BARISTAMATIC

1.0

Generated by Doxygen 1.8.17

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 CoffeeMachine Class Reference

3.1.1 Detailed Description

Definition at line 47 of file CoffeeMachine.h.

`#include <CoffeeMachine.h>`

Public Member Functions

- **CoffeeMachine** ()
  : default constructor which initializes all possible drinks.
- ∼**CoffeeMachine** ()
  : destructor which does not explicitly delete anything
- **CoffeeMachine** (**CoffeeMachine** const &other)
  : Copy constructor which copies over all member variables
- void **makeDrink** (long i)
  : if possible, makeDrink i and update the inventory
- bool **isStocked** (**Drink** &d)
  : determine if the drink can be made
- double **calcPrice** (std::string drinkName) const
  : calculate the price of the drink based on the ingredients
- void **displayInventory** ()
  : print out the current Inventory in alphabetical order
- void **displayMenu** ()
  : print out the Menu in alphabetical order
- void **restock** ()
  : restock the Inventory to 10 units for each ingredient
- void **processInput** (const std::string &userInput)
  Process the user supplied input.
- void **run** ()
  : run the machine, which handles all user input appropriately

Private Attributes

- std::vector< **Drink** > **drinks**

  vector of all possible drinks
- std::map< **INGREDIENTS**, int > **Inventory**

  Hash table for ingredients and their quantities.

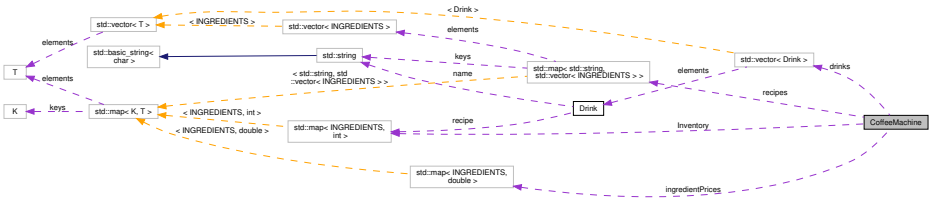
Static Private Attributes

- static const std::map< std::string, std::vector< **INGREDIENTS** > > **recipes** = **createDrinks**()

  Hash table for drinkNames and their required ingredients.
- static const std::map< **INGREDIENTS**, double > **ingredientPrices**

  Hash table for ingredient and their prices.

Collaboration diagram for CoffeeMachine:



3.1.2 Constructor & Destructor Documentation

3.1.2.1 CoffeeMachine() [1/2]

```
CoffeeMachine::CoffeeMachine ( )
```

Drink (p. 14) list is ordered by drink name

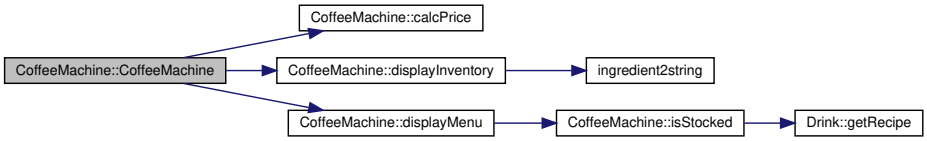
Definition at line 77 of file CoffeeMachine.cpp.

```
77                              {
78   // stock all inventory levels to 10 parts each
79   for (auto placeholder = ingredientPrices.begin();
80        placeholder != ingredientPrices.end(); ++placeholder) {
81     Inventory.insert(std::pair<INGREDIENTS, int>(placeholder->first, 10));
82   }
83   // iterate over DRINKS enum, create drinks obj for each
84   std::vector<std::string> tempVec(DRINKS_SIZE);
85   int idx = 0;
86   for (auto it = recipes.begin(); it != recipes.end(); ++it, ++idx) {
87     tempVec[idx] = it->first;
88   }
89   std::sort(tempVec.begin(), tempVec.end());
90
91   // cannot pre-reserve vec size and use assignment operator because price and
92   // name variables are named const
93   idx = 0;
94   for (auto it = recipes.begin(); it != recipes.end(); ++it, ++idx) {
95     this->drinks.emplace_back(Drink(calcPrice(it->first), it->first));
96     this->drinks[idx].setRecipe(it->second);
97   }
98
99   // display inventory and menu on machine startup
100   this->displayInventory();
101   this->displayMenu();
```

```
102 }
```

References calcPrice(), displayInventory(), displayMenu(), drinks, DRINKS_SIZE, ingredientPrices, Inventory, and recipes.

Here is the call graph for this function:



3.1.2.2 ~CoffeeMachine()

```
CoffeeMachine::~CoffeeMachine ( )
```

Definition at line 104 of file CoffeeMachine.cpp.

```
104                                {
105   // delete[] drinks;
106 }
```

3.1.2.3 CoffeeMachine() [2/2]

```
CoffeeMachine::CoffeeMachine (
              CoffeeMachine const & other )
```

Parameters

other	

Definition at line 108 of file CoffeeMachine.cpp.

```
109     : drinks(other.drinks), Inventory(other.Inventory) {}
```

3.1.3 Member Function Documentation

3.1.3.1 calcPrice()

```
double CoffeeMachine::calcPrice (
             std::string drinkName ) const
```

Parameters

drinkName	name of the drink which will be used to find the recipe

Returns

price of the drink

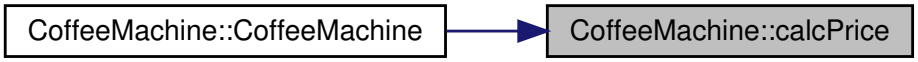
Definition at line 68 of file CoffeeMachine.cpp.

```
68                                                          {
69   double price = 0.;
70   auto currentRecipe = recipes.find(drinkName)->second;
71   for (auto &&it : currentRecipe) {
72     price += ingredientPrices.find(it)->second;
73   }
74   return price;
75 }
```

References ingredientPrices, and recipes.

Referenced by CoffeeMachine().

Here is the caller graph for this function:



3.1.3.2 displayInventory()

```
void CoffeeMachine::displayInventory ( )
```

Since hash's cannot be ordered, the sorting is done every time the function is called. This is potential room for improvement

Definition at line 160 of file CoffeeMachine.cpp.

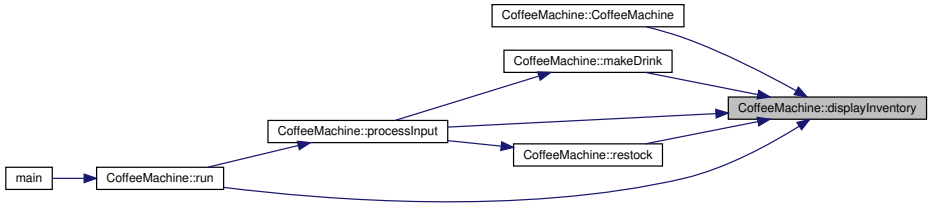
```
160                                          {
161   std::cout « "Inventory:";
162   std::vector<std::string> ingredientNames;
163   // sort the ingredient names
164   for (auto it = Inventory.begin(); it != Inventory.end(); ++it) {
165     ingredientNames.push_back(ingredient2string(it->first) + "," + std::to_string(it->second));
166   }
167   std::sort(ingredientNames.begin(), ingredientNames.end());
168   // print out the sorted list
169   for (auto it = ingredientNames.begin(); it != ingredientNames.end(); ++it) {
170     std::cout « "\n" « *it;
171   }
172   // flush buffer
173   std::cout « std::endl;
174 }
```

References ingredient2string(), and Inventory.

Referenced by CoffeeMachine(), makeDrink(), processInput(), restock(), and run().

Here is the call graph for this function:

Here is the caller graph for this function:



3.1.3.3 displayMenu()

```
void CoffeeMachine::displayMenu ( )
```

The function accounts for user added new drinks

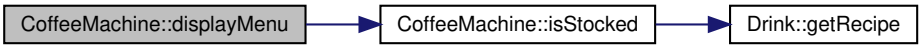
Definition at line 176 of file CoffeeMachine.cpp.

```
176                                 {
177   // Drinks are already sorted by name
178   int idx = 1;
179   std::cout « "Menu:";
180   for (auto it = drinks.begin(); it != drinks.end(); ++it) {
181     // <drink number>,<drink name>,<cost>,<in-stock>
182     std::cout « "\n"
183               « idx++ « "," « std::setprecision(2) « std::fixed
184               « it->getName() « ",$" « it->getPrice() « ","
185               « (this->isStocked(*it) ? "true" : "false");
186   }
187   // flush the buffer
188   std::cout « std::endl;
189 }
```

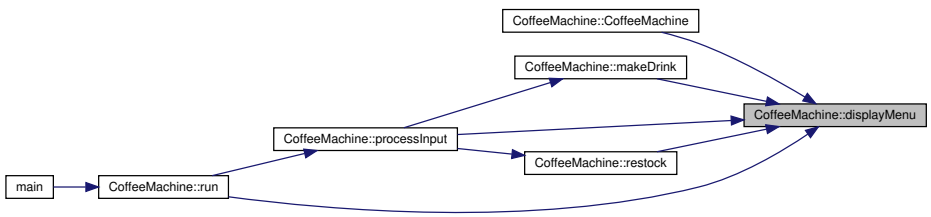
References drinks, and isStocked().

Referenced by CoffeeMachine(), makeDrink(), processInput(), restock(), and run().

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3.4 isStocked()

```
bool CoffeeMachine::isStocked (
            Drink & d )
```

If the quantity of the requested ingredients is greater than the amount in the Inventory, the drink can be made

Parameters

d	drink which is being checked

Returns

true or false based on Inventory stock

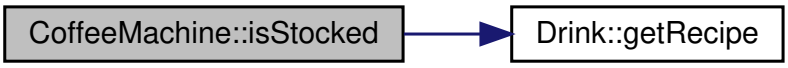
Definition at line 191 of file CoffeeMachine.cpp.

```
191                                     {
192   for( auto &&it : d.getRecipe()) {
193     // if the required quantity is greater than the current stock, return false
194     if((it.second > this->Inventory[it.first]) ) {
195       return false;
196     }
197   }
198   return true;
199 }
```

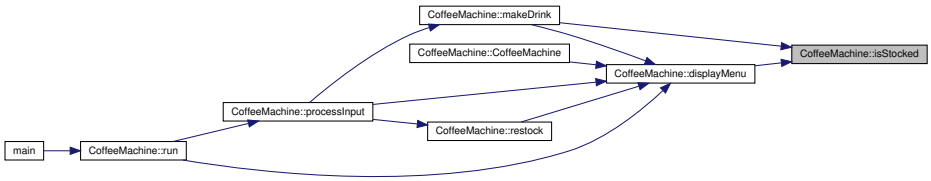
References Drink::getRecipe().

Referenced by displayMenu(), and makeDrink().

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3.5 makeDrink()

```
void CoffeeMachine::makeDrink (
            long i )
```

Parameters

i	index of the menu-item to be made

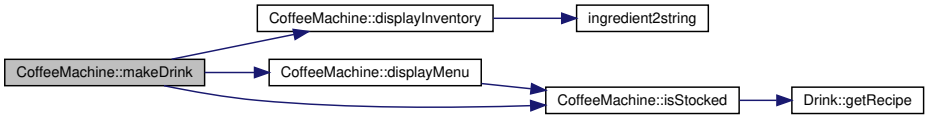
Definition at line 133 of file CoffeeMachine.cpp.

```
133                                     {
134   // menu is 1-based indexing, hence why using i-1
135   if (isStocked(drinks[i-1])) {
136     std::cout « "Dispensing: " « drinks[i-1].getName() « std::endl;
137     // get the vector of the ingredients which are needed by the drink
138     auto currentRecipe = drinks[i-1].getRecipe();
139     // remove the ingredients from the inventory
140     for (auto it = currentRecipe.begin(); it != currentRecipe.end(); ++it) {
141       Inventory[it->first] -= it->second;
142     }
143   } else {
144     std::cout « "Out of stock: " « drinks[i-1].getName() « std::endl;
145   }
146   // display inventory and menu whether drink is successfully made or not
147   this->displayInventory();
148   this->displayMenu();
149 }
```

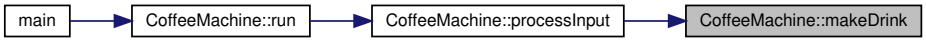
References displayInventory(), displayMenu(), drinks, Inventory, and isStocked().

Referenced by processInput().

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3.6 processInput()

```
void CoffeeMachine::processInput (
            const std::string & userInput )
```

If a user supplies: r/R-> restock the machine q/Q -> quit the machine 1....DRINKS_SIZE -> make drink ∗ if the ingredients are available anything else -> print invalid argument

Parameters

userInput	input argument take from the standard input stream

Definition at line 111 of file CoffeeMachine.cpp.

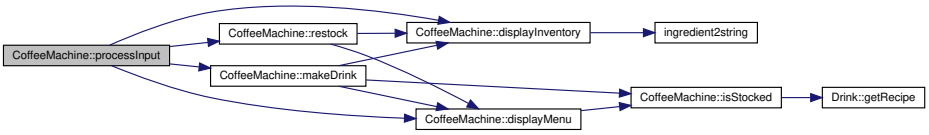
```
111                                                     {
112   if (userInput == "r" || userInput == "R") {
113     this->restock();
```

```
114   } else if (userInput == "q" || userInput == "Q") {
115     exit(0);
116   } else {
117     try {
118       char *end;
119       long attemptedConversion = std::strtol(userInput.c_str(), &end, 10);
120       if (attemptedConversion <= 0 || attemptedConversion > DRINKS_SIZE) {
121         throw std::out_of_range(
122             "User input is either out of bounds, or an invalid character");
123       }
124       this->makeDrink(attemptedConversion);
125     } catch (...) {
126       std::cout « "Invalid selection: " « userInput « std::endl;
127       this->displayInventory();
128       this->displayMenu();
129     }
130   }
131 }
```

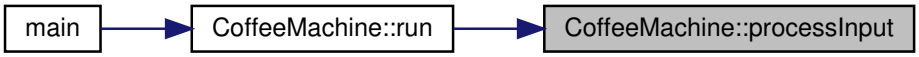
References displayInventory(), displayMenu(), DRINKS_SIZE, makeDrink(), and restock().

Referenced by run().

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3.7 restock()

```
void CoffeeMachine::restock ( )
```

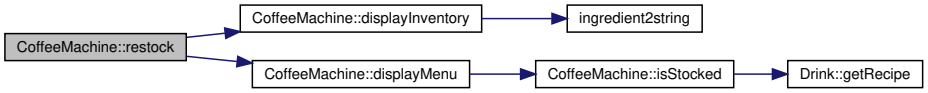
Definition at line 151 of file CoffeeMachine.cpp.

```
151                             {
152   for (auto it = Inventory.begin(); it != Inventory.end(); ++it) {
153     // restock all inventory levels to 10 parts each
154     it->second = 10;
155   }
156   this->displayInventory();
157   this->displayMenu();
158 }
```

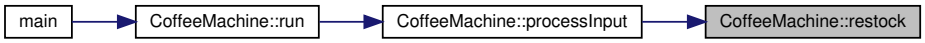
References displayInventory(), displayMenu(), and Inventory.

Referenced by processInput().

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3.8 run()

```
void CoffeeMachine::run ( )
```

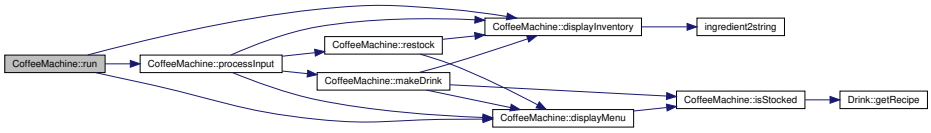
Definition at line 201 of file CoffeeMachine.cpp.

```
201                         {
202   std::string userInput;
203   // endless loop which will only terminate if the user provides a q or Q
204   do {
205     std::cin » userInput;
206     std::transform(userInput.begin(), userInput.end(), userInput.begin(),
207                    [](unsigned char c) { return std::tolower(c); });
208     this->processInput(userInput);
209     this->displayInventory();
210     this->displayMenu();
211   } while (true);
212 }
```

References displayInventory(), displayMenu(), and processInput().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

3.1.4 Member Data Documentation

3.1.4.1 drinks

```
std::vector< Drink> CoffeeMachine::drinks  [private]
```

Definition at line 132 of file CoffeeMachine.h.

Referenced by CoffeeMachine(), displayMenu(), and makeDrink().

3.1.4.2 ingredientPrices

```
const std::map< INGREDIENTS, double > CoffeeMachine::ingredientPrices [static], [private]
```

Initial value:

```
=
    createPriceMap()
```

Definition at line 138 of file CoffeeMachine.h.

Referenced by calcPrice(), and CoffeeMachine().

3.1.4.3 Inventory

```
std::map< INGREDIENTS, int> CoffeeMachine::Inventory [private]
```

Definition at line 134 of file CoffeeMachine.h.

Referenced by CoffeeMachine(), displayInventory(), makeDrink(), and restock().

3.1.4.4 recipes

```
const std::map< std::string, std::vector< INGREDIENTS > > CoffeeMachine::recipes = createDrinks() [static], [private]
```

Definition at line 136 of file CoffeeMachine.h.

Referenced by calcPrice(), and CoffeeMachine().

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

- **CoffeeMachine.h**
- **CoffeeMachine.cpp**

3.2 Drink Class Reference

3.2.1 Detailed Description

Definition at line 12 of file Drink.h.

```
#include <Drink.h>
```

Public Member Functions

- **Drink** ()

  : zero argument constructor, provide default values to mem vars
- ~**Drink** ()=default

  : default destructor, no memory is dynamically allocated
- **Drink** (**Drink** const &)

  : copy constructor
- **Drink** (double **price**, std::string **name**)

  *three parameter constructor for **Drink** (p. 14) Class*
- std::string **getName** () const

  : member variable name getter
- const double **getPrice** () const

  : member variable price getter
- void **setRecipe** (std::vector< **INGREDIENTS** > list)

  : member variable recipe setter
- std::map< **INGREDIENTS**, int > **getRecipe** ()

  : member variable recipe getter

Private Attributes

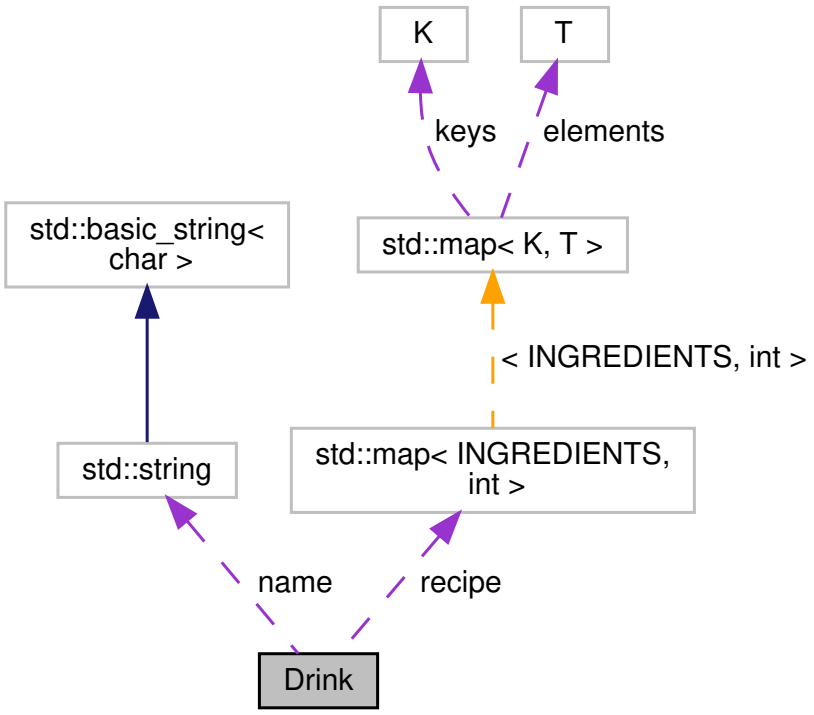
- const double **price**

  the price of the drink
- const std::string **name**

  name of the drink
- std::map< **INGREDIENTS**, int > **recipe**

  Quantity of ingredients need to make drink.

Collaboration diagram for Drink:



3.2.2 Constructor & Destructor Documentation

3.2.2.1 Drink() [1/3]

```
Drink::Drink ( )
```

Definition at line 9 of file Drink.cpp.

```
9 : price(0.), name("") {}
```

3.2.2.2 ~Drink()

```
Drink::~Drink ( )  [default]
```

3.2.2.3 Drink() [2/3]

```
Drink::Drink (
              Drink const & other )
```

Definition at line 11 of file Drink.cpp.

```
12     : price(other.price), name(other.name), recipe(other.recipe) {}
```

3.2.2.4 Drink() [3/3]

```
Drink::Drink (
             double price,
             std::string name )
```

Parameters

*price_*	price of the drink being constructed
*name_*	name of the drink, case sensitive

Definition at line 14 of file Drink.cpp.

```
15     : price(price_), name(std::move(name_)) {
16 }
```

3.2.3 Member Function Documentation

3.2.3.1 getName()

```
std::string Drink::getName ( ) const
```

Returns

name of the drink

Definition at line 18 of file Drink.cpp.

```
18                                {
19   return this->name;
20 }
```

References name.

3.2.3.2 getPrice()

```
const double Drink::getPrice ( ) const
```

Returns

price of the drink

Definition at line 22 of file Drink.cpp.

```
22                                       {
23   return this->price;
24 }
```

References price.

3.2.3.3 getRecipe()

```
std::map<  INGREDIENTS, int > Drink::getRecipe ( )
```

Returns

vector of enum INGREDIENTS

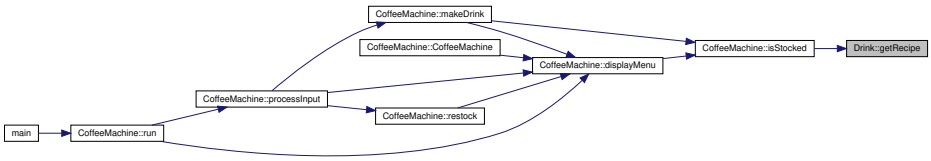
Definition at line 47 of file Drink.cpp.

```
47                                            {
48   return this->recipe;
49 }
```

References recipe.

Referenced by CoffeeMachine::isStocked().

Here is the caller graph for this function:



3.2.3.4 setRecipe()

```
void Drink::setRecipe (
            std::vector<  INGREDIENTS > list )
```

Parameters

list	vector of enum INGREDIENTS

Definition at line 26 of file Drink.cpp.

```
26                                                {
27   // we know everything in the list exists at-least once, we are looking
28   // for duplicates
29   int count = 1;
30   // it does not need to updated since the list shrinks in size
31   for (auto it = list.begin(); it != list.end();) {
32     // search for another instance of the current ingredient
33     auto search = std::find(it+1, list.end(), *it);
34     if (search != list.end()) {
35       // found another instance of current ingredient
36       ++count;
37       list.erase(search);
38     } else {
39       // no more other instances of current ingredient, add it to Hash Table
40       this->recipe.insert(std::pair<INGREDIENTS, int>(*it, count));
41       list.erase(it);
42       count = 1;
43     }
44   }
45 }
```

References recipe.

3.2.4 Member Data Documentation

3.2.4.1 name

```
const std::string Drink::name  [private]
```

Definition at line 63 of file Drink.h.

Referenced by getName().

3.2.4.2 price

```
const double Drink::price  [private]
```

Definition at line 62 of file Drink.h.

Referenced by getPrice().

3.2.4.3 recipe

```
std::map< INGREDIENTS, int> Drink::recipe  [private]
```

Definition at line 65 of file Drink.h.

Referenced by getRecipe(), and setRecipe().

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

- **Drink.h**
- **Drink.cpp**

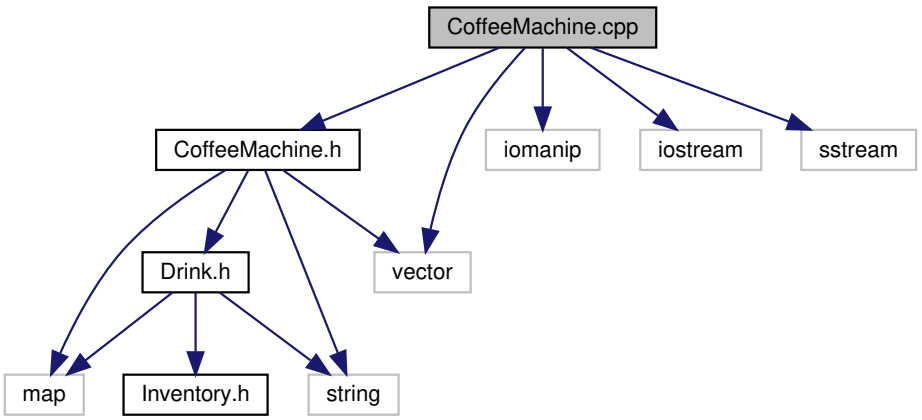
Chapter 4

File Documentation

4.1 CoffeeMachine.cpp File Reference

```
#include "CoffeeMachine.h"
#include <iomanip>
#include <iostream>
#include <sstream>
#include <vector>
```

Include dependency graph for CoffeeMachine.cpp:



Macros

- #define **DRINKS_SIZE** recipes.size()

Functions

- const std::map< **INGREDIENTS**, double > **createPriceMap** ()
- std::vector< **INGREDIENTS** > **defineDrink** (int numIngredients,...)
- const std::map< std::string, std::vector< **INGREDIENTS** > > **createDrinks** ()

4.1.1 Macro Definition Documentation

4.1.1.1 DRINKS_SIZE

```
#define DRINKS_SIZE recipes.size()
```

4.1.2 Function Documentation

4.1.2.1 createDrinks()

```
const std::map<std::string, std::vector< INGREDIENTS> > createDrinks ( )
```

Definition at line 38 of file CoffeeMachine.cpp.

```
38                                                                  {
39   const std::map<std::string, std::vector<INGREDIENTS» recipes = {
40       {"Coffee", defineDrink(5, INGREDIENTS::Coffee, INGREDIENTS::Coffee,
41                              INGREDIENTS::Coffee, INGREDIENTS::Sugar,
42                              INGREDIENTS::Cream)},
43       {"Decaf Coffee",
44        defineDrink(5, INGREDIENTS::DecafCoffee, INGREDIENTS::DecafCoffee,
45                    INGREDIENTS::DecafCoffee, INGREDIENTS::Sugar,
46                    INGREDIENTS::Cream)},
47       {"Caffe Latte",
48        defineDrink(3, INGREDIENTS::Espresso, INGREDIENTS::Espresso,
49                    INGREDIENTS::SteamedMilk)},
50       {"Caffe Americano",
51        defineDrink(3, INGREDIENTS::Espresso, INGREDIENTS::Espresso,
52                    INGREDIENTS::Espresso)},
53       {"Caffe Mocha",
54        defineDrink(4, INGREDIENTS::Espresso, INGREDIENTS::Cocoa,
55                    INGREDIENTS::SteamedMilk, INGREDIENTS::WhippedCream)},
56       {"Cappuccino",
57        defineDrink(4, INGREDIENTS::Espresso, INGREDIENTS::Espresso,
58                    INGREDIENTS::SteamedMilk, INGREDIENTS::FoamedMilk)}};
59 #define DRINKS_SIZE recipes.size()
60   return recipes;
61 }
```

References Cocoa, Coffee, Cream, DecafCoffee, defineDrink(), Espresso, FoamedMilk, SteamedMilk, Sugar, and WhippedCream.

Here is the call graph for this function:

4.1.2.2 createPriceMap()

```
const std::map< INGREDIENTS, double> createPriceMap ( )
```

Definition at line 11 of file CoffeeMachine.cpp.

```
11                                                  {
12   const std::map<INGREDIENTS , double> ingredientPriceMap = {
13       {INGREDIENTS::Coffee, 0.75},
14       {INGREDIENTS::DecafCoffee, 0.75},
15       {INGREDIENTS::Sugar, 0.25},
16       {INGREDIENTS::Cream, 0.25},
17       {INGREDIENTS::SteamedMilk, 0.35},
18       {INGREDIENTS::FoamedMilk, 0.35},
19       {INGREDIENTS::Espresso, 1.10},
20       {INGREDIENTS::Cocoa, 0.9},
21       {INGREDIENTS::WhippedCream, 1.00}
22   };
23   return ingredientPriceMap;
24 }
```

References Cocoa, Coffee, Cream, DecafCoffee, Espresso, FoamedMilk, SteamedMilk, Sugar, and Whipped↩ Cream.

4.1.2.3 defineDrink()

```
std::vector< INGREDIENTS> defineDrink (
              int numIngredients,
               ...  )
```

Definition at line 26 of file CoffeeMachine.cpp.

```
26                                                              {
27   std::vector<INGREDIENTS> v;
28   va_list vaList;
29   va_start(vaList, numIngredients); // initialize vaList for num number of arguments
30   // can't pre-allocate vector size, enums are all converted to lowest value
31   for (int i = 0; i < numIngredients; ++i) {
32     v.push_back(va_arg(vaList, INGREDIENTS));
33   }
34   va_end(vaList); // clean memory reserved for vaList
35   return v;
36 }
```

Referenced by createDrinks().

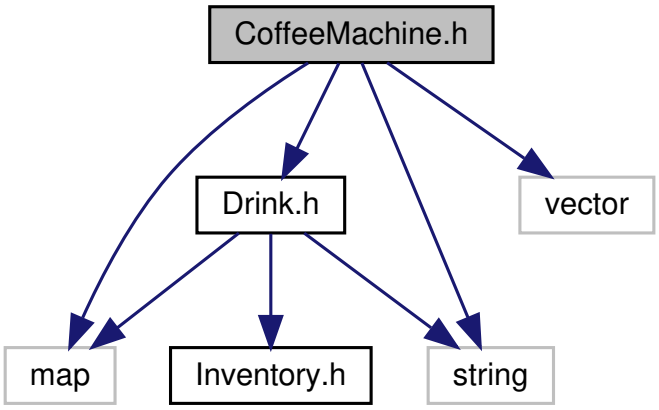
Here is the caller graph for this function:

4.2 CoffeeMachine.h File Reference

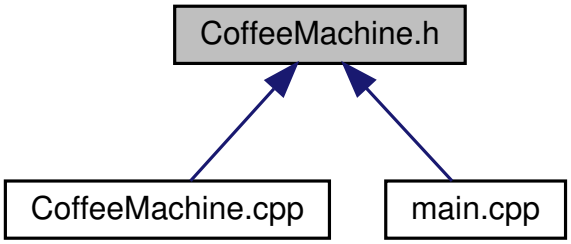
```
#include "Drink.h"
#include <map>
#include <string>
```

```
#include <vector>
```

Include dependency graph for CoffeeMachine.h:



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



Classes

- class **CoffeeMachine**

Functions

- std::string **ingredient2string** (**INGREDIENTS** i)
- const std::map< **INGREDIENTS**, double > **createPriceMap** ()
- const std::map< std::string, std::vector< **INGREDIENTS** > > **createDrinks** ()
- std::vector< **INGREDIENTS** > **defineDrink** (int numIngredients,...)

4.2.1 Function Documentation

4.2.1.1 createDrinks()

```
const std::map<std::string, std::vector< INGREDIENTS> > createDrinks ( )
```

Definition at line 38 of file CoffeeMachine.cpp.

```
38                                                                  {
39   const std::map<std::string, std::vector<INGREDIENTS» recipes = {
40       {"Coffee", defineDrink(5, INGREDIENTS::Coffee, INGREDIENTS::Coffee,
41                                 INGREDIENTS::Coffee, INGREDIENTS::Sugar,
42                                 INGREDIENTS::Cream)},
43       {"Decaf Coffee",
44        defineDrink(5, INGREDIENTS::DecafCoffee, INGREDIENTS::DecafCoffee,
45                    INGREDIENTS::DecafCoffee, INGREDIENTS::Sugar,
46                    INGREDIENTS::Cream)},
47       {"Caffe Latte",
48        defineDrink(3, INGREDIENTS::Espresso, INGREDIENTS::Espresso,
49                    INGREDIENTS::SteamedMilk)},
50       {"Caffe Americano",
51        defineDrink(3, INGREDIENTS::Espresso, INGREDIENTS::Espresso,
52                    INGREDIENTS::Espresso)},
53       {"Caffe Mocha",
54        defineDrink(4, INGREDIENTS::Espresso, INGREDIENTS::Cocoa,
55                    INGREDIENTS::SteamedMilk, INGREDIENTS::WhippedCream)},
56       {"Cappuccino",
57        defineDrink(4, INGREDIENTS::Espresso, INGREDIENTS::Espresso,
58                    INGREDIENTS::SteamedMilk, INGREDIENTS::FoamedMilk)}};
59 #define DRINKS_SIZE recipes.size()
60   return recipes;
61 }
```

References Cocoa, Coffee, Cream, DecafCoffee, defineDrink(), Espresso, FoamedMilk, SteamedMilk, Sugar, and WhippedCream.

Here is the call graph for this function:

4.2.1.2 createPriceMap()

```
const std::map< INGREDIENTS, double> createPriceMap ( )
```

Definition at line 11 of file CoffeeMachine.cpp.

```
11                                                          {
12   const std::map<INGREDIENTS , double> ingredientPriceMap = {
13       {INGREDIENTS::Coffee, 0.75},
14       {INGREDIENTS::DecafCoffee, 0.75},
15       {INGREDIENTS::Sugar, 0.25},
16       {INGREDIENTS::Cream, 0.25},
17       {INGREDIENTS::SteamedMilk, 0.35},
18       {INGREDIENTS::FoamedMilk, 0.35},
19       {INGREDIENTS::Espresso, 1.10},
20       {INGREDIENTS::Cocoa, 0.9},
21       {INGREDIENTS::WhippedCream, 1.00}
22   };
23   return ingredientPriceMap;
24 }
```

References Cocoa, Coffee, Cream, DecafCoffee, Espresso, FoamedMilk, SteamedMilk, Sugar, and Whipped↩ Cream.

4.2.1.3 defineDrink()

```
std::vector< INGREDIENTS> defineDrink (
              int numIngredients,
               ...  )
```

Definition at line 26 of file CoffeeMachine.cpp.

```
26                                                          {
27   std::vector<INGREDIENTS> v;
28   va_list vaList;
29   va_start(vaList, numIngredients); // initialize vaList for num number of arguments
30   // can't pre-allocate vector size, enums are all converted to lowest value
31   for (int i = 0; i < numIngredients; ++i) {
32     v.push_back(va_arg(vaList, INGREDIENTS));
33   }
34   va_end(vaList); // clean memory reserved for vaList
35   return v;
36 }
```

Referenced by createDrinks().

Here is the caller graph for this function:

4.2.1.4 ingredient2string()

```
std::string ingredient2string (
              INGREDIENTS i )
```

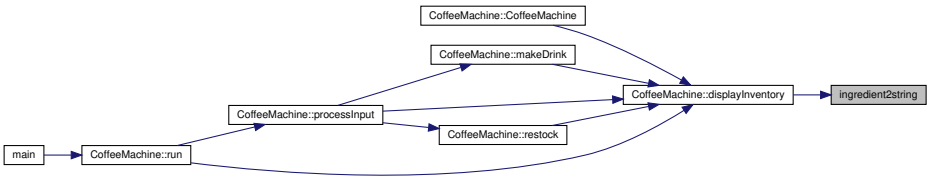
Definition at line 13 of file CoffeeMachine.h.

```
13                                                  {
14   switch (i) {
15   case INGREDIENTS::Coffee: {
16     return "Coffee";
17   }
18   case INGREDIENTS::DecafCoffee: {
19     return "Decaf Coffee";
20   }
21   case INGREDIENTS::WhippedCream: {
22     return "Whipped Cream";
23   }
24   case INGREDIENTS::Cocoa: {
25     return "Cocoa";
26   }
27   case INGREDIENTS::Espresso: {
28     return "Espresso";
29   }
30   case INGREDIENTS::Sugar: {
31     return "Sugar";
32   }
33   case INGREDIENTS::Cream: {
34     return "Cream";
35   }
36   case INGREDIENTS::SteamedMilk: {
37     return "Steamed Milk";
38   }
39   case INGREDIENTS::FoamedMilk: {
40     return "Foamed Milk";
41   }
42   default:
43     throw std::exception();
44   }
45 }
```

References Cocoa, Coffee, Cream, DecafCoffee, Espresso, FoamedMilk, SteamedMilk, Sugar, and Whipped↩ Cream.

Referenced by CoffeeMachine::displayInventory().

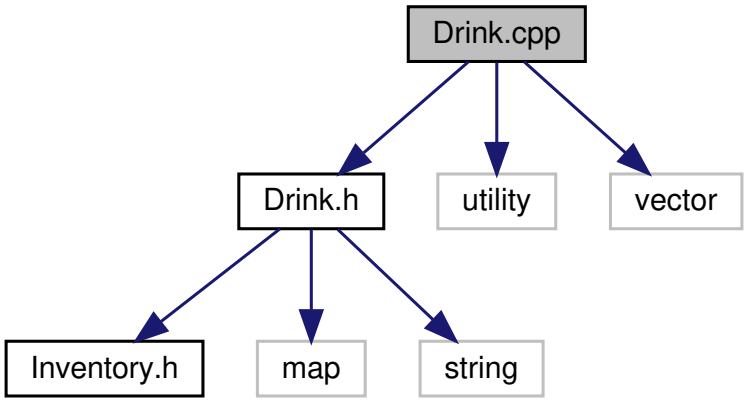
Here is the caller graph for this function:



4.3 Drink.cpp File Reference

```
#include "Drink.h"
#include <utility>
#include <vector>
```

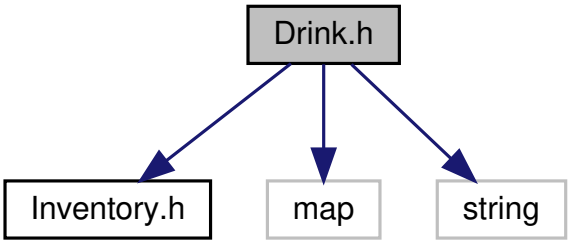
Include dependency graph for Drink.cpp:



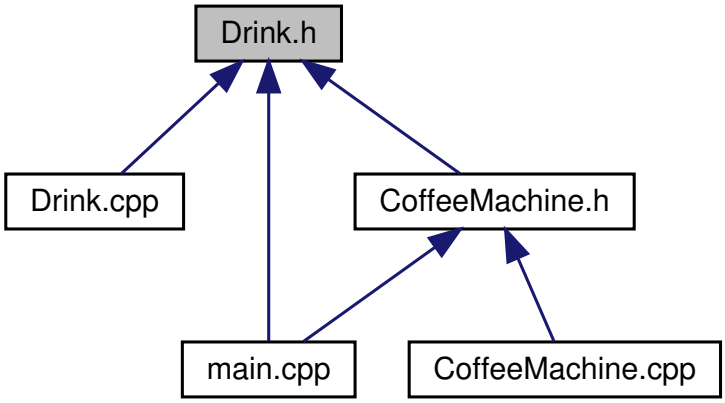
4.4 Drink.h File Reference

```
#include "Inventory.h"
#include <map>
#include <string>
```

Include dependency graph for Drink.h:



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

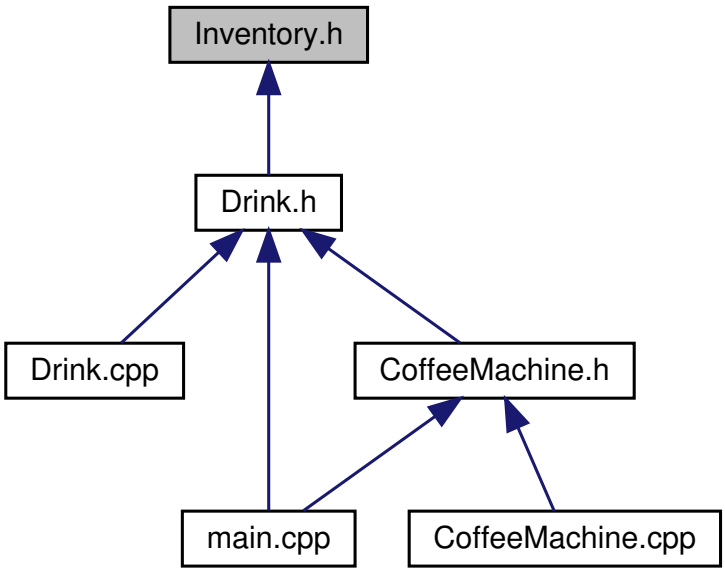


Classes

- class **Drink**

4.5 Inventory.h File Reference

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



Enumerations

- enum **INGREDIENTS** {
  Coffee, **DecafCoffee**, **Sugar**, **Cream**,
  SteamedMilk, **FoamedMilk**, **Espresso**, **Cocoa**,
  WhippedCream }

4.5.1 Enumeration Type Documentation

4.5.1.1 INGREDIENTS

enum **INGREDIENTS**

Enumerator

Coffee	
DecafCoffee	
Sugar	
Cream	
SteamedMilk	
FoamedMilk	
Espresso	
Cocoa	
WhippedCream	

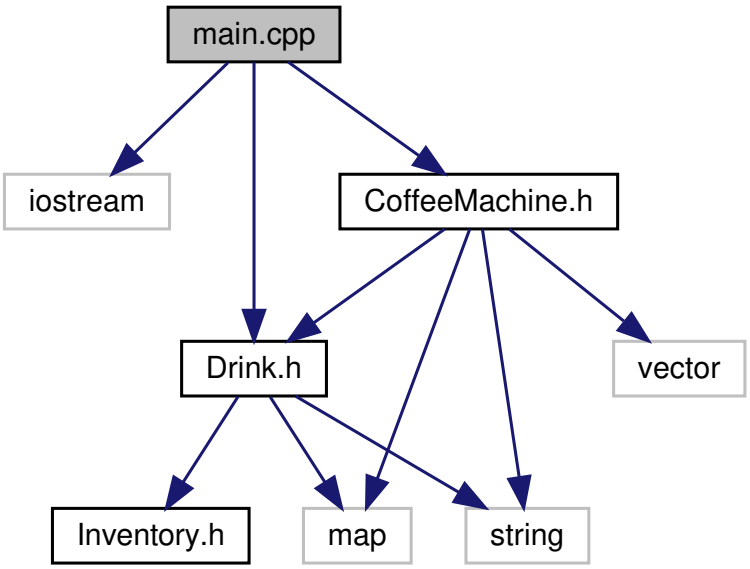
Definition at line 8 of file Inventory.h.

```
8                {
9   Coffee,
10   DecafCoffee,
11   Sugar,
12   Cream,
13   SteamedMilk,
14   FoamedMilk,
15   Espresso,
16   Cocoa,
17   WhippedCream
18 };
```

4.6 main.cpp File Reference

```
#include <iostream>
#include "CoffeeMachine.h"
#include "Drink.h"
```

Include dependency graph for main.cpp:



Functions

- int **main** ()

4.6.1 Function Documentation

4.6.1.1 main()

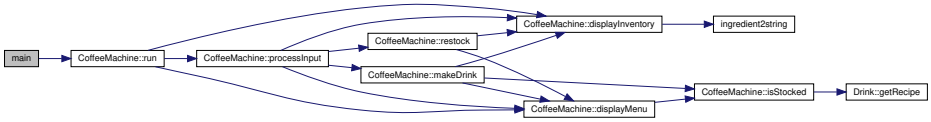
```
int main ( )
```

Definition at line 5 of file main.cpp.

```
5           {
6   CoffeeMachine c;
7   c.run();
8 }
```

References CoffeeMachine::run().

Here is the call graph for this function:



Index