Detailed Design

for

Brew Day!

Version 1.0 approved

Prepared by Zhenghao Wu, Anqin Zha, Renjie Deng and Ruichao Zhong

Dijkstra

Tuesday, April 9, 2019

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Zhenghao Wu, Anqin Zha, Renjie Deng and Ruichao Zhong	2019-04-09	The first version of the Detail Design	1.0

1. Overview

1.1 Project description

This project is to develop a desktop-base software for home brewer to brew beers. Home brewer can use the product to maintain recipe, ingredients and equipment information, he can also receive recommendation from this software and take notes on each brewing.

1.2 References

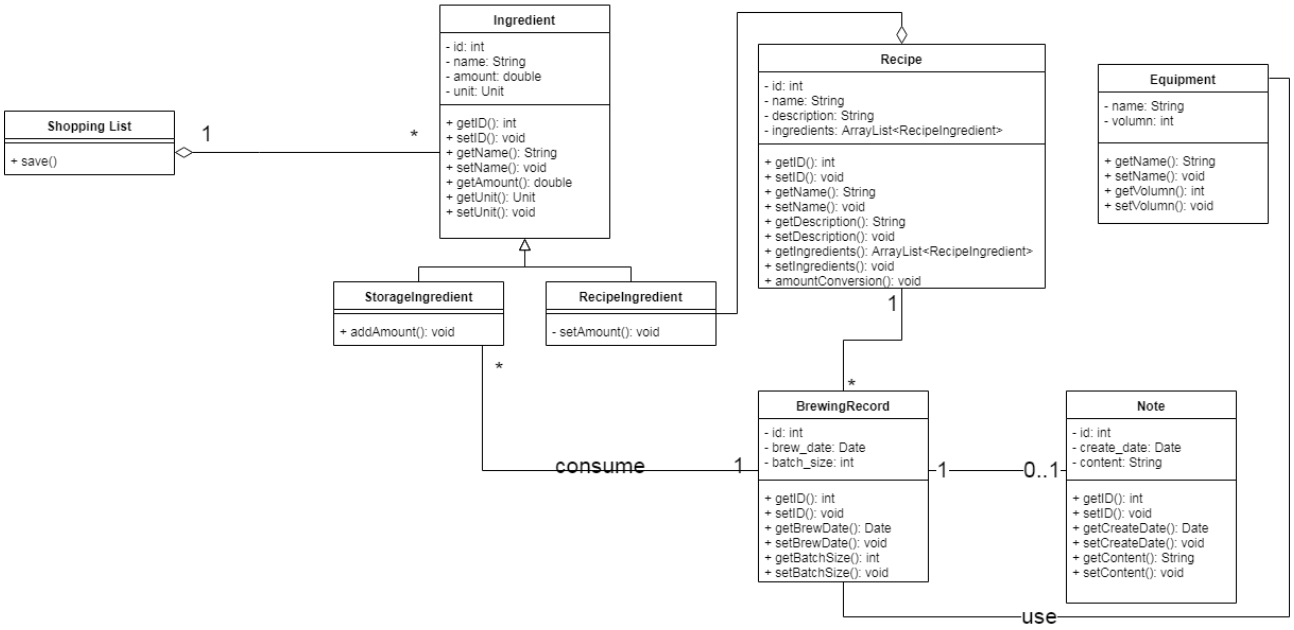
Z. H. Wu, et al., “Software Requirements Specification for Brew Day! Version 5.0” Apr. 2nd, 2019.
Z. H. Wu, et al., “Architecture Design for Brew Day! Version 1.0” Apr. 2nd, 2019.

1.3 Design purpose

This design is to restructure the previous design, refine each class in previous class diagram, give detailed information of each classes and implement interface design. Thus when conducting implementation, programmers can get a more detailed and non-ambiguity design to prevent misunderstanding the design and save time.

2. Overall description

2.1 Class diagram

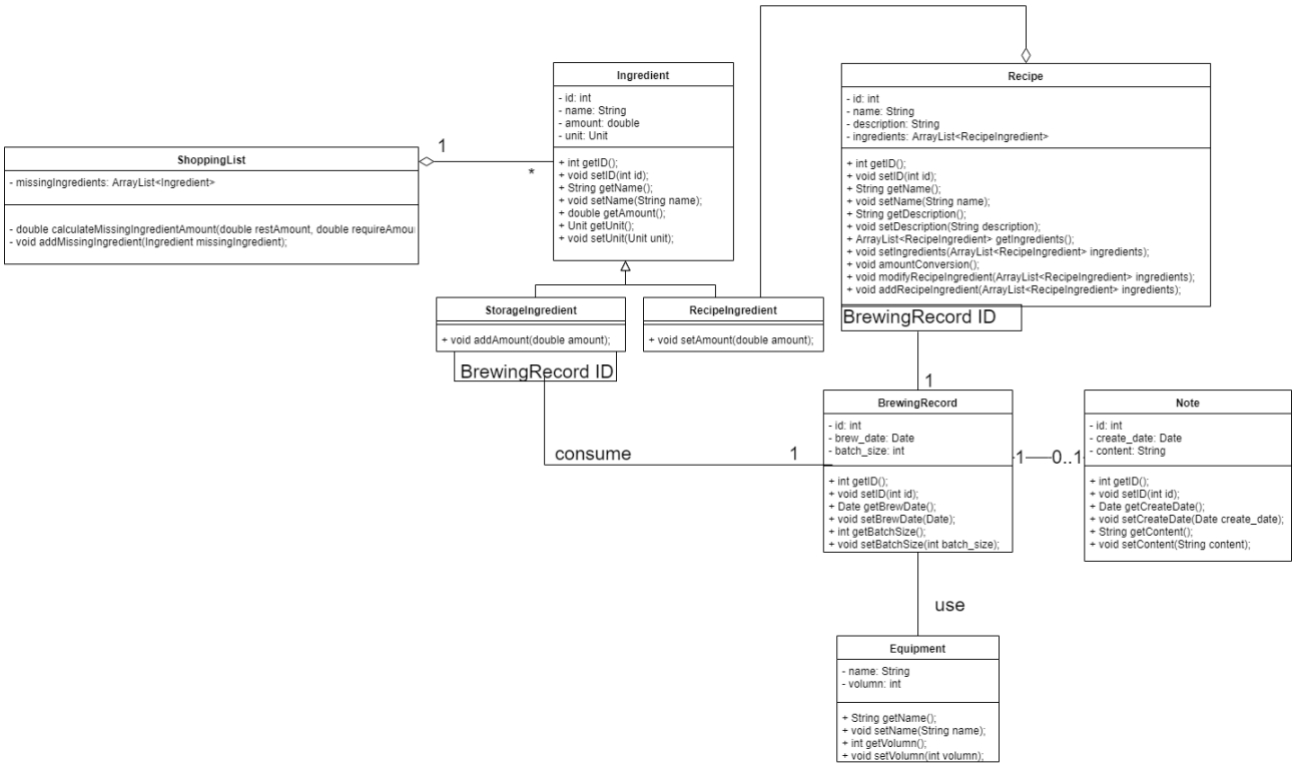


2.2 Refinements

In the refinements. Attributes and features for each class with visibility (private or public) and signatures (return type, parameter and type) are specified. And detail is shown in part 3.2(Classes) of this document.
For restructuring, two qualifiers “BrewingRecord ID” is add to Recipe and StorageIngredient Class in order to change the One-to-Many relationship to One-to-one relationship.

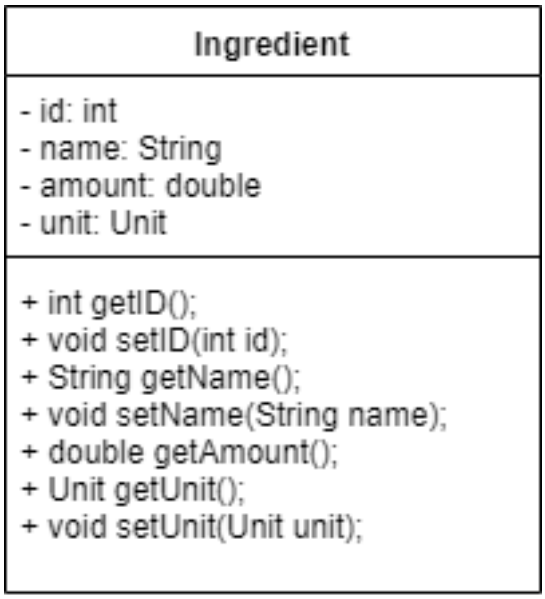
3. Detailed design

3.1 Class diagram



3.2 Classes

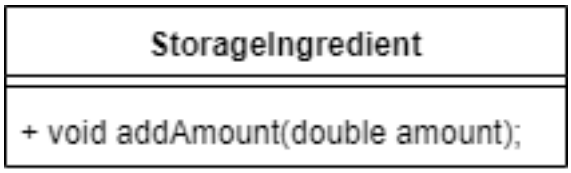
3.2.1 Ingredient



Explanations

For all attributes, there is no any change because all attributes have been with type in last version, and all the attributes are necessary, without needing to add anymore attributes.
For all operations, the changes were adding input parameters and type of specific parameter for the class.
1. For operation “setID();”: add a int type input parameter “id”.
2. For operation “setName();”: add a String type input parameter “name”.
3. For operation “setUnit();”: add a Unit type input parameter “unit”.

3.2.2 StorageIngredient



Explanations

For all attributes, there is no any change because all attributes have been with type in last version, and all the attributes are necessary, without needing anymore attributes.
For all operations, the changes were adding input parameters and type of specific parameter for the class.
1. For operation “addAmount();”: add a double type input parameter “amount”.

3.2.3 RecipeIngredient

RecipeIngredient
+ void setAmount(double amount);

Explanations

For all attributes, there is no any change because all attributes have been with type in last version, and all the attributes are necessary, without needing anymore attributes.
For all operations, the changes were adding input parameters and type of specific parameter for the class.
1. For operation “setAmount();”: add a double type input parameter “amount”.

3.2.4 Recipe

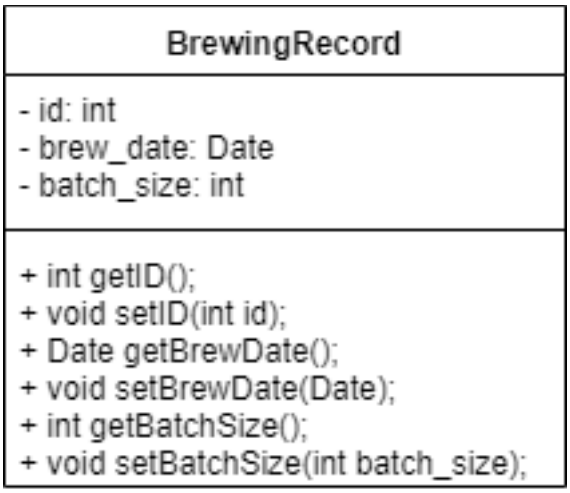
Recipe
- id: int - name: String - description: String - ingredients: ArrayList<RecipeIngredient>
+ int getID(); + void setID(int id); + String getName(); + void setName(String name); + String getDescription(); + void setDescription(String description); + ArrayList<RecipeIngredient> getIngredients(); + void setIngredients(ArrayList<RecipeIngredient> ingredients); + void amountConversion(); + void modifyRecipeIngredient(ArrayList<RecipeIngredient> ingredients); + void addRecipeIngredient(ArrayList<RecipeIngredient> ingredients);

Explanations

For all attributes, there is no any change because all attributes have been with type in last version, and all the attributes are necessary, without needing anymore attributes.
For the reason that user need to have operations to add or modify ingredient to recipe, modifyRecipeIngredient and addRecipeIngredient are added.
1. For operation “setID();”: add a int type input parameter “id”.
2. For operation “setName()”: add a String type input parameter “name”.
3. For operation “setDescription()”: add a String type input parameter “description”.
4. For operation “setIngredient()”: add a ArrayList<RecipeIngredient> type input parameter “ingredients”.

5. For operation “modifyRecipeIngredient()”: set void as return type and add a ArrayList<RecipeIngredient> type input parameter “ingredients”. Additionally, its visibility is public.
6. For operation “addRecipeIngredient()”: set void as return type and add a ArrayList<RecipeIngredient> type input parameter “ingredients”. Additionally, its visibility is public.

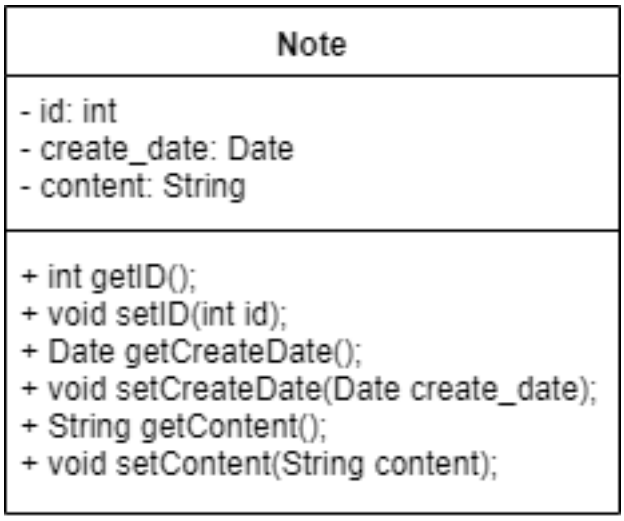
3.2.5 BrewingRecord



Explanations

For all attributes, we have not modified the latest version so that there is no changes on them.
For all operations, we have added parameters to all the “set” functions.
1. For operation “setID”: Give an id to the brewing record. The parameter id refers to the id which will be given to the new BrewingRecord.
2. For operation “setBrewDate”: Save the date of the brew record. The parameter date is the date that brewing record last modified.
3. For operation “setBatchSize”: Set the batch size of each brew. The parameter batch_size is the amount of beer that brew each time.

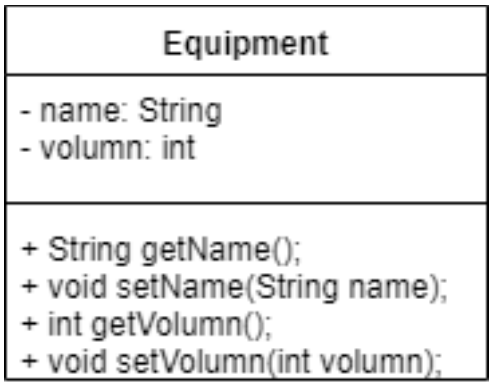
3.2.6 Note



Explanations

For all attributes, there is no any change because all attributes have been with type in last version, and all the attributes are necessary, without needing anymore attributes.
For all operations, the changes were adding input parameters and type of specific parameter for the class.
1. For operation “setID();”: add a int type input parameter “id”.
2. For operation “setCreateDate();”: add a Date type input parameter “create_date”.
3. For operation “setContent();”: add a String type input parameter “content”.

3.2.7 Equipment



Explanations

For all attributes, there is no any change because all attributes have been with type in last version, and all the attributes are necessary, without needing to add anymore attributes.
For all operations, the changes were adding input parameters and type of specific parameter for the class.
1. For operation “setName;”: add a String type input parameter “name”.
2. For operation “setVolumn();”: add a int type input parameter “volumn”.

3.2.8 ShoppingList

ShoppingList
- missingIngredients: ArrayList<Ingredient>
- int calculateMissingIngredientAmount(double restAmount, double requireAmount); - void addMissingIngredient(Ingredient missingIngredient);

Explanations

For all attributes, we have added a new ArrayList attribute called missingIngredients. The ArrayList contains all the ingredients which are needed and the amount of which are not enough in the storage to arrange a brew.
For all operations, we have change the origin save() into 2 totally different operation because shopping list do not need to store in database.
1. Add operation “calculateMissingIngredientAmount”: The method calculate the amount needed for each of the missing ingredient. The parameter restAmount is the remain amount of each ingredient in storage. The parameter requireAmount is the required amount for each ingredient.
2. Add operation “addMissingIngredient”: The method add missing ingredients to the ArrayList missingIngredients while initialing a new ShoppingList. The parameter missingIngredient refers to each ingredient which should be added to shopping list.

Constraints

double calculateMissingIngredientAmount(double restAmount, double requireAmount);

Pre-condition: restAmount and requireAmount should not be null, both parameter should greater than 0. restAmount should less than requireAmount.
Post-condition: the return value shows that what amount of a specific ingredient should buy in order to make a specific recipe. This value should always greater than 0.

4. More considerations

For more information about database design, please refer to Architecture Design for Brew Day! Version 1.0 (Z. H. Wu, et al. Apr. 2nd, 2019.)