Class name: Food

Parent Class: -- (abstract)
Clean Architecture Level: Entity

Responsibilities:

- Stores the name of food item
- Stores the quantity of a food
- Stores the unit of a food
- Method #1:

  changeQuantity - Method changes a specified quantity of the food item

Collaborates:

Children:

- PerishableFood
- NonPerishableFood

Class name: PerishableFood

Parent Class: Food
Clean Architecture Level: Entity

Responsibilities:

// This entity takes a name, quantity and expiry date and creates a PerishableFood object.

- updateExpired - Method updates the expiry date of a perishable food item?

- getExpiryStatus - Method returns the expiry status of the perishable food item (ex. Good vs Expired)

- getExpiryDate - Method returns expiry date

Collaborates:

- Food(parent)

Class name: NonPerishableFood

Parent Class: Food
Clean Architecture Level: Entity

Responsibilities:

- This entity takes the name, quantity, and unit of a NonPerishableFood

Collaborates:

- Food (parent)

Class name: Recipe

Parent Class: --
Clean Architecture Level: Entity

Responsibilities:

// Represents a recipe that will contain a name and its ingredients and instructions

- Stores the name of the recipe
- Stores/returns the type of ingredients needed for the recipe
- Stores/returns the quantity of ingredients needed for the recipe
- Stores/returns the instructions for a recipe

Collaborates:

Class name: RecipeHandler

Parent Class: --

Clean Architecture Level: Use Cases

1. On application boot:
   - Takes an array of array of: String name, HashMap (foodName, ArrayList(Integer foodQuantity, String foodUnits) String instructions
   - Create recipe objects and store them in an array
2. On needing a recipe recommendation:
   - Take an integer for parameter
   - Calls FoodHandler for string of all foods
   - Rank recipes stored based on current inventory vs what the recipe calls for
3. Has a method getAllRecipes that returns an ArrayList of all the recipes.
4. On adding on recipe
   - Not currently in-scope

Collaboration:

- Recipe
- FoodHandler

Class name: FoodHandler

Parent Class: --

Clean Architecture Level: Use Case

1. On application boot:
   - Take an array with data structure of: String foodName, String quantity, String measurement (and optional String day, String month, and String year)
   - Create perishable and non-perishable foods and store them in an array
2. On recording a new food item to inventory:
   - Takes a parameter with: String foodName, String quantity, String measurement (and optional String day, String month, String year) and creates a Food object
   - Adds it into the Array from boot
3. On needing all food:
   - Output an array of all foods stored
4. Alerts the user when the PerishableFood items have expired
5. Updates the PerishableFood status when expiry date has been reached

Collaboration:

- PerishableFood
- Non-perishableFood

Class name: RecipeController

Parent Class: --

Clean Architecture Level: Controller

1. On Application Boot:
   - Takes an array of strings and sorts it into an array of arrays of String name, String[] foodName, int[] foodQuantity, String[] foodUnits, String instructions for each string.
   - Calls RecipeHandler with that array to load the data into RecipeHandler
2. RecipeHandler outputting recipe recommendation, outputs the recipe recommendations formatted

Collaboration:

- RecipeHandler

Class name: FoodController

Parent Class: --

Clean Architecture Level: Controller

1. Creates a single food item by calling FoodHandler with an array of: String foodName, String quantity, String measurement, (optional String day, String month, String year). This creates a single food.
2. On application boot: Calls FoodHandler with an array of array of strings (optional String day, String month, String year). This loads multiple foods.

Collaboration:

- FoodHandler

Class name: DataParser

Parent Class: --

Clean Architecture Level: Frameworks and Drivers

- Refers to two different csv files in the data folder
- Can write with files with one function and read files with another
- Returns what it reads.

Collaboration:

Class name: CommandInput

Parent Class: --

Clean Architecture Level: Frameworks and Drivers

- Takes commands from the user as text input and runs commands
- Has a Run() method that will run the program in a loop, taking input.
- Stores possible commands and parses command that are sent by the input method
- Has methods to parse commands and executes correct action

Collaboration:

- FoodController(Controller)
- RecipeController(Controller)
- DataParser

Class name: CommandGUI

Parent Class: --

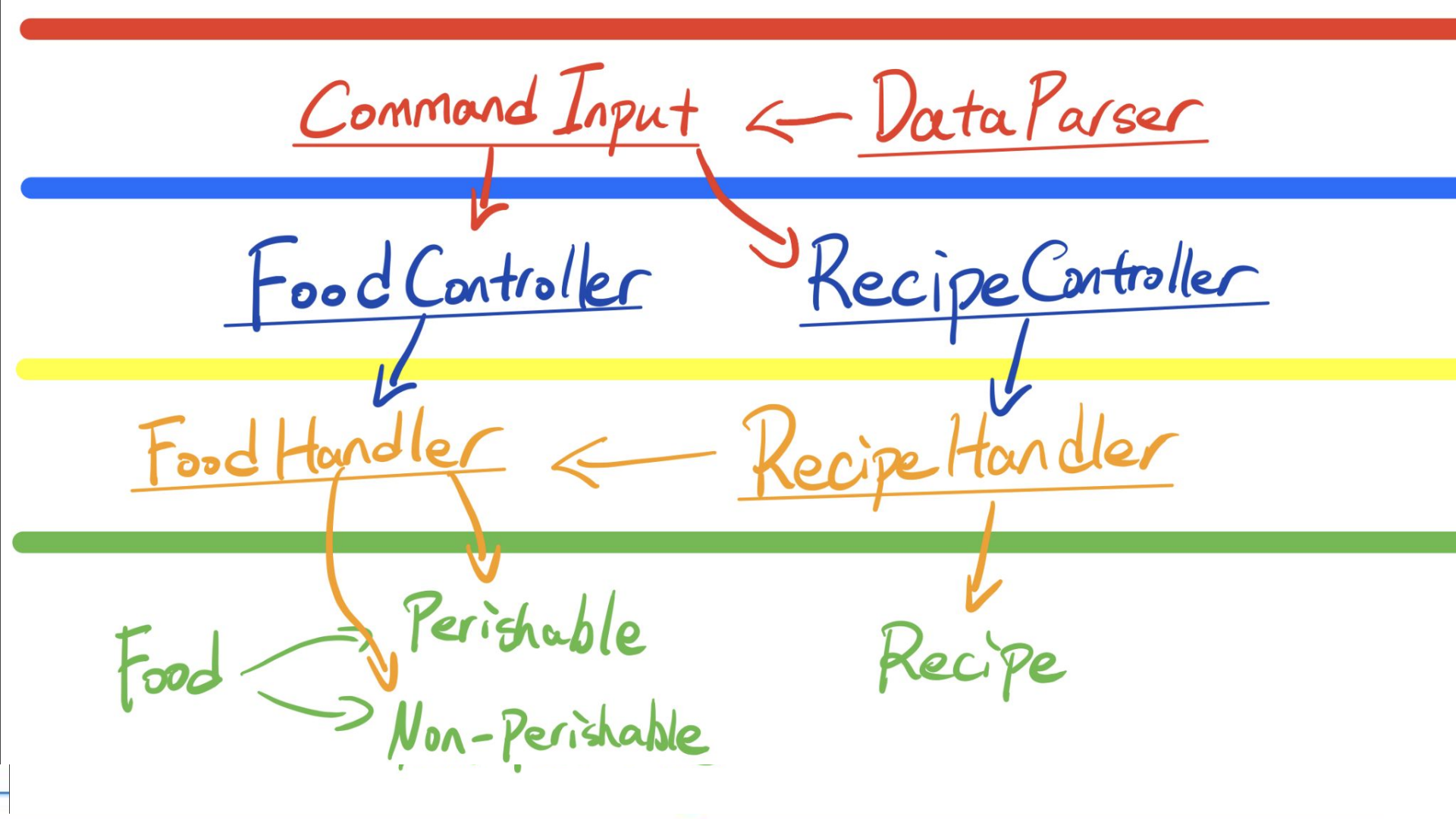
Clean Architecture Level: Frameworks and Drivers

- Is a graphical user interface that simplifies inputs into the program
- sends commands to the controllers separated from the commandinput
- Has buttons, text fields, etc.

NOT IN SCOPE ATM

Collaboration:

- CommandInput


Command Input
Data Parser
Food Controller
Recipe Controller
Food Handler
Recipe Handler
Food
Perishable
Non-Perishable
Recipe