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

   PANTRY

   RECIPE

   INGREDIENT

   HOLS

   LINE ITEM

   USES

   USES

   KNOWN INGREDIENTS

   ALL RECIPES

   HAS A

   O.M

   O.N

   O.M

   HAS A

   Common info not related to a recipe or amounts on hand

   One ingredient and an amount

2. Add containers that will help organize (set up housekeeping)

   NB: Possibly a singleton variable and not a class
3
ADD KEY ATTRIBUTES
AND HOUSE KEEPING METHODS
- GETTERS & SETTERS AS NEEDED
- CONSTRUCTOR(S)

WORK CLASS
BY CLASS
ATTRIBUTES GO WHERE THEY BELONG
"COHESION"

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

INGREDIENT

PRIVATE STRING NAME
PRIVATE STRING RECIPE UNIT // 1/2 CUP
PRIVATE STRING PURCHASE UNIT // 2 LB POUND
PRIVATE DOUBLE RECIPE QUANTITY // USUALLY 1
PRIVATE DOUBLE PURCHASE QUANTITY // 1/2 5
PRIVATE DOUBLE_RATIO_PURCHASE_TO_RECIPE // 1/4 1/5 1/15 1/50 1/500

PUBLIC INGREDIENT (... ALL ABOVE)

LINE ITEM

PRIVATE DOUBLE AMOUNT // IN RECIPE UNITS
PRIVATE INGREDIENT ITEM

PUBLIC LINE ITEM (... ALL ABOVE)
PUBLIC DOUBLE GET_AMOUNT()
PUBLIC INGREDIENT GET_ITEM()
PUBLIC STRING TO_STRING()

PANTRY

PRIVATE STRING PANTRY NAME
PRIVATE LIST<LINE ITEM> MY PANTRY

PUBLIC PANTRY (STRING NAME)
PUBLIC BOOL ADD_TO_PANTRY (INGREDIENT I, DOUBLE AMOUNT)
PUBLIC LIST<LINE ITEM> NEEDS_TO_BAKE (RECIPE R, DOUBLE PARTIAL) // 1 = FULL
PUBLIC BOOL USE (RECIPE R, DOUBLE PARTIAL)

REDUCES PANTRY BY AMOUNT OF EACH INGREDIENT
IN RECIPE.
PRECONDITION: SUFFICIENT STOCK, RETURNS FALSE OTHERWISE

RETURNS LIST OF NEEDED ADDITIONAL INGREDIENTS OR NULL IF NONE.
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)

Will Need A Helper Function

```
list<lineitem> findIngredientLine(list<lineitem> r, ingredient i)
```

// 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 My Pantry, Save In P

// If Found

// If L.GetAmount() > P.GetAmount (Recipe Needs) (Pantry Contents)

// Add New Line To Needed Items With Amount = Difference

// Else Continue

// If L.Count Of Needed Items > 0, RETURN IT

// Else Return 0

// To Do: If Partial != 1, Replace r With A New Recipe With Amounts Decreased.
To allow
foreach lineItem \textbackslash i \textbackslash n \textbackslash a \textbackslash recipe

**RECIPE : IEnumerable**

**PRIVATE STRING** RECIPE \textit{NAME}
**PRIVATE LIST \textless LINEITEM \textgreater** RECIPE \textit{LIST}
**PRIVATE INT** QUANTITY \textit{MADE}
**PRIVATE STRING** INSTRUCTIONS
**PRIVATE INT** BAKING \textit{TIME} //MINUTES
**PRIVATE INT** BAKING \textit{TEMP} //F°

**PUBLIC RECIPE (... ALL ABOVE...)**
**PUBLIC STRING** To \textit{STRING}()

**PUBLIC** K\textit{NOWN INGREDIENTS}
**PRIVATE LIST \textless INGREDIENT \textgreater** MY \textit{INGREDIENTS}

**PUBLIC** KN\textit{OWN INGREDIENTS (INT SIZE)}
**PUBLIC** INGREDIENT \textit{Find INGREDIENT (STRING NAME)}
**PUBLIC** BOOL \textit{Add INGREDIENT (INGREDIENT \textit{i})}
**PUBLIC STRING** To \textit{STRING}()

**ALLRECIPES**
**PRIVATE LIST \textless RECIPE \textgreater** MY \textit{RECIPES}

**PUBLIC ALLRECIPES (INT SIZE)**
**PUBLIC** RECIPE \textit{Find RECIPE (STRING NAME)}
**PUBLIC** BOOL \textit{Add RECIPE (RECIPE \textit{i})}
**PUBLIC STRING** To \textit{STRING}()

Provides string suitable
for printing recipe
card (name, one line
each ingredient,
instructions, etc.)

Find similar
names, ignore case, etc.