/*CRIT B IS DONE BEFORE PROGRAMMING*/

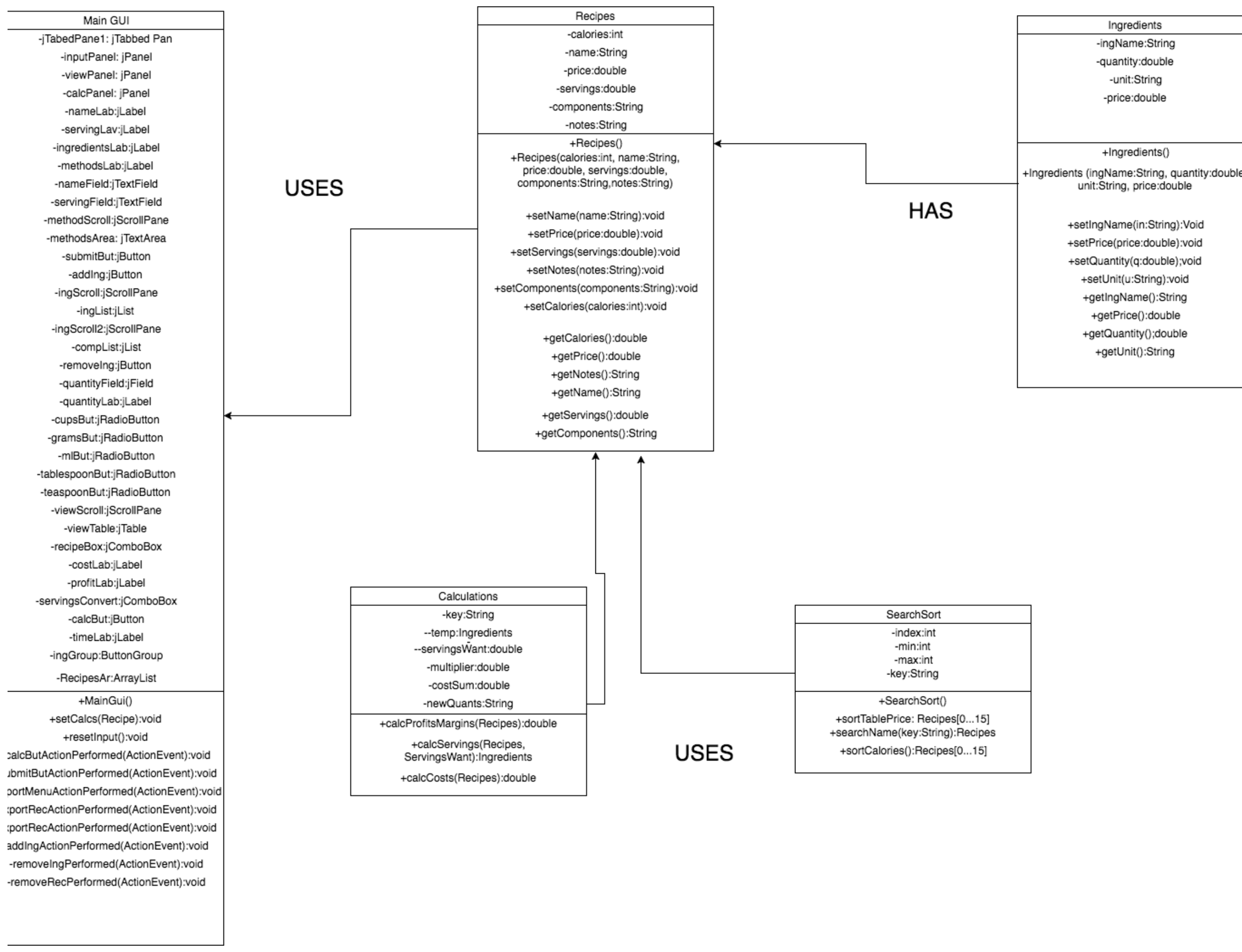
Input/Output Tables

Input	**Data Type**	**Example**
Name	String	"Brownies"
Servings	double	1.5
Ingredients	String	"1.5 Cups Flour"
Methods	String	"1. Preheat Oven"

Output	**Data Type**	**Example**
Name	String	"Chocolate Cake"
Servings	double	3.5
Ingredients	String	"3 tablespoons of Sugar"
Methods	String	"1. Mix all ingredients"
Price	double	530.0
ProfitMargin	double	43.5
Cost	Double	6.5
New Servings	Ingredients (Strings within multiple instances of class Ingredients)	“4.5 grams”

Testing Plan

Input	Normal Value	Border Value	Abnormal Value	Extreme Value	Other
Recipe Name	“Brownies” (String)	Empty: display jOptionPane Message“Please input Name”	No Letters(only numbers): jOptionMessage “Please input Alphabetical name”-not parsed	15+ Characters: jOptionMessage “Reduce name length to avoid format errors”-Not parsed	n/a
Method	“First Step… Second Step…”(String)	Empty: jOptionPane Confirm Dialogue “Are you certain there is not method?”	N/a	n/a	n/a
Ingredients Name JList ComboBox	Flour (String)	n/a	n/a	n/a	n/a
Ingredient Quantity	1.0 (double)	Empty: display jOptionPane Message“Please input quantity”	Non-numerical jOptionMessage “Please input numerical quantity”	n/a	n/a
Ingredient Unit ComboBox	“Grams” (String)	n/a	n/a	n/a	n/a
Servings ComboBox	1.5(double)	n/a	n/a	n/a	n/a
Ingredient cost	34.5	Empty: display jOptionPane Message“Please input cost”	Non-numerical jOptionMessage “Please input numerical quantity”	n/a	n/a
Ingredient name	“Flour” (String)	Empty: display jOptionPane Message“Please input Name”	No Letters(only numbers): jOptionMessage “Please input Alphabetical name”-not parsed	15+ Characters: jOptionMessage “Reduce name length to avoid format errors”-Not parsed	n/a


Main GUI
-jTabedPane1: jTabbed Pan
-inputPanel: jPanel
-viewPanel: jPanel
-calcPanel: jPanel
-nameLab:jLabel
-servingLav:jLabel
-ingredientsLab:jLabel
-methodsLab:jLabel
-nameField:jTextField
-servingField:jTextField
-methodScroll:jScrollPane
-methodsArea: jTextArea
-submitBut:jButton
-addIng:jButton
-ingScroll:jScrollPane
-ingList:jList
-ingScroll2:jScrollPane
-compList:jList
-removeIng:jButton
-quantityField:jField
-quantityLab:jLabel
-cupsBut:jRadioButton
-gramsBut:jRadioButton
-mlBut:jRadioButton
-tablespoonBut:jRadioButton
-teaspoonBut:jRadioButton
-viewScroll:jScrollPane
-viewTable:jTable
-recipeBox:jComboBox
-costLab:jLabel
-profitLab:jLabel
-servingsConvert:jComboBox
-calcBut:jButton
-timeLab:jLabel
-ingGroup:ButtonGroup
-RecipesAr:ArrayList
+MainGui()
+setCalcs(Recipe):void
+resetInput():void
calcButActionPerformed(ActionEvent):void
ubmitButActionPerformed(ActionEvent):void
portMenuActionPerformed(ActionEvent):void
portRecActionPerformed(ActionEvent):void
portRecActionPerformed(ActionEvent):void
addIngActionPerformed(ActionEvent):void
-removeIngPerformed(ActionEvent):void
-removeRecPerformed(ActionEvent):void
USES
Recipes
-calories:int
-name:String
-price:double
-servings:double
-components:String
-notes:String
+Recipes()
+Recipes(calories:int, name:String, price:double, servings:double, components:String,notes:String)
+setName(name:String):void
+setPrice(price:double):void
+setServings(servings:double):void
+setNotes(notes:String):void
+setComponents(components:String):void
+setCalories(calories:int):void
+getCalories():double
+getPrice():double
+getNotes():String
+getName():String
+getServings():double
+getComponents():String
HAS
Ingredients
-ingName:String
-quantity:double
-unit:String
-price:double
+Ingredients()
+Ingredients (ingName:String, quantity:double unit:String, price:double
+setIngName(in:String):Void
+setPrice(price:double):void
+setQuantity(q:double);void
+setUnit(u:String):void
+getIngName():String
+getPrice():double
+getQuantity();double
+getUnit():String
Calculations
-key:String
--temp:Ingredients
--servingsWant:double
-multiplier:double
-costSum:double
-newQuants:String
+calcProfitsMargins(Recipes):double
+calcServings(Recipes, ServingsWant):Ingredients
+calcCosts(Recipes):double
USES
SearchSort
-index:int
-min:int
-max:int
-key:String
+SearchSort()
+sortTablePrice: Recipes[0...15]
+searchName(key:String):Recipes
+sortCalories():Recipes[0...15]

1. Create List of *Recipes*
    - Create list of *Ingredients* to be implemented within *Recipes*
        - Implement inheritance separating Ingredients into *wet* and *dry*
    - Populate array with default recipes
2. Create process to save recipe list
    - Allow user to choose directory
    - Create save folder in directory including:
        i. Text document with Recipes
        ii. Text document with catalog of ingredients
    - Process:
        i. Traverse through Recipe array, write into a text document individually
        ii. Traverse through ingredients Jlist, write into a text document individually
3. Create process to load recipe list
    - Load recipes into created list
        i. Populate "View Recipes" Tab with loaded recipes
    - Load ingredients into Jlist in the ingredients builder
4. Create process for the Recipe builder
    - Create new GUI for user to insert ingredients+quantities
    - Create link between Ingredient GUI and main GUI
        i. Ingredients are passed to main GUI text box
5. Create method to input new recipe
    - Add recipe to list
    - Call upon save method
        i. Rewrite recipe text file to include new recipe
        ii. Rewrite ingredients text file if new ingredients were used
6. Create method to delete recipe
    - Delete from list
    - Call upon save method
        i. Rewrite recipe text file to not include deleted recipe.
7. Create method to perform basic calculations
    - Determine what recipes to perform calculations
    - Calculate profit, costs, servings etc. (Methods in the Recipe class,)
    - ****This is Pseudocode and therefore does not count in the word count****
        i. Profit Margins: *((price*servings)/cost)*100 ///Expressed as percentage///*
        ii. Servings Converter: reduces or increases ingredient quantities based on servings desired

```
temp=ing[1]
servingsWant=input
multiplier=servingsWant/servings
loop for i from 0 to ing.size
        temp.get(i).setQuantity(temp.get(i).getQuantity*multiplier)
```

[1] ing is an arraylist of Ingredients within each instance of Recipes

```
            newQuants=newQuants+”\n”+temp.get(i).getQuantity
        end loop
        Output newQuants
    iii.    Cost:
        costSum=0.0
        loop for i from 0 to ing.size
            costSum=ing.get(i).getIngPrice+costSum
        end loop
        Output costSum
```

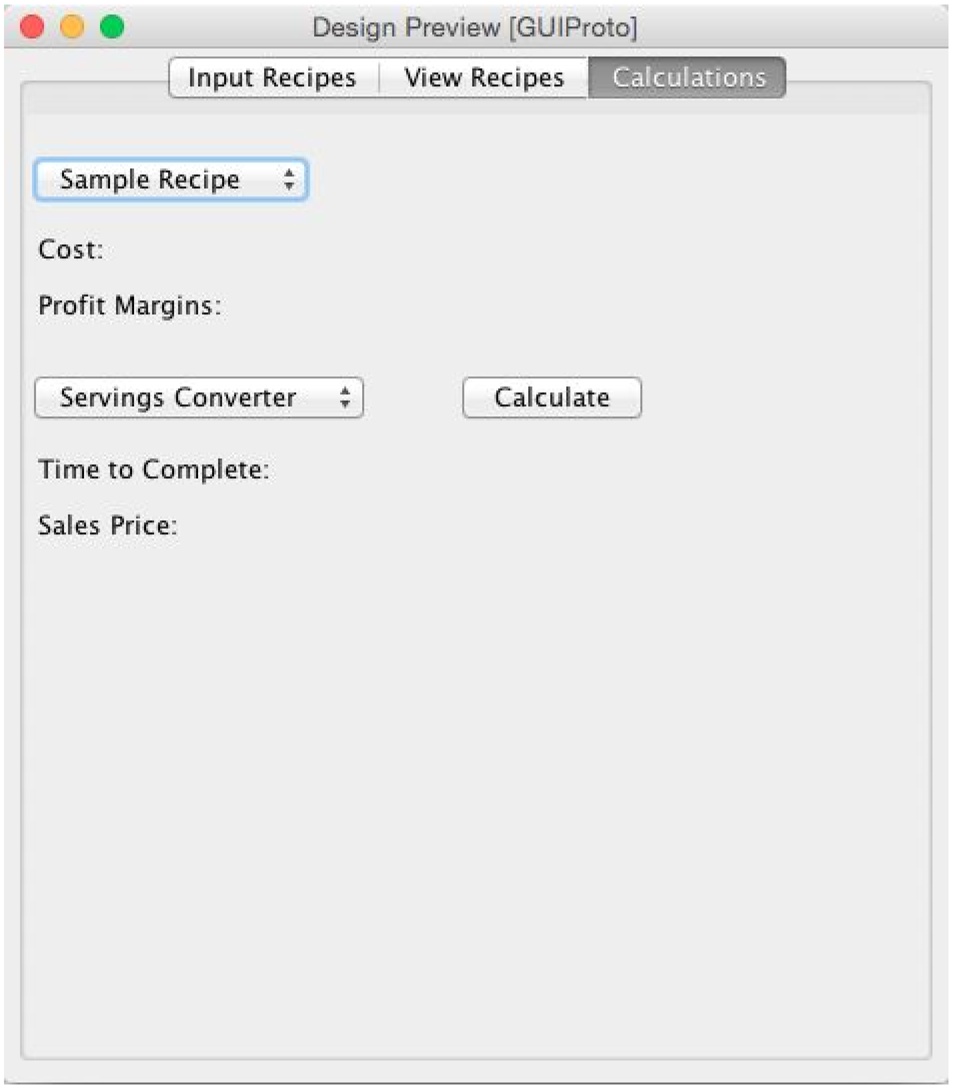
8. Sort Recipes Alphabetically

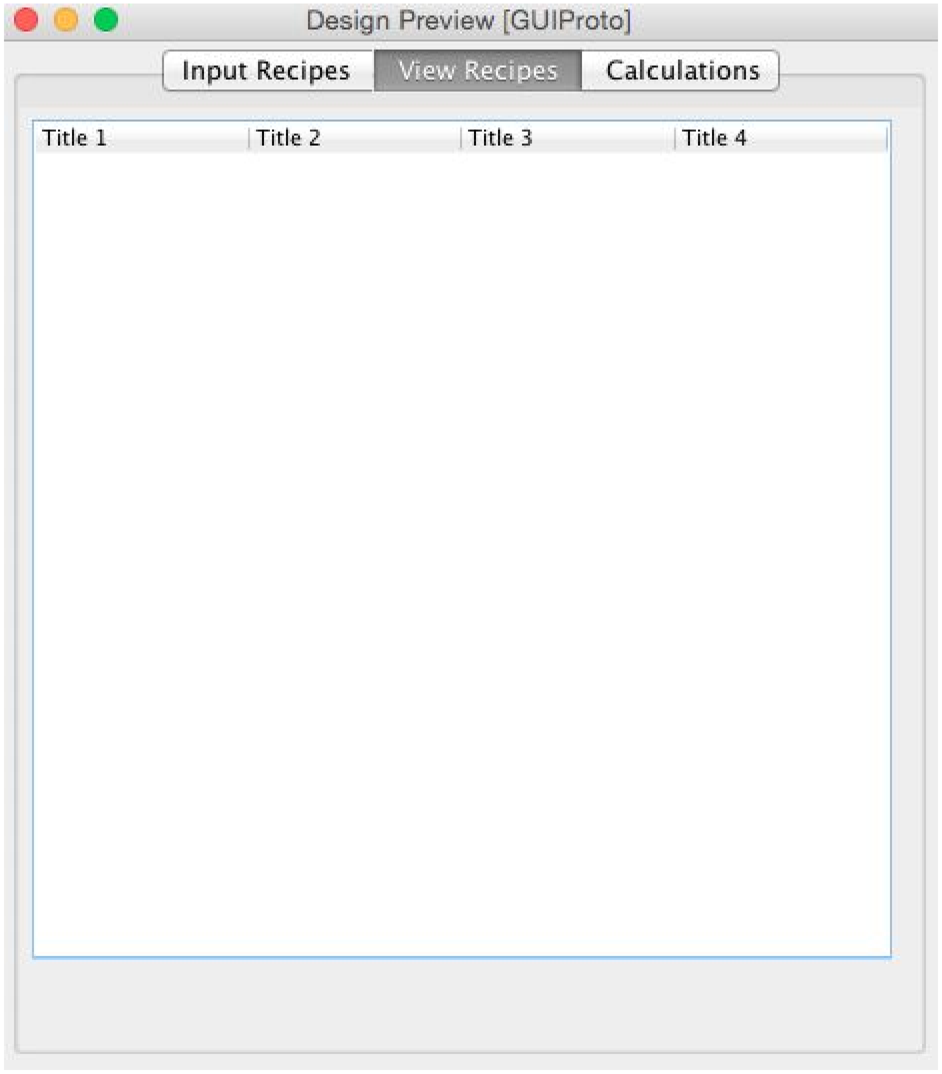
```
loop for j from 0 to numberRecipes
    loop for i  from 0 to numberRecipes
        if recipe[i].getName.compareTo(recipe[i+1].getName) > 0
            Recipe temp = recipe[i]
            recipe[i]=recipe[i+1]
            recipe[i+1]=temp
        end if
    end loop
end loops
```

9. Sort Recipes by profit margins (ascending)

```
loop for j from 0 to numberRecipes
    loop for i  from 0 to numberRecipes
        if recipe[i].getPMarg-recipe[i+1].getPMarg > 0
            Recipe temp = recipe[i]
            recipe[i]=recipe[i+1]
            recipe[i+1]=temp
        end if
    end loop
end loops
```

*For client opinions refer to second interview




Design Preview [GUIProto]
Input Recipes
View Recipes
Calculations
Title 1
Title 2
Title 3
Title 4