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)

PANTRY CONTRACTS

Contract CO2: viewPantry

Operation:	viewPantry()
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: addFoodToPantry

Operation:	addFoodToPantry(food, 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: editFoodInPantry

Operation:	editFoodInPantry(name, 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: removeFoodInPantry

Operation:	removeFoodInPantry(name)
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.

Contract CO6: searchPantry

Operation:	searchPantry(itemName)
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(foodItem)
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: viewShoppingList

Operation:	viewShoppingList()
Cross References:	Use Cases: 4.1 Manage Shopping List, 4.3 Mark Purchased Items
Preconditions:	User has been registered within the system and initialized
Postconditions:	● System provides user with their shopping list

Contract CO9: addShoppingItem

Operation:	addShoppingItem(foodType, quantity)
Cross References:	Use Cases: 4.1 Manage Shopping List
Preconditions:	● User is viewing the shopping list interface ● System is in "Modify Shopping List" mode ● The food type exists within the list of registered food types ● The quantity is valid (non-negative, etc)
Postconditions:	● Food type and quantity added to the shopping list

Contract CO10: editShoppingItem

Operation:	editShoppingItem(foodType, quantity)
Cross References:	Use Cases: 4.1 Manage Shopping List
Preconditions:	● User is viewing the shopping list interface ● System is in "Modify Shopping List" mode ● The food type exists within the shopping list ● The quantity is valid (non-negative, etc)
Postconditions:	● Quantity of given food type updated within the shopping list

Contract CO11: deleteShoppingItem

Operation:	deleteShoppingItem(foodType)
Cross References:	Use Cases: 4.1 Manage Shopping List
Preconditions:	● User is viewing the shopping list interface ● System is in "Modify Shopping List" mode ● The food type exists within the shopping list ● The quantity is valid (non-negative, etc)
Postconditions:	● Food type removed from shopping list

Contract CO12: exportShoppingList

Operation:	exportShoppingList(format)
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 CO13: printList

Operation:	printList()
Cross References:	Use Cases: 4.2 Export Shopping List
Preconditions:	● User has selected that they wish to export shopping list ● Shopping list instance exists ● Format is selected as print
Postconditions:	● Shopping list has been opened in print preview ● User can confirm printing at their printer

Contract CO14: purchaseAllItems

Operation:	purchaseAllItems()
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 items on the shopping list

Postconditions:	● System moves all items from shopping list to pantry

Contract CO15: purchaseItem

Operation:	purchaseItem(foodType)
Cross References:	Use Cases: 4.3 Mark Purchased Items
Preconditions:	● User is viewing the shopping list interface ● Given food type is registered in the shopping list ● Given food type is registered in the list of food types ● User purchased the given item
Postconditions:	● System moves item of given food type from shopping list to pantry

RECIPE CONTRACTS

Contract CO16: viewRecipes

Operation:	viewRecipes()
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 CO17: addRecipe

Operation:	addRecipe(name, ingredients, 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 CO18: cookRecipe

Operation:	cookRecipe(recipe, method)
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 CO19: 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 CO20: selectRecipe

Operation:	selectRecipe(recipeID)
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 CO21: recommendRecipes

Operation:	recommendRecipes()
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 CO22: searchRecipe

Operation:	searchRecipe(text)
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 CO23: recipeToCart

Operation:	recipeToCart(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

NUTRITION LOG CONTRACTS

Contract CO24: logFood

Operation:	logFood(food, quantity)
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 CO25: selectNutritionTime

Operation:	selectNutritionTime(time, type)
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 CO26: editNutritionItem

Operation:	editNutritionItem(item)
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 which the user wants to edit
Postconditions:	● The selected item has had its values changed as per user request

Contract CO27: deleteNutritionItem

Operation:	deleteNutritionItem()
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 which the user wants to delete
Postconditions:	● The selected item has been removed from the log

Contract CO28: createGoal

Operation:	createGoal(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 CO29: 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 CO30: deleteGoal

Operation:	deleteGoal(goal)
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 CO31: 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 CO32: 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 CO33: 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 CO34: viewFoodTypes

Operation:	viewFoodTypes()
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
Postconditions:	● System provides user with the list of food types

Contract CO35: addFoodType

Operation:	addFoodType(name, details), 6.2 Create New Food Types
Cross References:	Use Cases: 6.1 Manage 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 CO36: editFoodType

Operation:	editFoodType(id, newData)
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 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

CO37: deleteFoodType

Operation:	deleteFoodType(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