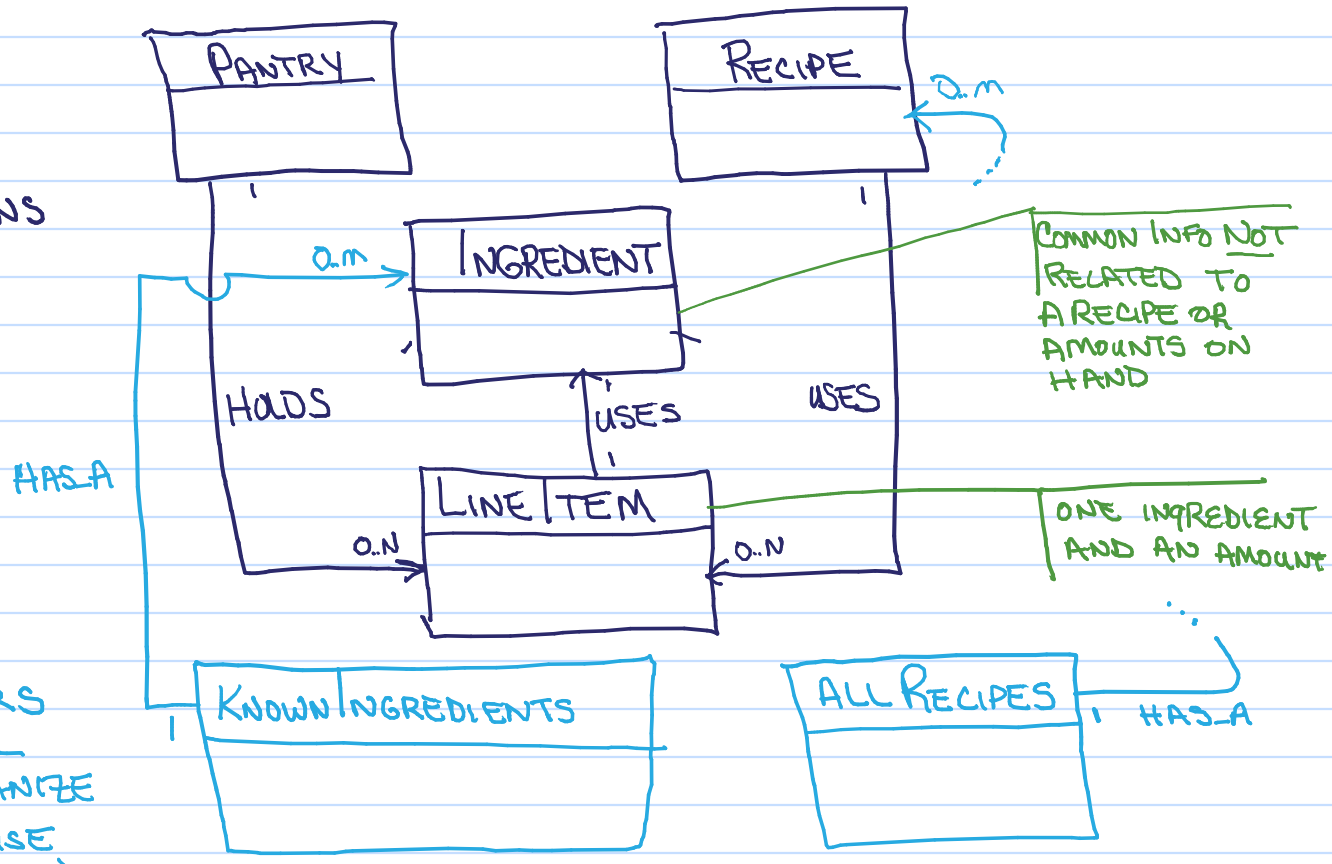
IDEA: KEEP TRACK OF INGREDIENTS ON HAND AND DETERMINE IF A CERTAIN COOKIE RECIPE CAN BE MADE (OR IF NOT, WHAT NEED TO BUY)

(1) HIGH LEVEL CONCEPTS & ASSOCIATIONS



(2) ADD CONTAINERS THAT WILL HELP ORGANIZE (SET UP HOUSE KEEPING)

NB: POSSIBLY A SINGLETON VARIABLE AND NOT A CLASS

(3)

ADD KEY ATTRIBUTES AND HOUSEKEEPING METHODS

- GETTERS & SETTERS AS NEEDED
- CONSTRUCTOR(S)

WORK CLASS BY CLASS

ATTRIBUTES GO WHERE THEY BELONG

"COHESION"

INGREDIENT

```
PRIVATE STRING NAME
PRIVATE STRING RECIPEUNIT          //eg CUP
PRIVATE STRING PURCHASEUNIT        //eg POUND
PRIVATE DOUBLE RECIPEQUANTITY      // USUALLY 1
PRIVATE DOUBLE PURCHASEQUANTITY    //eg 5
PRIVATE DOUBLE RATIOPURCHASETORECIPE   //eg 15C IN 5 POUNDS
```

```
PUBLIC INGREDIENT(... ALL ABOVE)
```

LINE ITEM

```
PRIVATE DOUBLE AMOUNT          //IN RECIPEUNITS
PRIVATE INGREDIENT ITEM
```

```
PUBLIC LINEITEM(...ALL ABOVE)
PUBLIC DOUBLE GETAMOUNT()
PUBLIC INGREDIENT GETITEM()
PUBLIC STRING TOSTRING()
```

(4) INVENT METHODS FOR THE "PAYOFF" OR "HEAVY LIFTING"

PANTRY

```
PRIVATE STRING PANTRYNAME
PRIVATE LIST<LINEITEM> MYPANTRY
```

```
PUBLIC PANTRY(STRING N)
PUBLIC BOOL ADDTOPANTRY(Ingredient i,
                        DOUBLE AMOUNT)
PUBLIC LIST<LINEITEM> NEEDEDTOBAKE(
                        RECIPE r, DOUBLE PARTIAL)
PUBLIC BOOL USE(RECIPE r, DOUBLE PARTIAL)
```

RETURNS LIST OF NEEDED ADDITIONAL INGREDIENTS OR NULL IF NONE

//1 = full Recipe

REDUCES PANTRY BY AMOUNTS OF EACH INGRIDENT IN RECIPE.

PRECONDITION: SUFFICIENT STOCK, RETURNS FALSE OTHERWISE

(5) WRITE HOWTOS FOR THE HARD PARTS

...

AND ADJUST STEPS (1) TO (4) UNTIL YOU CAN "SEE THE FOOTSTEPS FOR THE PROGRAMMER"

AND LEAVE SOME THINGS IN STUBS FOR NOW IF YOU WISH

DO THIS TOP DOWN FOR COMPLEX METHODS: START IN ALL ENGLISH, THEN REFINE INTO C# TERMS (VARIABLES, CONTROL STRUCTURES)

```
// List<LineItem> NeededToBake(Recipe r, Double partial)
//
// Create Empty NeededItems List To Hold
        Shopping Needs
// For Each Line In Recipe: (One Needed Ingredient)
//         Lookup L.getItem() in myPantry, save in P
//         If Found
//             If L.getAmount > P.getAmount
//                 (Recipe needs)   (Pantry Contents)
//                 Add new line to NeededItems
//                 with Amount = Difference
//             Else Continue
//                 (Have sufficient of than ingredient)
//         Else Add new line to Needed Items
//                 with Full Amount of Recipe
// If count of NeededItems > 0, return it
// Else return n
// TODO: If partial != 1, replace r with a
        new Recipe with Amounts
        Decreased.
```

// Will need a helper function

LineItem FindIngredientLine(List<LineItems> l, Ingredient i)

... some more of step (3) & (4)

RECIPE : IEnumerable

To allow foreach lineItem l in aRecipe

```
PRIVATE STRING RECIPENAME
PRIVATE LIST<LINEITEM> RECIPELIST
PRIVATE INT QUANTITYMADE
PRIVATE STRING INSTRUCTIONS
PRIVATE INT BAKINGTIME      //MINUTES
PRIVATE INT BAKINGTEMP      // F°
```

```
PUBLIC RECIPE (... ALL ABOVE ...)
PUBLIC STRING TOSTRING()
```

Provides String Suitable For Printing Recipe Card (Name, one line each ingredient, instructions, etc.)

KnownIngredients

```
PRIVATE LIST<INGREDIENT> MYINGREDIENTS
```

```
PUBLIC KNOWNINGREDIENTS(INT SIZE)
PUBLIC INGREDIENT FINDINGREDIENT(STRING NAME)
PUBLIC BOOL ADDINGREDIENT(INGREDIENT i)
PUBLIC STRING TOSTRING()
```

AllRecipes

```
PRIVATE LIST<RECIPE> MYRECIPIES
```

```
PUBLIC ALLRECIPIES(INT SIZE)
PUBLIC RECIPE FINDRECIPE(STRING NAME)
PUBLIC BOOL ADDRECIPE(RECIPE r)
PUBLIC STRING TOSTRING
```

Find similar names, ignore case, etc