Diet

Write an application to manage a diet by means of nutritional values computation.
The application must allow the definition of raw materials and their use as ingredients for recipes. All the classes must be in the package "`Diet`".

R1 - Raw Material

The system works though the facade class `Food`.
To define a raw material, we can use the function `defineRawMaterial()` that accepts as arguments the name, the kilo-calories, the amount of proteins, carbohydrates (carbs), and fat; all the values refer to 100 grams of raw material. The name of the raw material can be considered unique.
To retrieve some information about the raw materials we can use the function `rawMaterials()` which returns a list of raw materials, sorted by name in alphabetic order. To get info about a specific material, we can use the function `getRawMaterial()` that accepts the name of the raw material and returns the corresponding raw material.
The raw materials returned by the above methods are objects implementing the protocol `NutritionalElement`, which provides the functions `getName()`, `getCalories()`, `getProteins()`, `getCarbs()`, `getFat()`. Calories are expressed in KCal, while proteins, carbs, and fat are expressed in grams.
Moreover the protocol includes the function `per100g()` that indicates whether the values refer to 100 grams of nutritional element or represent an absolute value. For raw materials the nutritional values are always expressed per 100 grams, therefore the method returns `true`.

R2 - Products

The diet may include also pre-packaged products (e.g. an ice cream or a snack). Products are defined by means of the method `defineProduct()` of class Food accepting as arguments the name, the kilo-calories, the amount of proteins, carbohydrates (carbs), and fat. Such values express the value for the whole product, therefore the method `per100g()` returns `false`. The name of the product can be considered unique.
To retrieve information about the products we can use the function `products()` of class Food that returns a collection of products sorted by name. To get information about a specific product, function `getProduct()` is available that accepts the name of the product and returns the corresponding object, which implements the protocol *`NutritionalElement`*.

R3 - Recipes

Raw materials can be combined as ingredients of recipes. Toe define a recipe we ought to create a new **`Recipe`** object that will be added with the ingredients. The **`init()`** function of class *`Recipe`* accepts as arguments the name of the recipe and the Food object that contains the ingredients. The name of the recipe can be considered unique.
A new ingredient can be added to a recipe by means of function **`addIngredient()`** that accepts as arguments the name of the raw material and the amount in grams.
Class *`Recipe`* implements the protocol *`NutritionalElement`* and the values are expressed per 100 grams.
When the recipe is created, it is automatically added to the the content of the corresponding *`Food`* object. An to retrieve the information about the recipes we can use the function **`recipes()`**, of class Food, that returns a collection of recipes sorted by name.
To get information regarding a specific recipe we can use the function **`getRecipe()`** that accepts as argument the name of the recipe and return the corresponding recipe.

Warning: While the sum of the amounts of ingredients (in grams) of a recipe is not necessarily equal to 100g, the nutritional values of the recipe must refer to an hypothetical portion of 100 grams.

R4 - Menu

A menu consists of portions of either recipes or pre-packaged products.
Menus are represented by class **`Menu`** whose **`init()`** accepts as arguments the name and the reference to the Food object containing the information about products and recipes.
To add a portion of a recipe to the menu we can use the function **`addRecipe()`** that accepts as argument the name of the recipe and the size of the portion, in grams.
To add an item of a pre-packaged product, we can use the function **`addProduct()`** that accepts as argument the name of the product.
Class *`Menu`* implements the *`NutritionalElement`* protocol; in this case the values are referred to the whole menu and not to 100 grams.