3. Design

Refrigerator with U

22213485 / 박차오름 / 22213485@yu.ac.kr

[Revision history]

Revision date	Version #	Description	Author
MM/DD/YYYY	0.00	Type brief description here	Author name
	0.1		

= Contents =

1. Introduction 4

2. Class diagram 5

3. Sequence diagram 9

4. State machine diagram 22

5. Implementation requirements 23

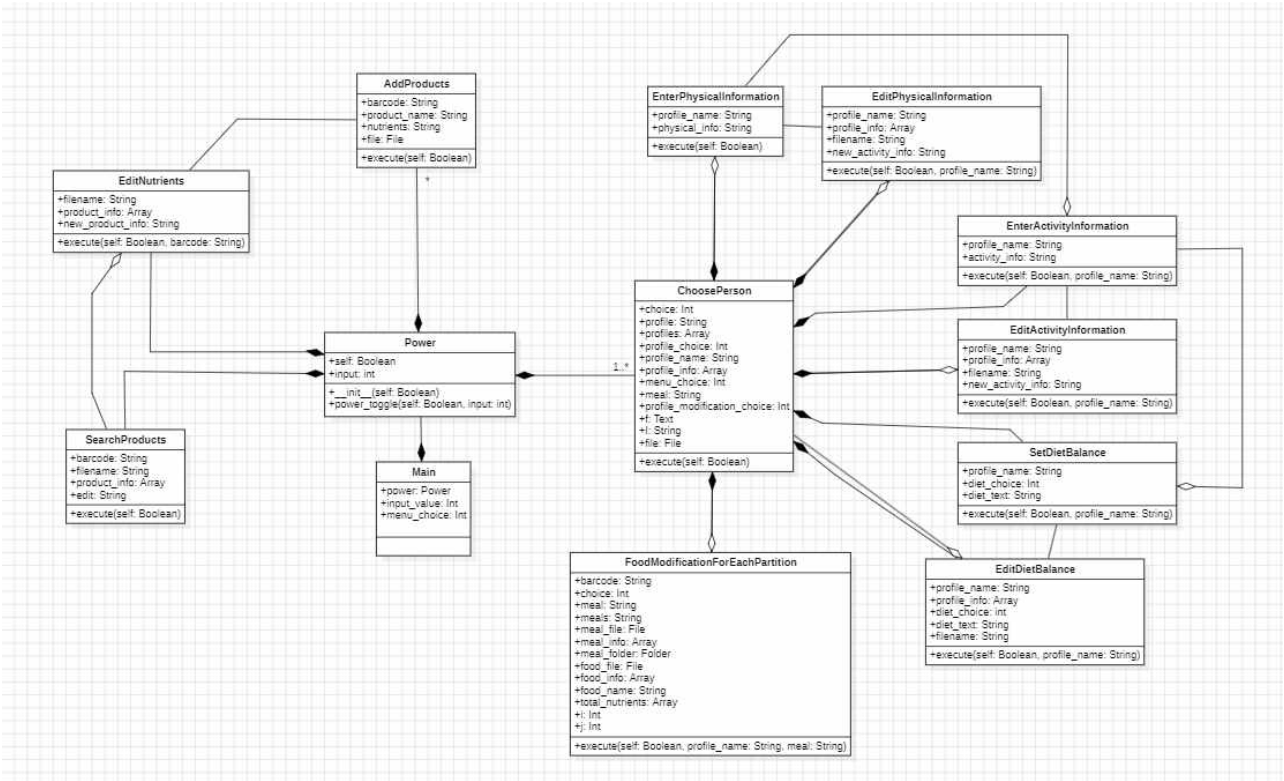
6. Glossary 23

7. References 23

1. Introduction

지난번 analysis에서는 각 클래스 별로 무엇을 하는지에 대해 클래스들의 기능을 분석했다. 이번 Design에서는 analysis를 바탕으로 실제 Refrigerator with U의 세부 기능들을 어떻게 구현할 것인지, 추후 구현단계에서의 틀이 될 내용들이 작성되었다. 최종단계는 콘솔로 개발할 계획이다.

2. Class diagram



[사진 1] Class diagram

2.1. Main

Attributes
power: Power() : Power 함수 실행 변수 input_value: Int : 프로그램 실행 값 입력 변수 menu_choice: Int : 메뉴 선택 입력 변수
Methods

2.2. Power

Attributes
self: Boolean : 프로그램 on/off input: int : 1이 입력되면 프로그램을 실행하고, 0이 입력되면 프로그램을 종료함.
Methods
__init__(self: Boolean) : 기본 상태는 작동하지 않는 것으로 설정함. power_toggle(self: Boolean, input int) : 프로그램 동작을 위한 함수.

2.3. AddProducts

Attributes
barcode: String : 추가하고자 하는 제품의 바코드

product_name: String : 추가하고자 하는 제품의 제품명 nutrients: String : 추가하고자 하는 제품의 영양성분 file: File : 파일
Methods
execute(self: Boolean) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함.

2.4. EditNutrients

Attributes
filename: String : 바코드.txt 파일의 존재 확인 product_info: Array : file의 내용을 배열로 split new_product_info: String : 기존 제품의 수정된 영양성분
Methods
execute(self: Boolean, barcode: String) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함과 입력받은 바코드를 통해 제품 파일을 찾기 위함.

2.5. SearchProducts

Attributes
barcode: String : 찾고자 하는 제품의 바코드. filename: String : 바코드.txt 파일의 존재 확인 product_info: Array : file의 내용을 배열로 split edit: String : 제품의 수정여부
Methods
execute(self: Boolean) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함.

2.6. ChoosePerson

Attributes
choice: Int : 프로필 메뉴 입력 profile: String : profile 이름을 가진 폴더 profiles: Array : profile 배열 profile_choice: Int : 프로필 선택 profile_name: String : 프로필 profile_info: Array : 프로필 정보들을 변수에 split 한 상태로 입력받음 menu_choice: Int : 메뉴 션택 meal: String : 식사 시간을 저장하는 변수 profile_modification_choice: Int : 프로필에서 수정할 메뉴 선택 f: Text : function I: String : profile 순서 입력 file: File : 파일

Methods
execute(self: Boolean) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함.

2.7. EnterPhysicalInformation

Attributes
profile_name: String : 프로필 이름 입력 physical_info: String : 신체정보 입력
Methods
execute(self: Boolean) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함.

2.8. EnterActivityInformation

Attributes
profile_name: String : 생성하고자 하는 프로필에 저장하기 위함 activity_info: String : 활동 정보를 입력받기 위함
Methods
execute(self: Boolean, profile_name: String) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함과 생성하는 프로필에 활동 정보를 입력하기 위함

2.9. SetDietBalance

Attributes
profile_name: String : 생성하고자 하는 프로필에 저장하기 위함 diet_choice: Int : 식단 균형을 선택하기 위함 diet_text: String : 선택한 식단 균형을 저장하기 위함
Methods
execute(self: Boolean, profile_name: String) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함과 생성하는 프로필에 식단 균형 정보를 입력하기 위함

2.10. EditPhysicalInformation

Attributes
profile_name: String : 변경하고자 하는 프로필.txt 파일을 선택하기 위해 입력받음 profile_info: Array : 프로필의 정보들을 변수에 split 한 상태로 입력받음 filename: String : profile_name.txt 파일을 의미함 new_physical_info: String : 수정하고자 하는 신체정보를 입력받음
Methods
execute(self: Boolean, profile_name: String) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함과 선택한 프로필의 신체정보를 수정하기 위함

2.11. EditActivityInformation

Attributes
profile_name: String : 변경하고자 하는 프로필.txt 파일을 선택하기 위해 입력받음

profile_info: Array : 프로필의 정보들을 변수에 split 한 상태로 입력받음 filename: String : profile_name.txt 파일을 의미함 new_activity_info: String : 수정하고자 하는 신체정보를 입력받음
Methods
execute(self: Boolean, profile_name: String) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함과 선택한 프로필의 활동 정보를 수정하기 위함

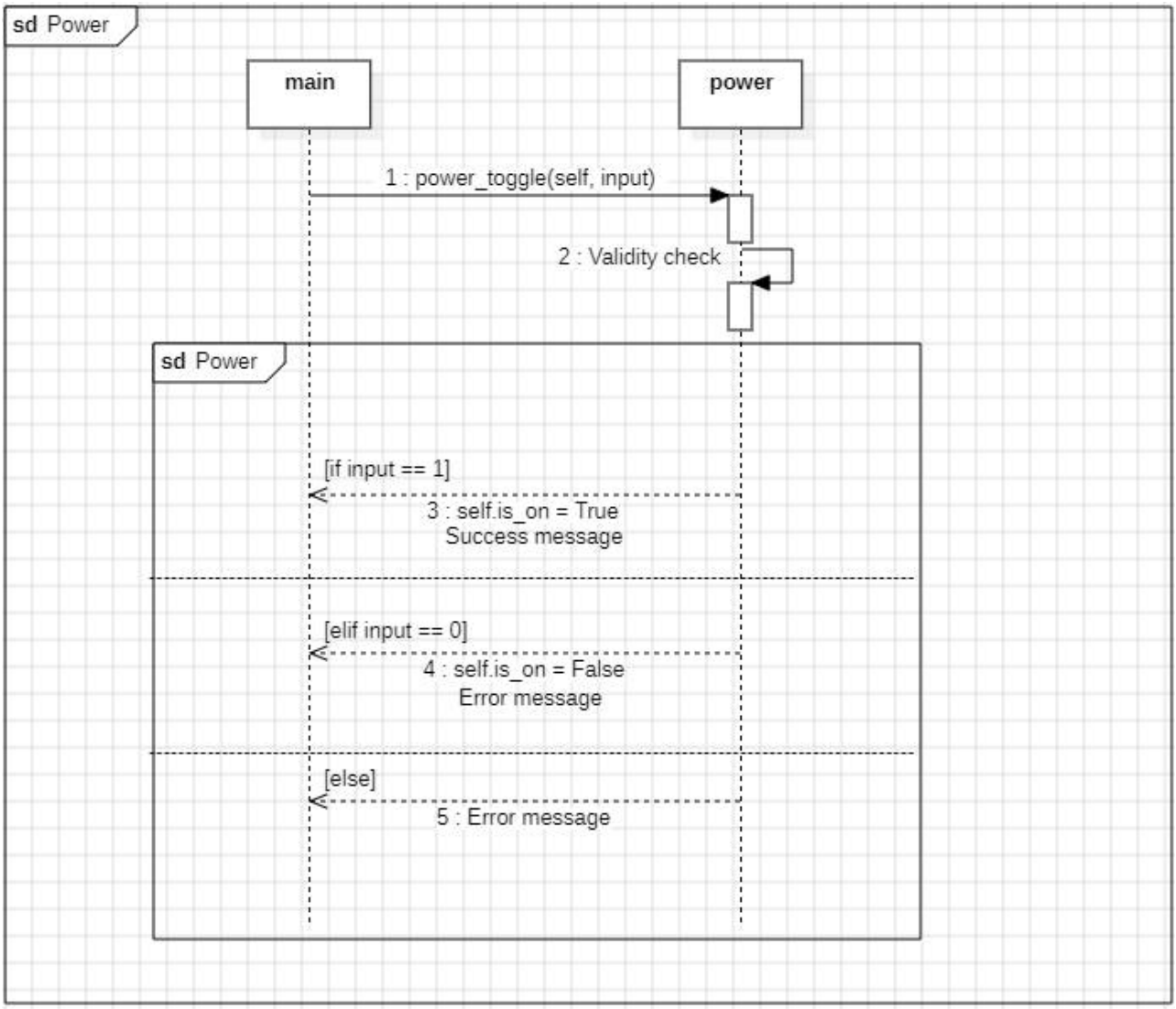
2.12. EditDietBalance

Attributes
profile_name: String : 변경하고자 하는 프로필.txt 파일을 선택하기 위해 입력받음 profile_info: Array : 프로필의 정보들을 변수에 split 한 상태로 입력받음 diet_choice: Int : 식단 균형 선택 diet_text: String : 선택한 식단 균형을 저장하기 위함 filename: String : profile_name.txt 파일을 의미함
Methods
execute(self: Boolean, profile_name: String) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함과 선택한 프로필의 식단 균형을 수정하기 위함

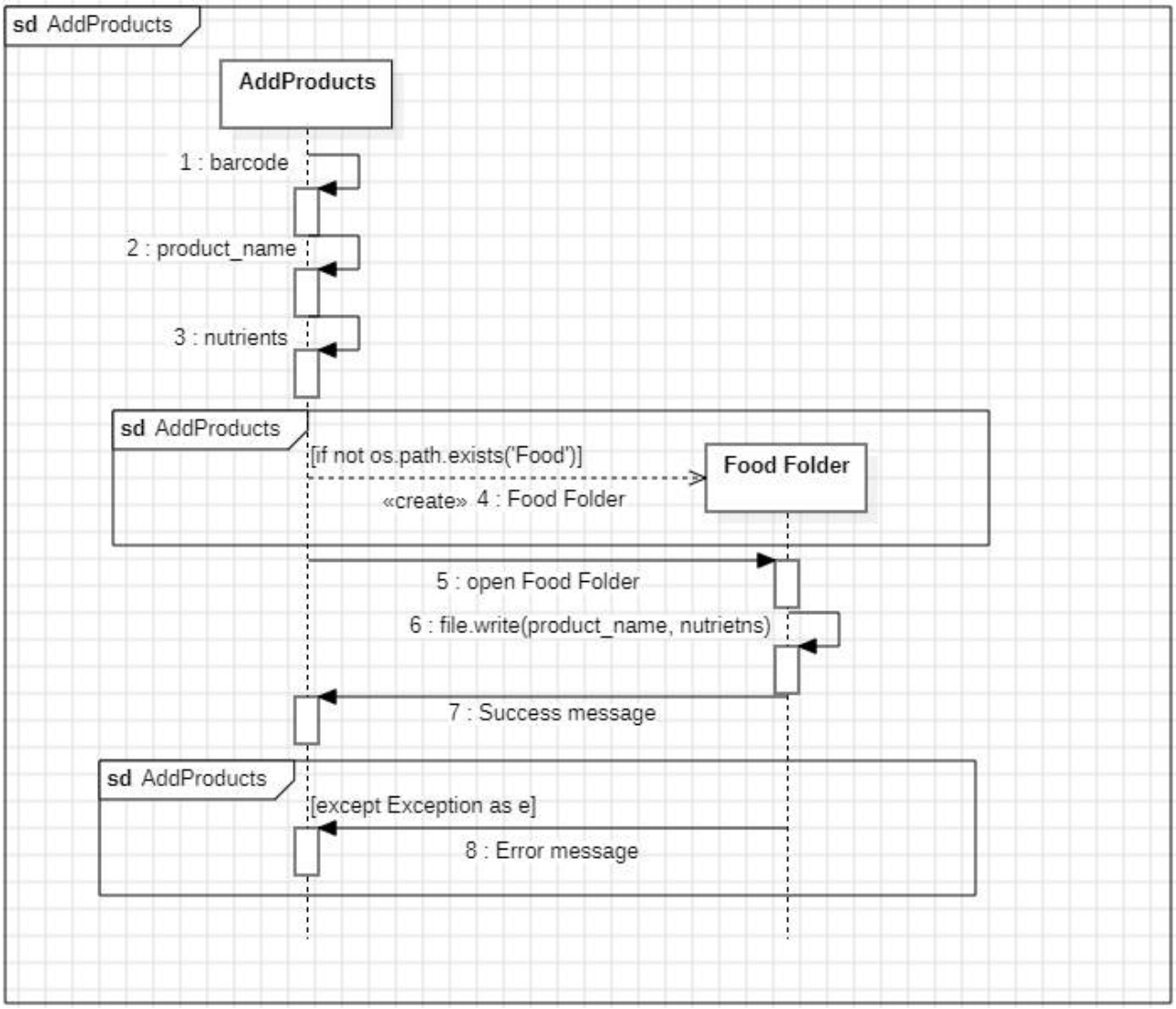
2.13. FoodModificationForEachPartition

Attributes
barcode: String : 먹은 음식을 추가하기 위함 choice: Int : 식사에 관련된 기능을 선택 meal: String : 아침, 점심, 저녁과 같은 식사 시간 meals: String : meal_file에서 한 줄씩 불러옴 meal_file: File : 파일 meal_info: Array : meals 정보들을 변수에 split 한 상태로 입력받음 meal_folder: Folder : 폴더 food_file: File : 파일 food_info: Array : food_file에서 불러온 정보를 변수에 split 한 상태로 입력받음 food_name: String : 삭제할 식사의 제품명을 입력받기 위함 total_nutrients: Array : 섭취한 총 영양소 I: Int : 배열 순서 입력 j: Int : 배열 순서 입력
Methods
execute(self: Boolean, profile_name: String, meal: String) : 프로그램이 True(on) 상태일 때만 동작하게 하기 위함과 선택한 프로필에서의 선택한 식사 시간에서 메뉴 선택을 하기 위함.

3. Sequence diagram

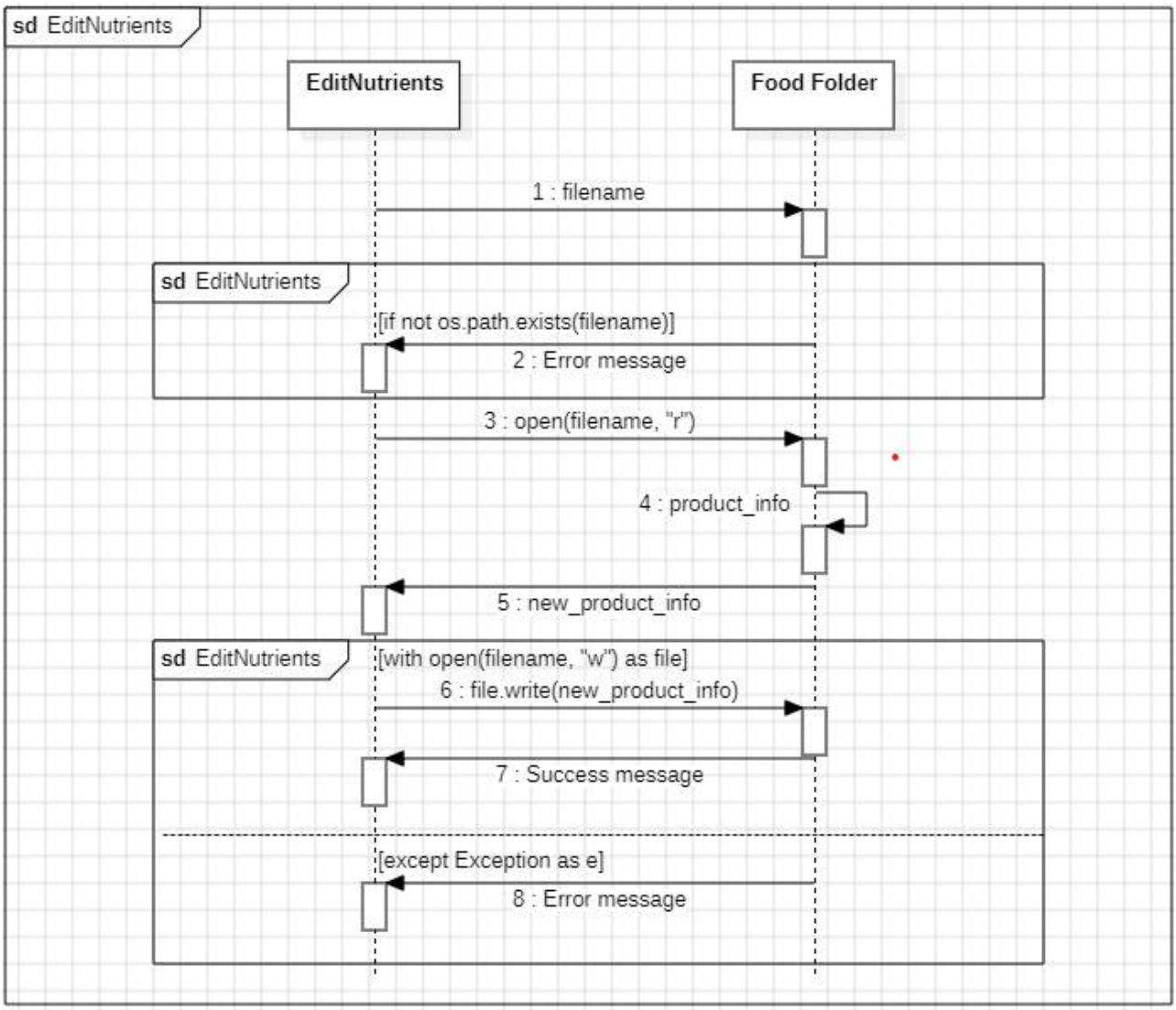


위의 다이어그램은 Power diagram이다.
프로그램을 실행하면 power_toggle이 실행된다, input을 입력받고, 1을 입력하면 성공 메시지의 반환과 함께 프로그램이 실행되고, 0을 입력하면 에러 메시지의 반환과 함께 프로그램이 종료된다. 1과 0이 아닌 다른 숫자나 문자가 입력되면 에러 메시지가 출력된다.

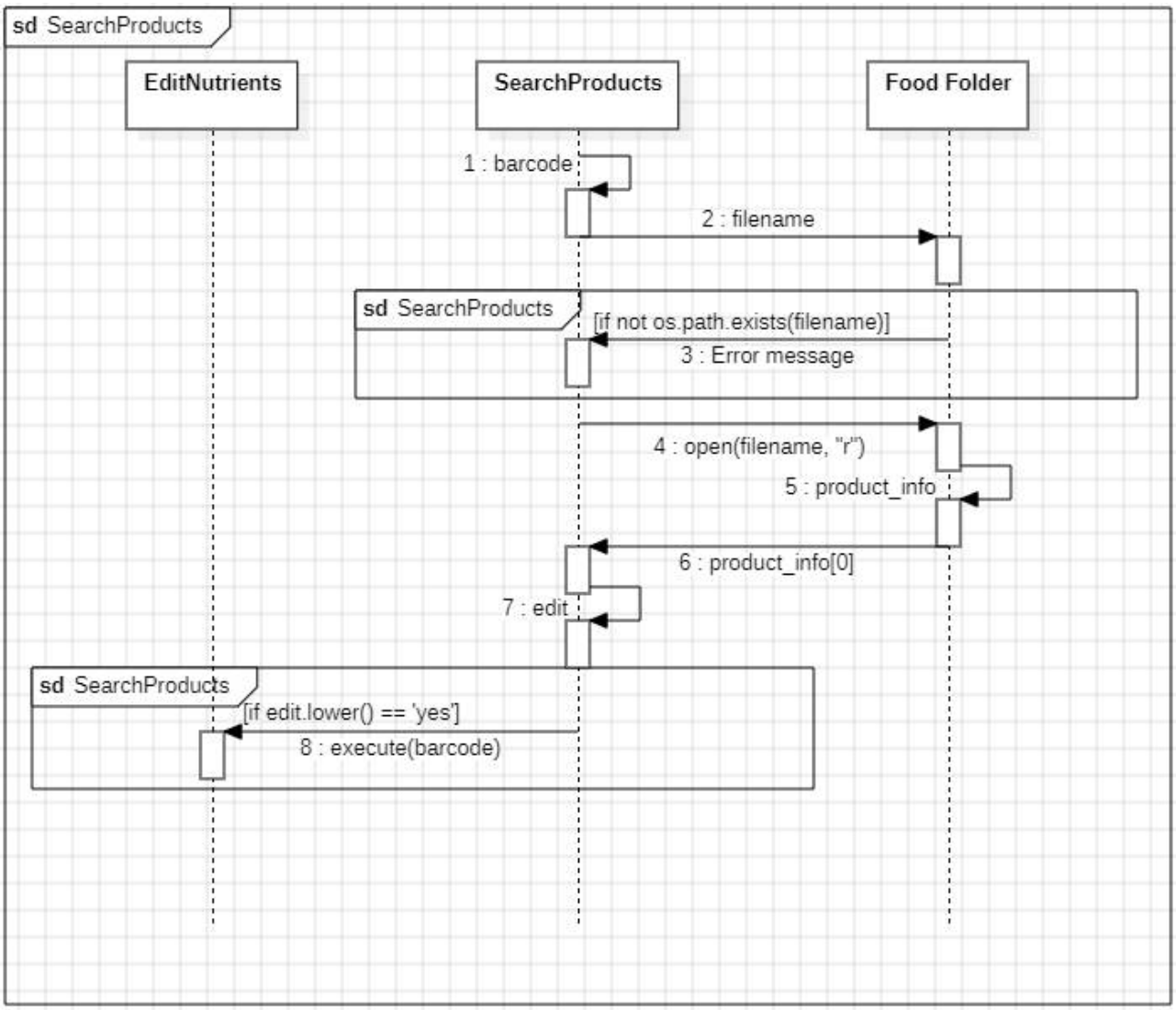


위의 다이어그램은 AddProducts diagram이다.

제품 추가를 원할 때 AddProducts class에서 barcode, product_namem, nutrients변수에 정보를 순서대로 입력하고 Food 폴더에 입력된 값들을 txt파일로 작성한다. Food 폴더가 존재하지 않으면 Food 폴더를 생성하게 한다. txt파일로 작성이 완료되었으면 성공 메시지를 반환하고, 정상적으로 작성이 되지 않았으면 에러 메시지를 반환한다.

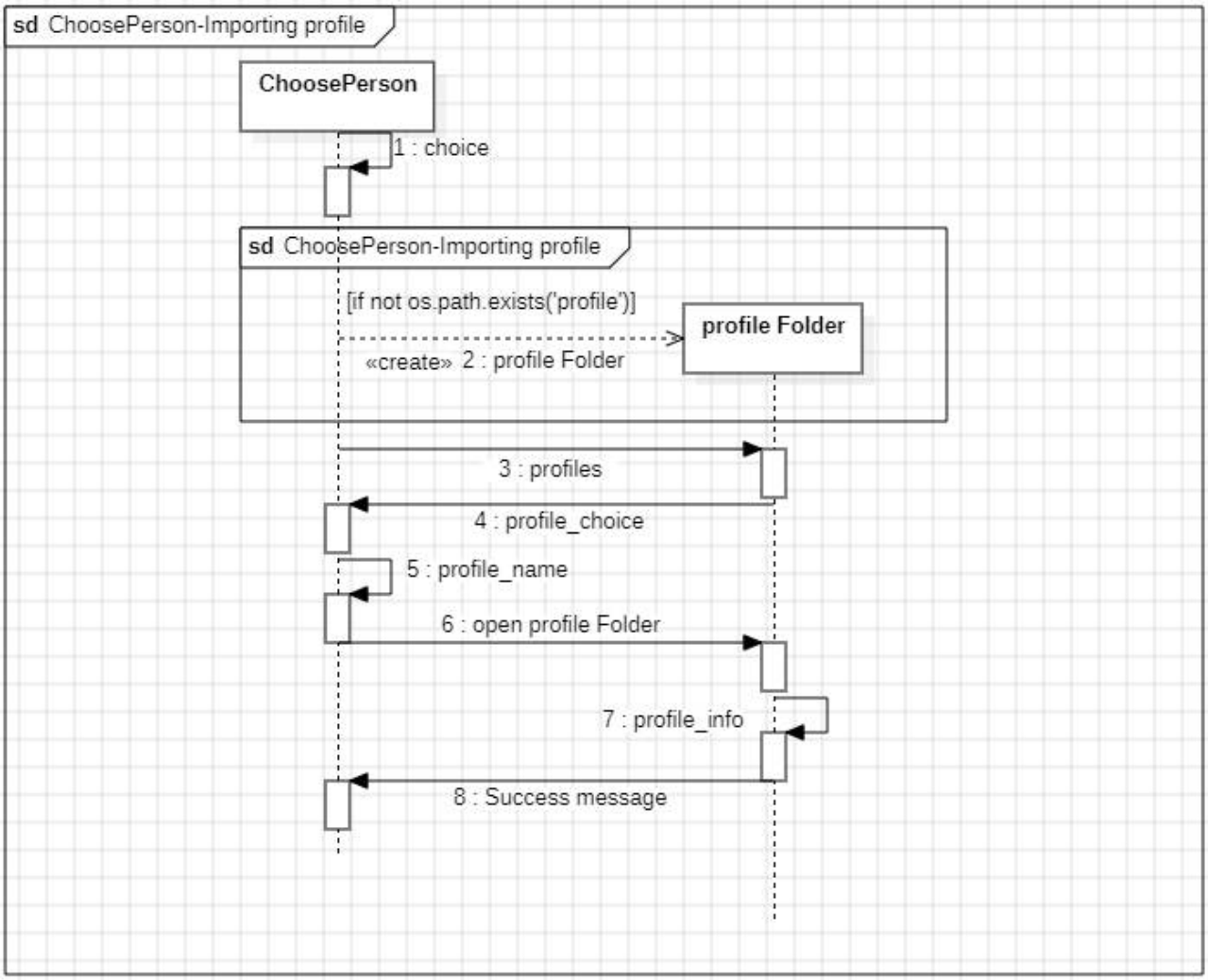


위의 다이어그램은 EditNutrients diagram이다.
영양 정보 수정을 원할 때 EditNutrients class에서 Food 폴더로 filename을 전송한다. 이때 일치하는 filename이 없으면 에러 메시지를 반환한다. 일치하는 filename이 있다면 파일을 open하고 split하여 product_info 변수에 저장한다. 수정하고자 하는 정보를 new_product_info 에 input 받는다. 다시 일치하는 filename의 파일을 열어 new_product_info 의 정보를 입력한다. 입력이 완료되면 성공 메시지를 반환하고, 실패하면 에러 메시지를 반환한다.

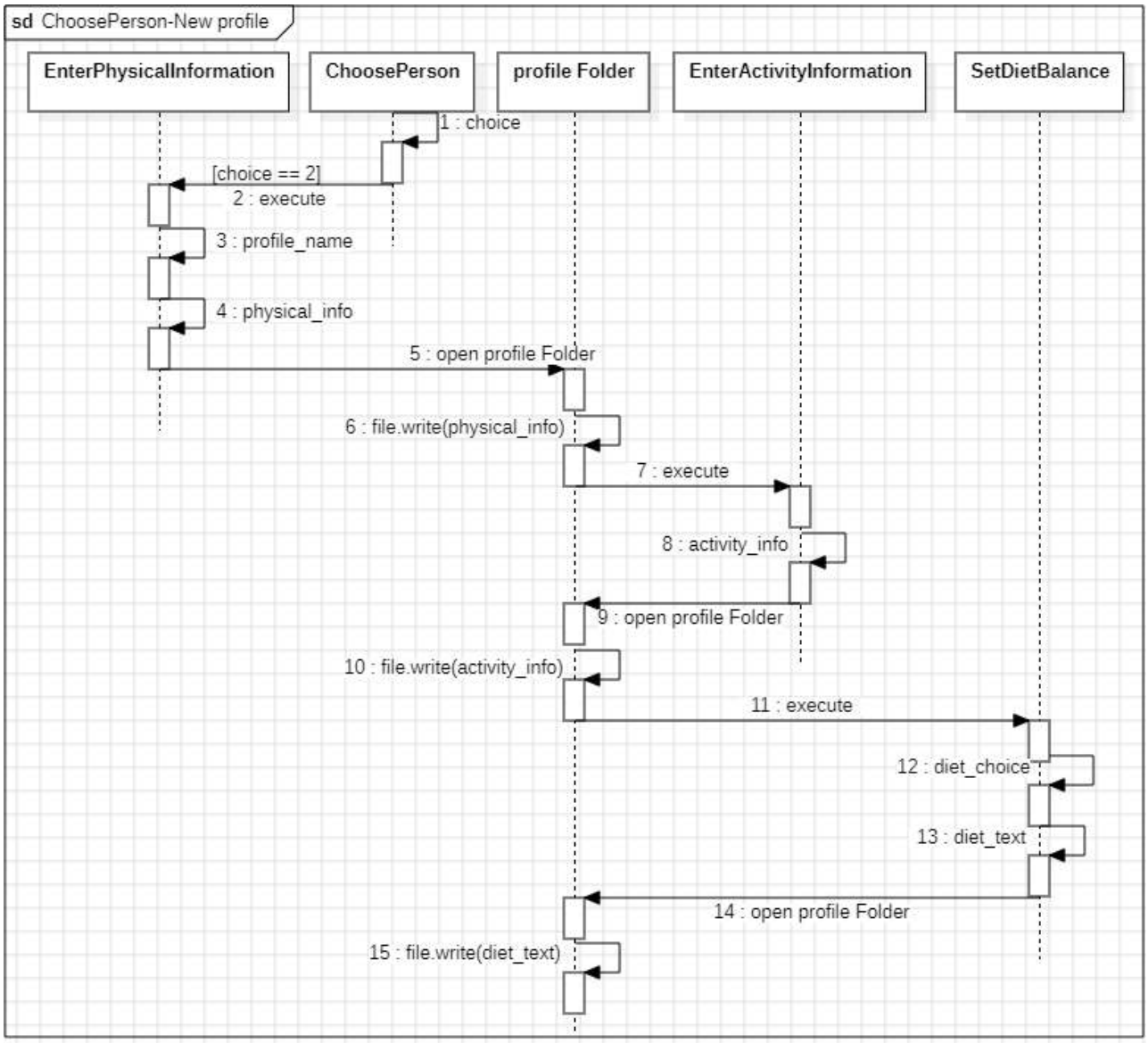


위의 다이어그램은 SearchProducts diagram이다.

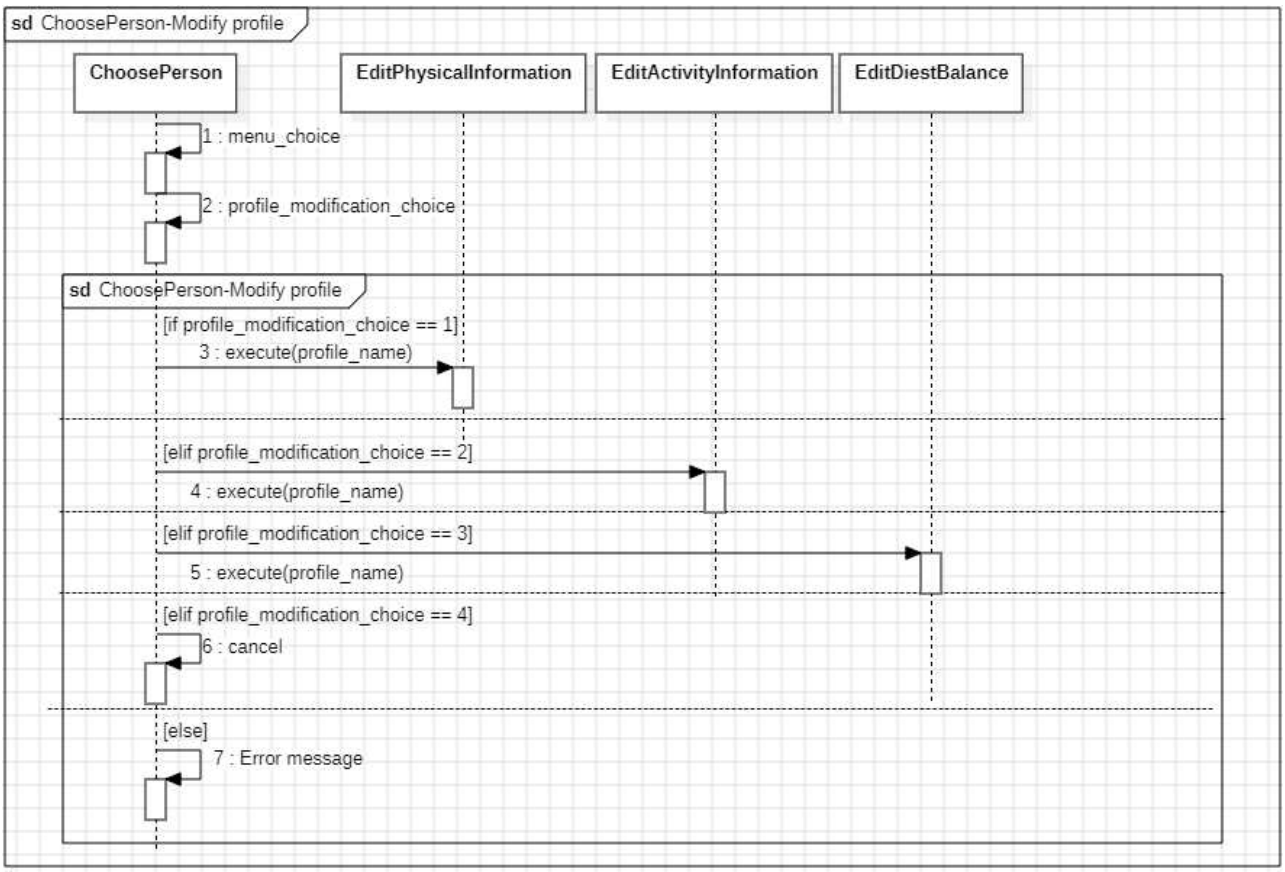
제품을 찾고자 할 때, SearchProducts class에서 barcode를 입력한다. barcode는 Food Folder에 해당하는 filename을 가진 파일을 열거나, 없다면 Error message를 반환한다. 해당하는 filename을 open 후 product_info 변수에 정보를 분리하여 저장한다. 제품명을 출력한 후, 영양 정보 수정 여부를 묻는다. 이때, yes를 입력하면 EditNutrients class로 barcode를 넘겨준다.



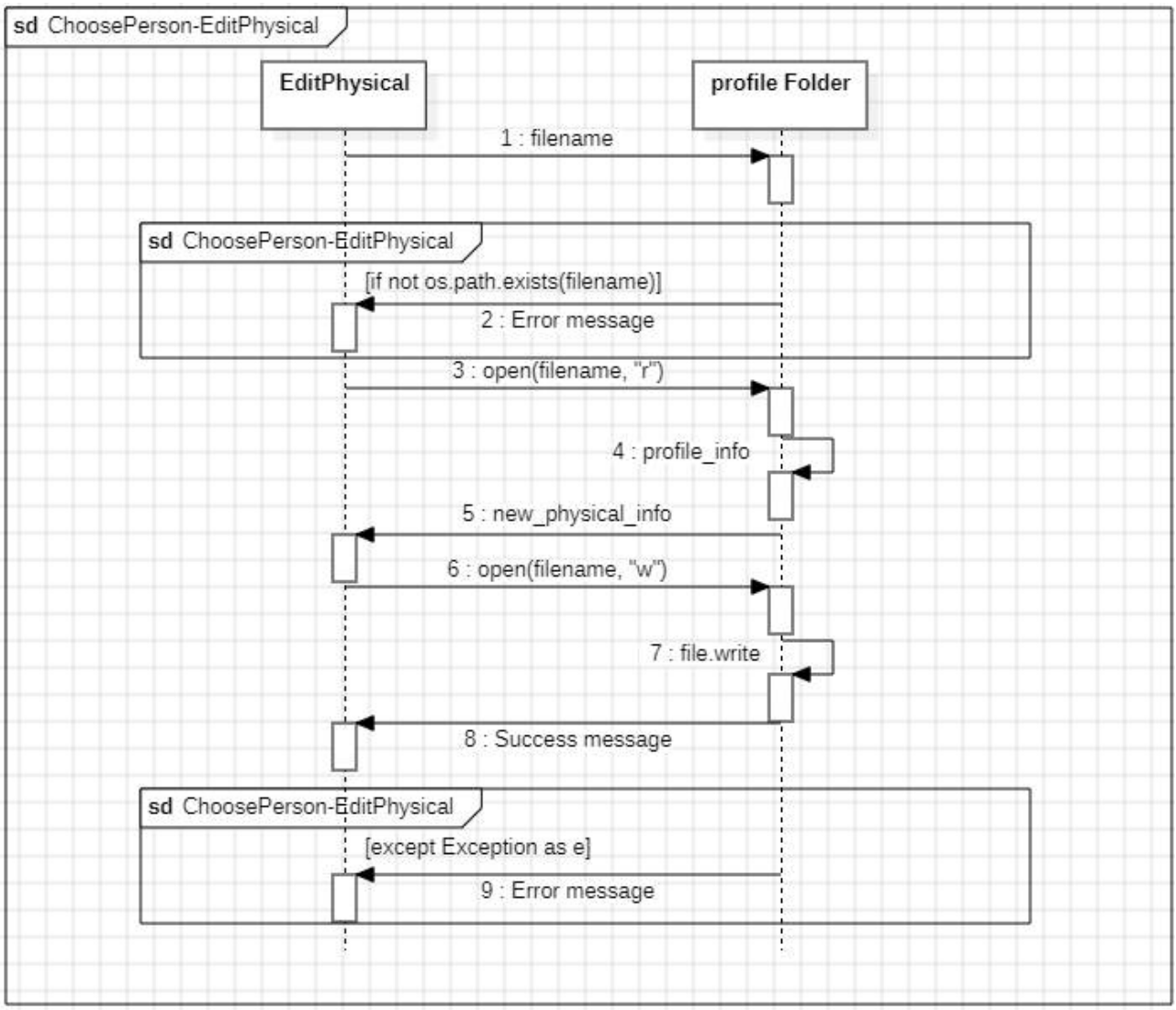
위의 다이어그램은 ChoosePerson diagram중 프로필을 불러오는 과정이다.
저장된 프로필을 불러오고자 할 때, ChoosePerson class에서 choice 변수에 1을 입력하면 profile Folder에서 profiles 변수를 통해 프로필들을 불러온다. 원하는 프로필을 profile_choice 변수에 입력하면, profile_name 변수에 저장이 된다. profile Folder를 open하고 profile_info 변수에 프로필 정보들을 분리하여 저장한다. 이후 Success message를 출력한다.



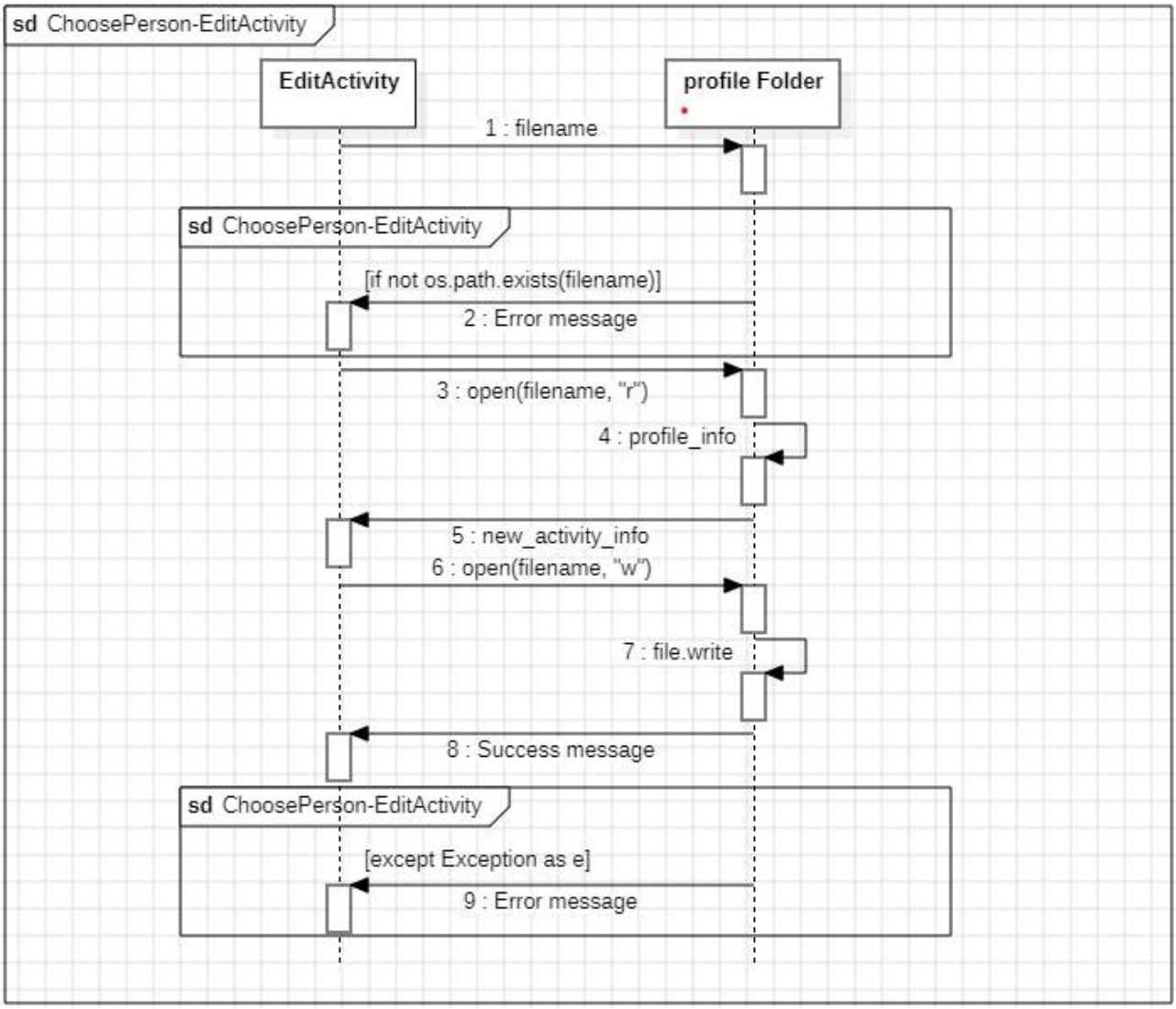
위의 다이어그램은 ChoosePerson diagram중 프로필을 생성하는 과정이다.
새 프로필을 생성하고자 할 때, ChoosePerson class에서 choice 변수에 2를 입력하고 프로필의 이름, 신체정보를 각 변수에 저장한다. profile Folder 폴더에 profilie_name 파일을 열어 physical_info를 입력한다. 마찬가지로 activity_info를 입력, 저장하고 diet_choice 변수에 식사 밸런스를 입력해서 profile_name에 저장한다.



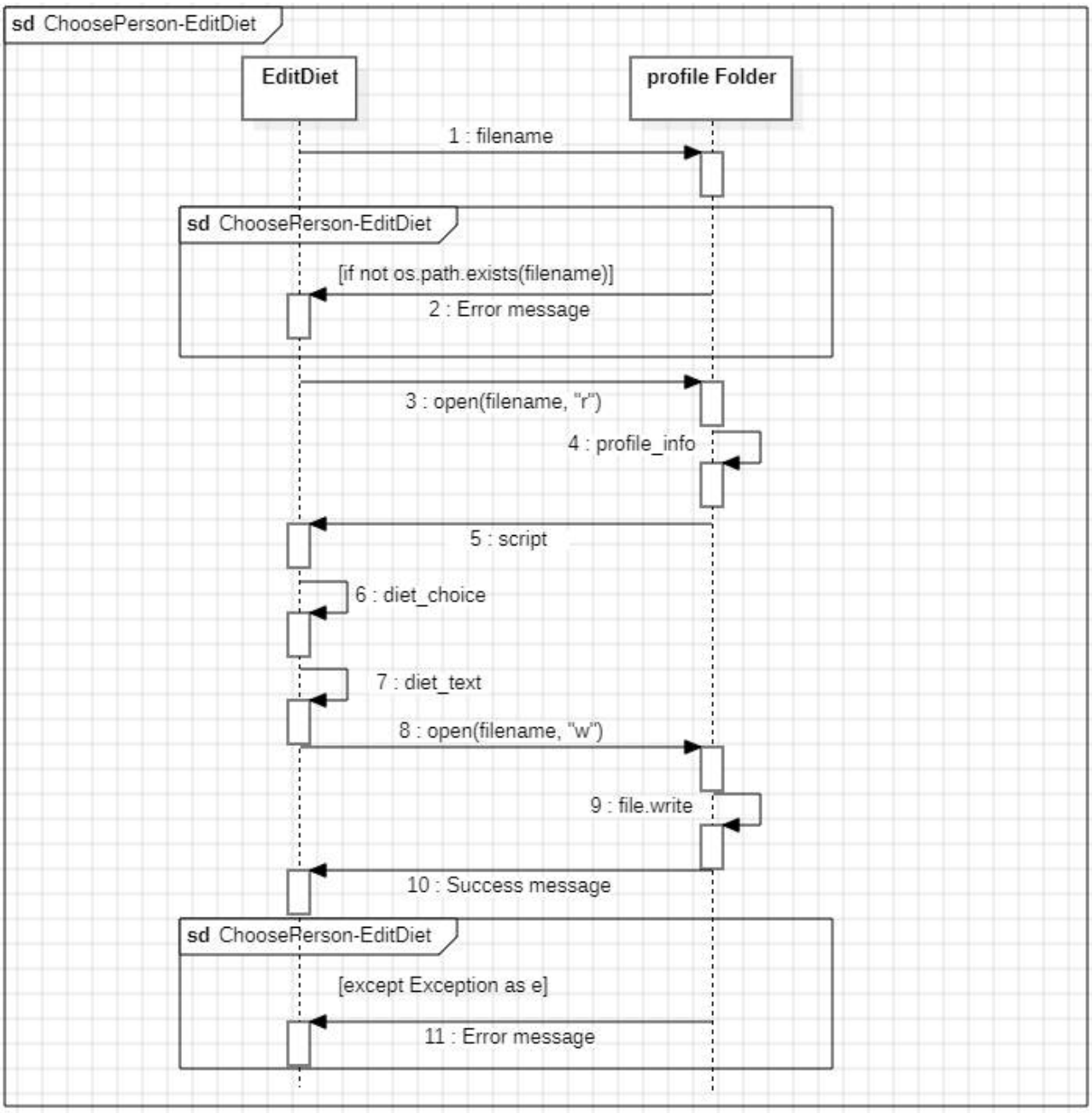
위의 다이어그램은 ChoosePerson diagram중 프로필을 수정하는 과정이다.
프로필을 수정할 때, ChoosePerson class에서 menu_choice 변수에 n을 입력한다. 그 후 수정할 메뉴를 입력받는다. 1을 입력하면 신체정보, 2를 입력하면 활동정보, 3을 입력하면 식사밸런스를 수정 할 수 있다. 4를 입력하면 이전 메뉴로 돌아가고 그 외의 숫자를 입력하면 Error message가 출력된다.



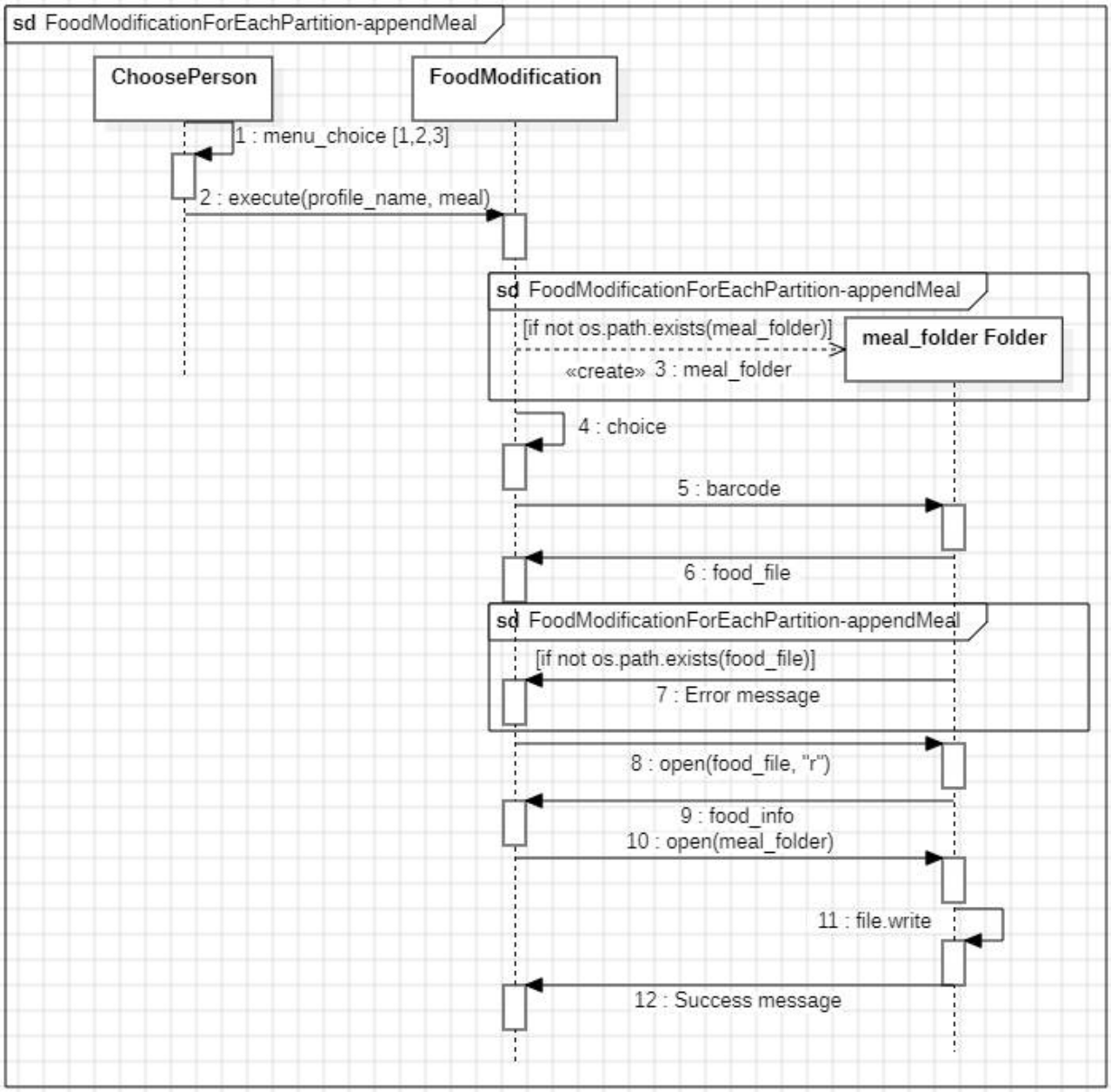
위의 다이어그램은 ChoosePerson diagram의 프로필 수정 중 신체정보 부분을 수정하는 과정이다. EditPhysical class에서 profile Folder에 filename에 해당하는 파일을 찾아 연다. 해당하는 파일이 없다면 Error message를 반환한다. profile_info 변수에 각 데이터들을 분리해서 저장하고, 새로운 데이터를 입력 및 저장한다. 정상적으로 성공했으면 Success message를 반환하고, 실패하면 Error message를 반환한다.



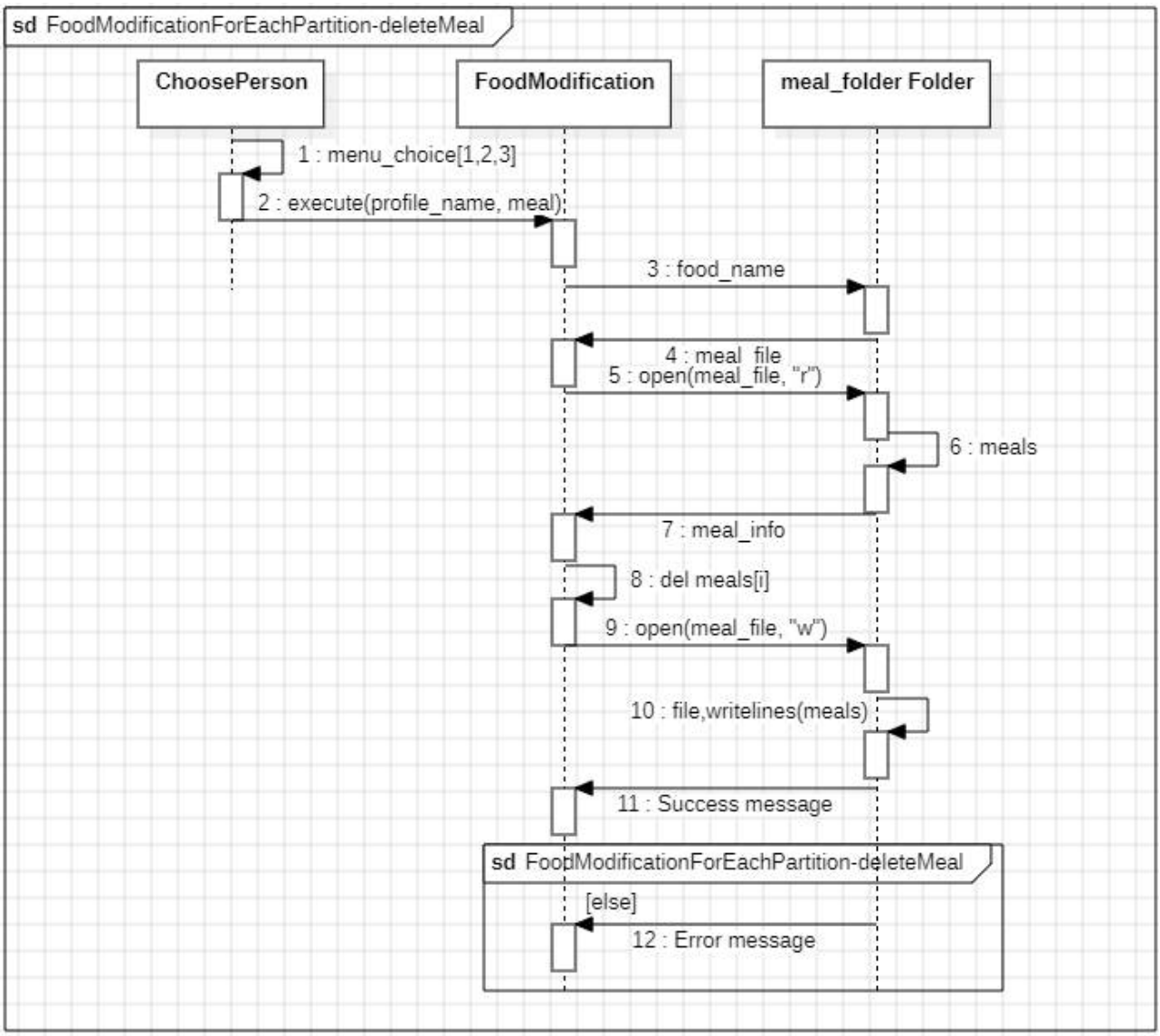
위의 다이어그램은 ChoosePerson diagram의 프로필 수정 중 신체활동 부분을 수정하는 과정이다. EditActivity class에서 profile Folder에 filename에 해당하는 파일을 찾아 연다. 해당하는 파일이 없다면 Error message를 반환한다. profile_info 변수에 각 데이터들을 분리해서 저장하고, 새로운 데이터를 입력 및 저장한다. 정상적으로 성공했으면 Success message를 반환하고, 실패하면 Error message를 반환한다.



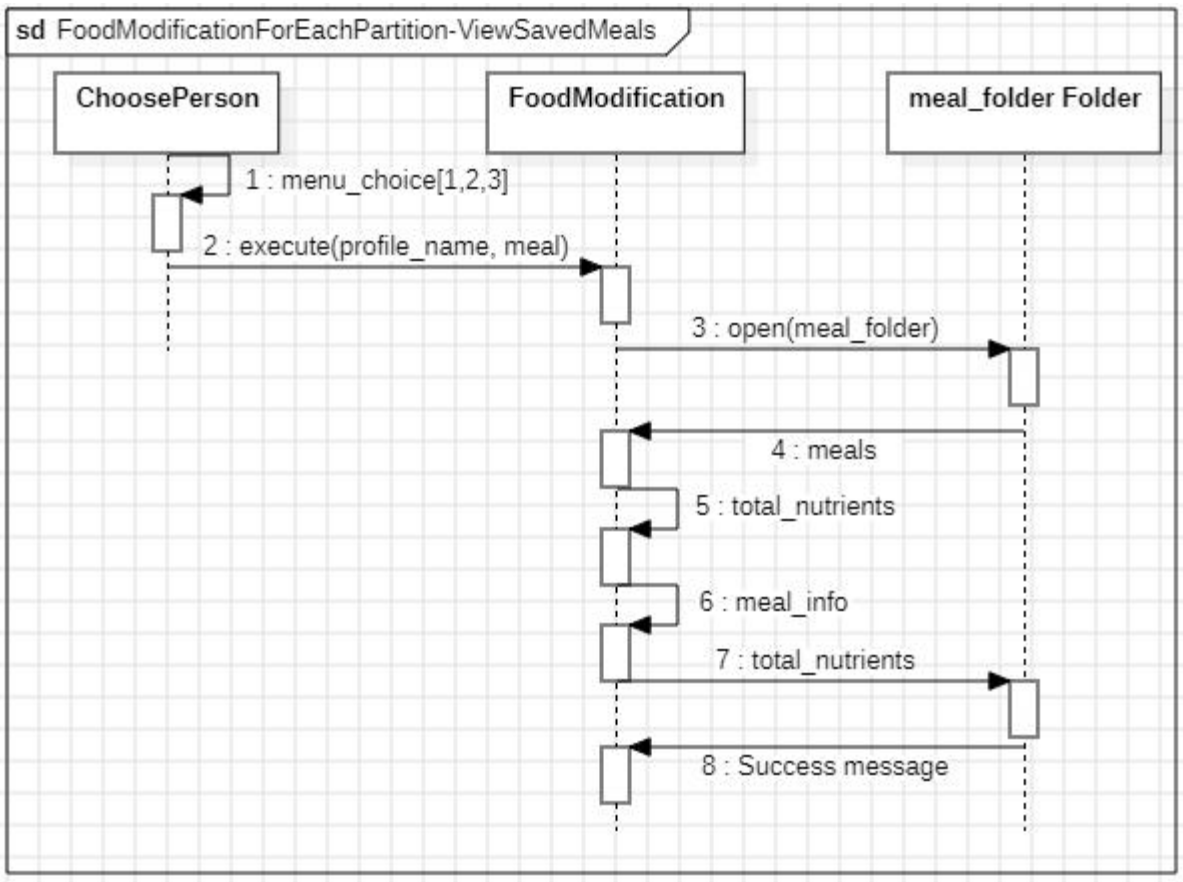
위의 다이어그램은 ChoosePerson diagram의 프로필 수정 중 식사밸런스 부분을 수정하는 과정이다. EditActivity class에서 profile Folder에 filename에 해당하는 파일을 찾아 연다. 해당하는 파일이 없다면 Error message를 반환한다. profile_info 변수에 각 데이터들을 분리해서 저장하고, 새로운 데이터를 입력 및 저장한다. 정상적으로 성공 했으면 Success message를 반환하고, 실패하면 Error message를 반환한다.



위의 다이어그램은 FoodModificationForEachPartition diagram중 먹은/을 식사를 추가하는 과정이다. ChoosePerson class에서 식사 추가를 의미하는 1을 입력한다. 입력한 숫자는 meal 변수에 저장되고, FoodModificationForEachPartition class로 전달된다. meal_folder 폴더가 없으면 생성한다. barcode를 입력해서 저장되어있는 음식 데이터를 불러와 meal_folder 폴더에 추가한다. 정상적으로 완료되었으면 Success message를 반환하고, 실패하면 Error message를 반환한다.

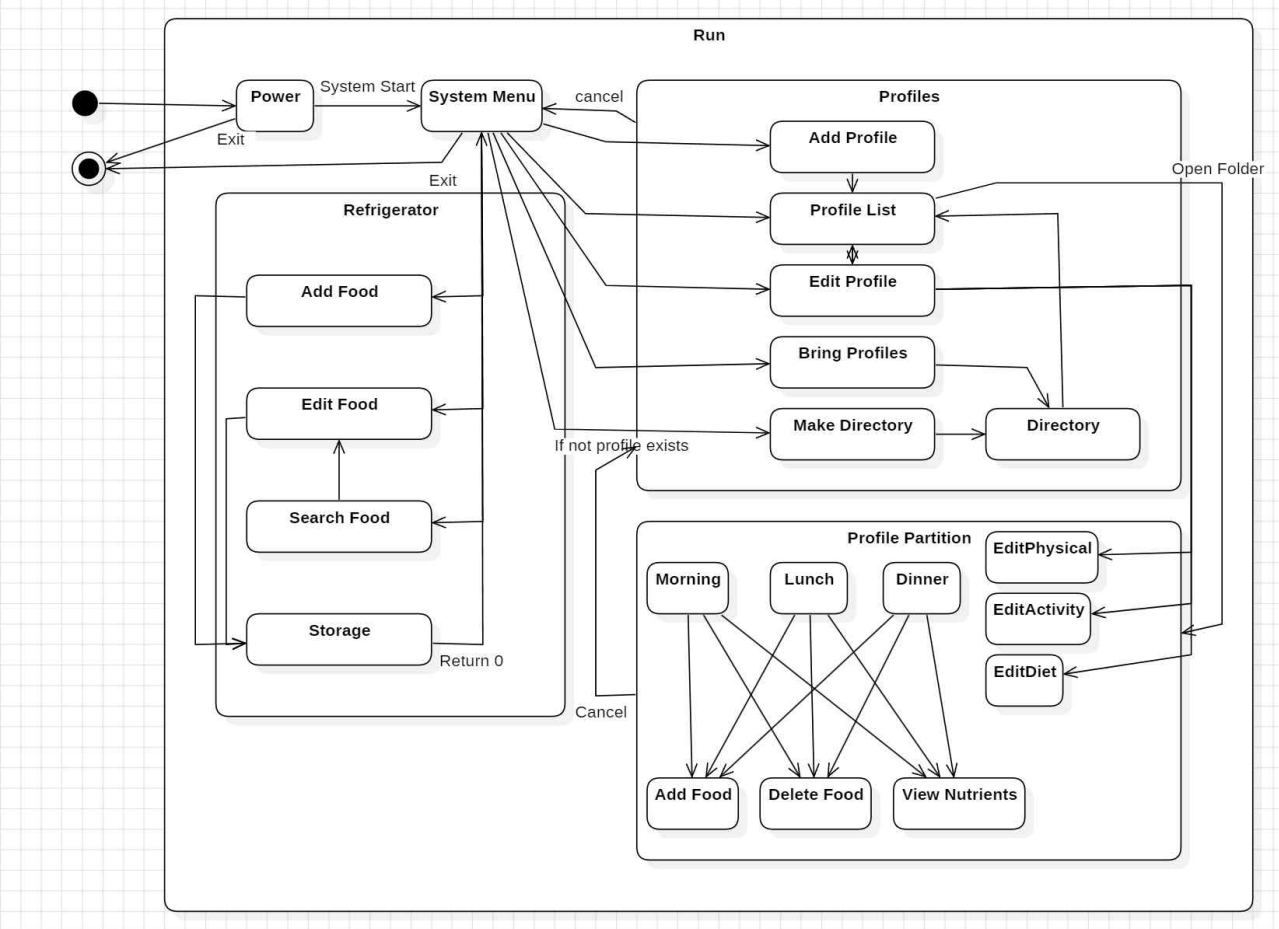


위의 다이어그램은 FoodModificationForEachPartition diagram중 먹은 식사를 삭제하는 과정이다. ChoosePerson class에서 먹은 식사를 제거하고자 할 때, menu_choice 변수에 2를 입력한다. 입력한 숫자는 meal 변수에 저장되고, FoodModificationForEachPartition class로 전달된다. 제거하고자 하는 제품명을 food_name 변수에 입력한다. 해당하는 제품이 추가되어 있었을 때, 성공적으로 제거되었다면 Success message를 반환하고, 실패하면 Error message를 반환한다.



위의 다이어그램은 FoodModificationForEachPartition diagram중 먹은 식사의 총 영양소를 보여주는 과정이다. ChoosePerson class에서 저장된 식사 정보를 보고자 할 때, menu_choice 변수에 3를 입력한다. 입력한 숫자는 meal 변수에 저장되고, FoodModificationForEachPartition class로 전달된다. 불러온 식사는 meals 변수에 저장되고, total_nutrients 변수에 탄수화물, 단백질, 지방, 나트륨 순서로 해당하는 시간대에 먹은 영양소의 총합을 저장하여 보여준다. 정상적으로 완료되면 Success message를 반환한다.

4. State machine diagram



이 장은 시스템이나 객체의 상태 변화를 시각적으로 표현한 State Machine Diagram 이다. Power에서 프로그램 동작을 하거나 바로 종료할 수 있다. 동작을 하면 System Menu를 통해 Refrigerator에서 음식을 추가/수정/찾기를 할 수 있고, Profiles에서 프로필 추가, 수정, 프로필 폴더 생성을 할 수 있다. 원하는 프로필을 선택하면 프로필 폴더 내에 저장되어있는 섭취한 식사를 삭제/추가/먹은 영양소를 확인 할 수 있다.

5. Implementation requirements

5.1 H/W platform requirements

cpu : intel i3+

ram : 4G+

HDD / SSD : 10G+

5.2 S/W platformm requirements

OS : Windows 7+

Implement Language : Python

6. Glossary

Terms	Description
Attribute	클래스의 속성, 객체의 상태를 나타내는 데이터
Method	클래스의 동작을 정의하는 함수, 객체의 상태를 변경하거나, 객체 간의 상호작용 제어
Class Diagram	객체 지향 시스템의 구조를 시각화하는 다이어그램. 클래스, 인터페이스들의 사이 관계 표현
Sequence Diagram	시스템의 동작을 시간 순서에 따라 시각화한 다이어그램.
State Machine Diagram	시스템의 행동을 표현하는 다이어그램. 시스템이나 객체의 상태 변화를 시각적으로 표현함.

7. References

-강의 자료: Structural Modeling I, II

-참고 자료: Always be there, 있고 없고, 전 세계의 음식들, Market Nearby You, Lost & Found App, Real Time Location System