
```

A function to be called when the customer leaves to properly delete/clean the customer and their related objects.

**Parameters**

<i>customer</i>	a <a href="#">Customer</a> pointer
-----------------	------------------------------------

**Authors**

Aidan Chapman (u22738917)

**4.28.2.2 initialise()**

```
void Restaurant::initialise ( )
```

A function to be called directly after the constructor for [Restaurant](#) has been called in order to link all member variables properly.

**Authors**

Aidan Chapman (u22738917)

### 4.28.2.3 placeOrder()

```
void Restaurant::placeOrder (
    OrderContainer * order )
```

A method used to send the order to the kitchen.

#### Parameters

<i>orderContainer</i>	an <a href="#">OrderContainer</a> pointer
-----------------------	---

#### Authors

Aidan Chapman (u22738917), Douglas Porter (u21797545)

### 4.28.2.4 seatCustomer()

```
void Restaurant::seatCustomer (
    Customer * customer )
```

implements the functionality to seat a customer, link a [Waiter](#) observer to the customer and take the customer's order

#### Parameters

<i>customer</i>	a <a href="#">Customer</a> pointer
-----------------	------------------------------------

#### Authors

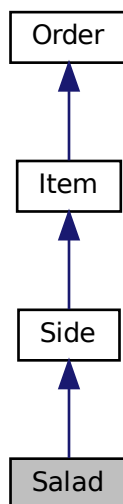
Aidan Chapman (u22738917)

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

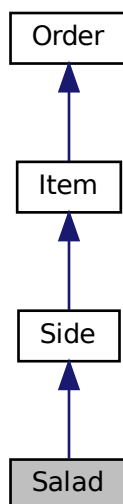
- [Restaurant.h](#)
- [Restaurant.cpp](#)

## 4.29 Salad Class Reference

Inheritance diagram for Salad:



Collaboration diagram for Salad:





## Public Member Functions

- [Salad](#) ()  
*Salad* Constructor.
- [~Salad](#) ()  
*Salad* Destructor.

## Public Attributes

- bool [washedLettuce](#) = false  
*Whether the lettuce has been washed.*
- bool [cutLettuce](#) = false  
*Whether the lettuce has been cut.*
- bool [washedTomato](#) = false  
*Whether the tomato has been washed.*
- bool [cutTomato](#) = false  
*Whether the tomato has been cut.*
- bool [washedCucumber](#) = false  
*Whether the cucumber has been washed.*
- bool [cutCucumber](#) = false  
*Whether the cucumber has been cut.*
- bool [cutFeta](#) = false  
*Whether the feta has been cut.*
- bool [assembledSalad](#) = false  
*Whether the salad has been assembled.*
- bool [platedSalad](#) = false  
*Whether the salad has been plated.*

## Additional Inherited Members

### 4.29.1 Constructor & Destructor Documentation

#### 4.29.1.1 Salad()

```
Salad::Salad ( )
```

[Salad](#) Constructor.

#### Authors

Aidan Chapman (u22738917)

#### 4.29.1.2 ~Salad()

```
Salad::~~Salad ( )
```

[Salad](#) Destructor.

#### Authors

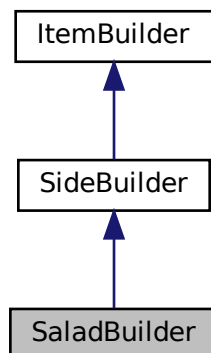
Aidan Chapman (u22738917)

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

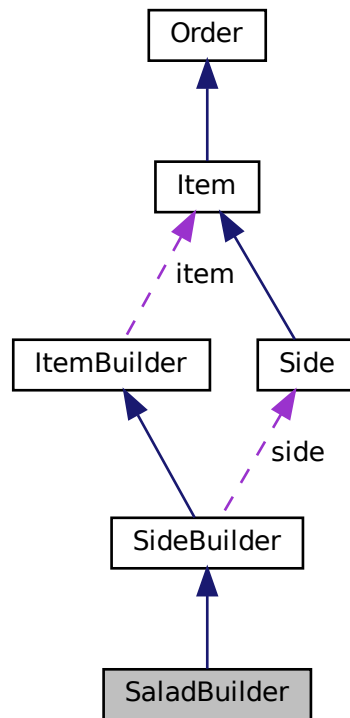
- [Salad.h](#)
- [Salad.cpp](#)

## 4.30 SaladBuilder Class Reference

Inheritance diagram for SaladBuilder:



Collaboration diagram for SaladBuilder:



## Public Member Functions

- [SaladBuilder](#) ()  
Construct a new *Salad Builder*:: *Salad Builder* object.
- [~SaladBuilder](#) ()  
Destroy the *Salad Builder*:: *Salad Builder* object.
- void [washVegetables](#) ()  
Wash the vegetables.
- void [chopVegetables](#) ()  
Chop the vegetables.
- void [assembleSide](#) ()  
Assemble the side.
- void [plateSide](#) ()  
Plate the side.
- void [washLettuce](#) ()  
Wash the lettuce.
- void [cutLettuce](#) ()  
Cut the lettuce.
- void [washTomato](#) ()  
Wash the tomato.
- void [cutTomato](#) ()

- Cut the tomato.*
  - void [washCucumber](#) ()
- Wash the cucumber.*
  - void [cutCucumber](#) ()
- Cut the cucumber.*
  - void [cutFeta](#) ()
- Cut the feta cheese.*
  - void [assembleSalad](#) ()
- Assemble the salad.*

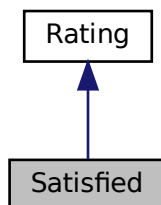
## Additional Inherited Members

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

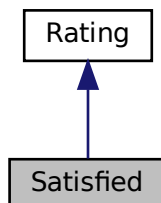
- [SaladBuilder.h](#)
- [SaladBuilder.cpp](#)

## 4.31 Satisfied Class Reference

Inheritance diagram for Satisfied:



Collaboration diagram for Satisfied:



## Public Member Functions

- [Satisfied](#) ()  
*Default constructor for the [Satisfied](#) class.*
- [~Satisfied](#) ()  
*Destructor for the [Satisfied](#) class.*
- float [calculateTip](#) ()  
*Function for calculating the tip of the customer.*
- void [changeState](#) ([Customer](#) \*customer)  
*Function for changing the rating of the customer.*
- string [getRating](#) ()  
*Function for getting the rating of the customer.*

### 4.31.1 Member Function Documentation

#### 4.31.1.1 [calculateTip\(\)](#)

```
float Satisfied::calculateTip ( ) [virtual]
```

Function for calculating the tip of the customer.

##### Returns

float

Implements [Rating](#).

#### 4.31.1.2 [changeState\(\)](#)

```
void Satisfied::changeState (  
    Customer * customer ) [virtual]
```

Function for changing the rating of the customer.

##### Parameters

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

Implements [Rating](#).

#### 4.31.1.3 [getRating\(\)](#)

```
string Satisfied::getRating ( )
```

Function for getting the rating of the customer.

#### Returns

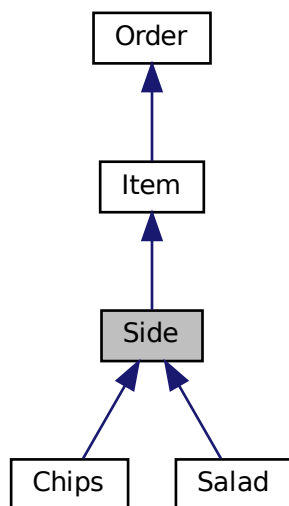
string

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

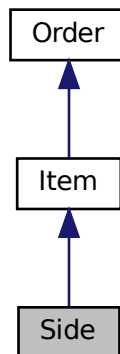
- [Satisfied.h](#)
- [Satisfied.cpp](#)

## 4.32 Side Class Reference

Inheritance diagram for Side:



Collaboration diagram for Side:



## Public Member Functions

- `Side` (float `price`)  
*Constructor for the `Side` class.*
- `~Side` ()  
*Destructor for the `Side` class.*

## Public Attributes

- bool `washedVegetables` = false  
*Whether the vegetables have been washed.*
- bool `cutVegetables` = false  
*Whether the vegetables have been cut.*
- bool `assembledSide` = false  
*Whether the side has been assembled.*
- bool `platedSide` = false  
*Whether the side has been plated.*

## Additional Inherited Members

### 4.32.1 Constructor & Destructor Documentation

#### 4.32.1.1 Side()

```
Side::Side (
    float price )
```

Constructor for the `Side` class.

**Parameters**

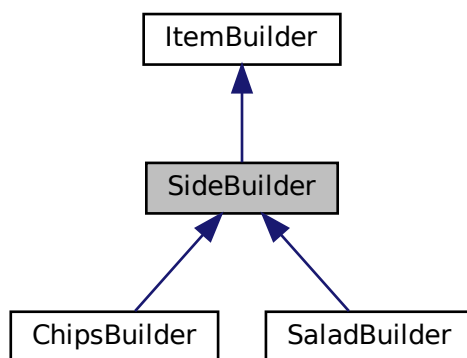
<i>price</i>	The price of the side
--------------	-----------------------

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

- [Side.h](#)
- [Side.cpp](#)

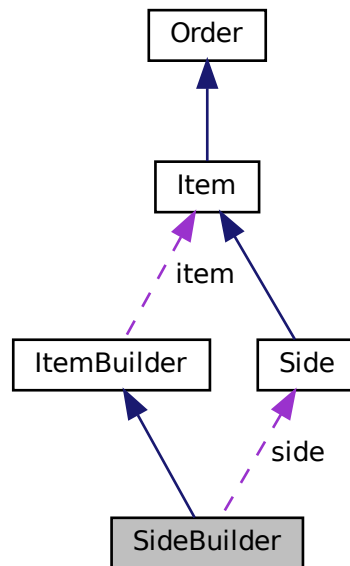
## 4.33 SideBuilder Class Reference

Inheritance diagram for SideBuilder:





Collaboration diagram for SideBuilder:



## Public Member Functions

- virtual void [prepareIngredients](#) ()  
*Prepares the ingredients for the side dish by washing and chopping the vegetables.*
- virtual void [assembleItem](#) ()  
*Assembles the side dish by calling the [assembleSide\(\)](#) and [plateSide\(\)](#) functions.*
- virtual [Item](#) \* [getItem](#) ()  
*Returns the item that was built by the [SideBuilder](#).*
- virtual void **washVegetables** ()=0
- virtual void **chopVegetables** ()=0
- virtual void **assembleSide** ()=0
- virtual void **plateSide** ()=0

## Protected Attributes

- [Side](#) \* [side](#)  
*[Side](#) object.*

### 4.33.1 Member Function Documentation

#### 4.33.1.1 getItem()

```
Item * SideBuilder::getItem ( ) [virtual]
```

Returns the item that was built by the [SideBuilder](#).

##### Returns

Item\* A pointer to the item that was built.

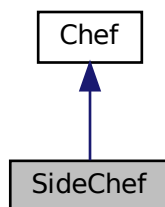
Implements [ItemBuilder](#).

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

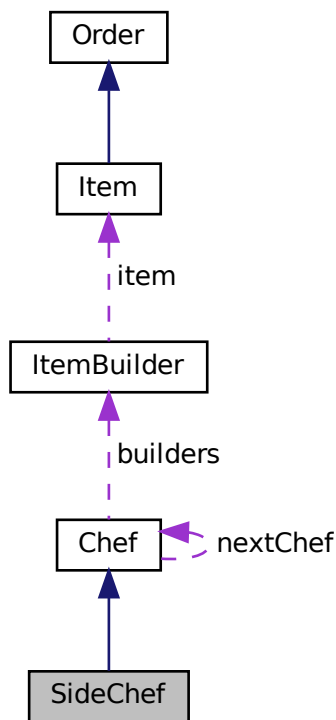
- [SideBuilder.h](#)
- [SideBuilder.cpp](#)

## 4.34 SideChef Class Reference

Inheritance diagram for SideChef:



Collaboration diagram for SideChef:



## Public Member Functions

- [SideChef](#) ()  
*Constructor of the [SideChef](#) class.*
- [~SideChef](#) ()  
*Destructor of the [SideChef](#) class.*
- void [preparePart](#) (string order, [Order](#) \*o)  
*The Chain of responsibility handle() method.*

## Additional Inherited Members

### 4.34.1 Constructor & Destructor Documentation

#### 4.34.1.1 SideChef()

```
SideChef::SideChef ( )
```

Constructor of the [SideChef](#) class.

##### Authors

Aidan Chapman (u22738917)

#### 4.34.1.2 ~SideChef()

```
SideChef::~~SideChef ( )
```

Destructor of the [SideChef](#) class.

##### Authors

Aidan Chapman (u22738917)

### 4.34.2 Member Function Documentation

#### 4.34.2.1 preparePart()

```
void SideChef::preparePart (
    string order,
    Order * o ) [virtual]
```

The Chain of responsibility handle() method.

##### Parameters

<i>order</i>	a string
<i>o</i>	an <a href="#">Order</a> pointer

##### Authors

Aidan Chapman (u22738917)

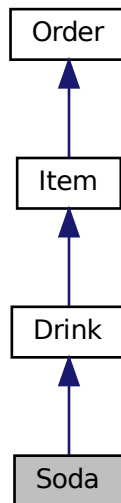
Implements [Chef](#).

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

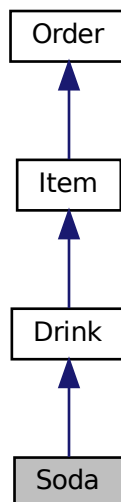
- [SideChef.h](#)
- [SideChef.cpp](#)

## 4.35 Soda Class Reference

Inheritance diagram for Soda:



Collaboration diagram for Soda:



## Public Member Functions

- [Soda\(\)](#)  
*Soda Constructor.*
- [~Soda\(\)](#)  
*Soda Destructor.*

## Public Attributes

- bool [gotSodaGlass](#) = false  
*Whether a soda glass has been obtained.*
- bool [pouredSoda](#) = false  
*Whether soda has been poured into the glass.*
- bool [assembledSoda](#) = false  
*Whether the soda has been assembled.*

## Additional Inherited Members

### 4.35.1 Constructor & Destructor Documentation

#### 4.35.1.1 Soda()

```
Soda::Soda ( )
```

[Soda](#) Constructor.

##### Authors

Aidan Chapman (u22738917)

#### 4.35.1.2 ~Soda()

```
Soda::~~Soda ( )
```

[Soda](#) Destructor.

##### Authors

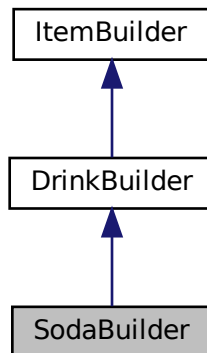
Aidan Chapman (u22738917)

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

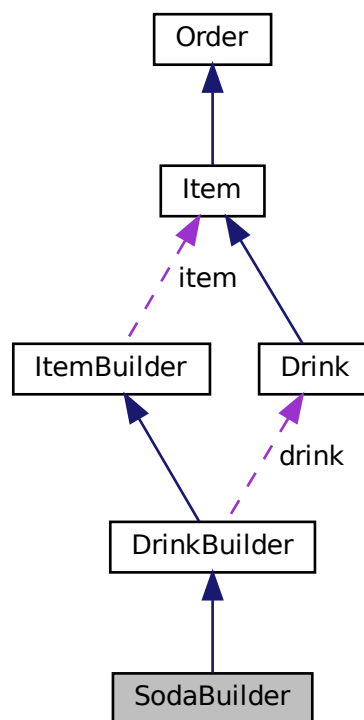
- [Soda.h](#)
- [Soda.cpp](#)

## 4.36 SodaBuilder Class Reference

Inheritance diagram for SodaBuilder:



Collaboration diagram for SodaBuilder:



## Public Member Functions

- [SodaBuilder](#) ()  
*Construct a new [Soda](#) Builder:: [Soda](#) Builder object.*
- [~SodaBuilder](#) ()  
*Destroy the [Soda](#) Builder:: [Soda](#) Builder object.*
- void [getGlass](#) ()  
*Get the [Glass](#) object.*
- void [pourDrink](#) ()  
*Pour [Drink](#) object.*
- void [assembleDrink](#) ()  
*Assemble [Drink](#) object.*
- void [getSodaGlass](#) ()  
*Get the [Soda](#) Glass object.*
- void [pourSoda](#) ()  
*Pour [Soda](#) object.*
- void [assembleSoda](#) ()  
*Assemble [Soda](#) object.*

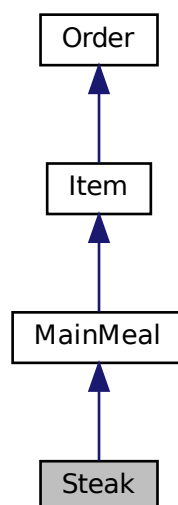
## Additional Inherited Members

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

- [SodaBuilder.h](#)
- [SodaBuilder.cpp](#)

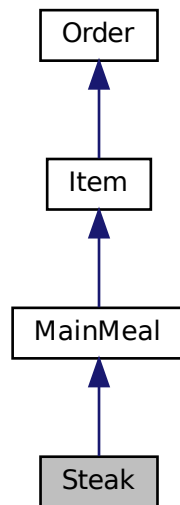
## 4.37 Steak Class Reference

Inheritance diagram for Steak:





Collaboration diagram for Steak:



## Public Member Functions

- [Steak\(\)](#)  
*Steak Constructor.*
- [~Steak\(\)](#)  
*Steak Destructor.*

## Public Attributes

- bool [tenderisedSteak](#) = false  
*Whether the steak has been tenderised.*
- bool [seasonedSteak](#) = false  
*Whether the steak has been seasoned.*
- bool [cookedSteak](#) = false  
*Whether the steak has been cooked.*
- bool [platedSteak](#) = false  
*Whether the steak has been plated.*

## Additional Inherited Members

### 4.37.1 Constructor & Destructor Documentation

#### 4.37.1.1 Steak()

```
Steak::Steak ( )
```

[Steak](#) Constructor.

##### Authors

Aidan Chapman (u22738917)

#### 4.37.1.2 ~Steak()

```
Steak::~~Steak ( )
```

[Steak](#) Destructor.

##### Authors

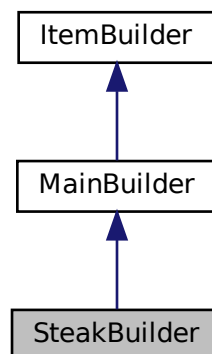
Aidan Chapman (u22738917)

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

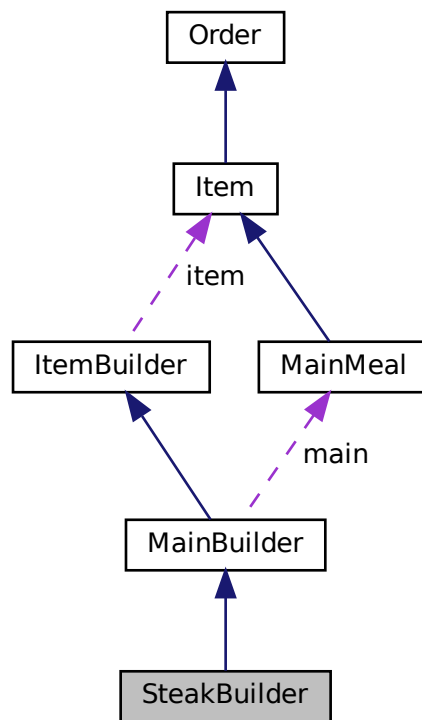
- [Steak.h](#)
- [Steak.cpp](#)

## 4.38 SteakBuilder Class Reference

Inheritance diagram for SteakBuilder:



Collaboration diagram for SteakBuilder:



## Public Member Functions

- [SteakBuilder](#) ()  
Construct a new *Steak Builder*:: *Steak Builder* object.
- [~SteakBuilder](#) ()  
Destroy the *Steak Builder*:: *Steak Builder* object.
- void [prepareMeat](#) ()  
Prepare the main meal.
- void [seasonMeat](#) ()  
Season the main meal.
- void [cookMeat](#) ()  
Cook the main meal.
- void [plateMain](#) ()
- void [tenderiseSteak](#) ()  
Tenderise the steak.
- void [seasonSteak](#) ()  
Season the steak.
- void [cookSteak](#) ()  
Cook the steak.
- void [plateSteak](#) ()  
Plate the steak.

## Additional Inherited Members

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

- [SteakBuilder.h](#)
- [SteakBuilder.cpp](#)

## 4.39 Table Class Reference

### Public Member Functions

- [Table](#) ()  
*Constructor for the [Table](#) Class.*
- [~Table](#) ()  
*Destructor for the [Table](#) Class.*
- void [addCustomer](#) ([Customer](#) \*customer)  
*Sets the customer member variable.*
- [Customer](#) \* [getCustomer](#) ()  
*A getter for the customer member variable.*
- void [cleanUp](#) ()  
*A function to be called after the customer has received their food. This is used to properly remove the customer object from the table.*

### 4.39.1 Constructor & Destructor Documentation

#### 4.39.1.1 [Table](#)()

```
Table::Table ( )
```

Constructor for the [Table](#) Class.

##### Authors

Aidan Chapman (u22738917)

#### 4.39.1.2 [~Table](#)()

```
Table::~~Table ( )
```

Destructor for the [Table](#) Class.

##### Authors

Aidan Chapman (u22738917)

## 4.39.2 Member Function Documentation

### 4.39.2.1 addCustomer()

```
void Table::addCustomer (
    Customer * customer )
```

Sets the customer member variable.

#### Parameters

<i>customer</i>	A <a href="#">Customer</a> pointer
-----------------	------------------------------------

#### Authors

Aidan Chapman (u22738917)

### 4.39.2.2 cleanUp()

```
void Table::cleanUp ( )
```

A function to be called after the customer has received their food. This is used to properly remove the customer object from the table.

#### Authors

Aidan Chapman (u22738917)

### 4.39.2.3 getCustomer()

```
Customer * Table::getCustomer ( )
```

A getter for the customer member variable.

#### Authors

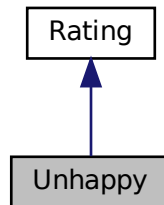
Aidan Chapman (u22738917)

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

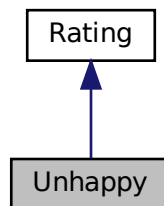
- [Table.h](#)
- [Table.cpp](#)

## 4.40 Unhappy Class Reference

Inheritance diagram for Unhappy:



Collaboration diagram for Unhappy:



### Public Member Functions

- [Unhappy \(\)](#)  
*Construct a new `Unhappy::Unhappy` object.*
- [~Unhappy \(\)](#)  
*Destroy the `Unhappy::Unhappy` object.*
- float [calculateTip \(\)](#)  
*Function for calculating the tip of the customer.*
- void [changeState \(Customer \\*customer\)](#)  
*Function for changing the state of the customer.*
- string [getRating \(\)](#)  
*Function for getting the rating of the customer.*

#### 4.40.1 Member Function Documentation

#### 4.40.1.1 calculateTip()

```
float Unhappy::calculateTip ( ) [virtual]
```

Function for calculating the tip of the customer.

##### Returns

float

Implements [Rating](#).

#### 4.40.1.2 changeState()

```
void Unhappy::changeState (
    Customer * customer ) [virtual]
```

Function for changing the state of the customer.

##### Parameters

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

Implements [Rating](#).

#### 4.40.1.3 getRating()

```
string Unhappy::getRating ( )
```

Function for getting the rating of the customer.

##### Returns

string

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

- [Unhappy.h](#)
- [Unhappy.cpp](#)

## 4.41 Waiter Class Reference

### Public Member Functions

- [Waiter](#) ([Restaurant](#) \*restaurant)  
*Constructor for the [Waiter](#) Class.*
- [~Waiter](#) ()  
*Destructor for the [Waiter](#) Class.*
- void **visitCustomer** ([Customer](#) \*customer)
- void **takeOrder** ([OrderContainer](#) \*orderContainer)  
*A member function of the [Waiter](#) Class. Used to take the customer's order.*
- void **serveCustomer** ([Order](#) \*order)  
*A member function of the [Waiter](#) Class. Calls the customer's receiveOrder([Order](#)\* order) function.*
- [Customer](#) \* **getCustomer** ()  
*A member function of the [Waiter](#) class. A getter for the [Customer](#) member variable.*
- [Restaurant](#) \* **getRestaurant** ()  
*A member function of the [Waiter](#) class. A getter for the [Restaurant](#) member variable.*
- void **cleanUp** ()  
*A member function of the [Waiter](#) class. Resets customer member variable.*

### 4.41.1 Constructor & Destructor Documentation

#### 4.41.1.1 Waiter()

```
Waiter::Waiter (
    Restaurant * restaurant )
```

Constructor for the [Waiter](#) Class.

##### Parameters

<i>restaurant</i>	a <a href="#">Restaurant</a> pointer
-------------------	--------------------------------------

##### Authors

Aidan Chapman (u22738917), Douglas Porter (u21797545)

#### 4.41.1.2 ~Waiter()

```
Waiter::~~Waiter ( )
```

Destructor for the [Waiter](#) Class.

##### Authors

Aidan Chapman (u22738917), Douglas Porter (u21797545)



## Parameters

<i>customer</i>	a <a href="#">Customer</a> pointer
-----------------	------------------------------------

## Authors

Aidan Chapman (u22738917), Douglas Porter (u21797545)

## 4.41.2 Member Function Documentation

### 4.41.2.1 `cleanUp()`

```
void Waiter::cleanUp ( )
```

A member function of the [Waiter](#) class. Resets customer member variable.

## Authors

Aidan Chapman (u22738917)

### 4.41.2.2 `getCustomer()`

```
Customer * Waiter::getCustomer ( )
```

A member function of the [Waiter](#) class. A getter for the [Customer](#) member variable.

## Returns

customer reference

## Authors

Aidan Chapman (u22738917)

### 4.41.2.3 `getRestaurant()`

```
Restaurant * Waiter::getRestaurant ( )
```

A member function of the [Waiter](#) class. A getter for the [Restaurant](#) member variable.

## Returns

[Restaurant](#) reference

## Authors

Aidan Chapman (u22738917)

### 4.41.2.4 `serveCustomer()`

```
void Waiter::serveCustomer (
    Order * order )
```

A member function of the [Waiter](#) Class. Calls the customer's `receiveOrder(Order* order)` function.

**Parameters**

<i>order</i>	an <a href="#">Order</a> pointer
--------------	----------------------------------

**Authors**

Aidan Chapman (u22738917), Douglas Porter (u21797545)

**4.41.2.5 takeOrder()**

```
void Waiter::takeOrder (  
    OrderContainer * orderContainer )
```

A member function of the [Waiter](#) Class. Used to take the customer's order.

**Parameters**

<i>orderContainer</i>	an <a href="#">OrderContainer</a> pointer
-----------------------	---

**Authors**

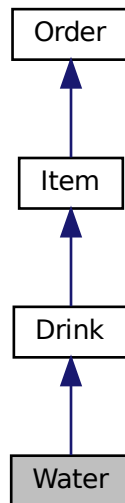
Aidan Chapman (u22738917), Douglas Porter (u21797545)

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

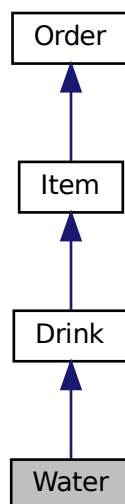
- [Waiter.h](#)
- [Waiter.cpp](#)

## 4.42 Water Class Reference

Inheritance diagram for Water:



Collaboration diagram for Water:



## Public Member Functions

- [Water](#) ()  
*[Water](#) Constructor.*
- [~Water](#) ()  
*[Water](#) Destructor.*

## Public Attributes

- bool [gotWaterGlass](#) = false  
*Whether a water glass has been obtained.*
- bool [pouredWater](#) = false  
*Whether water has been poured into the glass.*
- bool [assembledWater](#) = false  
*Whether the water has been assembled.*

## Additional Inherited Members

### 4.42.1 Constructor & Destructor Documentation

#### 4.42.1.1 [Water](#)()

```
Water::Water ( )
```

[Water](#) Constructor.

##### Authors

Aidan Chapman (u22738917)

#### 4.42.1.2 [~Water](#)()

```
Water::~~Water ( )
```

[Water](#) Destructor.

##### Authors

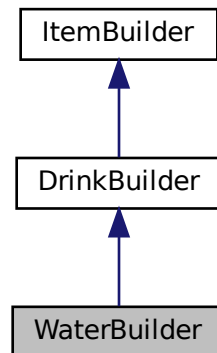
Aidan Chapman (u22738917)

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

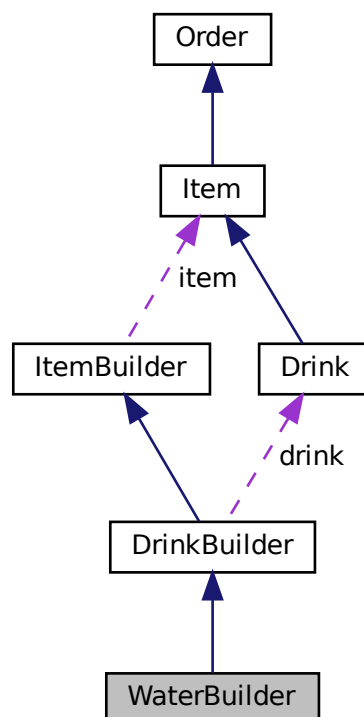
- [Water.h](#)
- [Water.cpp](#)

## 4.43 WaterBuilder Class Reference

Inheritance diagram for WaterBuilder:



Collaboration diagram for WaterBuilder:



## Public Member Functions

- [WaterBuilder](#) ()  
*Construct a new [Water](#) Builder:: [Water](#) Builder object.*
- [~WaterBuilder](#) ()  
*Destroy the [Water](#) Builder:: [Water](#) Builder object.*
- void [getGlass](#) ()  
*prepare the glass*
- void [pourDrink](#) ()  
*Pour the [Drink](#) object.*
- void [assembleDrink](#) ()  
*Assemble the [Drink](#) object.*
- void [getWaterGlass](#) ()  
*Get the [Water](#) Glass object.*
- void [pourWater](#) ()  
*Pour the [Water](#) object.*
- void [assembleWater](#) ()  
*Assemble the [Water](#) object.*

## Additional Inherited Members

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

- [WaterBuilder.h](#)
- [WaterBuilder.cpp](#)

## Chapter 5

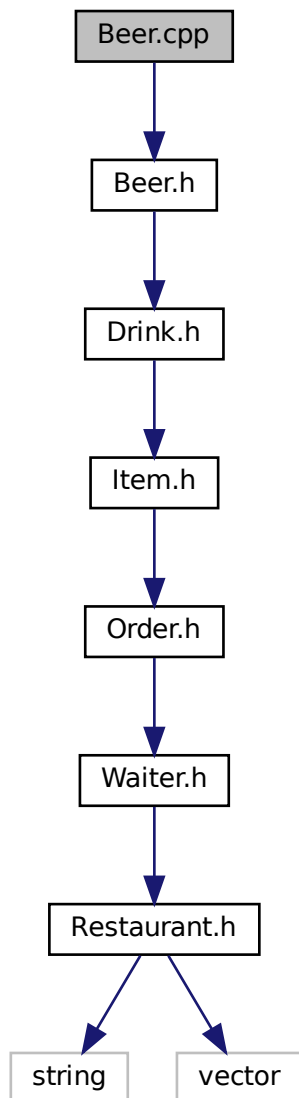
# File Documentation

### 5.1 Beer.cpp File Reference

Contains implementation for the [Beer](#) class.

```
#include "Beer.h"
```

Include dependency graph for Beer.cpp:



### 5.1.1 Detailed Description

Contains implementation for the [Beer](#) class.

#### Authors

Aidan Chapman (u22738917)

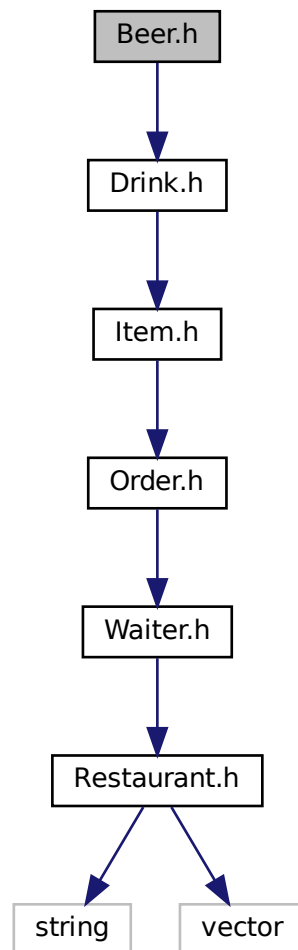


## 5.2 Beer.h File Reference

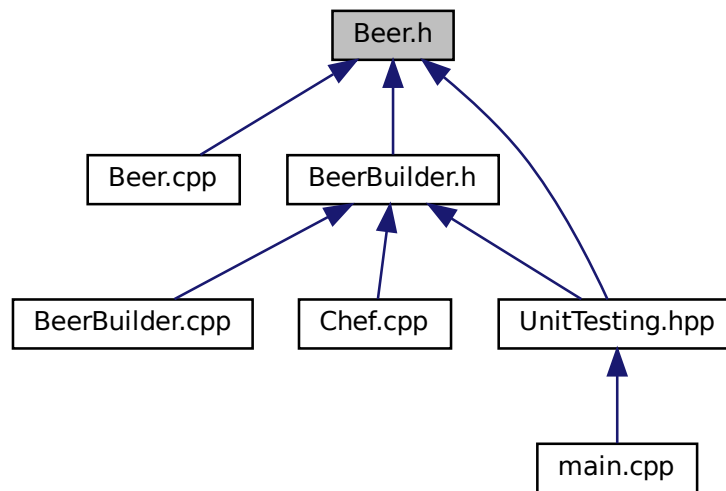
Contains the declaration for the [Beer](#) class.

```
#include "Drink.h"
```

Include dependency graph for Beer.h:



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



## Classes

- class [Beer](#)

### 5.2.1 Detailed Description

Contains the declaration for the [Beer](#) class.

This file defines the [Beer](#) class, which is a subclass of the [Drink](#) class. It contains boolean variables to keep track of whether a beer glass has been obtained, whether beer has been poured into the glass, and whether the beer has been assembled.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

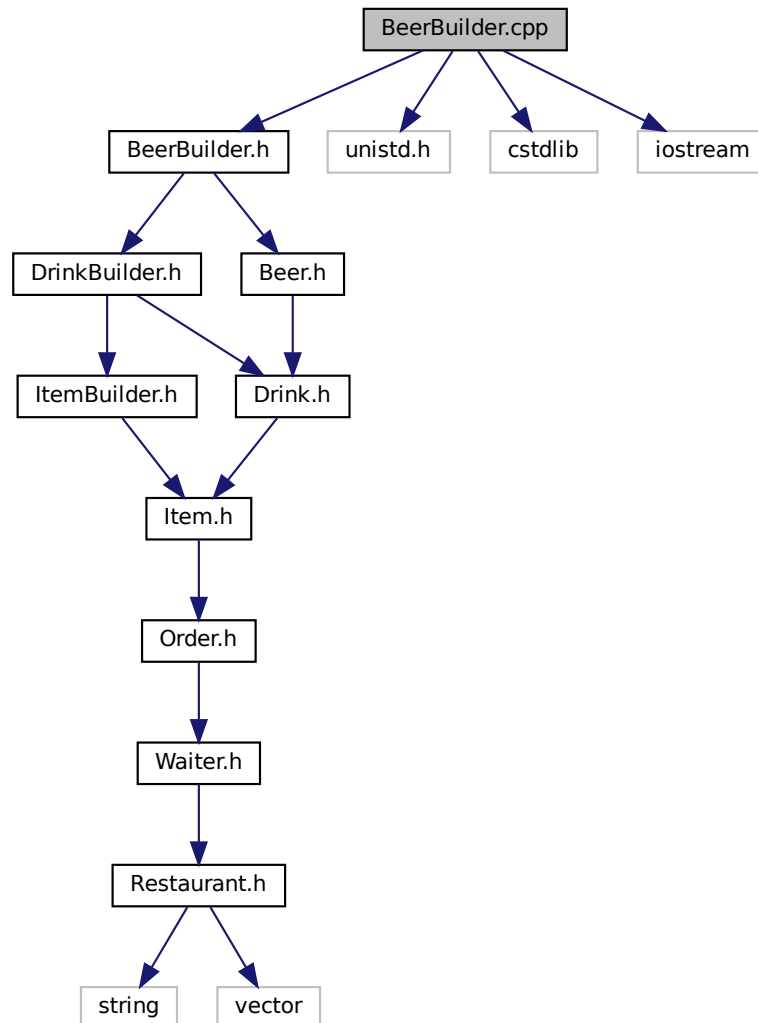
## 5.3 BeerBuilder.cpp File Reference

Implementation of the [BeerBuilder](#) class.

```
#include "BeerBuilder.h"  
#include <unistd.h>  
#include <cstdlib>
```

```
#include <iostream>
```

Include dependency graph for BeerBuilder.cpp:



### 5.3.1 Detailed Description

Implementation of the [BeerBuilder](#) class.

This file contains the implementation of the [BeerBuilder](#) class, which is responsible for building a [Beer](#) object. The class defines methods for getting the glass, pouring the drink, and assembling the drink. It also defines methods for getting the beer glass, pouring the beer, and assembling the beer.

#### Author

- Graeme Blain (u22625462)

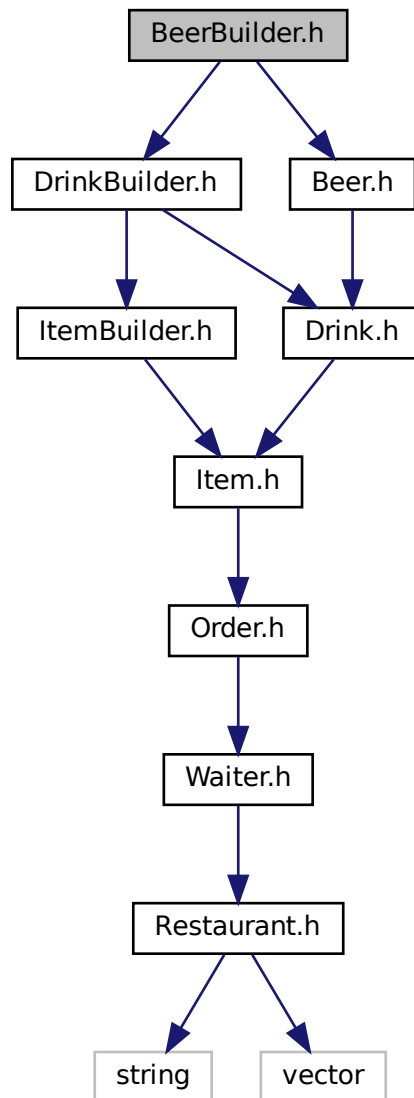
## 5.4 BeerBuilder.h File Reference

Contains declaration for the [BeerBuilder](#) class.

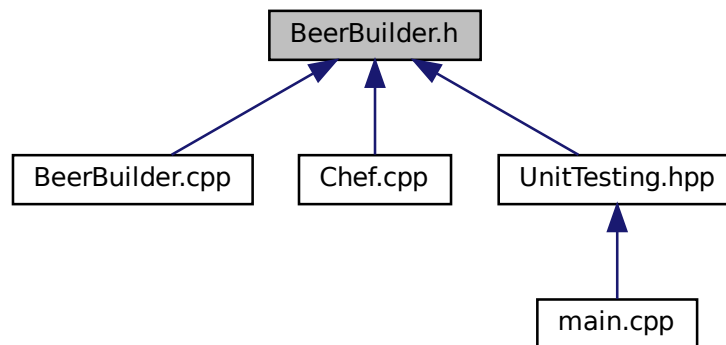
```
#include "DrinkBuilder.h"
```

```
#include "Beer.h"
```

Include dependency graph for BeerBuilder.h:



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



## Classes

- class [BeerBuilder](#)

### 5.4.1 Detailed Description

Contains declaration for the [BeerBuilder](#) class.

This file defines the [BeerBuilder](#) class, which is a subclass of the [DrinkBuilder](#) class. [BeerBuilder](#) is used to build a [Beer](#) object, and contains functions to get a beer glass,

#### Author

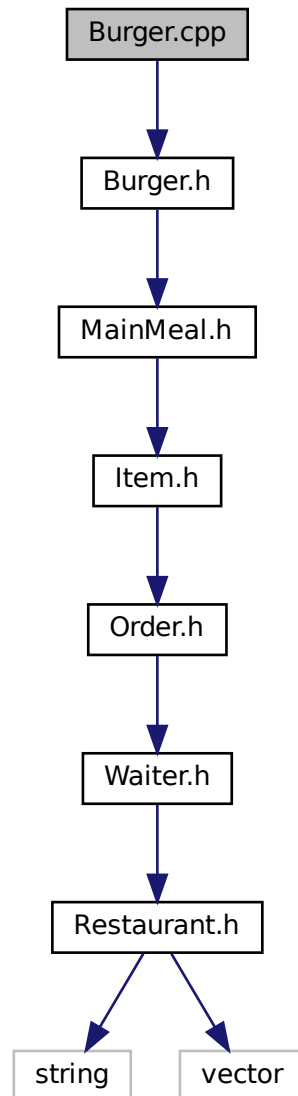
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.5 Burger.cpp File Reference

Contains implementation for the [Burger](#) class.

```
#include "Burger.h"
```

Include dependency graph for Burger.cpp:



### 5.5.1 Detailed Description

Contains implementation for the [Burger](#) class.

#### Authors

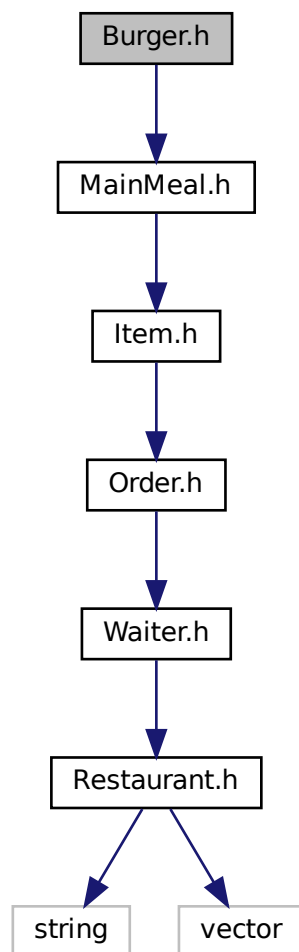
Aidan Chapman (u22738917)

## 5.6 Burger.h File Reference

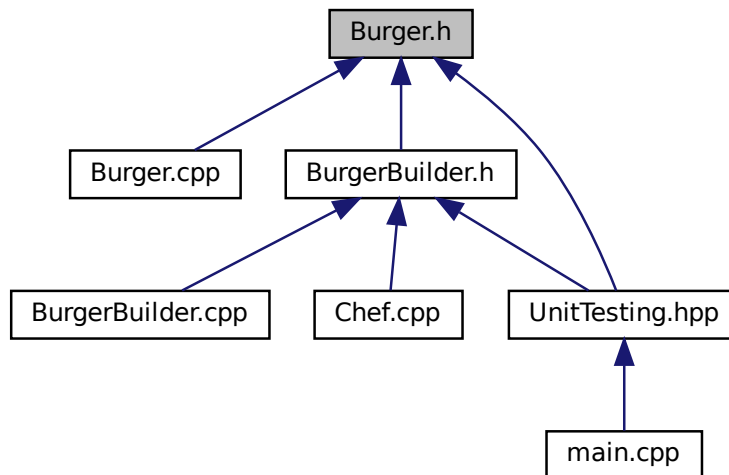
Contains declaration for the [Burger](#) class.

```
#include "MainMeal.h"
```

Include dependency graph for Burger.h:



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



## Classes

- class [Burger](#)

### 5.6.1 Detailed Description

Contains declaration for the [Burger](#) class.

The [Burger](#) class is a subclass of the [MainMeal](#) class. It represents a burger

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)
- Douglas Porter (u21797545)

## 5.7 BurgerBuilder.cpp File Reference

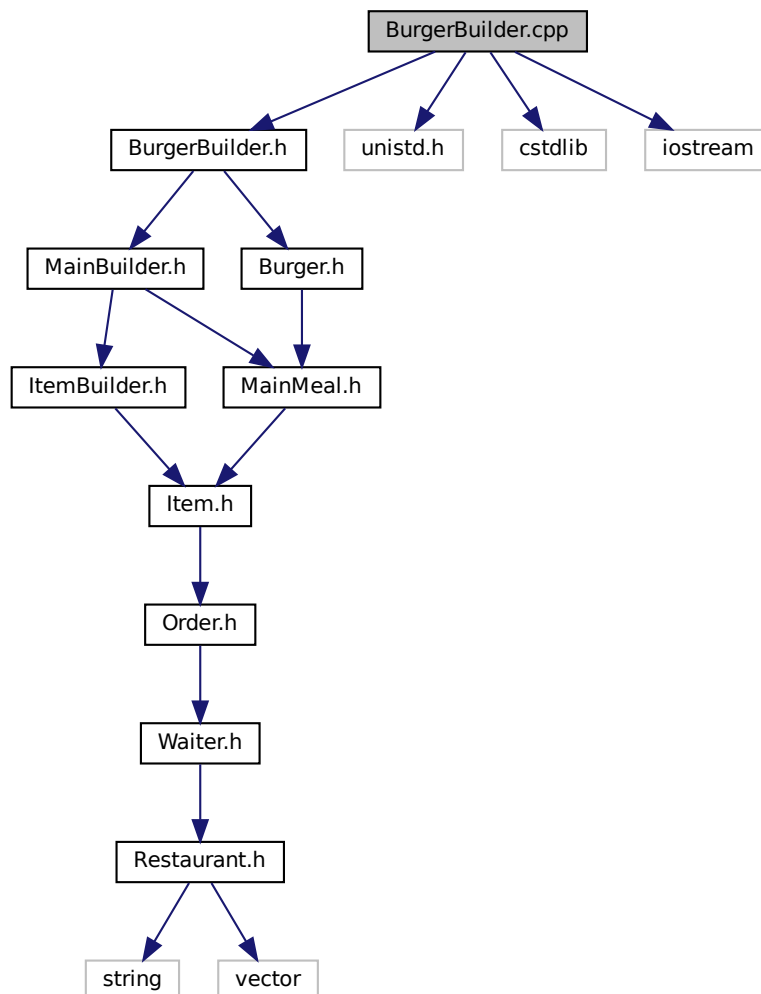
Contains the implementation for the [BurgerBuilder](#) class.

```
#include "BurgerBuilder.h"
#include <unistd.h>
#include <cstdlib>
```



```
#include <iostream>
```

Include dependency graph for BurgerBuilder.cpp:



### 5.7.1 Detailed Description

Contains the implementation for the [BurgerBuilder](#) class.

This file contains the implementation of the [BurgerBuilder](#) class, which is responsible for building a [Burger](#) object.

#### Authors

- Douglas Porter (u21797545)
- Graeme Blain (u22625462)

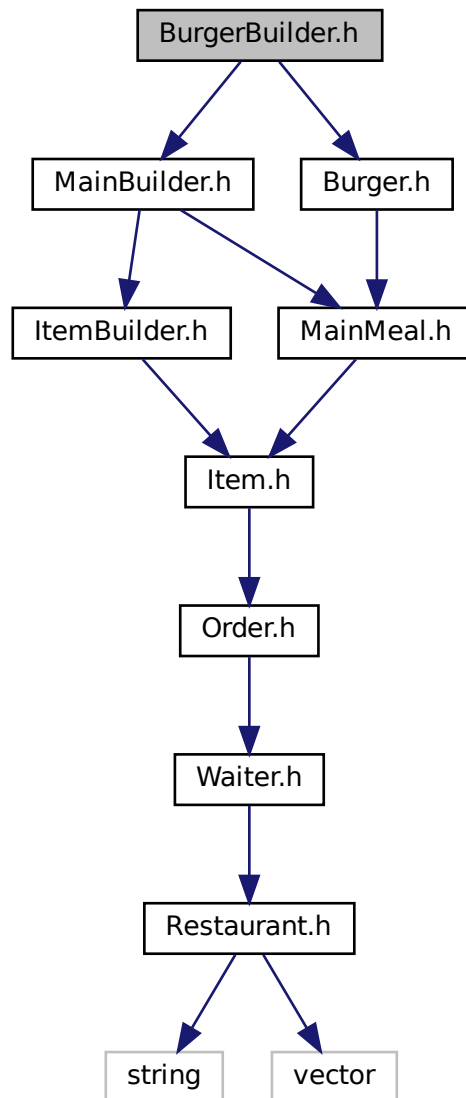
## 5.8 BurgerBuilder.h File Reference

Contains declaration for the [BurgerBuilder](#) class.

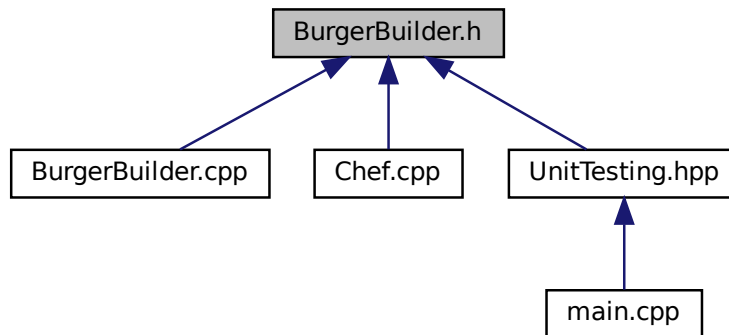
```
#include "MainBuilder.h"
```

```
#include "Burger.h"
```

Include dependency graph for BurgerBuilder.h:



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



## Classes

- class [BurgerBuilder](#)

### 5.8.1 Detailed Description

Contains declaration for the [BurgerBuilder](#) class.

This file contains the declaration for the [BurgerBuilder](#) class. [BurgerBuilder](#) is a concrete builder class that inherits from the [MainBuilder](#) class. It is responsible for building a [Burger](#) object using the template method pattern.

See also

[MainBuilder](#)  
[Burger](#)

Authors

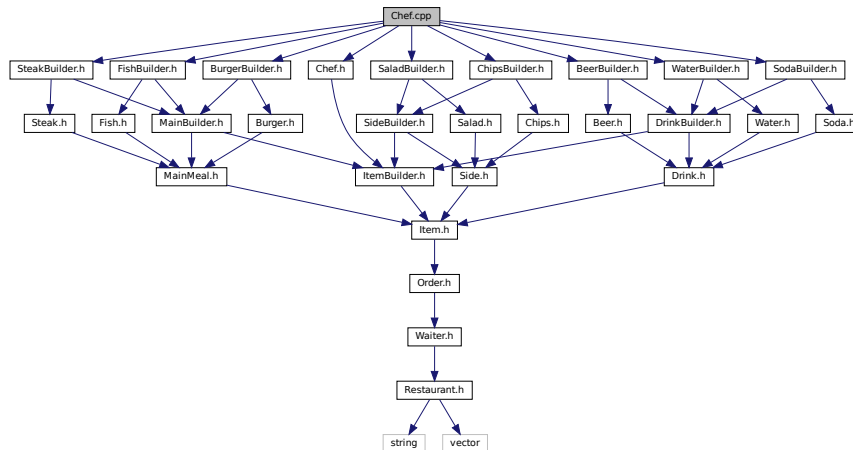
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)
- Douglas Porter (u21797545)

## 5.9 Chef.cpp File Reference

Contains implementation for the [Chef](#) class.

```
#include "Chef.h"  
#include "SteakBuilder.h"  
#include "BurgerBuilder.h"  
#include "FishBuilder.h"
```

```
#include "ChipsBuilder.h"
#include "SaladBuilder.h"
#include "BeerBuilder.h"
#include "WaterBuilder.h"
#include "SodaBuilder.h"
Include dependency graph for Chef.cpp:
```



### 5.9.1 Detailed Description

Contains implementation for the [Chef](#) class.

This file contains the implementation for the [Chef](#) class. The [Chef](#) class is responsible for building the order.

#### Authors

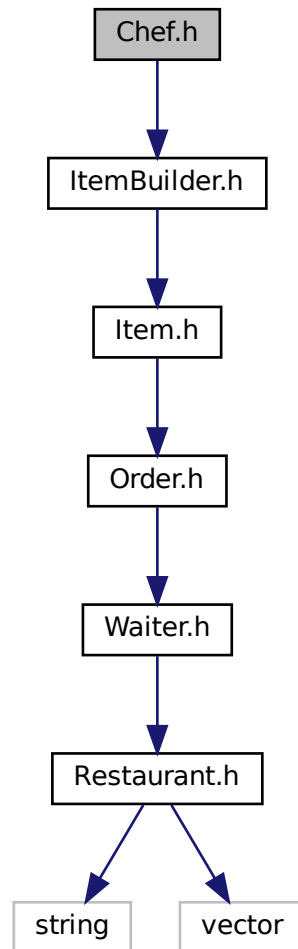
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.10 Chef.h File Reference

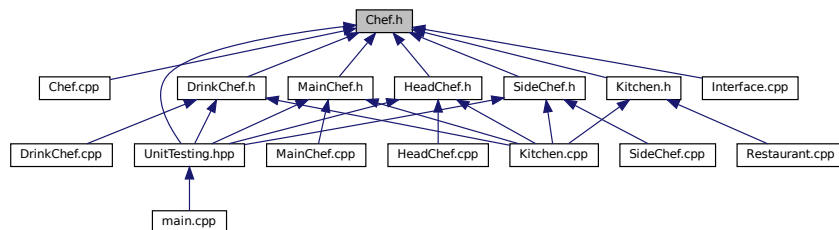
Contains declaration for the [Chef](#) class.

```
#include "ItemBuilder.h"
```

Include dependency graph for Chef.h:



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



## Classes

- class [Chef](#)

### 5.10.1 Detailed Description

Contains declaration for the [Chef](#) class.

This file contains the declaration for the [Chef](#) class, which is responsible for preparing different parts of an order using various [ItemBuilder](#) objects.

#### Note

This class is abstract and cannot be instantiated directly.

#### Author

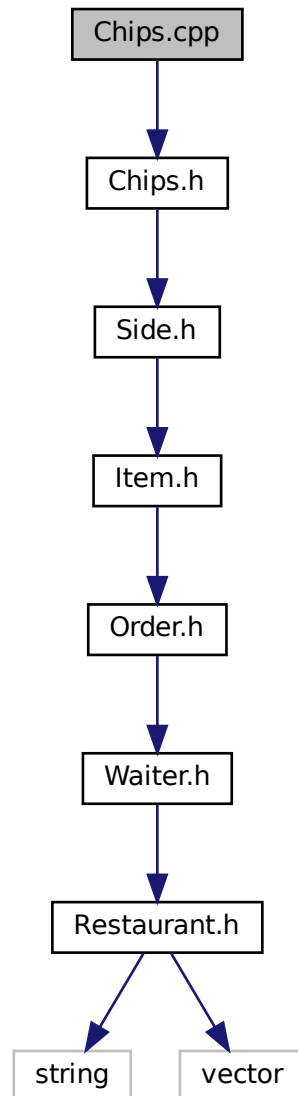
Aidan Chapman

## 5.11 Chips.cpp File Reference

Contains implementation for the [Chips](#) class.

```
#include "Chips.h"
```

Include dependency graph for Chips.cpp:



### 5.11.1 Detailed Description

Contains implementation for the [Chips](#) class.

#### Authors

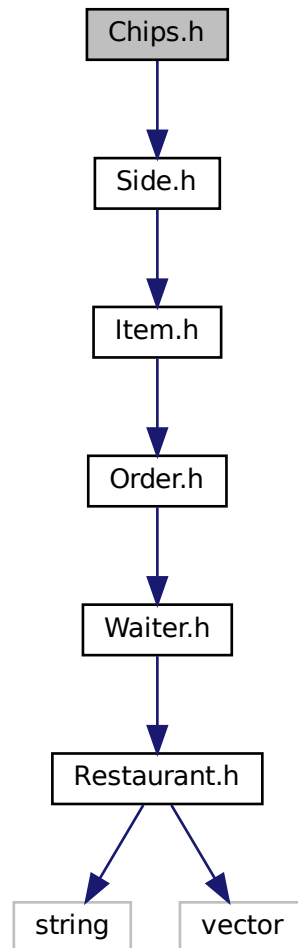
Aidan Chapman (u22738917)

## 5.12 Chips.h File Reference

Contains declaration for the [Chips](#) class.

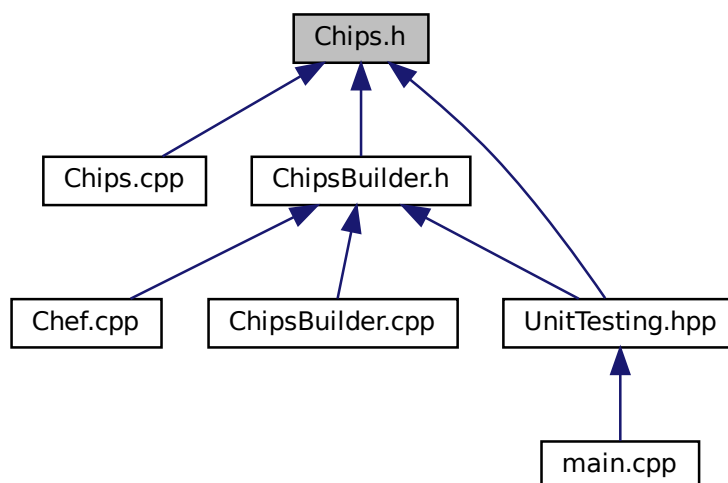
```
#include "Side.h"
```

Include dependency graph for Chips.h:





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



## Classes

- class [Chips](#)

### 5.12.1 Detailed Description

Contains declaration for the [Chips](#) class.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

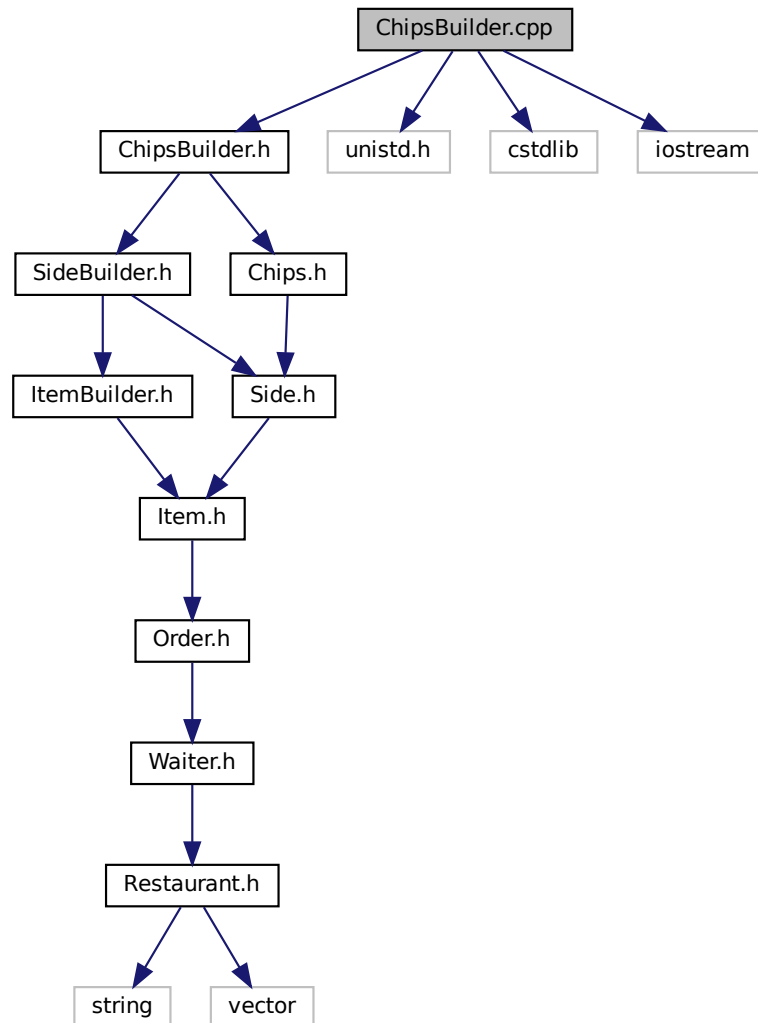
## 5.13 ChipsBuilder.cpp File Reference

Implementation of the [ChipsBuilder](#) class.

```
#include "ChipsBuilder.h"  
#include <unistd.h>  
#include <cstdlib>
```

```
#include <iostream>
```

Include dependency graph for ChipsBuilder.cpp:



### 5.13.1 Detailed Description

Implementation of the [ChipsBuilder](#) class.

This file contains the implementation of the [ChipsBuilder](#) class, which is responsible for building chips as a side dish. The class defines methods for washing, chopping, assembling, and plating the side dish, as well as for washing, cutting, frying, and seasoning the potatoes.

Author

- Graeme Blain (u22625462)

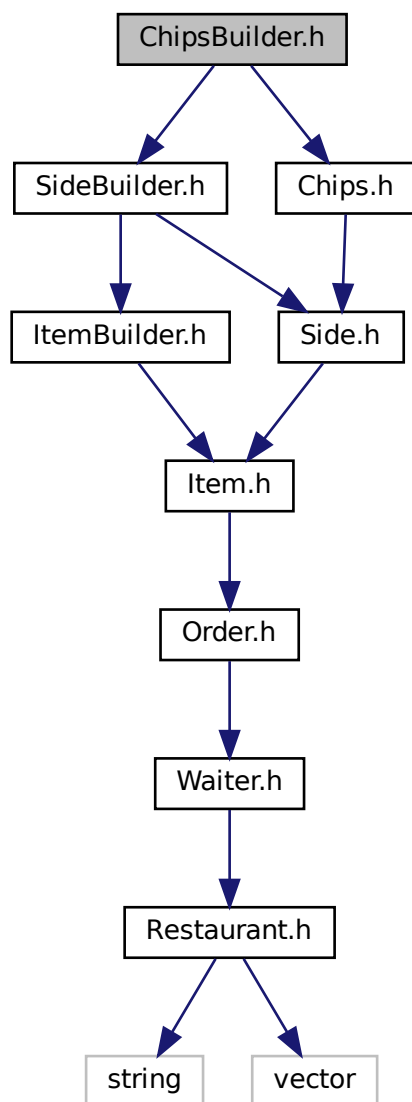
## 5.14 ChipsBuilder.h File Reference

Contains declaration for the [ChipsBuilder](#) class.

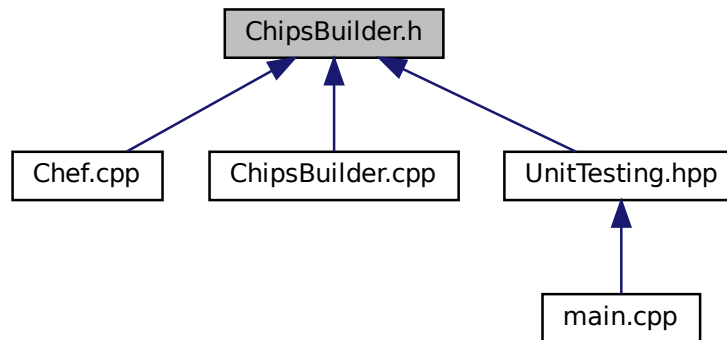
```
#include "SideBuilder.h"
```

```
#include "Chips.h"
```

Include dependency graph for ChipsBuilder.h:



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



## Classes

- class [ChipsBuilder](#)

### 5.14.1 Detailed Description

Contains declaration for the [ChipsBuilder](#) class.

Represents a concrete builder in the builder pattern. Responsible for building a [Chips](#) object.

#### Author

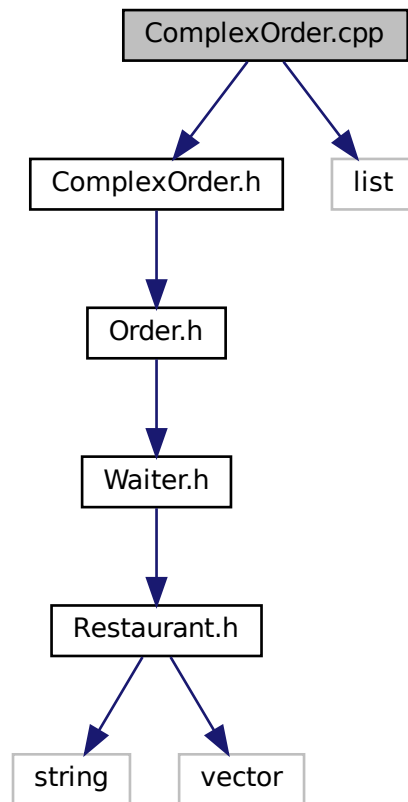
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.15 ComplexOrder.cpp File Reference

Contains the implementation for the [ComplexOrder](#) class.

```
#include "ComplexOrder.h"  
#include <list>
```

Include dependency graph for ComplexOrder.cpp:



### 5.15.1 Detailed Description

Contains the implementation for the [ComplexOrder](#) class.

This file contains the implementation for the [ComplexOrder](#) class, which is a concrete implementation of the [Order](#) abstract class. The [ComplexOrder](#) class represents a complex order that can contain multiple orders, including other complex orders.

#### Author

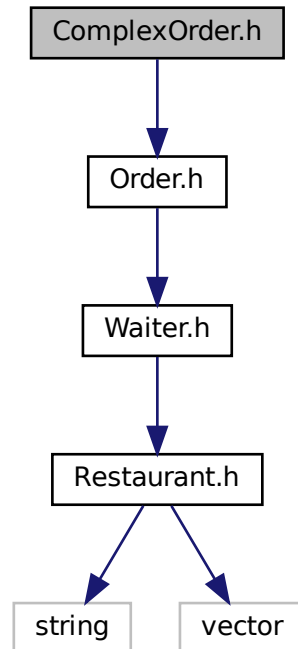
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)
- Sange Tshakumane (u21479748)

## 5.16 ComplexOrder.h File Reference

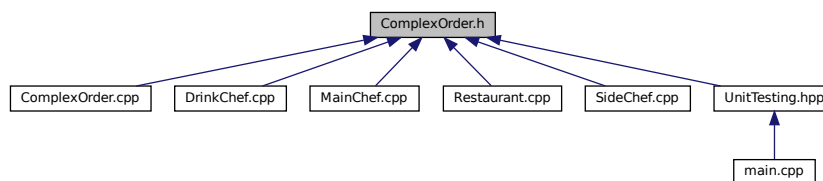
Contains declaration for the [ComplexOrder](#) class.

```
#include "Order.h"
```

Include dependency graph for ComplexOrder.h:



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



## Classes

- class [ComplexOrder](#)

### 5.16.1 Detailed Description

Contains declaration for the [ComplexOrder](#) class.

[ComplexOrder](#) is a concrete class which inherits from [Order](#). It is a composite in the composite pattern. It is responsible for storing a list of orders and calculating the total price of the order.

## Authors

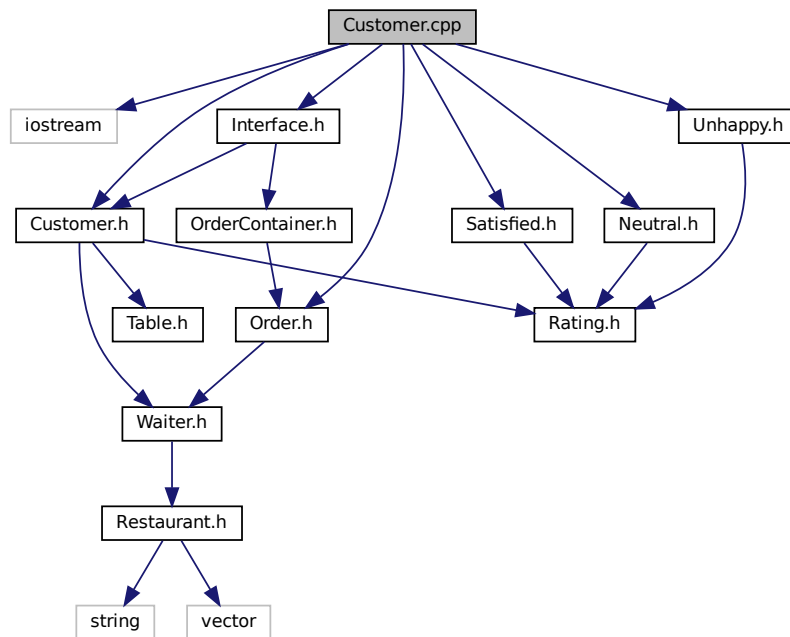
- Aidan Chapman (u22738917)

## 5.17 Customer.cpp File Reference

Contains implementation for the [Customer](#) class.

```
#include <iostream>
#include "Customer.h"
#include "Order.h"
#include "Interface.h"
#include "Satisfied.h"
#include "Neutral.h"
#include "Unhappy.h"
```

Include dependency graph for Customer.cpp:



### 5.17.1 Detailed Description

Contains implementation for the [Customer](#) class.

This file contains the implementation of the [Customer](#) class, which is responsible for representing a customer in the restaurant.

## Authors

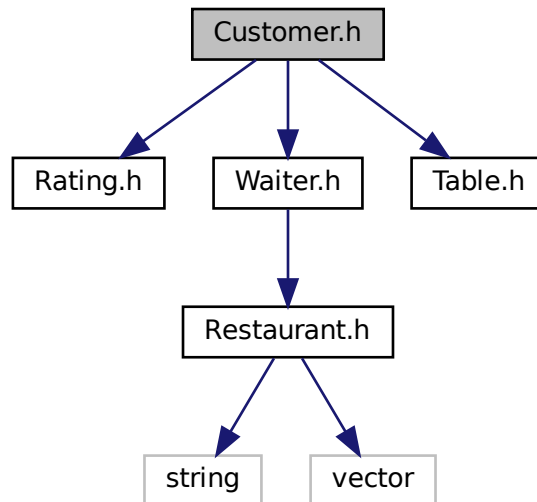
Aidan Chapman (u22738917)

## 5.18 Customer.h File Reference

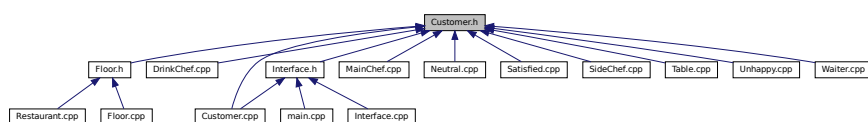
Contains declaration for the [Customer](#) class.

```
#include "Rating.h"
#include "Waiter.h"
#include "Table.h"
```

Include dependency graph for Customer.h:



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



### Classes

- class [Customer](#)

#### 5.18.1 Detailed Description

Contains declaration for the [Customer](#) class.

Contains the declaration of the [Customer](#) class, which represents a customer

#### Authors

- Aidan Chapman (u22738917)

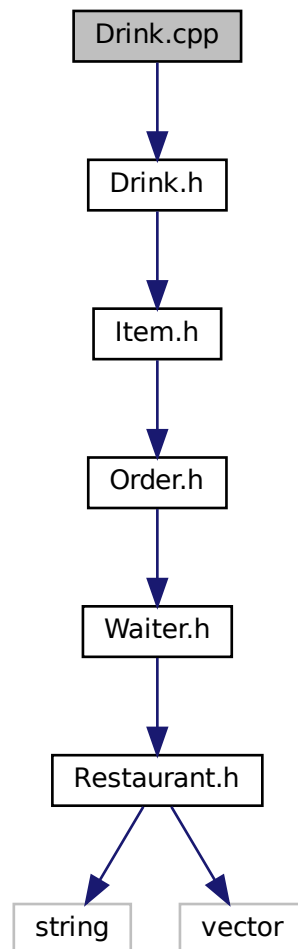


## 5.19 Drink.cpp File Reference

Contains implementation for the [Drink](#) class.

```
#include "Drink.h"
```

Include dependency graph for Drink.cpp:



### 5.19.1 Detailed Description

Contains implementation for the [Drink](#) class.

This file contains the implementation of the [Drink](#) class, which is responsible for building a [Drink](#) object.

#### Authors

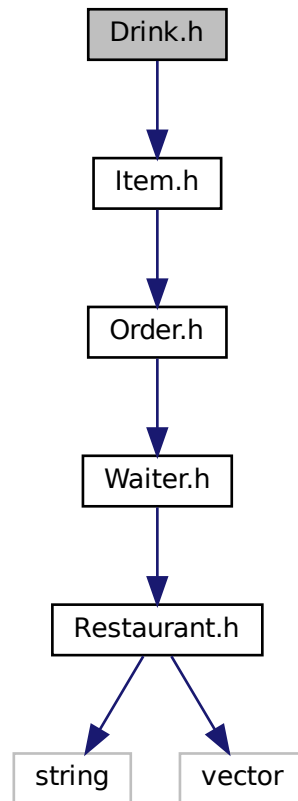
- Sange Tshakumane (u21479748)
- Aidan Chapman (u22738917)

## 5.20 Drink.h File Reference

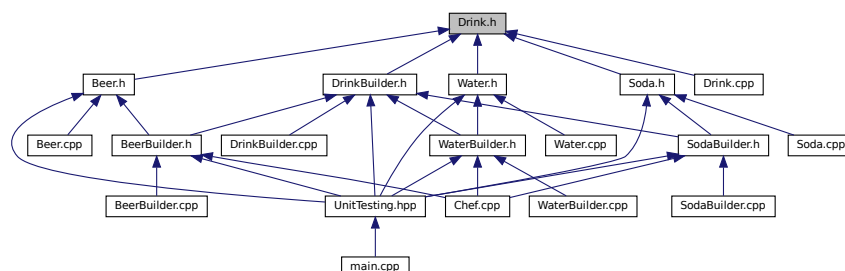
Contains declaration for the [Drink](#) class.

```
#include "Item.h"
```

Include dependency graph for Drink.h:



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



## Classes

- class [Drink](#)

### 5.20.1 Detailed Description

Contains declaration for the [Drink](#) class.

[Drink](#) is a derived class of [Item](#), representing a drink item on the menu.

#### Authors

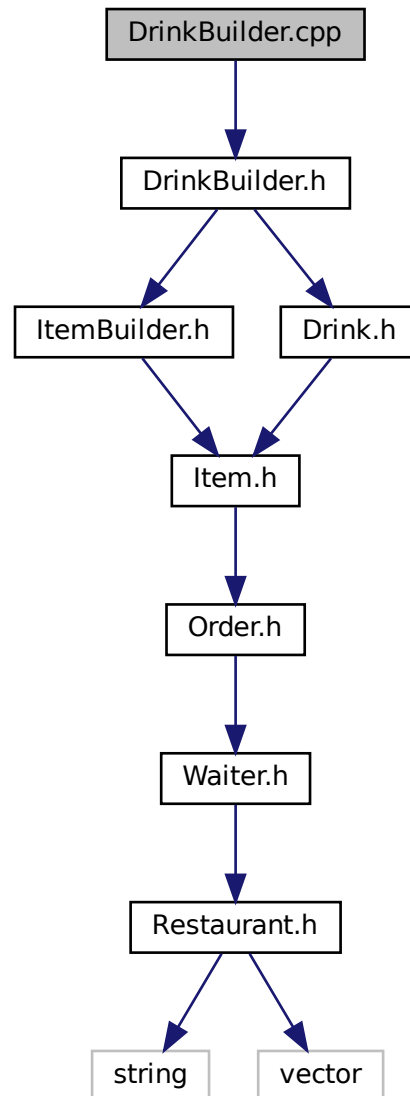
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.21 DrinkBuilder.cpp File Reference

Implementation of the [DrinkBuilder](#) class.

```
#include "DrinkBuilder.h"
```

Include dependency graph for DrinkBuilder.cpp:



### 5.21.1 Detailed Description

Implementation of the [DrinkBuilder](#) class.

This file contains the implementation of the [DrinkBuilder](#) class, which is responsible for building drinks. It defines the functions to prepare the ingredients, assemble the drink, and return the built item.

#### Authors

- Graeme Blain (u22625462)

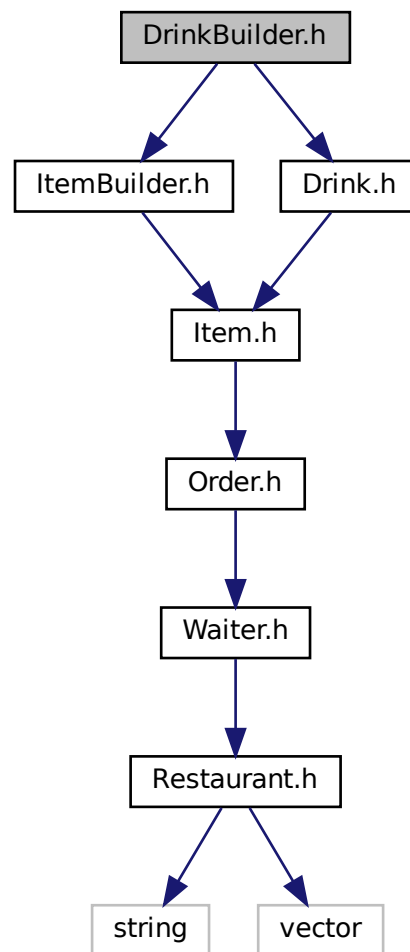
## 5.22 DrinkBuilder.h File Reference

Contains declaration for the [DrinkBuilder](#) class.

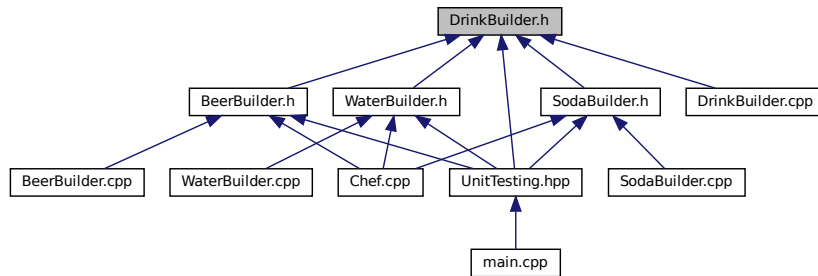
```
#include "ItemBuilder.h"
```

```
#include "Drink.h"
```

Include dependency graph for DrinkBuilder.h:



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



## Classes

- class [DrinkBuilder](#)

### 5.22.1 Detailed Description

Contains declaration for the [DrinkBuilder](#) class.

This file defines the [DrinkBuilder](#) class, which is a subclass of the [ItemBuilder](#) class. [DrinkBuilder](#) is used to build a [Drink](#) object, and contains functions to prepare ingredients,

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

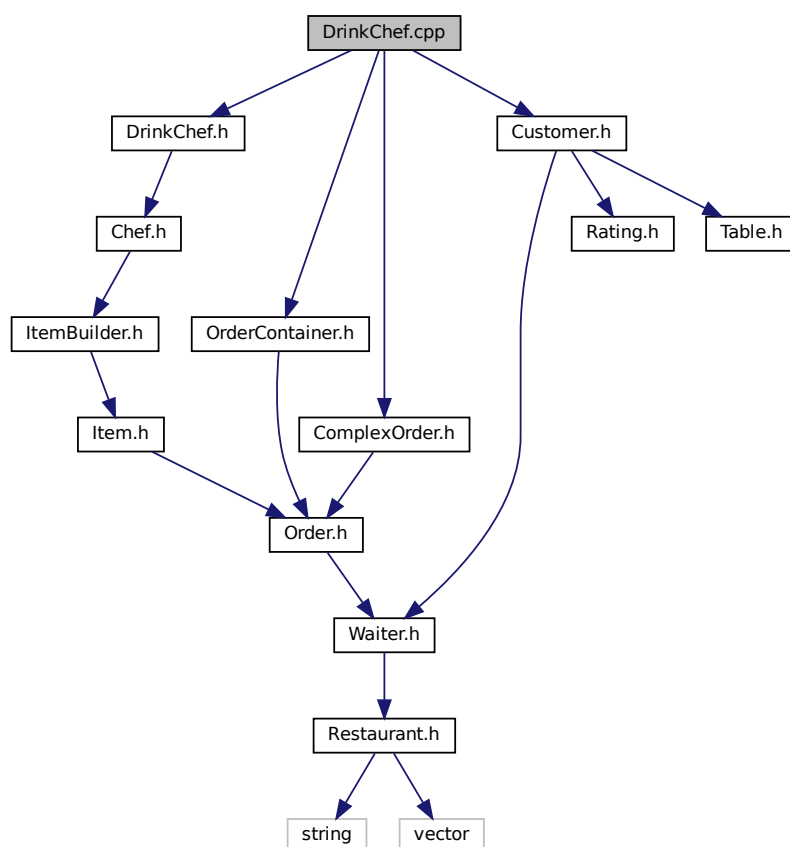
## 5.23 DrinkChef.cpp File Reference

Contains implementation for the [DrinkChef](#) class.

```
#include "DrinkChef.h"  
#include "Customer.h"  
#include "OrderContainer.h"
```

```
#include "ComplexOrder.h"
```

Include dependency graph for DrinkChef.cpp:



### 5.23.1 Detailed Description

Contains implementation for the [DrinkChef](#) class.

#### Authors

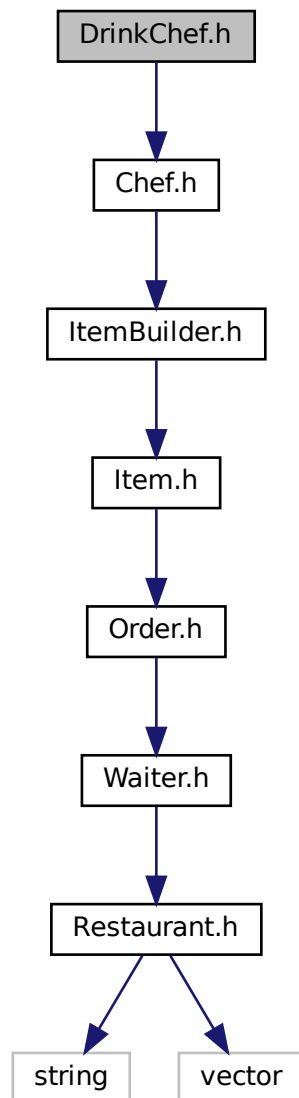
Aidan Chapman (u22738917)

## 5.24 DrinkChef.h File Reference

Contains declaration for the [DrinkChef](#) class.

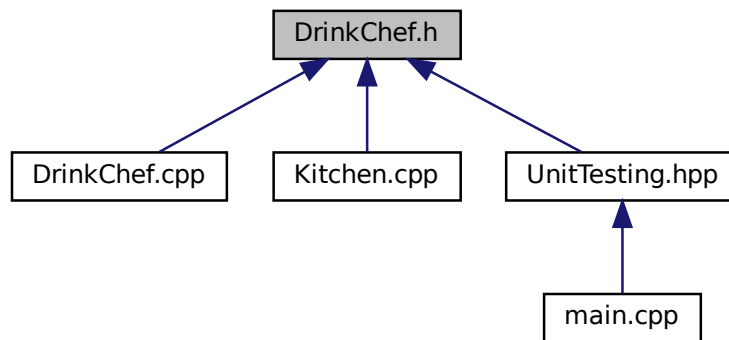
```
#include "Chef.h"
```

Include dependency graph for DrinkChef.h:





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



## Classes

- class [DrinkChef](#)

### 5.24.1 Detailed Description

Contains declaration for the [DrinkChef](#) class.

[DrinkChef](#) is a derived class of [Chef](#), representing a chef that prepares drinks.

#### Authors

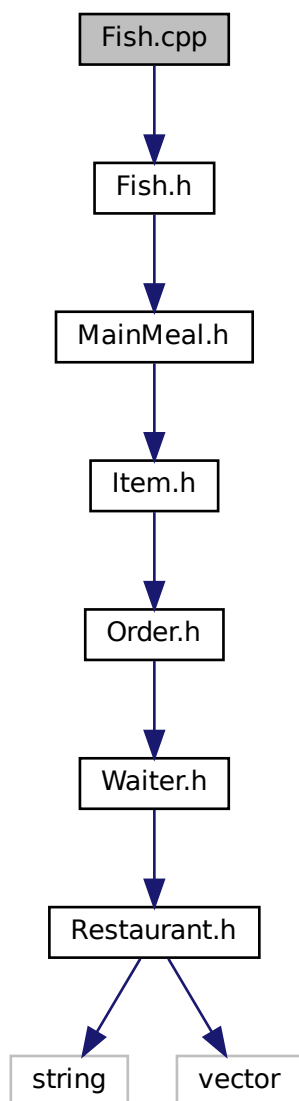
Aidan Chapman (u22738917)

## 5.25 Fish.cpp File Reference

Contains implementation for the [Fish](#) class.

```
#include "Fish.h"
```

Include dependency graph for Fish.cpp:



### 5.25.1 Detailed Description

Contains implementation for the [Fish](#) class.

#### Authors

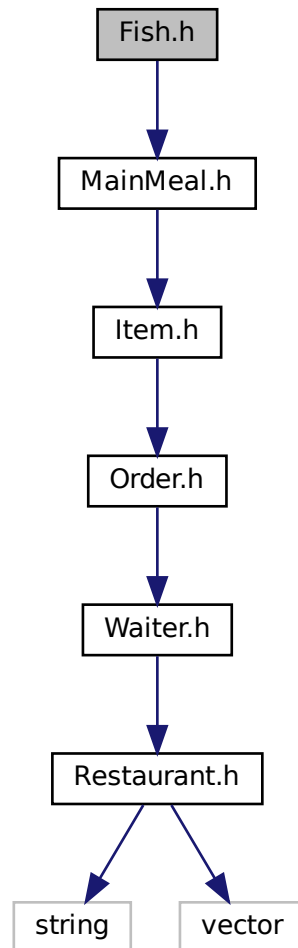
Aidan Chapman (u22738917)

## 5.26 Fish.h File Reference

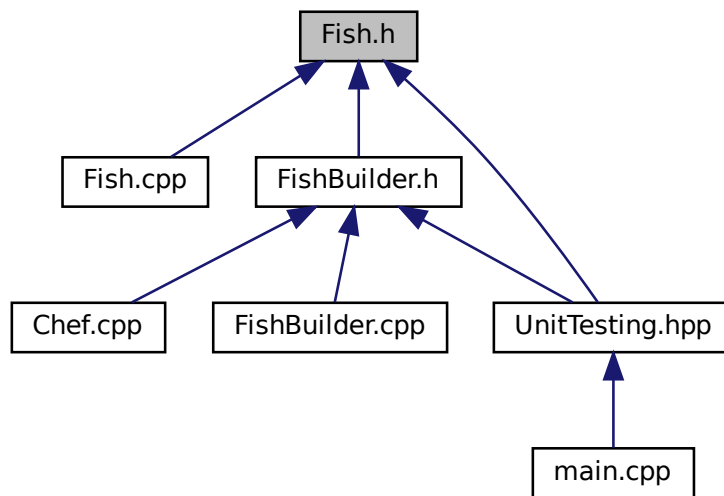
Contains declaration for the [Fish](#) class.

```
#include "MainMeal.h"
```

Include dependency graph for Fish.h:



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



## Classes

- class [Fish](#)

### 5.26.1 Detailed Description

Contains declaration for the [Fish](#) class.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

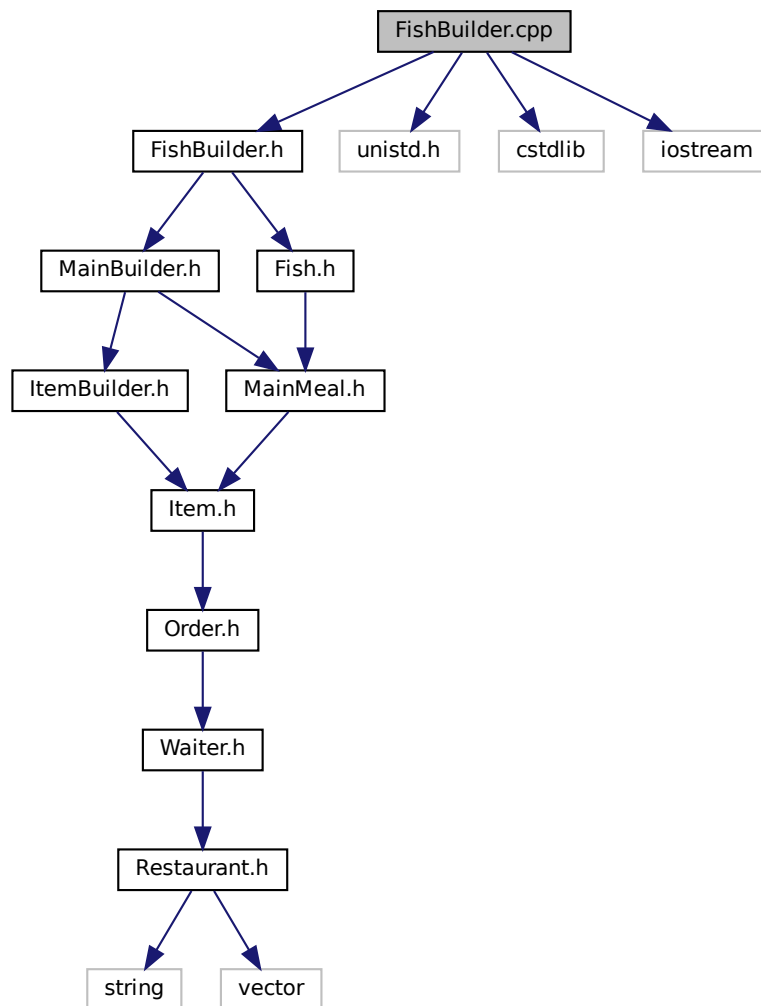
## 5.27 FishBuilder.cpp File Reference

Contains the implementation for the [FishBuilder](#) class.

```
#include "FishBuilder.h"  
#include <unistd.h>  
#include <cstdlib>
```

```
#include <iostream>
```

Include dependency graph for FishBuilder.cpp:



### 5.27.1 Detailed Description

Contains the implementation for the [FishBuilder](#) class.

#### Authors

- Graeme Blain (u22625462)
- Douglas Porter (u21797545)

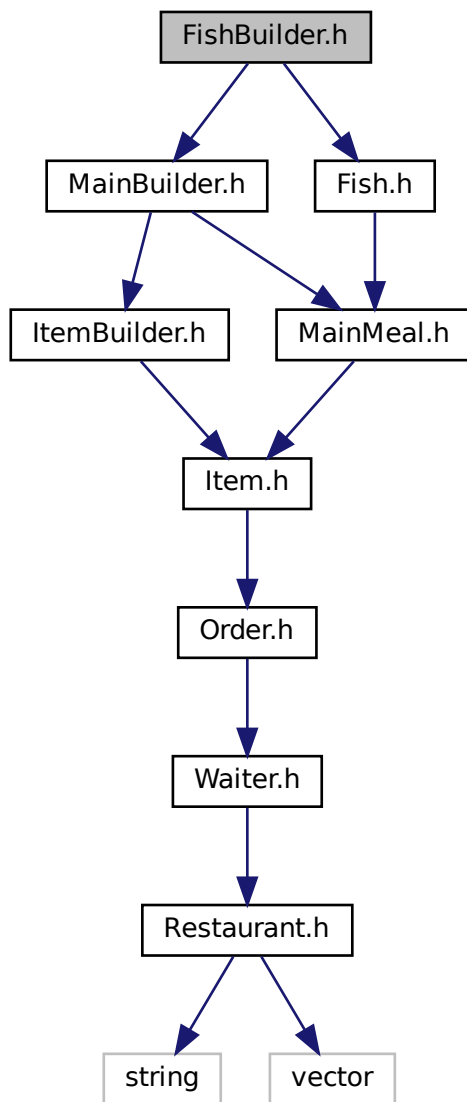
## 5.28 FishBuilder.h File Reference

Contains declaration for the [FishBuilder](#) class.

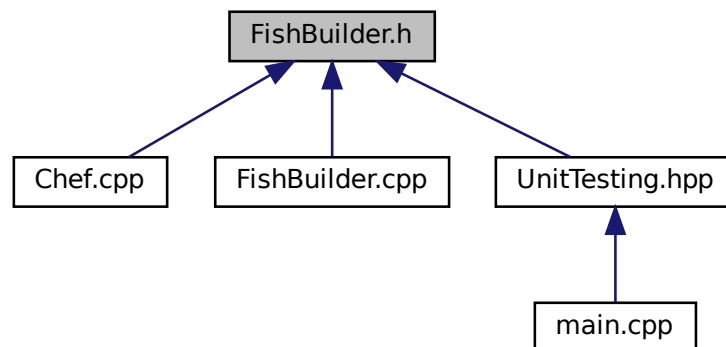
```
#include "MainBuilder.h"
```

```
#include "Fish.h"
```

Include dependency graph for FishBuilder.h:



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



## Classes

- class [FishBuilder](#)

### 5.28.1 Detailed Description

Contains declaration for the [FishBuilder](#) class.

[FishBuilder](#) is a derived class of [MainBuilder](#), representing a chef that prepares fish. [FishBuilder](#) is a concrete builder in the Builder design pattern.

#### Authors

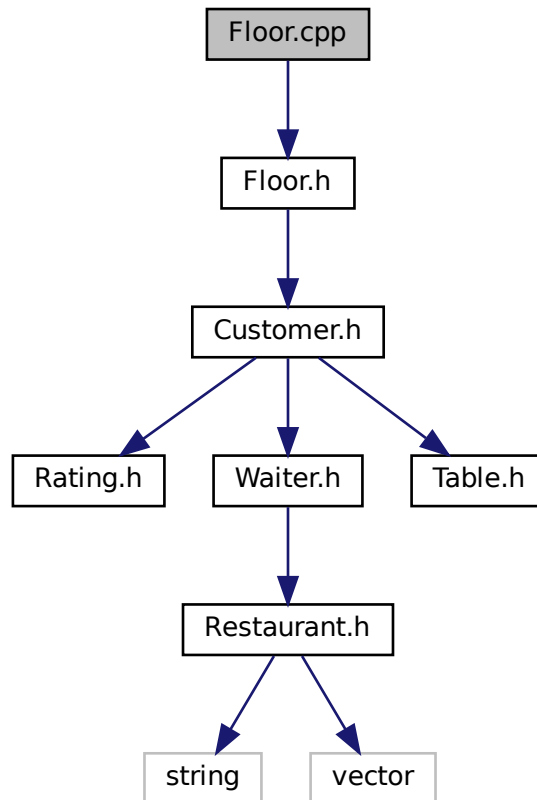
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.29 Floor.cpp File Reference

Contains implementation for the [Floor](#) class.

```
#include "Floor.h"
```

Include dependency graph for Floor.cpp:



### 5.29.1 Detailed Description

Contains implementation for the [Floor](#) class.

#### Authors

Aidan Chapman (u22738917)

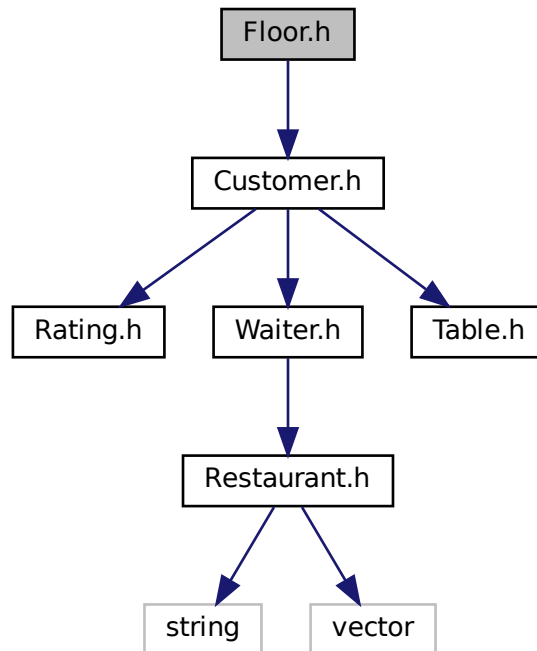
## 5.30 Floor.h File Reference

Contains declaration for the [Floor](#) class.

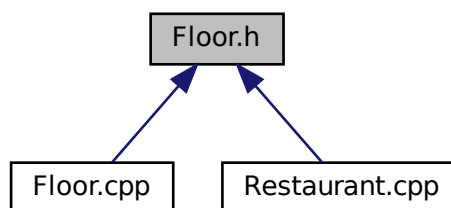


```
#include "Customer.h"
```

Include dependency graph for Floor.h:



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



## Classes

- class [Floor](#)

### 5.30.1 Detailed Description

Contains declaration for the [Floor](#) class.

[Floor](#) is a class that represents a floor in a restaurant. It contains a vector of [Table](#) pointers. It also contains a pointer to a [Restaurant](#) object.

#### Authors

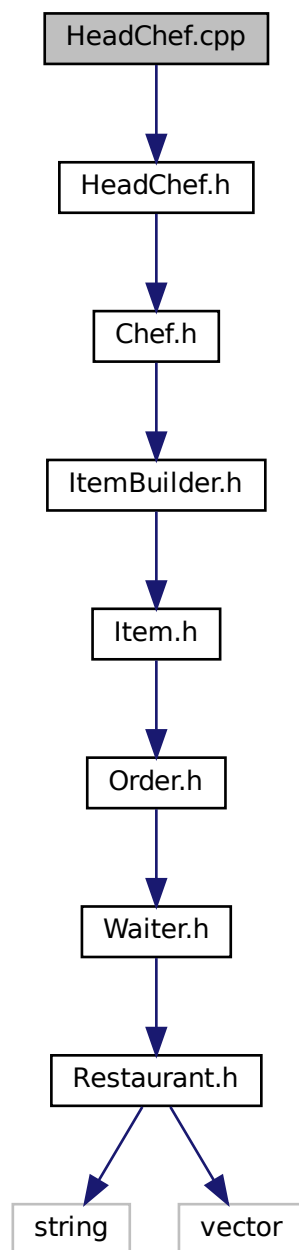
Aidan Chapman (u22738917)

## 5.31 HeadChef.cpp File Reference

Contains implementation for the [HeadChef](#) class.

```
#include "HeadChef.h"
```

Include dependency graph for HeadChef.cpp:



### 5.31.1 Detailed Description

Contains implementation for the [HeadChef](#) class.

#### Authors

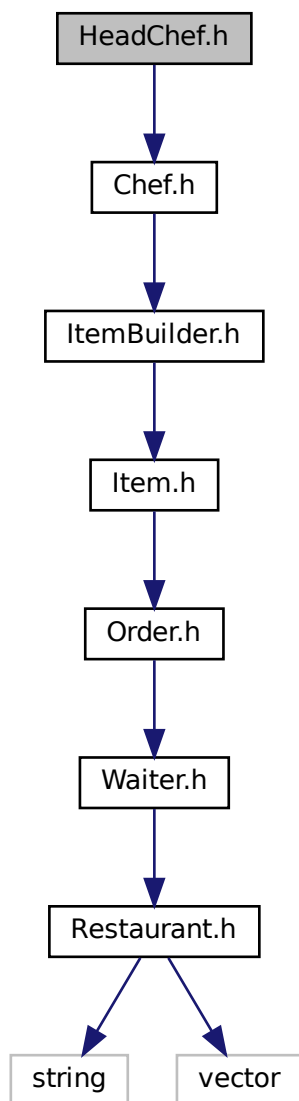
Aidan Chapman (u22738917)

## 5.32 HeadChef.h File Reference

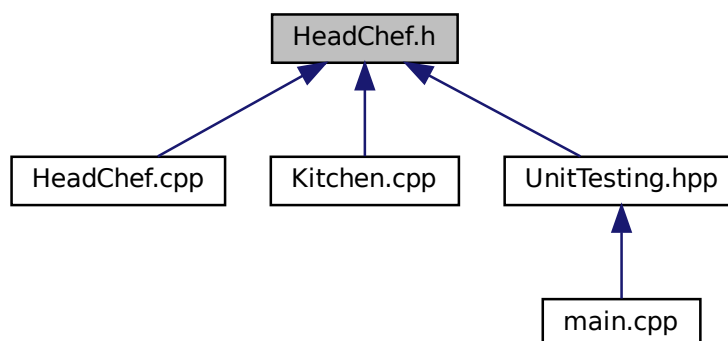
Contains declaration for the [HeadChef](#) class.

```
#include "Chef.h"
```

Include dependency graph for HeadChef.h:



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



## Classes

- class [HeadChef](#)

### 5.32.1 Detailed Description

Contains declaration for the [HeadChef](#) class.

[HeadChef](#) is a derived class of [Chef](#), representing a chef that prepares finishes meals.

#### Authors

Aidan Chapman (u22738917)

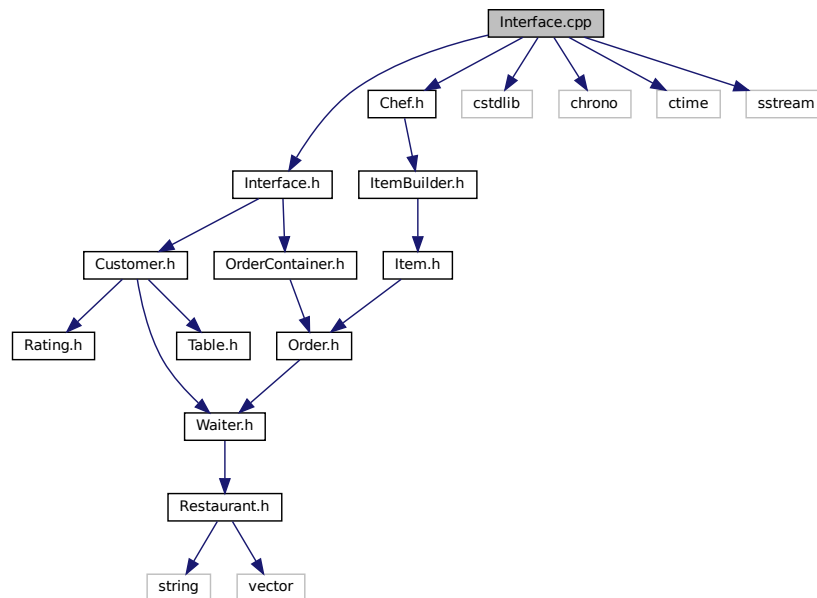
## 5.33 Interface.cpp File Reference

Contains implementation for the [Interface](#) class.

```
#include "Interface.h"
#include "Chef.h"
#include <cstdlib>
#include <chrono>
#include <ctime>
```

```
#include <sstream>
```

Include dependency graph for Interface.cpp:



### 5.33.1 Detailed Description

Contains implementation for the [Interface](#) class.

#### Authors

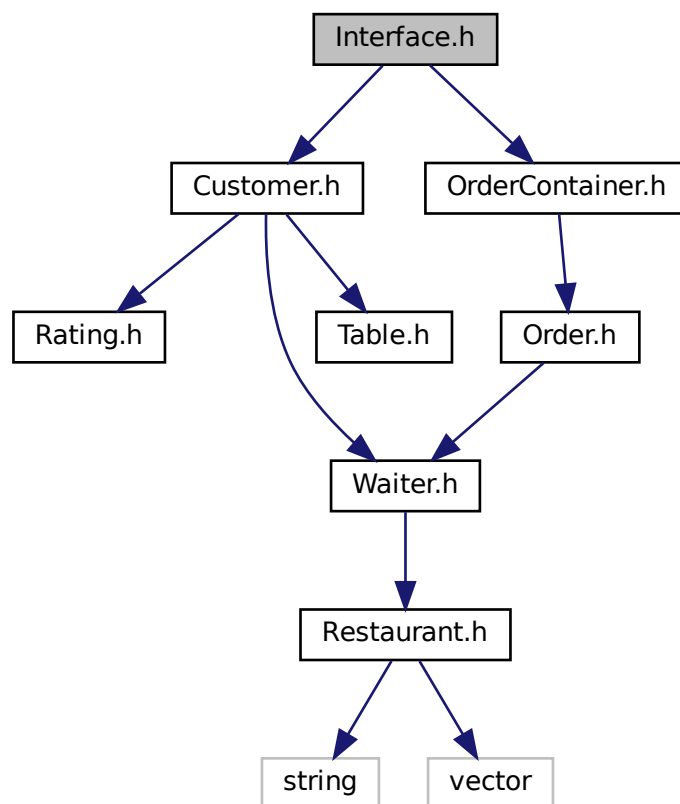
Aidan Chapman (u22738917), Douglas Porter (u21797545), Kabelo Chuene(u14046492)

## 5.34 Interface.h File Reference

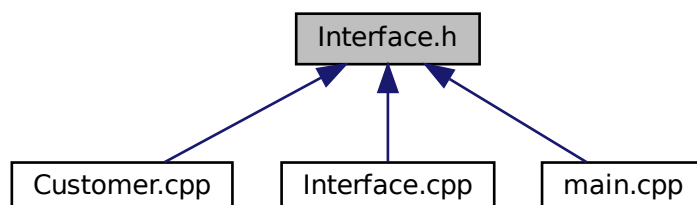
Contains declaration for the [Interface](#) class.

```
#include "Customer.h"
#include "OrderContainer.h"
```

Include dependency graph for Interface.h:



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



## Classes

- class [Interface](#)

### 5.34.1 Detailed Description

Contains declaration for the [Interface](#) class.

[Interface](#) is a class that handles the user interface for the program. Aggregates [Restaurant](#) and [Customer](#) objects and their functions.

#### Authors

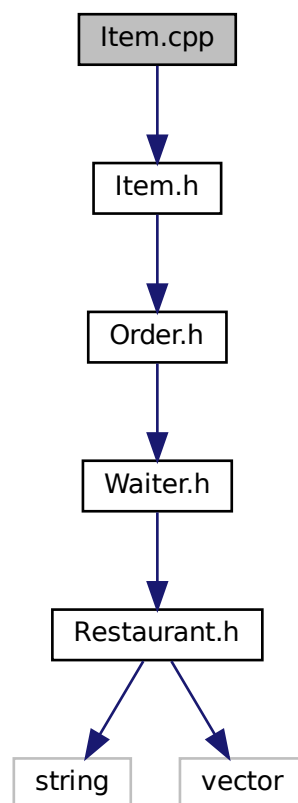
Aidan Chapman (u22738917)

## 5.35 Item.cpp File Reference

Contains implementation for the [Item](#) class.

```
#include "Item.h"
```

Include dependency graph for Item.cpp:





### 5.35.1 Detailed Description

Contains implementation for the [Item](#) class.

#### Authors

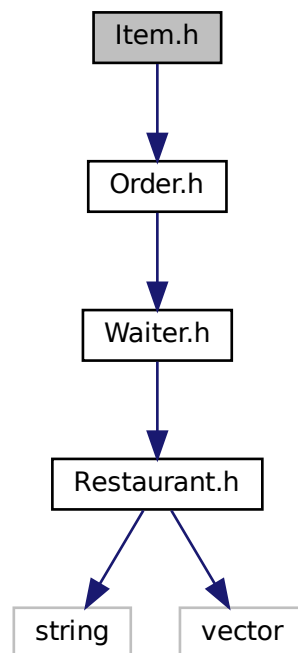
- Aidan Chapman (u22738917)
- Sange Tshakumane (u21479748)

## 5.36 Item.h File Reference

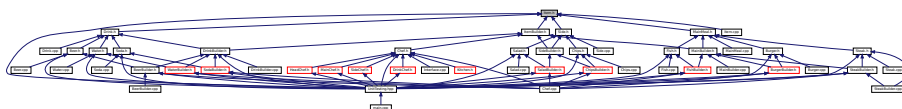
Contains declaration for the [Item](#) class.

```
#include "Order.h"
```

Include dependency graph for Item.h:



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



## Classes

- class [Item](#)

### 5.36.1 Detailed Description

Contains declaration for the [Item](#) class.

#### Authors

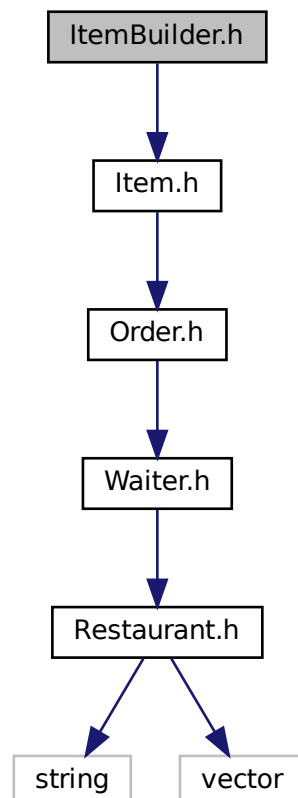
Aidan Chapman (u22738917)

## 5.37 ItemBuilder.h File Reference

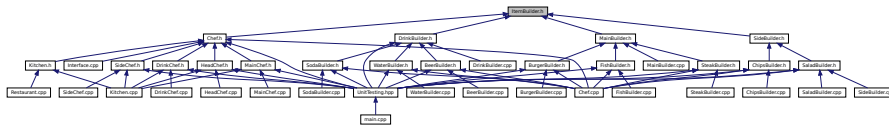
Contains declaration for the [ItemBuilder](#) class.

```
#include "Item.h"
```

Include dependency graph for ItemBuilder.h:



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



## Classes

- class [ItemBuilder](#)

### 5.37.1 Detailed Description

Contains declaration for the [ItemBuilder](#) class.

Superclass for all ItemBuilders, which are used to build Items

#### Authors

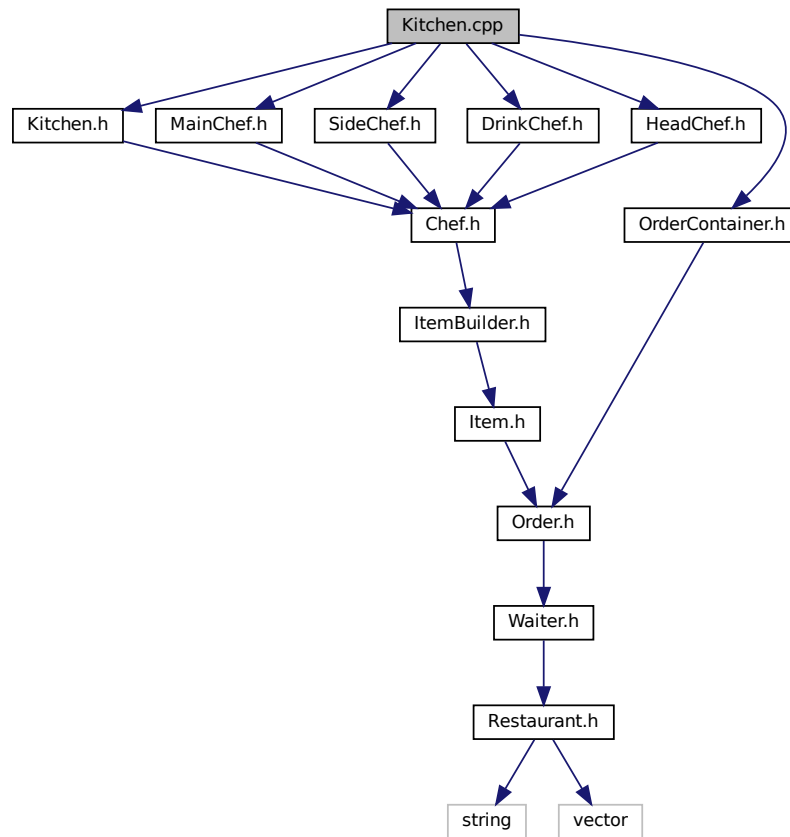
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.38 Kitchen.cpp File Reference

Contains implementation for the [Kitchen](#) class.

```
#include "Kitchen.h"
#include "MainChef.h"
#include "SideChef.h"
#include "DrinkChef.h"
#include "HeadChef.h"
#include "OrderContainer.h"
```

Include dependency graph for Kitchen.cpp:



### 5.38.1 Detailed Description

Contains implementation for the [Kitchen](#) class.

#### Authors

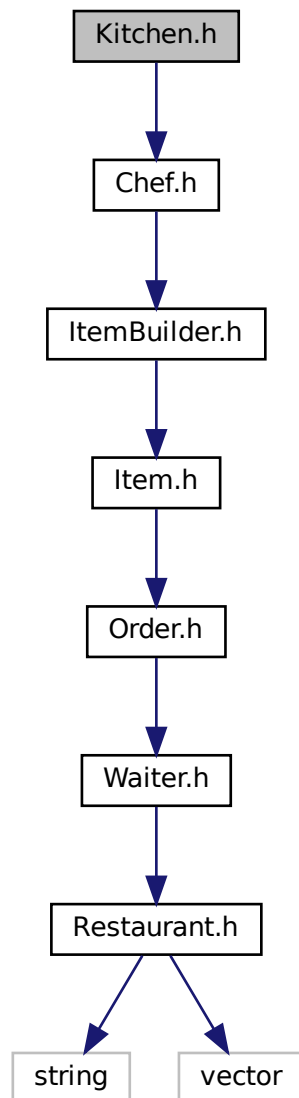
Aidan Chapman (u22738917)

## 5.39 Kitchen.h File Reference

Contains declaration for the [Kitchen](#) class.

```
#include "Chef.h"
```

Include dependency graph for Kitchen.h:





## Functions

- void **run** ()
- int **main** ()

*A function used to run the program.*

### 5.40.1 Detailed Description

This is the file that the user will interact with.

#### Authors

Aidan Chapman (u22738917)

### 5.40.2 Function Documentation

#### 5.40.2.1 main()

```
int main ( )
```

A function used to run the program.

#### Authors

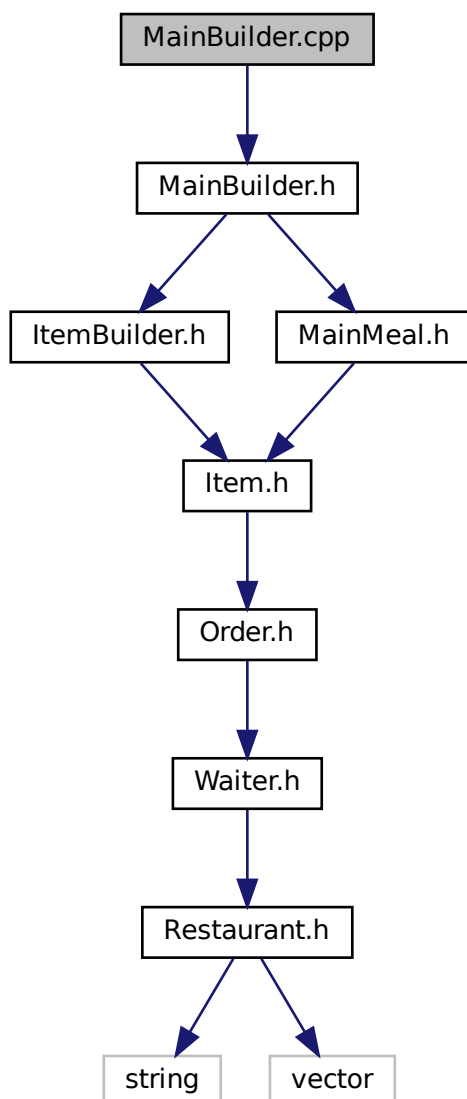
Aidan Chapman (u22738917)

## 5.41 MainBuilder.cpp File Reference

Contains the implementation for the [MainBuilder](#) class.

```
#include "MainBuilder.h"
```

Include dependency graph for MainBuilder.cpp:



### 5.41.1 Detailed Description

Contains the implementation for the [MainBuilder](#) class.

#### Authors

Douglas Porter (u21797545)



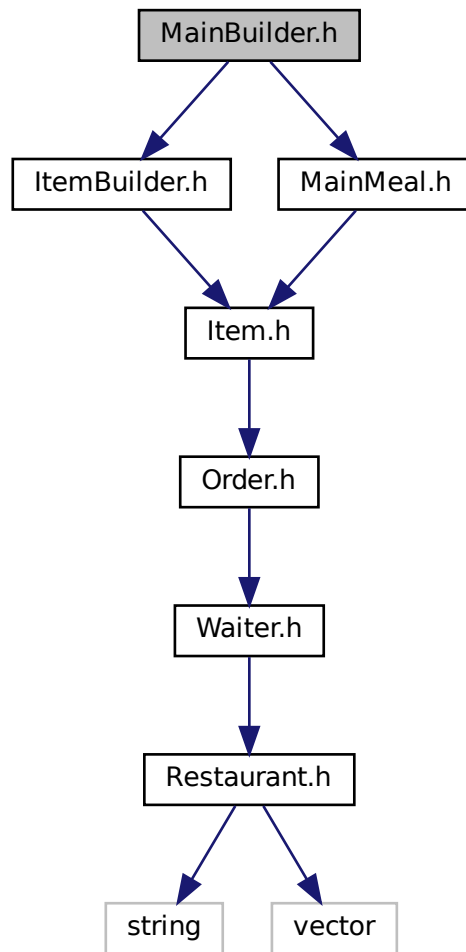
## 5.42 MainBuilder.h File Reference

Contains declaration for the [MainBuilder](#) class.

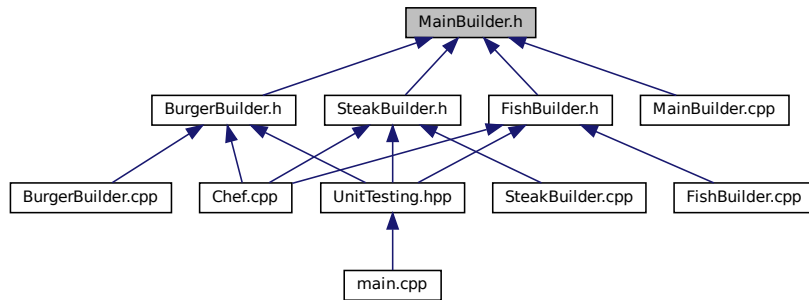
```
#include "ItemBuilder.h"
```

```
#include "MainMeal.h"
```

Include dependency graph for MainBuilder.h:



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



## Classes

- class [MainBuilder](#)

### 5.42.1 Detailed Description

Contains declaration for the [MainBuilder](#) class.

[MainBuilder](#) is a derived class of [ItemBuilder](#), representing a builder that prepares main meals.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)
- Douglas Porter (u21797545)

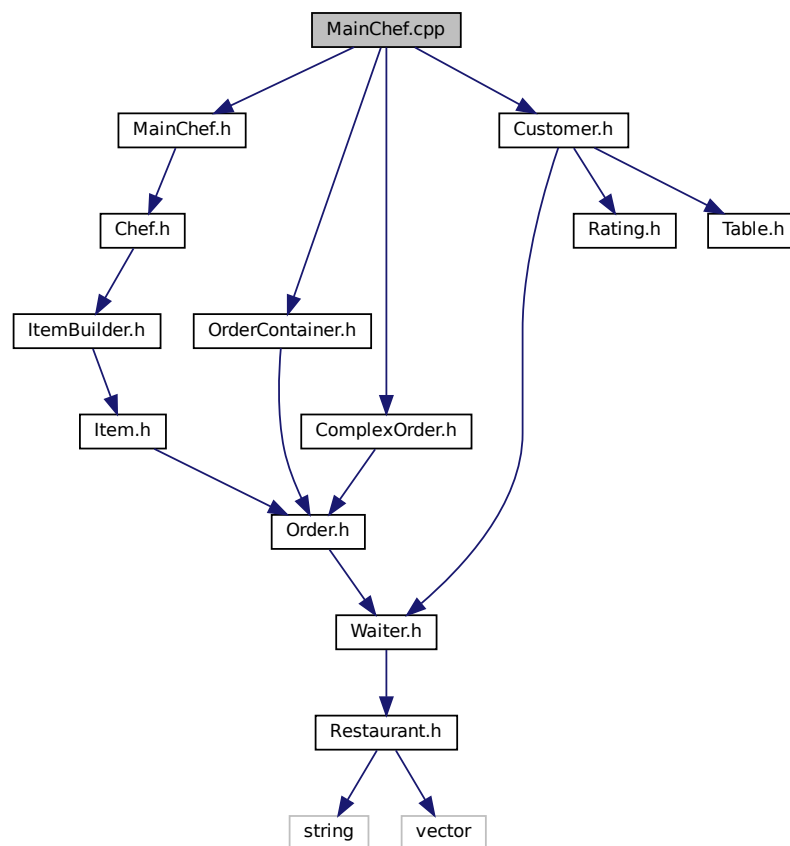
## 5.43 MainChef.cpp File Reference

Contains implementation for the [MainChef](#) class.

```
#include "MainChef.h"
#include "Customer.h"
#include "OrderContainer.h"
```

```
#include "ComplexOrder.h"
```

Include dependency graph for MainChef.cpp:



### 5.43.1 Detailed Description

Contains implementation for the [MainChef](#) class.

#### Authors

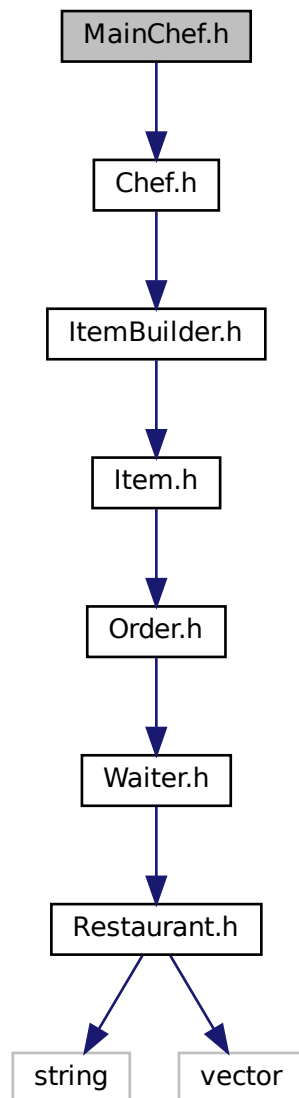
Aidan Chapman (u22738917)

## 5.44 MainChef.h File Reference

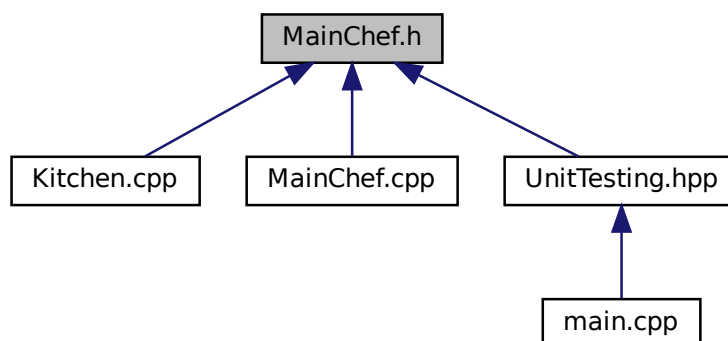
Contains declaration for the [MainChef](#) class.

```
#include "Chef.h"
```

Include dependency graph for MainChef.h:



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



## Classes

- class [MainChef](#)

### 5.44.1 Detailed Description

Contains declaration for the [MainChef](#) class.

The [MainChef](#) class is a concrete class which inherits from the [Chef](#) class. It represents a chef that prepares main meals.

#### Authors

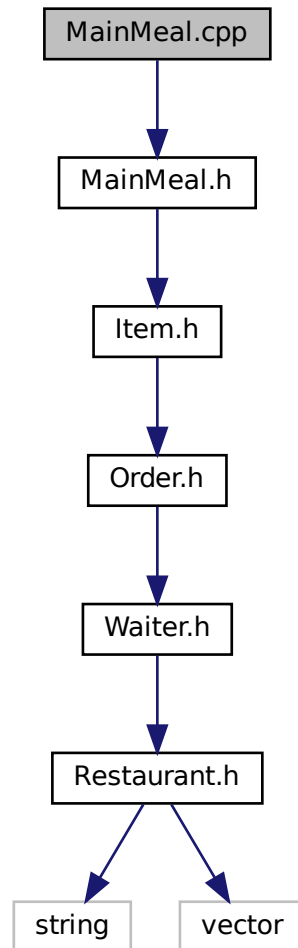
Aidan Chapman (u22738917)

## 5.45 MainMeal.cpp File Reference

Contains implementation for the [MainMeal](#) class.

```
#include "MainMeal.h"
```

Include dependency graph for MainMeal.cpp:



### 5.45.1 Detailed Description

Contains implementation for the [MainMeal](#) class.

#### Authors

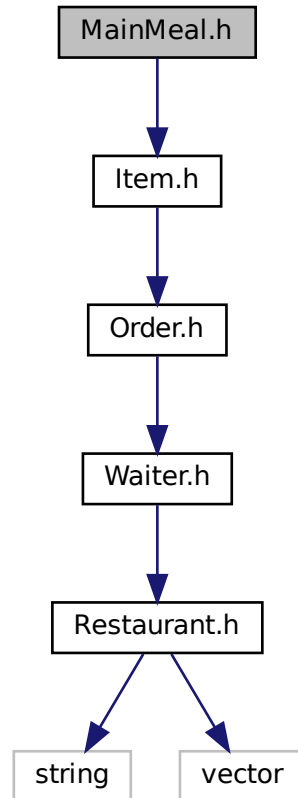
- Aidan Chapman (u22738917)
- Sange Tshakumane (u21479748)

## 5.46 MainMeal.h File Reference

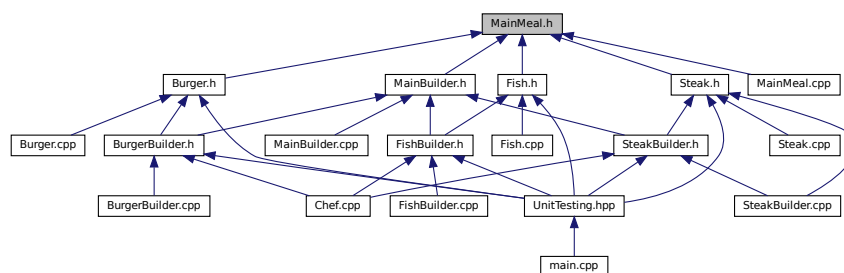
Contains declaration for the [MainMeal](#) class.

```
#include "Item.h"
```

Include dependency graph for MainMeal.h:



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



## Classes

- class [MainMeal](#)

### 5.46.1 Detailed Description

Contains declaration for the [MainMeal](#) class.

[MainMeal](#) is a derived class of [Item](#), representing a main meal.

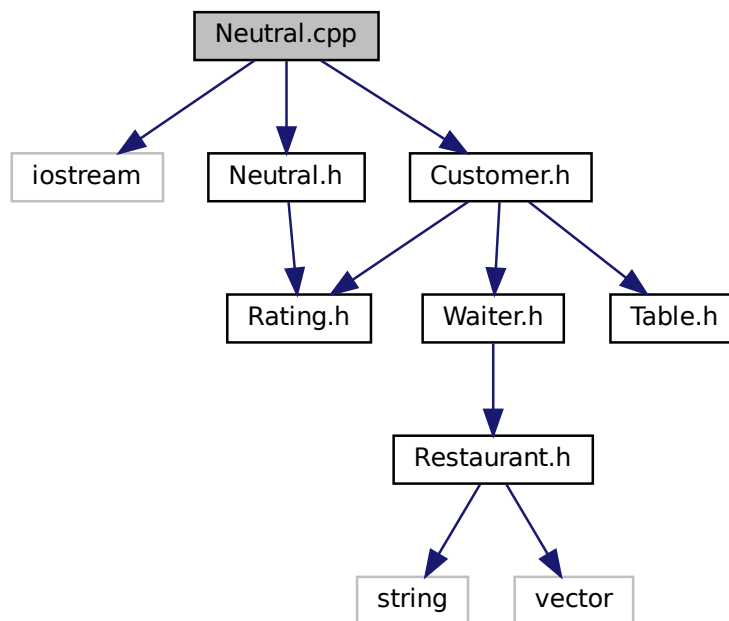
#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.47 Neutral.cpp File Reference

Contains implementation for the [Neutral](#) class.

```
#include <iostream>
#include "Neutral.h"
#include "Customer.h"
Include dependency graph for Neutral.cpp:
```



### 5.47.1 Detailed Description

Contains implementation for the [Neutral](#) class.

#### Authors

Sange Tshakumane (u21479748)

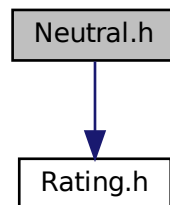


## 5.48 Neutral.h File Reference

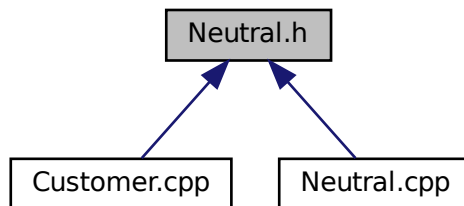
Contains declaration for the [Neutral](#) class.

```
#include "Rating.h"
```

Include dependency graph for Neutral.h:



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



### Classes

- class [Neutral](#)

#### 5.48.1 Detailed Description

Contains declaration for the [Neutral](#) class.

[Neutral](#) is a derived class of [Rating](#), representing a neutral rating. [Neutral](#) ratings do not affect the tip.

#### Authors

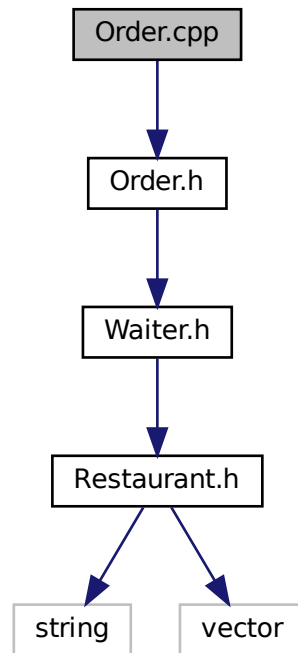
- Aidan Chapman (u22738917)
- Sange Tshakumane (u21479748)

## 5.49 Order.cpp File Reference

Contains implementation for the [Order](#) class.

```
#include "Order.h"
```

Include dependency graph for Order.cpp:



### 5.49.1 Detailed Description

Contains implementation for the [Order](#) class.

#### Authors

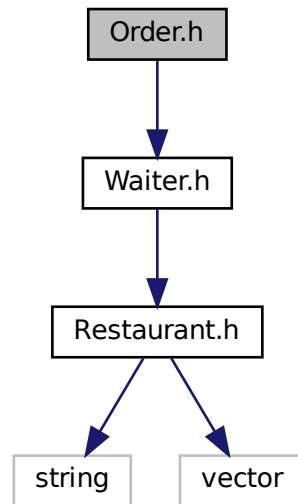
Sange Tshakumane (u21479748)

## 5.50 Order.h File Reference

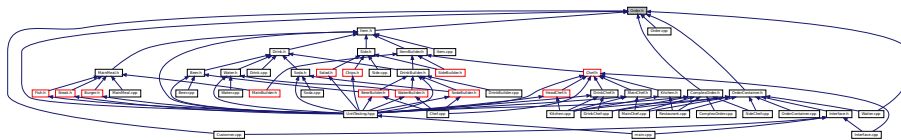
Contains declaration for the [Order](#) class.

```
#include "Waiter.h"
```

Include dependency graph for Order.h:



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



## Classes

- class [Order](#)

### 5.50.1 Detailed Description

Contains declaration for the [Order](#) class.

This file contains the declaration for the [Order](#) class. [Order](#) is an abstract class that is used to represent an order.

#### Authors

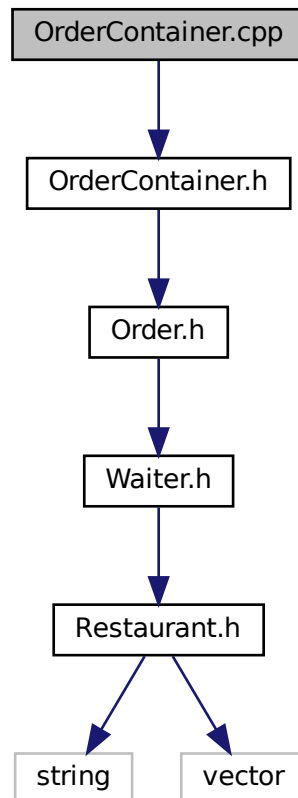
Aidan Chapman (u22738917)

## 5.51 OrderContainer.cpp File Reference

Contains implementation for the [OrderContainer](#) class.

```
#include "OrderContainer.h"
```

Include dependency graph for OrderContainer.cpp:



### 5.51.1 Detailed Description

Contains implementation for the [OrderContainer](#) class.

#### Authors

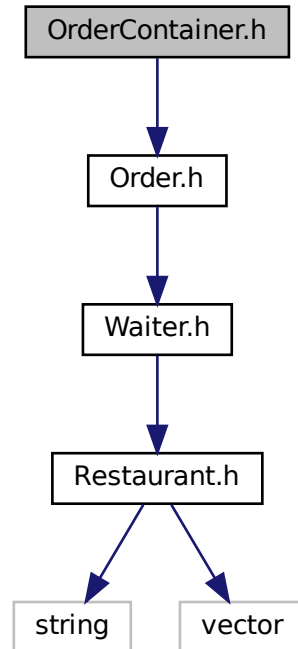
Aidan Chapman (u22738917)

## 5.52 OrderContainer.h File Reference

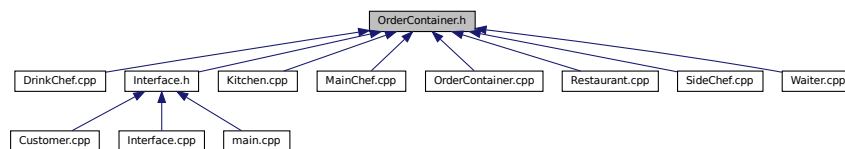
The [OrderContainer](#) class represents a container for an [Order](#) object and its corresponding requested order string.

```
#include "Order.h"
```

Include dependency graph for OrderContainer.h:



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



## Classes

- class [OrderContainer](#)

### 5.52.1 Detailed Description

The [OrderContainer](#) class represents a container for an [Order](#) object and its corresponding requested order string.

The [OrderContainer](#) class is used to pass orders to the [Chef](#) class.

#### Authors

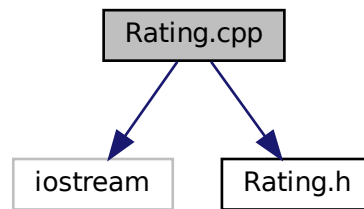
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)
- Sange Tshakumane (u21479748)

## 5.53 Rating.cpp File Reference

Contains implementation for the [Rating](#) class.

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

Include dependency graph for Rating.cpp:



### 5.53.1 Detailed Description

Contains implementation for the [Rating](#) class.

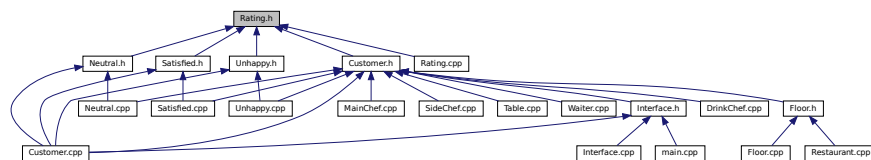
#### Authors

Sange Tshakumane (u21479748)

## 5.54 Rating.h File Reference

Contains declaration for the [Rating](#) class.

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



### Classes

- class [Rating](#)

### 5.54.1 Detailed Description

Contains declaration for the [Rating](#) class.

[Rating](#) is an abstract class that represents a rating. Ratings are used to calculate the tip for a customer.

#### Authors

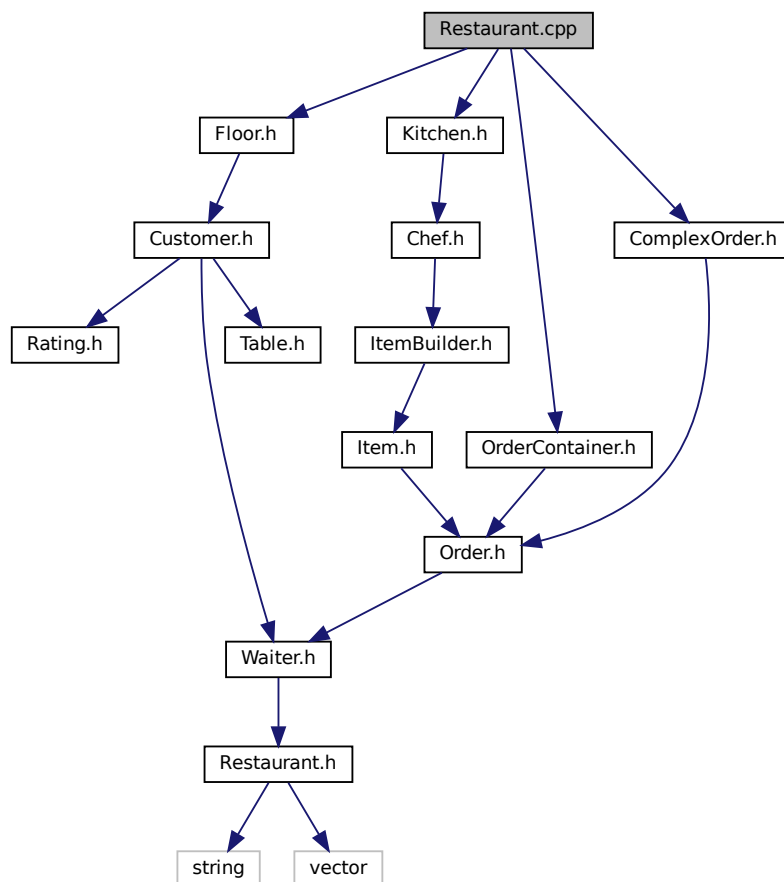
- Aidan Chapman (u22738917)
- Sange Tshakumane (u21479748)

## 5.55 Restaurant.cpp File Reference

Contains implementation for the [Restaurant](#) class.

```
#include "Floor.h"  
#include "Kitchen.h"  
#include "OrderContainer.h"  
#include "ComplexOrder.h"
```

Include dependency graph for Restaurant.cpp:



### 5.55.1 Detailed Description

Contains implementation for the [Restaurant](#) class.

#### Authors

Aidan Chapman (u22738917)

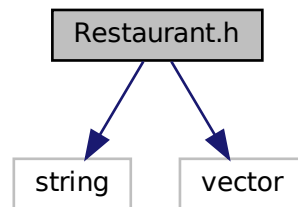
## 5.56 Restaurant.h File Reference

Contains declaration for the [Restaurant](#) class.

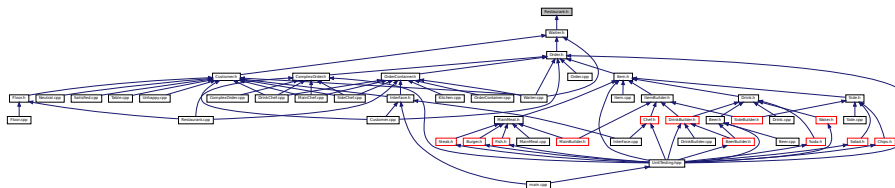
```
#include <string>
```

```
#include <vector>
```

Include dependency graph for Restaurant.h:



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



## Classes

- class [Restaurant](#)

### 5.56.1 Detailed Description

Contains declaration for the [Restaurant](#) class.

[Restaurant](#) class is the mediator class of the program. It contains the floor and kitchen and is responsible for seating customers, placing orders, and handling the flow of the program.

#### Authors

Aidan Chapman (u22738917)

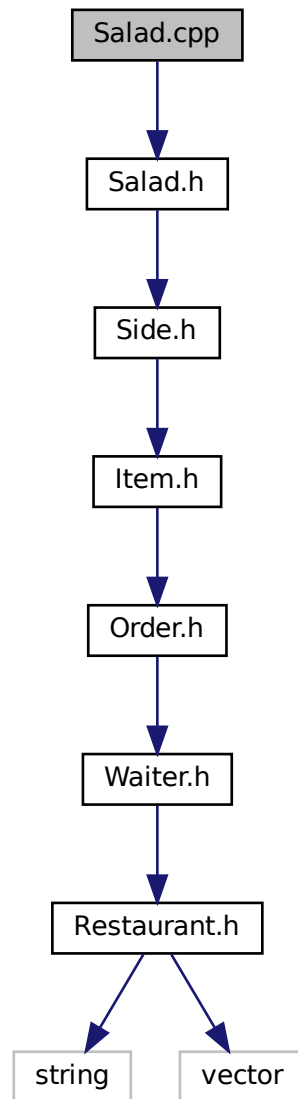


## 5.57 Salad.cpp File Reference

Contains implementation for the [Salad](#) class.

```
#include "Salad.h"
```

Include dependency graph for Salad.cpp:



### 5.57.1 Detailed Description

Contains implementation for the [Salad](#) class.

#### Authors

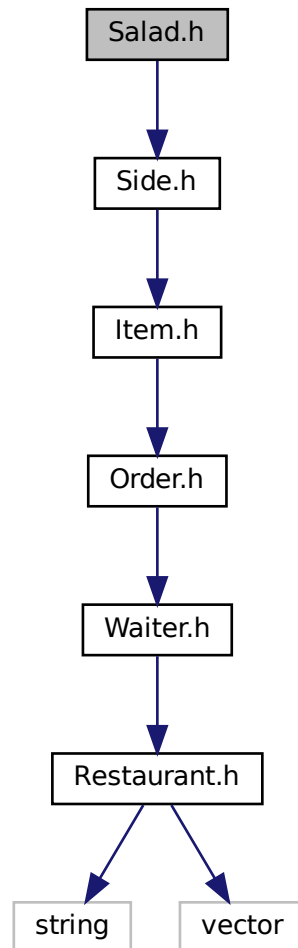
Aidan Chapman (u22738917)

## 5.58 Salad.h File Reference

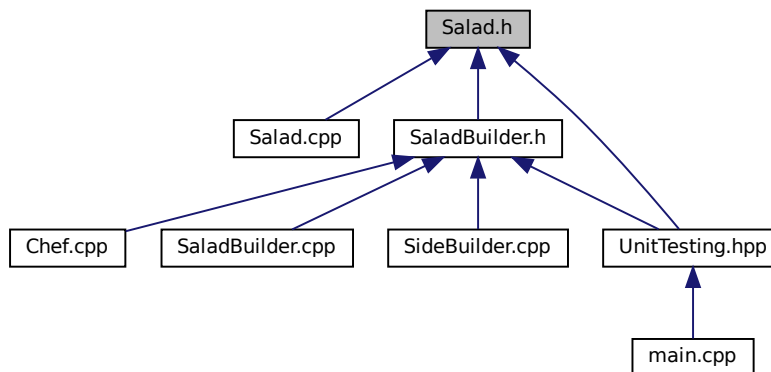
Contains declaration for the [Salad](#) class.

```
#include "Side.h"
```

Include dependency graph for Salad.h:



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



## Classes

- class [Salad](#)

### 5.58.1 Detailed Description

Contains declaration for the [Salad](#) class.

#### Authors

Aidan Chapman (u22738917)

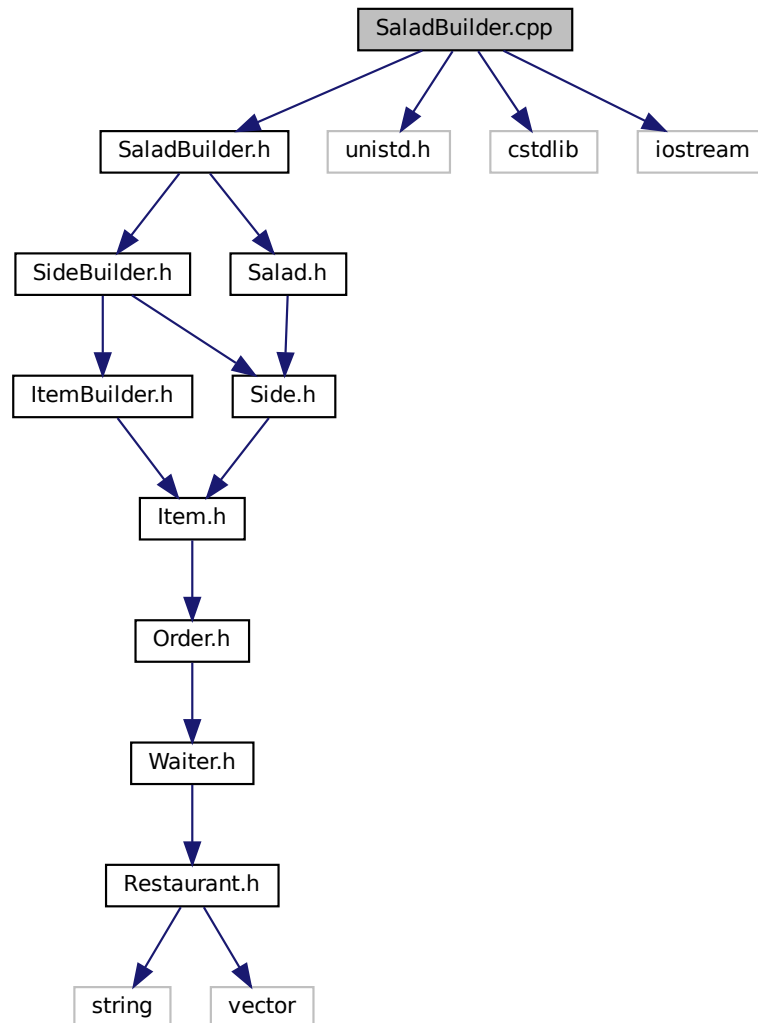
## 5.59 SaladBuilder.cpp File Reference

Contains implementation for the [SaladBuilder](#) class.

```
#include "SaladBuilder.h"
#include <unistd.h>
#include <cstdlib>
```

```
#include <iostream>
```

Include dependency graph for SaladBuilder.cpp:



### 5.59.1 Detailed Description

Contains implementation for the [SaladBuilder](#) class.

#### Authors

- Graeme Blain (u22625462)

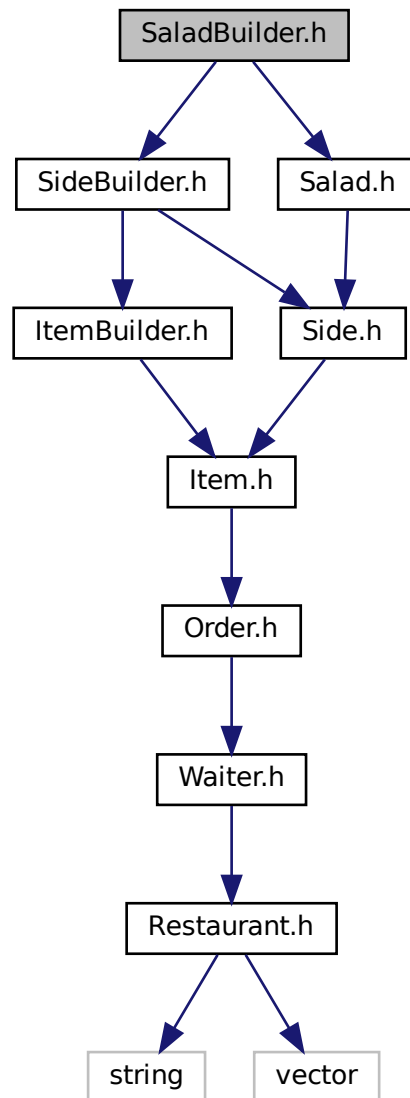
## 5.60 SaladBuilder.h File Reference

Contains declaration for the [SaladBuilder](#) class.

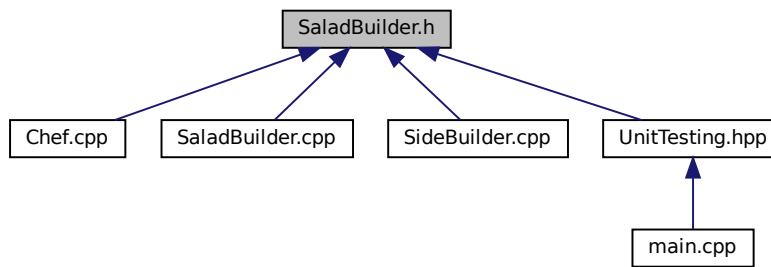
```
#include "SideBuilder.h"
```

```
#include "Salad.h"
```

Include dependency graph for SaladBuilder.h:



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



## Classes

- class [SaladBuilder](#)

### 5.60.1 Detailed Description

Contains declaration for the [SaladBuilder](#) class.

[SaladBuilder](#) is a concrete builder for the [Salad](#) class and inherits from [SideBuilder](#). It is responsible for building a [Salad](#) object using the template method pattern.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

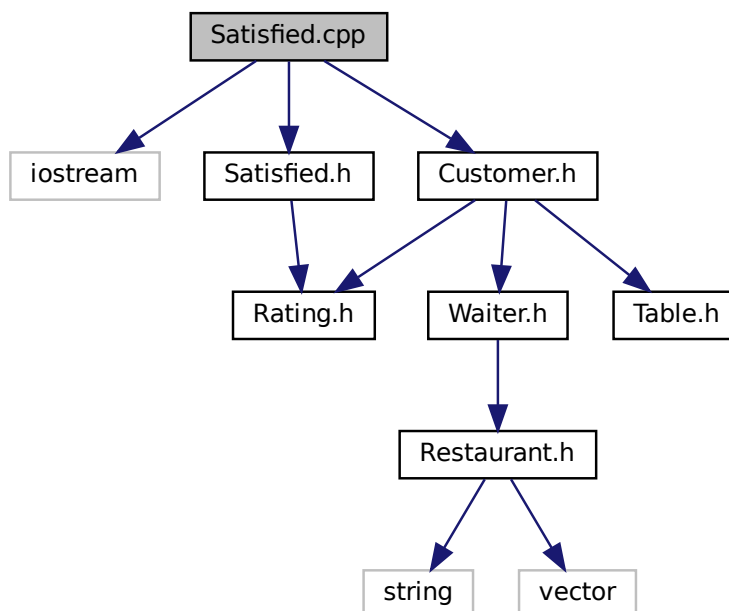
## 5.61 Satisfied.cpp File Reference

Contains implementation for the [Satisfied](#) class.

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

```
#include "Customer.h"
```

Include dependency graph for Satisfied.cpp:



### 5.61.1 Detailed Description

Contains implementation for the [Satisfied](#) class.

#### Authors

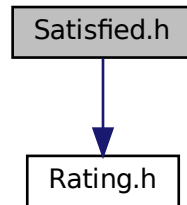
Sange Tshakumane (u21479748)

## 5.62 Satisfied.h File Reference

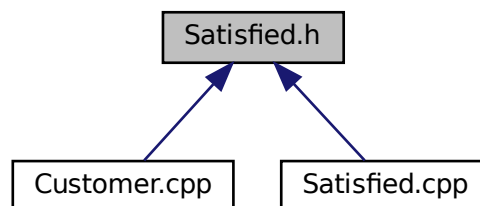
Contains declaration for the [Satisfied](#) class.

```
#include "Rating.h"
```

Include dependency graph for Satisfied.h:



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



## Classes

- class [Satisfied](#)

### 5.62.1 Detailed Description

Contains declaration for the [Satisfied](#) class.

This class is a concrete implementation of the [Rating](#) class. It is used to represent a satisfied customer. It is used to calculate the tip for a customer. Satisfied customers tip 25% of their bill.

#### Authors

Aidan Chapman (u22738917), Sange Tshakumane (u21479748)

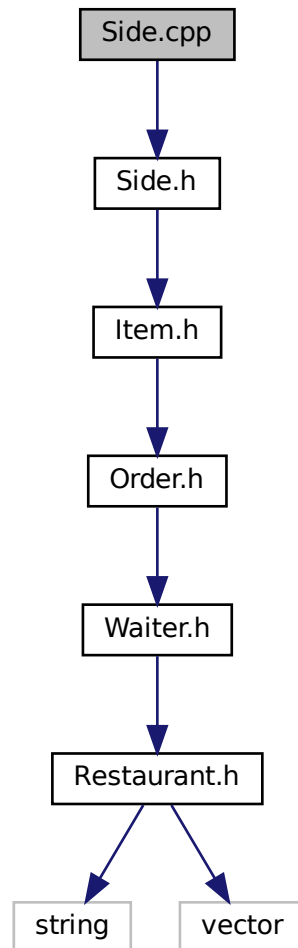


## 5.63 Side.cpp File Reference

Contains implementation for the [Side](#) class.

```
#include "Side.h"
```

Include dependency graph for Side.cpp:



### 5.63.1 Detailed Description

Contains implementation for the [Side](#) class.

#### Authors

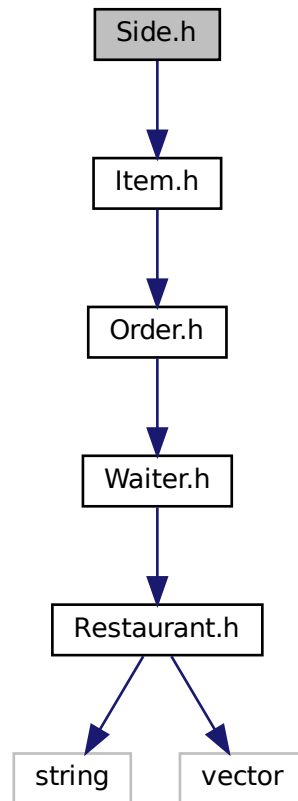
- Aidan Chapman (u22738917)
- Sange Tshakumane (u21479748)

## 5.64 Side.h File Reference

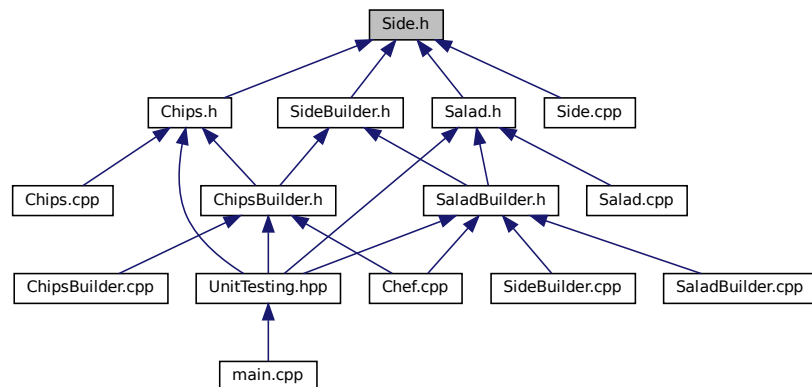
Contains declaration for the [Side](#) class.

```
#include "Item.h"
```

Include dependency graph for Side.h:



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



## Classes

- class [Side](#)

### 5.64.1 Detailed Description

Contains declaration for the [Side](#) class.

#### Authors

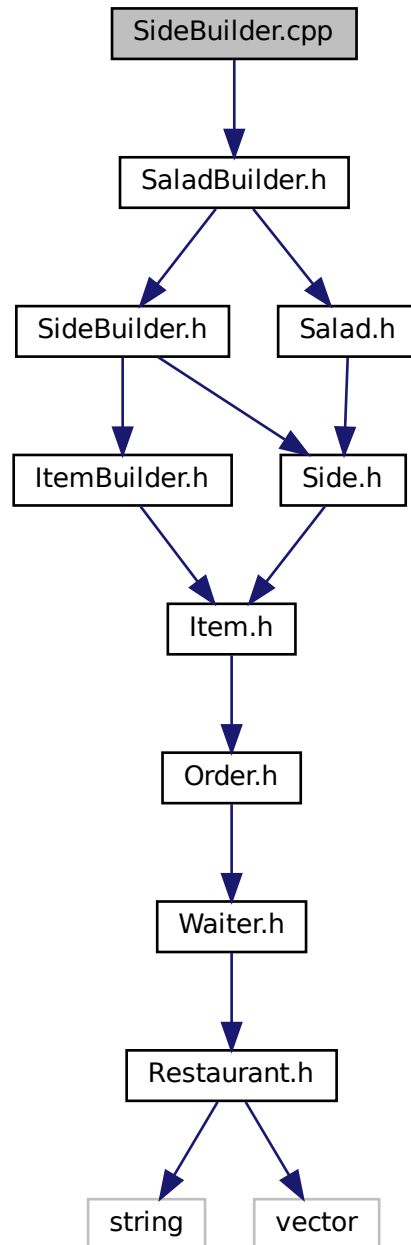
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.65 SideBuilder.cpp File Reference

Contains implementation for the [SideBuilder](#) class.

```
#include "SaladBuilder.h"
```

Include dependency graph for SideBuilder.cpp:



### 5.65.1 Detailed Description

Contains implementation for the [SideBuilder](#) class.

#### Authors

- Graeme Blain (u22625462)

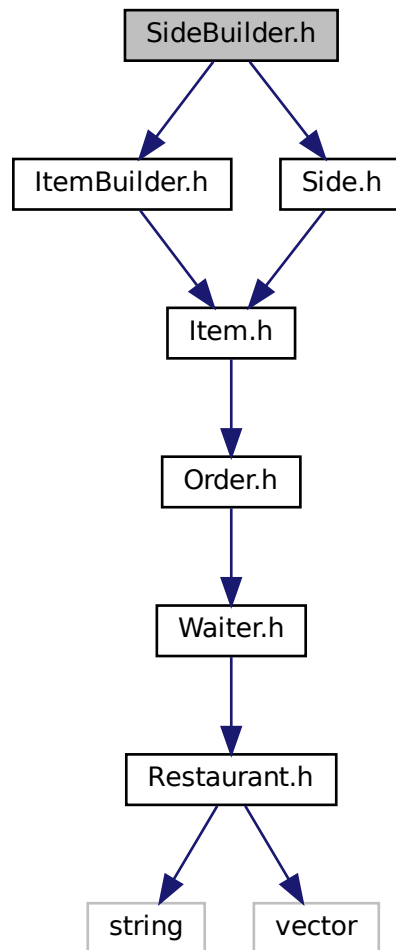
## 5.66 SideBuilder.h File Reference

Contains declaration for the [SideBuilder](#) class.

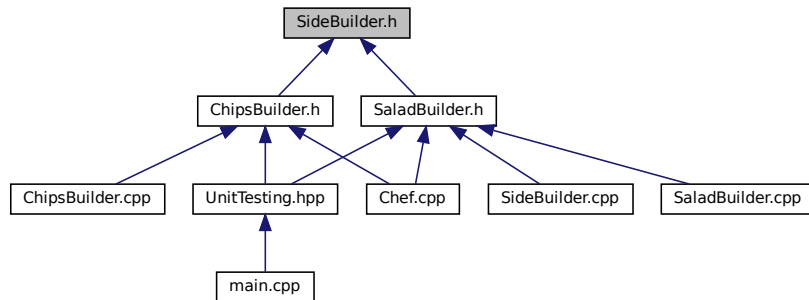
```
#include "ItemBuilder.h"
```

```
#include "Side.h"
```

Include dependency graph for SideBuilder.h:



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



## Classes

- class [SideBuilder](#)

### 5.66.1 Detailed Description

Contains declaration for the [SideBuilder](#) class.

[SideBuilder](#) is an abstract class that inherits from [ItemBuilder](#). It is used to build [Side](#) objects. It contains functions to prepare ingredients, assemble the item, and get the item.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

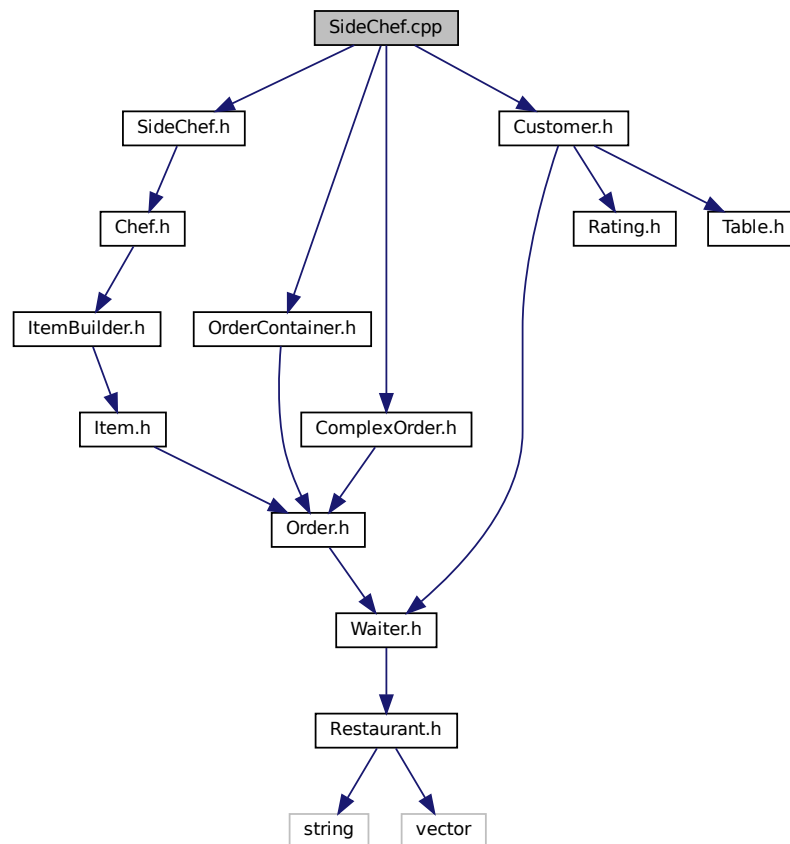
## 5.67 SideChef.cpp File Reference

Contains implementation for the [SideChef](#) class.

```
#include "SideChef.h"
#include "Customer.h"
#include "OrderContainer.h"
```

```
#include "ComplexOrder.h"
```

Include dependency graph for SideChef.cpp:



### 5.67.1 Detailed Description

Contains implementation for the [SideChef](#) class.

#### Authors

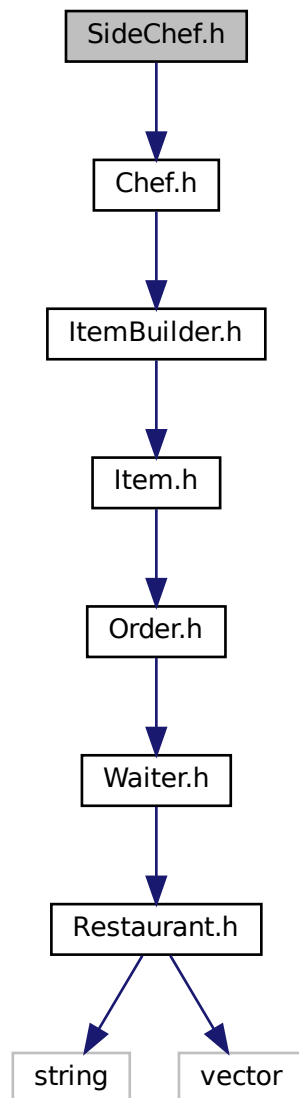
Aidan Chapman (u22738917)

## 5.68 SideChef.h File Reference

Contains declaration for the [SideChef](#) class.

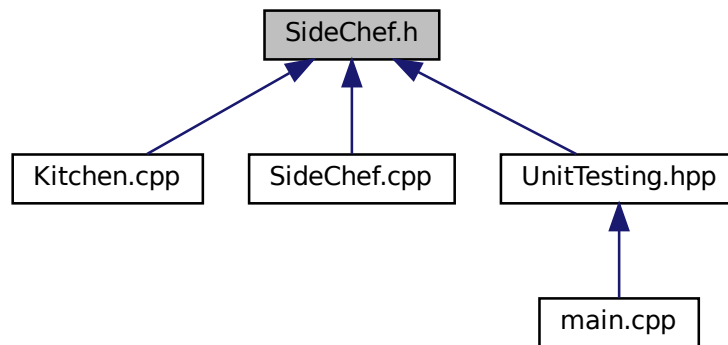
```
#include "Chef.h"
```

Include dependency graph for SideChef.h:





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



## Classes

- class [SideChef](#)

### 5.68.1 Detailed Description

Contains declaration for the [SideChef](#) class.

[SideChef](#) is a concrete class which inherits from [Chef](#). It is responsible for preparing the side part of the order.

#### Authors

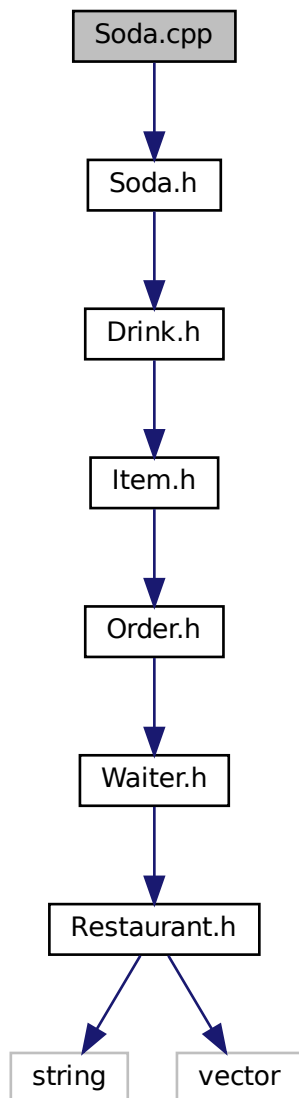
Aidan Chapman (u22738917)

## 5.69 Soda.cpp File Reference

Contains implementation for the [Soda](#) class.

```
#include "Soda.h"
```

Include dependency graph for Soda.cpp:



### 5.69.1 Detailed Description

Contains implementation for the [Soda](#) class.

#### Authors

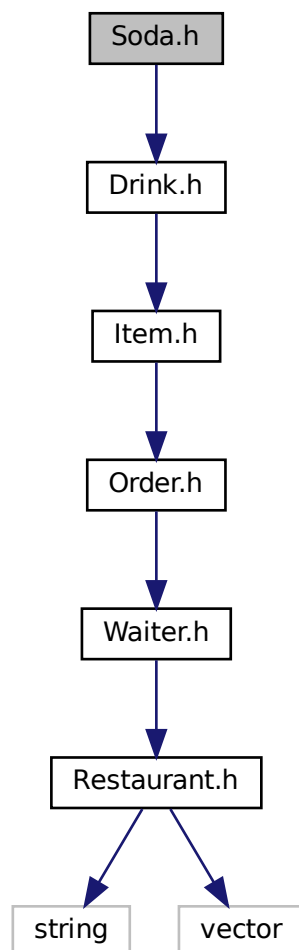
Aidan Chapman (u22738917)

## 5.70 Soda.h File Reference

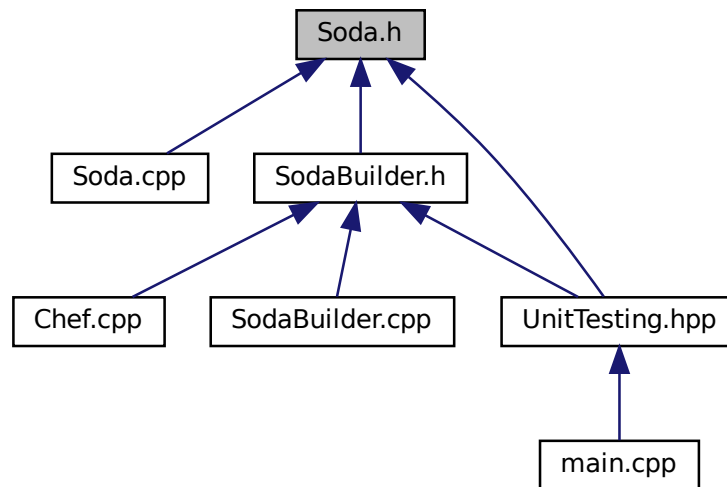
Contains declaration for the [Soda](#) class.

```
#include "Drink.h"
```

Include dependency graph for Soda.h:



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



## Classes

- class [Soda](#)

### 5.70.1 Detailed Description

Contains declaration for the [Soda](#) class.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

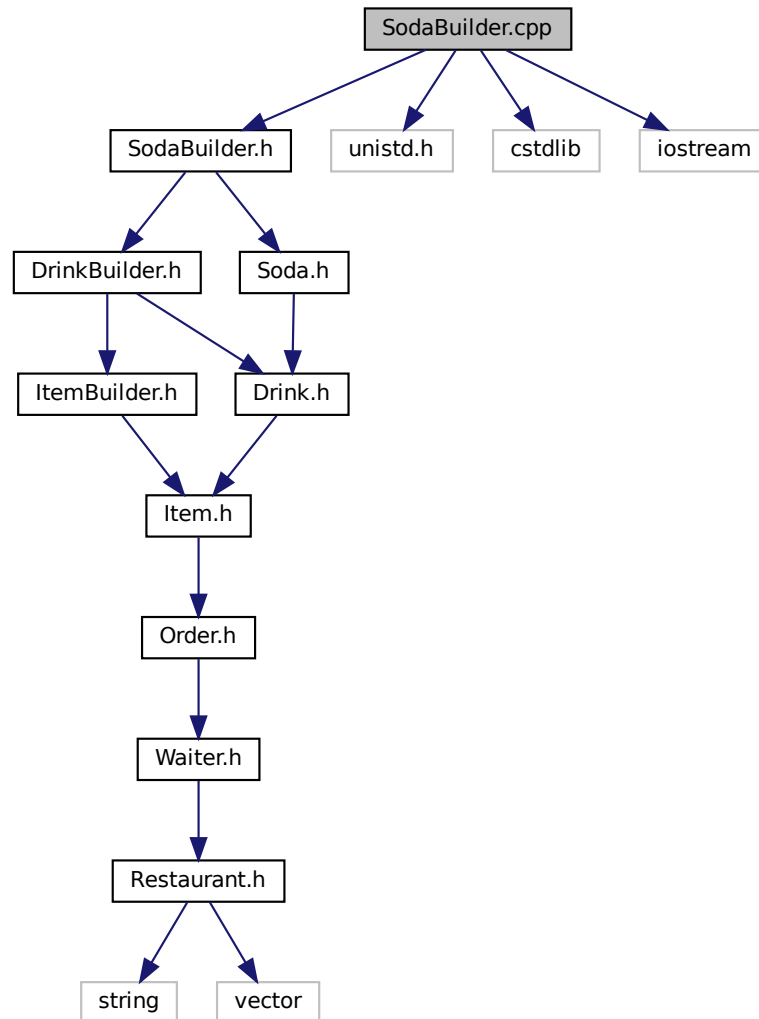
## 5.71 SodaBuilder.cpp File Reference

Contains implementation for the [SodaBuilder](#) class.

```
#include "SodaBuilder.h"  
#include <unistd.h>  
#include <cstdlib>
```

```
#include <iostream>
```

Include dependency graph for SodaBuilder.cpp:



### 5.71.1 Detailed Description

Contains implementation for the [SodaBuilder](#) class.

#### Authors

- Graeme Blain (u22625462)

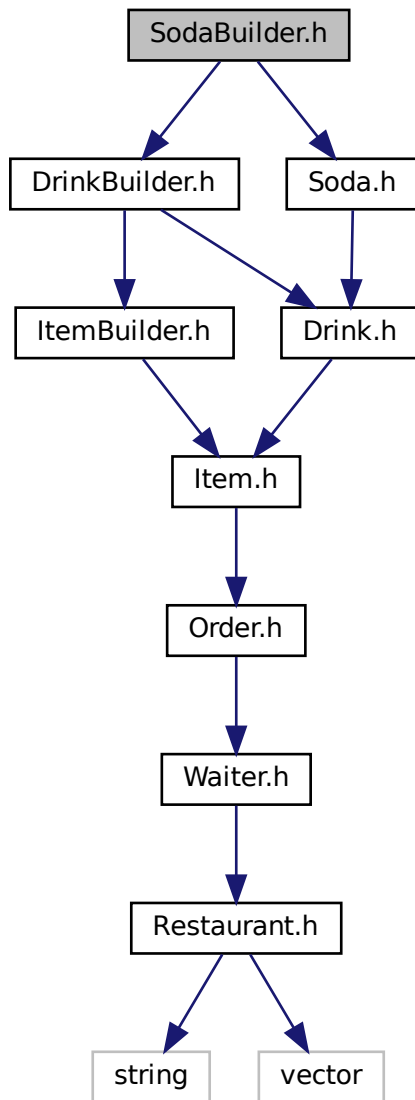
## 5.72 SodaBuilder.h File Reference

Contains declaration for the [SodaBuilder](#) class.

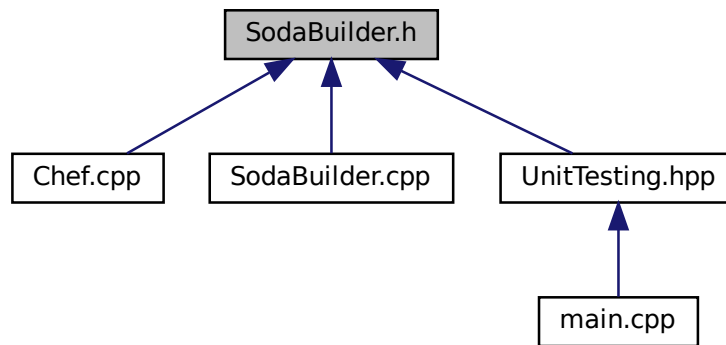
```
#include "DrinkBuilder.h"
```

```
#include "Soda.h"
```

Include dependency graph for SodaBuilder.h:



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



## Classes

- class [SodaBuilder](#)

### 5.72.1 Detailed Description

Contains declaration for the [SodaBuilder](#) class.

[SodaBuilder](#) is a concrete builder for the [DrinkBuilder](#) interface. It is used to create a [Soda](#) object. It is responsible for building a [Soda](#) object using the template method pattern.

#### Authors

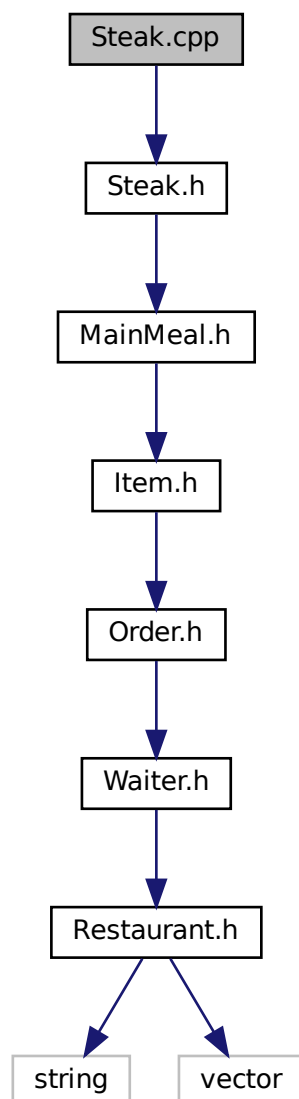
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.73 Steak.cpp File Reference

Contains implementation for the [Steak](#) class.

```
#include "Steak.h"
```

Include dependency graph for Steak.cpp:



### 5.73.1 Detailed Description

Contains implementation for the [Steak](#) class.

#### Authors

Aidan Chapman (u22738917)

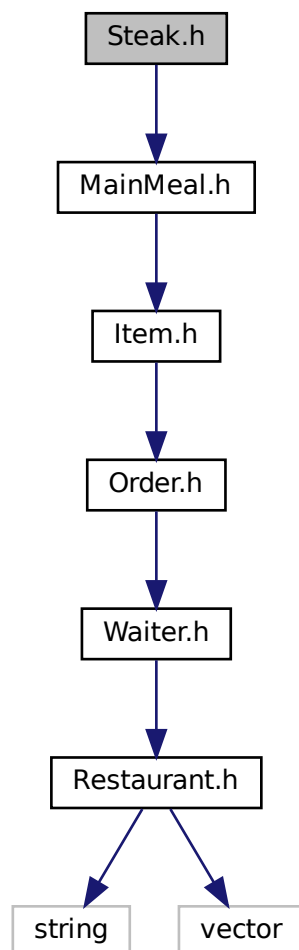


## 5.74 Steak.h File Reference

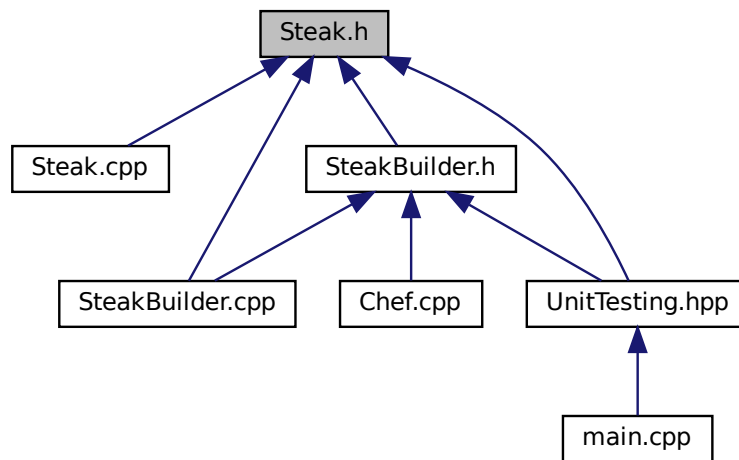
Contains declaration for the [Steak](#) class.

```
#include "MainMeal.h"
```

Include dependency graph for Steak.h:



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



## Classes

- class [Steak](#)

### 5.74.1 Detailed Description

Contains declaration for the [Steak](#) class.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

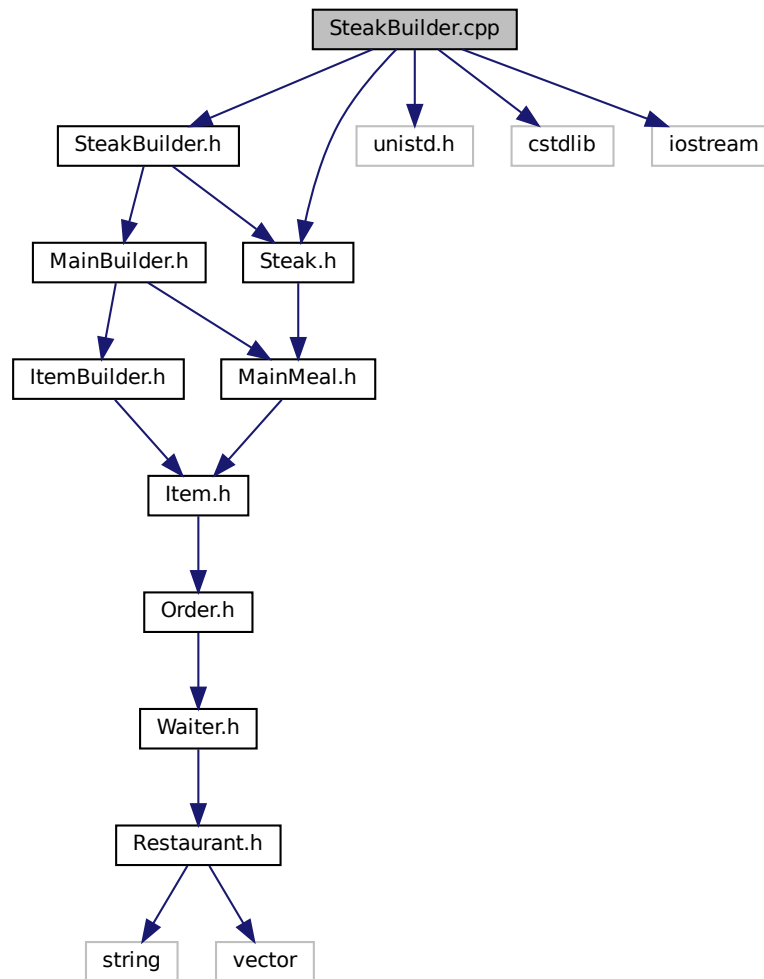
## 5.75 SteakBuilder.cpp File Reference

Contains the implementation for the [SteakBuilder](#) class.

```
#include "SteakBuilder.h"  
#include "Steak.h"  
#include <unistd.h>  
#include <cstdlib>
```

```
#include <iostream>
```

Include dependency graph for SteakBuilder.cpp:



### 5.75.1 Detailed Description

Contains the implementation for the [SteakBuilder](#) class.

#### Authors

- Graeme Blain (u22625462)
- Douglas Porter (u21797545)

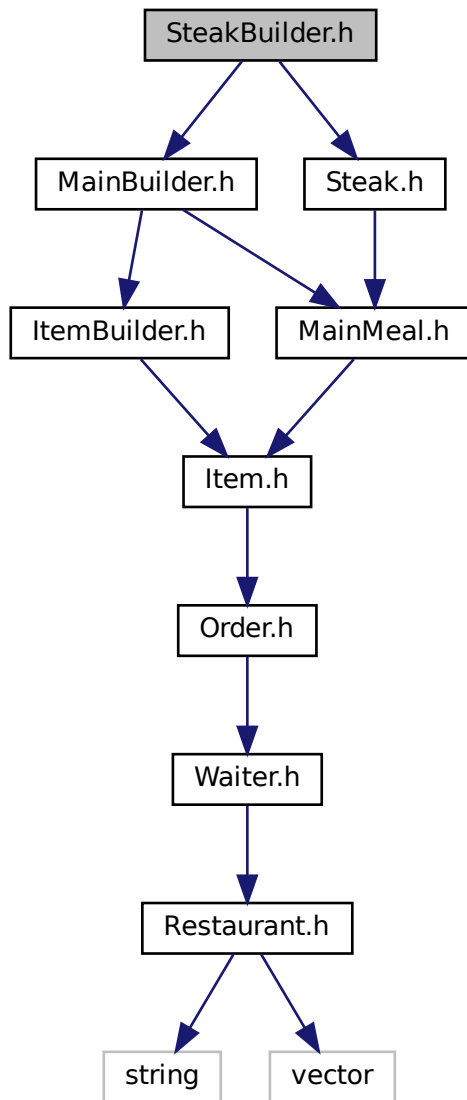
## 5.76 SteakBuilder.h File Reference

Contains declaration for the [SteakBuilder](#) class.

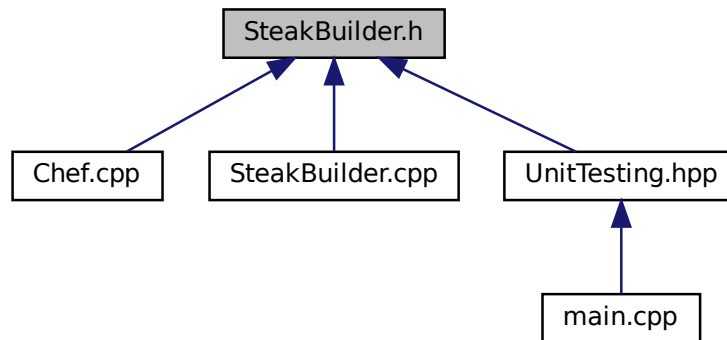
```
#include "MainBuilder.h"
```

```
#include "Steak.h"
```

Include dependency graph for SteakBuilder.h:



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



## Classes

- class [SteakBuilder](#)

### 5.76.1 Detailed Description

Contains declaration for the [SteakBuilder](#) class.

[SteakBuilder](#) is a concrete builder for the [Steak](#) class. Responsible for creating a [Steak](#) object.

#### Authors

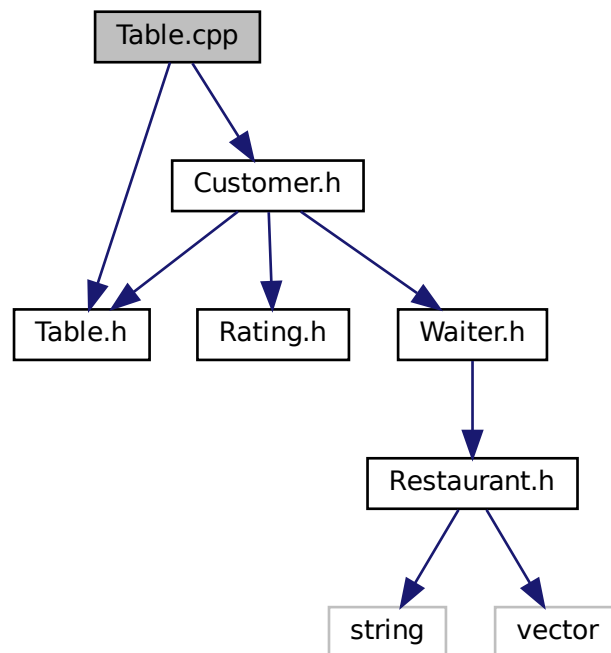
- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.77 Table.cpp File Reference

Contains implementation for the [Table](#) class.

```
#include "Table.h"  
#include "Customer.h"
```

Include dependency graph for Table.cpp:



### 5.77.1 Detailed Description

Contains implementation for the [Table](#) class.

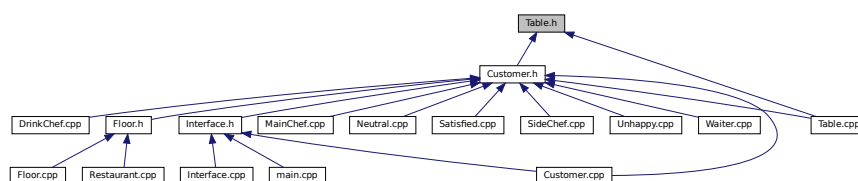
#### Authors

Aidan Chapman (u22738917)

## 5.78 Table.h File Reference

Contains declaration for the [Table](#) class.

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



## Classes

- class [Table](#)

### 5.78.1 Detailed Description

Contains declaration for the [Table](#) class.

The [Table](#) class is used to represent a table in the restaurant. It contains a pointer to a [Customer](#) object, which is used to represent the customer sitting at the table.

#### Authors

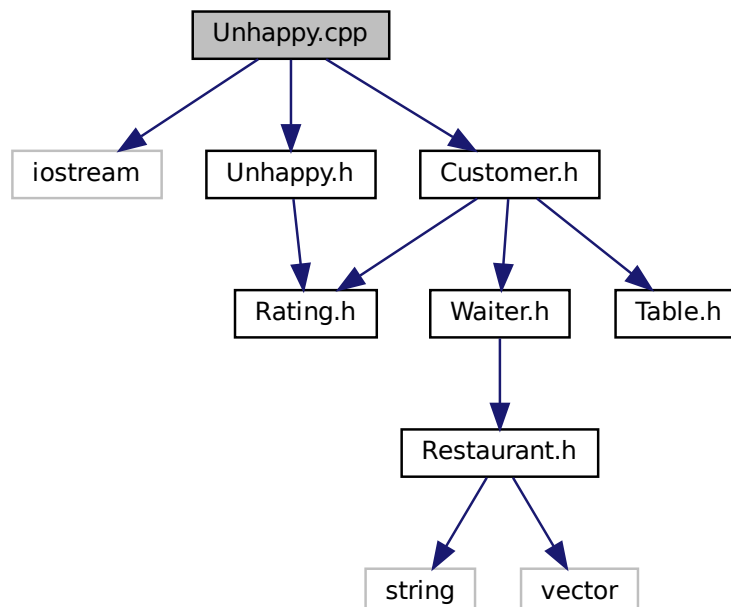
Aidan Chapman (u22738917)

## 5.79 Unhappy.cpp File Reference

Contains implementation for the [Unhappy](#) class.

```
#include <iostream>
#include "Unhappy.h"
#include "Customer.h"
```

Include dependency graph for Unhappy.cpp:



### 5.79.1 Detailed Description

Contains implementation for the [Unhappy](#) class.

#### Authors

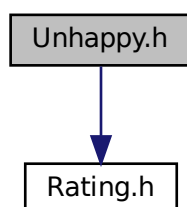
Sange Tshakumane (u21479748)

## 5.80 Unhappy.h File Reference

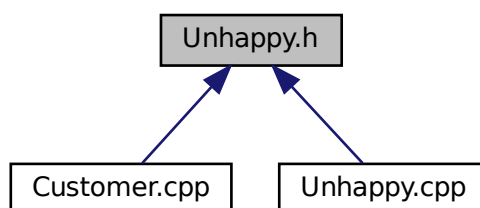
Contains declaration for the [Unhappy](#) class.

```
#include "Rating.h"
```

Include dependency graph for Unhappy.h:



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



## Classes

- class [Unhappy](#)



### 5.80.1 Detailed Description

Contains declaration for the [Unhappy](#) class.

[Unhappy](#) is a derived class of [Rating](#). It is one of the states of the State Pattern. It is used to represent a customer who is unhappy with the service they received. [Unhappy](#) customers do not tip.

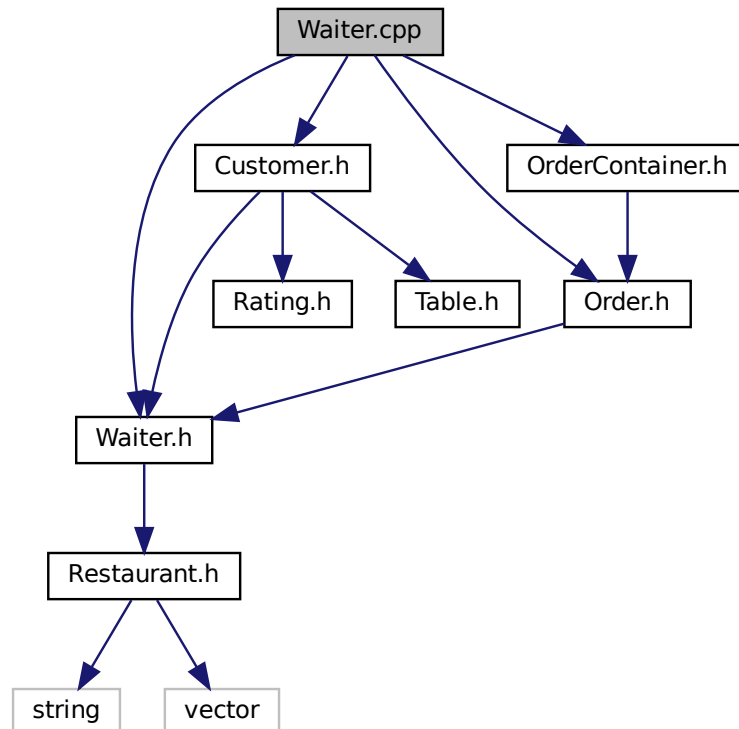
#### Authors

- Aidan Chapman (u22738917)
- Sange Tshakumane (u21479748)

## 5.81 Waiter.cpp File Reference

Contains implementation for the [Waiter](#) class.

```
#include "Waiter.h"
#include "Customer.h"
#include "Order.h"
#include "OrderContainer.h"
Include dependency graph for Waiter.cpp:
```



### 5.81.1 Detailed Description

Contains implementation for the [Waiter](#) class.

#### Authors

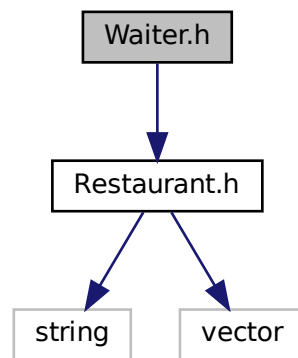
Aidan Chapman (u22738917)

## 5.82 Waiter.h File Reference

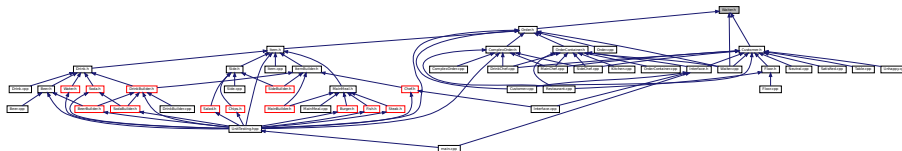
Contains declaration for the [Waiter](#) class.

```
#include "Restaurant.h"
```

Include dependency graph for Waiter.h:



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



## Classes

- class [Waiter](#)

### 5.82.1 Detailed Description

Contains declaration for the [Waiter](#) class.

The [Waiter](#) class is used to represent a waiter in the restaurant. It contains a pointer to a [Customer](#) object, which is used to represent the customer that the waiter is serving.

#### Authors

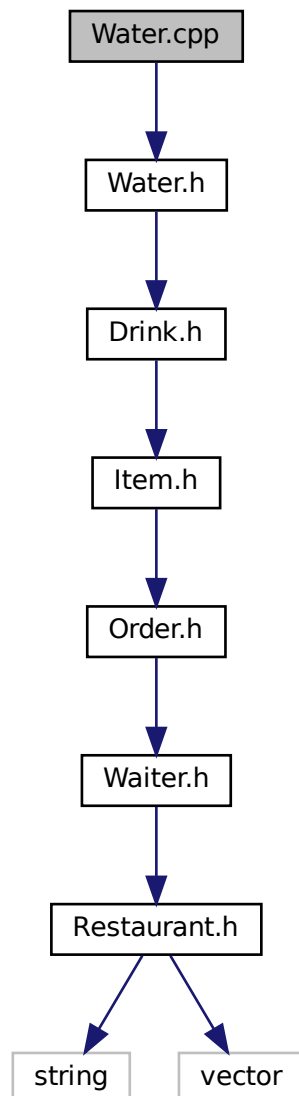
Aidan Chapman (u22738917)

## 5.83 Water.cpp File Reference

Contains implementation for the [Water](#) class.

```
#include "Water.h"
```

Include dependency graph for Water.cpp:



### 5.83.1 Detailed Description

Contains implementation for the [Water](#) class.

#### Authors

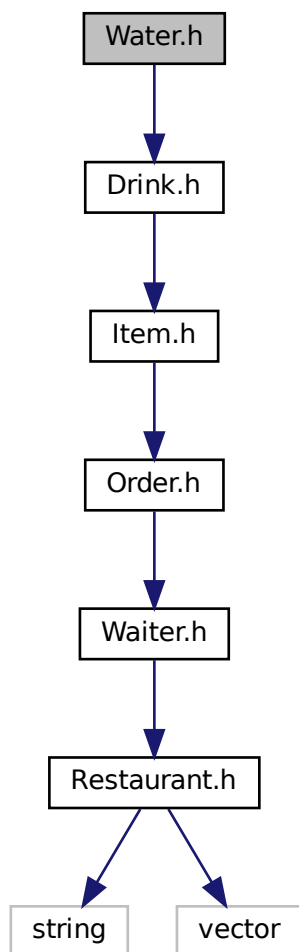
Aidan Chapman (u22738917)

## 5.84 Water.h File Reference

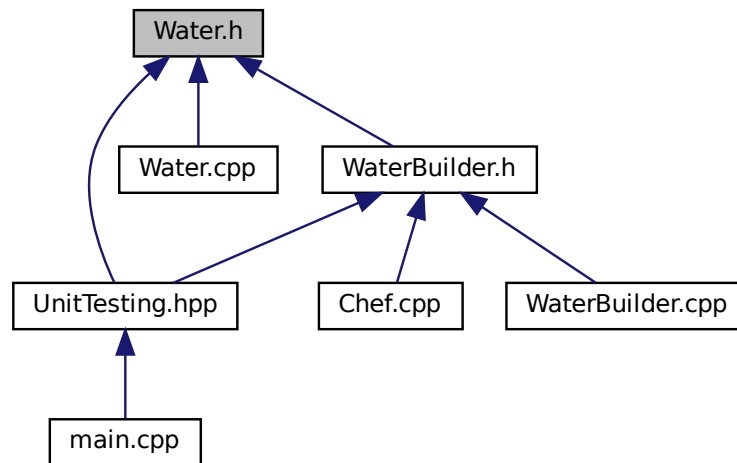
Contains declaration for the [Water](#) class.

```
#include "Drink.h"
```

Include dependency graph for Water.h:



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



## Classes

- class [Water](#)

### 5.84.1 Detailed Description

Contains declaration for the [Water](#) class.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

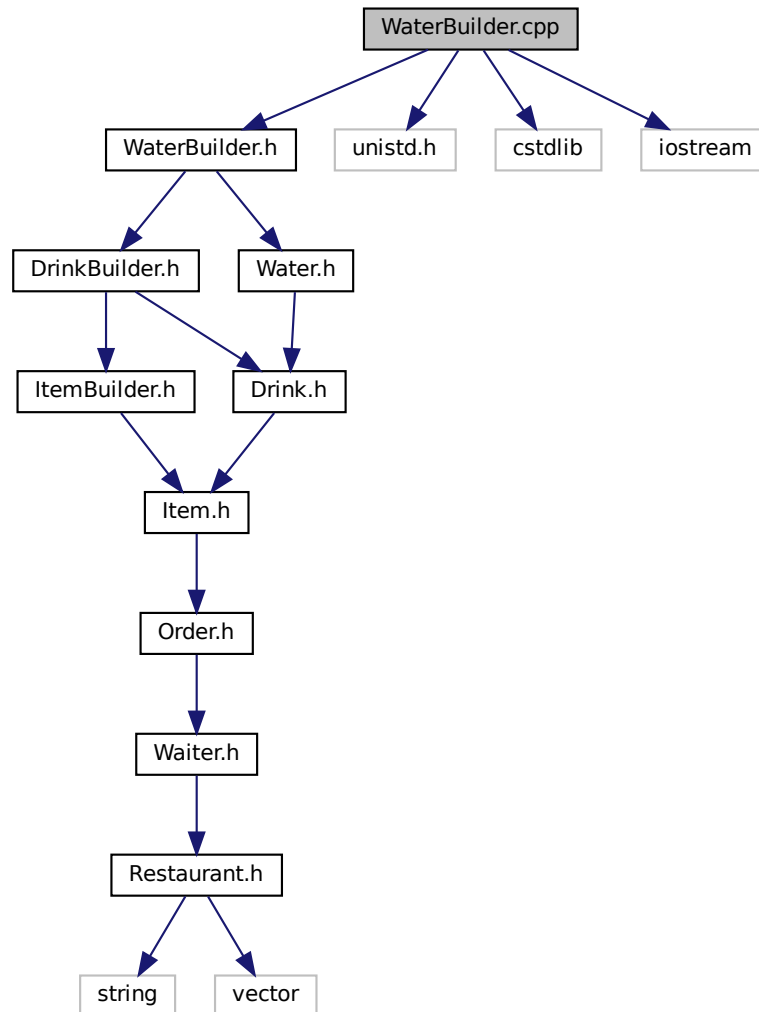
## 5.85 WaterBuilder.cpp File Reference

Contains implementation for the [WaterBuilder](#) class.

```
#include "WaterBuilder.h"
#include <unistd.h>
#include <cstdlib>
```

```
#include <iostream>
```

Include dependency graph for WaterBuilder.cpp:



### 5.85.1 Detailed Description

Contains implementation for the [WaterBuilder](#) class.

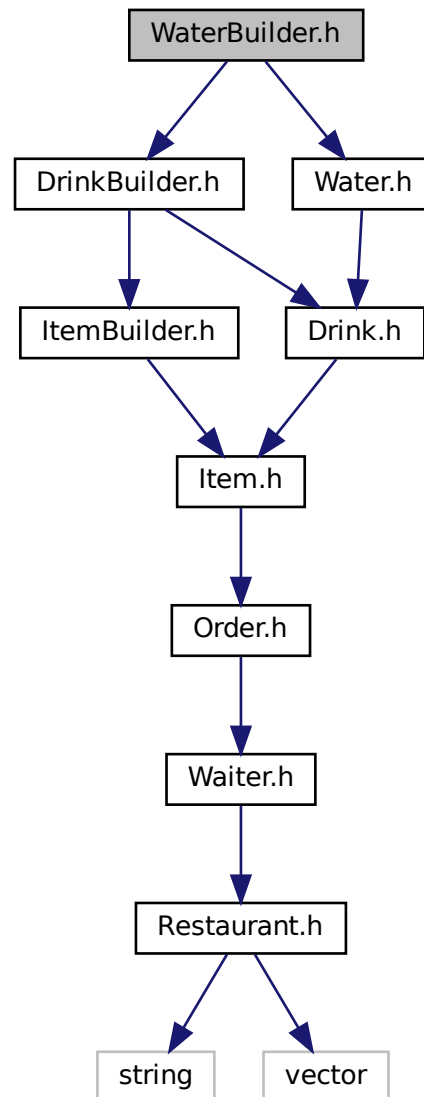
#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)

## 5.86 WaterBuilder.h File Reference

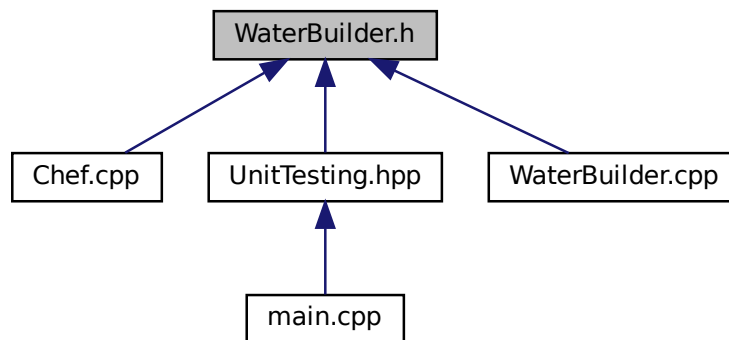
Contains declaration for the [WaterBuilder](#) class.

```
#include "DrinkBuilder.h"  
#include "Water.h"  
Include dependency graph for WaterBuilder.h:
```





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



## Classes

- class [WaterBuilder](#)

### 5.86.1 Detailed Description

Contains declaration for the [WaterBuilder](#) class.

This file contains the declarations for the functions to build a [Water](#) object using the Builder design pattern. [WaterBuilder](#) is a concrete builder class that inherits from the [DrinkBuilder](#) class. It is responsible for building a [Water](#) object using the template method pattern.

#### Authors

- Aidan Chapman (u22738917)
- Graeme Blain (u22625462)



# Index

- ~Beer
  - Beer, [11](#)
- ~Burger
  - Burger, [15](#)
- ~Chef
  - Chef, [18](#)
- ~Chips
  - Chips, [21](#)
- ~ComplexOrder
  - ComplexOrder, [24](#)
- ~Customer
  - Customer, [27](#)
- ~DrinkChef
  - DrinkChef, [36](#)
- ~Fish
  - Fish, [39](#)
- ~Floor
  - Floor, [41](#)
- ~HeadChef
  - HeadChef, [45](#)
- ~Interface
  - Interface, [46](#)
- ~MainChef
  - MainChef, [56](#)
- ~Restaurant
  - Restaurant, [66](#)
- ~Salad
  - Salad, [69](#)
- ~SideChef
  - SideChef, [80](#)
- ~Soda
  - Soda, [82](#)
- ~Steak
  - Steak, [86](#)
- ~Table
  - Table, [88](#)
- ~Waiter
  - Waiter, [92](#)
- ~Water
  - Water, [96](#)
- acceptWaiter
  - Customer, [27](#)
- addCustomer
  - Table, [89](#)
- addToOrder
  - ComplexOrder, [25](#)
  - Order, [62](#)
- appendToOrder
  - ComplexOrder, [25](#)
  - Order, [62](#)
- Beer, [9](#)
  - ~Beer, [11](#)
  - Beer, [10](#)
  - Beer.cpp, [99](#)
  - Beer.h, [101](#)
  - BeerBuilder, [11](#)
  - BeerBuilder.cpp, [102](#)
  - BeerBuilder.h, [104](#)
  - Burger, [13](#)
    - ~Burger, [15](#)
    - Burger, [14](#)
  - Burger.cpp, [105](#)
  - Burger.h, [107](#)
  - BurgerBuilder, [15](#)
  - BurgerBuilder.cpp, [108](#)
  - BurgerBuilder.h, [110](#)
- calculatePayment
  - Customer, [27](#)
- calculatePrice
  - ComplexOrder, [25](#)
  - Item, [49](#)
  - Order, [62](#)
- calculateTip
  - Neutral, [60](#)
  - Satisfied, [73](#)
  - Unhappy, [90](#)
- changeRating
  - Customer, [28](#)
- changeState
  - Neutral, [60](#)
  - Satisfied, [73](#)
  - Unhappy, [91](#)
- Chef, [17](#)
  - ~Chef, [18](#)
  - Chef, [18](#)
  - request, [19](#)
- Chef.cpp, [111](#)
- Chef.h, [112](#)
- Chips, [20](#)
  - ~Chips, [21](#)
  - Chips, [21](#)
- Chips.cpp, [114](#)
- Chips.h, [116](#)
- ChipsBuilder, [22](#)
- ChipsBuilder.cpp, [117](#)
- ChipsBuilder.h, [119](#)
- cleanUp

- Restaurant, 66
- Table, 89
- Waiter, 93
- ComplexOrder, 23
  - ~ComplexOrder, 24
  - addToOrder, 25
  - appendToOrder, 25
  - calculatePrice, 25
  - ComplexOrder, 24
- ComplexOrder.cpp, 120
- ComplexOrder.h, 121
- Customer, 26
  - ~Customer, 27
  - acceptWaiter, 27
  - calculatePayment, 27
  - changeRating, 28
  - Customer, 26
  - getOrder, 28
  - getOrderRequest, 28
  - getTable, 29
  - getTimestamp, 29
  - getWaiter, 29
  - receiveOrder, 30
- Customer.cpp, 123
- Customer.h, 124
- Drink, 31
  - Drink, 32
- Drink.cpp, 125
- Drink.h, 126
- DrinkBuilder, 33
  - getItem, 34
- DrinkBuilder.cpp, 127
- DrinkBuilder.h, 129
- DrinkChef, 35
  - ~DrinkChef, 36
  - DrinkChef, 36
  - preparePart, 36
- DrinkChef.cpp, 130
- DrinkChef.h, 131
- Fish, 37
  - ~Fish, 39
  - Fish, 38
- Fish.cpp, 133
- Fish.h, 135
- FishBuilder, 39
- FishBuilder.cpp, 136
- FishBuilder.h, 137
- Floor, 41
  - ~Floor, 41
  - Floor, 41
  - getNumTables, 42
  - getTable, 42
  - seatCustomer, 43
  - setRestaurant, 43
- Floor.cpp, 139
- Floor.h, 140
- generateNumberOfCustomers
  - Interface, 46
- generateOrderString
  - Interface, 46
- getCurrentUnixTime
  - Interface, 47
- getCustomer
  - Table, 89
  - Waiter, 93
- getItem
  - DrinkBuilder, 34
  - MainBuilder, 54
  - SideBuilder, 77
- getNumTables
  - Floor, 42
- getOrder
  - Customer, 28
  - OrderContainer, 64
- getOrderRequest
  - Customer, 28
- getRating
  - Neutral, 60
  - Satisfied, 73
  - Unhappy, 91
- getRequestedOrder
  - OrderContainer, 64
- getRestaurant
  - Waiter, 93
- getTable
  - Customer, 29
  - Floor, 42
- getTimestamp
  - Customer, 29
- getWaiter
  - Customer, 29
  - Order, 62
- HeadChef, 44
  - ~HeadChef, 45
  - HeadChef, 45
- HeadChef.cpp, 142
- HeadChef.h, 144
- initialise
  - Restaurant, 66
- Interface, 45
  - ~Interface, 46
  - generateNumberOfCustomers, 46
  - generateOrderString, 46
  - getCurrentUnixTime, 47
  - Interface, 46
  - runCustomer, 47
- Interface.cpp, 145
- Interface.h, 146
- Item, 48
  - calculatePrice, 49
  - Item, 49
- Item.cpp, 148
- Item.h, 149

ItemBuilder, 50  
ItemBuilder.h, 150

Kitchen, 51  
    Kitchen, 51  
    makeNextOrder, 51  
    receiveOrder, 52  
    setRestaurant, 52  
Kitchen.cpp, 151  
Kitchen.h, 152

main  
    main.cpp, 155  
main.cpp, 154  
    main, 155  
MainBuilder, 53  
    getItem, 54  
MainBuilder.cpp, 155  
MainBuilder.h, 157  
MainChef, 55  
    ~MainChef, 56  
    MainChef, 56  
    preparePart, 56  
MainChef.cpp, 158  
MainChef.h, 159  
MainMeal, 57  
    MainMeal, 58  
MainMeal.cpp, 161  
MainMeal.h, 162  
makeNextOrder  
    Kitchen, 51

Neutral, 59  
    calculateTip, 60  
    changeState, 60  
    getRating, 60  
Neutral.cpp, 164  
Neutral.h, 165

Order, 61  
    addToOrder, 62  
    appendToOrder, 62  
    calculatePrice, 62  
    getWaiter, 62  
    Order, 61  
Order.cpp, 166  
Order.h, 166  
OrderContainer, 63  
    getOrder, 64  
    getRequestedOrder, 64  
    OrderContainer, 63  
OrderContainer.cpp, 168  
OrderContainer.h, 168

placeOrder  
    Restaurant, 66  
preparePart  
    DrinkChef, 36  
    MainChef, 56  
    SideChef, 80

Rating, 64  
Rating.cpp, 170  
Rating.h, 170  
receiveOrder  
    Customer, 30  
    Kitchen, 52  
request  
    Chef, 19  
Restaurant, 65  
    ~Restaurant, 66  
    cleanUp, 66  
    initialise, 66  
    placeOrder, 66  
    Restaurant, 65  
    seatCustomer, 67  
Restaurant.cpp, 171  
Restaurant.h, 172  
runCustomer  
    Interface, 47

Salad, 68  
    ~Salad, 69  
    Salad, 69  
Salad.cpp, 173  
Salad.h, 174  
SaladBuilder, 70  
SaladBuilder.cpp, 175  
SaladBuilder.h, 177  
Satisfied, 72  
    calculateTip, 73  
    changeState, 73  
    getRating, 73  
Satisfied.cpp, 178  
Satisfied.h, 179  
seatCustomer  
    Floor, 43  
    Restaurant, 67  
serveCustomer  
    Waiter, 93  
setRestaurant  
    Floor, 43  
    Kitchen, 52  
Side, 74  
    Side, 75  
Side.cpp, 181  
Side.h, 182  
SideBuilder, 76  
    getItem, 77  
SideBuilder.cpp, 183  
SideBuilder.h, 185  
SideChef, 78  
    ~SideChef, 80  
    preparePart, 80  
    SideChef, 79  
SideChef.cpp, 186  
SideChef.h, 187  
Soda, 81

- [~Soda](#), 82
  - [Soda](#), 82
- [Soda.cpp](#), 189
- [Soda.h](#), 191
- [SodaBuilder](#), 83
- [SodaBuilder.cpp](#), 192
- [SodaBuilder.h](#), 193
- [Steak](#), 84
  - [~Steak](#), 86
  - [Steak](#), 85
- [Steak.cpp](#), 195
- [Steak.h](#), 197
- [SteakBuilder](#), 86
- [SteakBuilder.cpp](#), 198
- [SteakBuilder.h](#), 199
- [Table](#), 88
  - [~Table](#), 88
  - [addCustomer](#), 89
  - [cleanUp](#), 89
  - [getCustomer](#), 89
  - [Table](#), 88
- [Table.cpp](#), 201
- [Table.h](#), 202
- [takeOrder](#)
  - [Waiter](#), 94
- [Unhappy](#), 90
  - [calculateTip](#), 90
  - [changeState](#), 91
  - [getRating](#), 91
- [Unhappy.cpp](#), 203
- [Unhappy.h](#), 204
- [Waiter](#), 92
  - [~Waiter](#), 92
  - [cleanUp](#), 93
  - [getCustomer](#), 93
  - [getRestaurant](#), 93
  - [serveCustomer](#), 93
  - [takeOrder](#), 94
  - [Waiter](#), 92
- [Waiter.cpp](#), 205
- [Waiter.h](#), 206
- [Water](#), 95
  - [~Water](#), 96
  - [Water](#), 96
- [Water.cpp](#), 207
- [Water.h](#), 209
- [WaterBuilder](#), 97
- [WaterBuilder.cpp](#), 210
- [WaterBuilder.h](#), 211