STARTUP CONTRACTS

Contract CO1: register

Operation:	register(name, info)
Cross References:	Use Cases: 5.1 Startup
Preconditions:	none
Postconditions:	● A user instance has been instantiated ● User attributes have been initialized (name, opt. weight, gender)

CONTROLLER CONTRACTS (USED BY PANTRY, NUTRITION LOG, AND SHOPPING LIST)

Contract CO2: getItems

Operation:	getItems()
Cross References:	Use Cases: 5.1 Startup, 1.1 Manage Pantry, 1.2 Search Pantry, 1.3 Consume Pantry Item
Preconditions:	User has been registered within the system and initialized
Postconditions:	● System provides pantry data to user

Contract CO3: addItem

Operation:	addItem(foodId, quantity)
Cross References:	Use Cases: 5.1 Startup, 1.1 Manage Pantry
Preconditions:	User is viewing their pantry interface.
Postconditions:	● Added food exists within the food database ● Food exists within the user's pantry ● Pantry quantity of food item has been incremented to quantity + old quantity

Contract CO4: editItem

Operation:	editItem(foodId, quantity)
Cross References:	Use Cases: 1.1 Manage Pantry

Preconditions:	User is viewing their pantry interface. User has selected an existing food item.
Postconditions:	● Food item in pantry with given name is updated with new quantity ● Pantry view is updated.

Contract CO5: removeItem

Operation:	removeItem(foodId)
Cross References:	Use Cases: 5.1 Startup, 1.1 Manage Pantry
Preconditions:	User is viewing their pantry interface. User has selected an existing food item.
Postconditions:	● Food item with given name has its instance removed from the pantry. ● Pantry view is updated.

PANTRY CONTRACTS

Contract CO6: searchByFoodName

Operation:	searchByFoodName(query)
Cross References:	Use Cases: 1.2 Search Pantry
Preconditions:	User is viewing their pantry interface.
Postconditions:	● List of food items containing itemName are returned or an empty list if no pantry items match the itemName

Contract CO7: consume

Operation:	consume(foodId, quantity)
Cross References:	Use Cases: 1.3 Consume Pantry Item
Preconditions:	User is viewing their pantry interface. FoodItem passed in exists in the pantry.
Postconditions:	● Food in pantry with type foodItem has its quantity decremented ● If new quantity is 0, foodItem is removed from pantry

SHOPPING LIST CONTRACTS

Contract CO8: export

Operation:	export(format, destination)
Cross References:	Use Cases: 4.1 Manage Shopping List, 4.2 Export Shopping List
Preconditions:	● User has selected that they wish to export shopping list ● Shopping list instance exists
Postconditions:	● Shopping list has been exported to desired format

Contract CO9: purchaseItems

Operation:	purchaseItems(foodIds)
Cross References:	Use Cases: 4.3 Mark Purchased Items
Preconditions:	● User is viewing the shopping list interface ● Shopping list food types are registered in the list of food types ● User purchased all passed in items on the shopping list
Postconditions:	● System moves all items from shopping list to pantry

RECIPE CONTRACTS

Contract CO10: getRecipes

Operation:	getRecipes()
Cross References:	Use Cases: 3.1 Create Recipe, 3.2 View Recipe, 3.3 Add Recipe to Shopping List, 3.4 Modify Recipe, 3.5 Recommend Recipes
Preconditions:	The user has been initialized and registered within the system.
Postconditions:	● System provides the user's list of recipes ● System fetches and displays a few recommended recipes

Contract CO11: addRecipe

Operation:	addRecipe(name, ingredients, instructions, servings)
Cross References:	Use Cases: 5.1 Startup, 3.4 Modify Recipe
Preconditions:	The user has been initialized and registered within the system.
Postconditions:	● Ingredients have been stored has food instances within the database ● The recipe represents a new recipe instance within the recipe system ● Recipe appears within the recipe list

Contract CO12: produceCookedRecipe

Operation:	produceCookedRecipe(recipeId, usingPantry, consumedServings, leftoverServings)
Cross References:	Use Cases: 3.6 Cook Recipe
Preconditions:	● Recipe exists within the recipe system ● Ingredients exist within the food type database ● Method specifies how to cook the recipe (use all pantry items, use no pantry items, use only on hand).
Postconditions:	● User is notified of any errors preventing cooking ● If errors have been resolved, user can specify serving amounts and leftover amounts ● Recipe is logged to food log ● Pantry is updated to contain leftovers

Contract CO13: editRecipe

Operation:	editRecipe(recipeID)
Cross References:	Use Cases: 3.4 Modify Recipe
Preconditions:	● Method specifies what fields of recipe are to be modified
Postconditions:	● User is notified of any errors preventing modification ● If errors have been resolved, user can save the modified recipe ● Recipe is saved to Recipe List ● Food Database is updated with any new Food Types

Contract CO14: getRecipe

Operation:	getRecipe(id)

Cross References:	Use Cases: 3.5 Recommend Recipes
Preconditions:	● Method specifies what recipe is to be displayed ● Recipe exists within the recipe system
Postconditions:	● Recipe is displayed to the user

Contract CO15: getRecommendedRecipes

Operation:	getRecommendedRecipes()
Cross References:	Use Cases: 3.5 Recommend Recipe
Preconditions:	● Digital Pantry is non-empty ● Food Database is non-empty ● Recipe List is non-empty
Postconditions:	● The user is presented with all recommended recipes from the Recipe List based off of items in their pantry

Contract CO16: searchByRecipeName

Operation:	searchByRecipeName(query)
Cross References:	Use Cases: 3.2 View Recipes
Preconditions:	● User is registered within the system ● User has created at least one recipe
Postconditions:	● List of matching recipes are displayed to the user

Contract CO17: addIngredientsToCart

Operation:	addIngredientsToCart(recipeId)
Cross References:	Use Cases: 3.3 Add Recipe to Shopping Cart
Preconditions:	● User is registered within the system ● User has created at least one recipe
Postconditions:	● Recipe ingredients added to shopping cart

Contract CO18: addMissingIngredientsToCart

Operation:	addMissingIngredientsToCart(recipeId)

Cross References:	Use Cases: 3.3 Add Recipe to Shopping Cart
Preconditions:	● User is registered within the system ● User has created at least one recipe
Postconditions:	● Missing recipe ingredients added to shopping cart

NUTRITION LOG CONTRACTS

Contract CO19: addItem

Operation:	logFood(foodId, quantity, consumedAt)
Cross References:	Use Cases: 3.6 Cook Recipe, 2.1 Manage Nutrition Log
Preconditions:	● Recipe exists within the recipe system ● Ingredients exist within the food type database with nutrition info ● Recipe cook has been completed
Postconditions:	● New food instance is created from the cooked recipe ● Nutrition log is updated with new food instance

Contract CO20: selectNutritionTimethis needs to be refactored

Operation:	selectNutritionTime(time, type) return a list of NutritionItems in the time window
Cross References:	Use Cases: 2.1 Manage Nutrition Log
Preconditions:	● The user is in the nutrition log ● At least one item exists within the log
Postconditions:	● The selected item has been displayed within the timeline

Contract CO21: addGoal

Operation:	addGoal(field, value, isMax, hasProgressAlerts, timeAlerts)
Cross References:	Use Cases: 2.2 Nutrition Goals
Preconditions:	● The user is in the nutrition goals menu ● The input values are valid and logical
Postconditions:	● A nutrition goal has been created using the given fields

Contract CO22: editGoal

Operation:	editGoal(goal)
Cross References:	Use Cases: 2.2 Nutrition Goals
Preconditions:	● The user is in the nutrition goals menu ● At least one goal exists within the menu which the user wants to edit
Postconditions:	● The selected goal has had its values changed as per user request

Contract CO23: removeGoal

Operation:	removeGoal(id)
Cross References:	Use Cases: 2.2 Nutrition Goals
Preconditions:	● The user is in the nutrition goal menu ● At least one goal exists within the menu which the user wants to delete
Postconditions:	● The selected item has been removed from the log

Contract CO24: createReport

Operation:	createReport(chartType, fields)
Cross References:	Use Cases: 2.3 Manage Nutrition Report
Preconditions:	● The user is in the nutrition reports menu ● The input values are valid and logical
Postconditions:	● A nutrition report has been created using the given fields

Contract CO25: editReport

Operation:	editReport(report)
Cross References:	Use Cases: 2.3 Manage Nutrition Report
Preconditions:	● The user is in the nutrition reports menu ● At least one report exists within the menu which the user wants to edit
Postconditions:	● The selected report has had its values changed as per user request

Contract CO26: deleteReport

Operation:	deleteReport(report)
Cross References:	Use Cases: 2.3 Manage Nutrition Report
Preconditions:	● The user is in the nutrition report menu ● At least one report exists within the menu which the user wants to delete
Postconditions:	● The selected report has been removed from the report menu

FOODTYPE CONTRACTS

Contract CO27: getFood

Operation:	getFood(id)
Cross References:	Use Cases: 6.1 Manage Food Types, 6.2 Create New Food Types, 2.1 Manage Nutrition Log
Preconditions:	● User has been registered within the system and initialized ● Given id is a valid id in foods map
Postconditions:	● System provides user with the food type associated with id

Contract CO28: addFood

Operation:	addFood(name, category, nutrition)
Cross References:	Use Cases: 6.1 Manage Food Types, 6.2 Create New Food Types, 4.1 Manage Shopping List, 1.1 Manage Pantry, 1.2 Search Pantry, 2.1 Manage Nutrition Log
Preconditions:	● User is viewing the food types interface ● System is in “Modify Food Types” mode ● The name is not already in use by another food type ● The name is not empty ● The details are valid (non-negative weight, etc)
Postconditions:	● System adds food type to list of food types

Contract CO29: editFood

Operation:	editFood(id, category, nutrition)
Cross References:	Use Cases: 6.1 Manage Food Types
Preconditions:	● User is viewing the food ty ● pes interface ● System is in “Modify Food Types” mode ● The new data is valid ● The id refers to an already registered food type
Postconditions:	● System updates food type in list of food types to have a new name and/or details

Contract CO30: removeFood

Operation:	removeFood(id)
Cross References:	Use Cases: 6.1 Manage Food Types
Preconditions:	● User is viewing the food types interface ● System is in “Modify Food Types” mode ● The id refers to an already registered food type
Postconditions:	● System removes given food type from the list of food types