МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ГОУ НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМ. Р.Е. АЛЕКСЕЕВА

ИНСТИТУТ ЭКОНОМИКА И УПРАВЛЕНИЕ

Курсовая работа. Вариант 16

Дисциплина "Технология программирования"

Выполнил: Иванов Михаил Егорович
студент группы: 21- САИ
Проверил: Жевнерчук Дмитрий Валерьевич

Нижний Новгород, 2022

Оглавление

Введение

Программные проекты зачастую достаточно сложны, и их декомпозиция - основная стратегия борьбы со сложностью. Она состоит в разбиении проблемы на мелкие управляемые элементы. ООП — это методология программирования, которая основана на представлении программы в виде совокупности объектов, каждый из которых является реализацией определенного класса (типа особого вида), а классы образуют иерархию на принципах наследуемости». Основная идея объектно-ориентированного анализа и проектирования информационных систем состоит в рассмотрении предметной области и логического решения задачи с точки зрения объектов (понятий или сущностей). В процессе объектно-ориентированного анализа основное внимание уделяется определению и описанию объектов (или понятий) в терминах предметной области. В процессе объектно-ориентированного проектирования определяются логические программные объекты, которые будут реализованы средствами объектно-ориентированного языка программирования. Эти программные объекты включают в себя атрибуты и методы. Шаблоны проектирования — это один из инструментов, который помогает сэкономить время и сделать более качественное решение. Благодаря паттернам разработчику легко не только понимать чужой код, но и расширять его своими решениями, а также добавлять новые.

Необходимо реализовать консольное приложение, позволяющее получать и использовать пользовательские данные для расчёта необходимого суточного потребления питательных веществ. Полученные данные необходимо представить в совместимом виде для

реализации расчётов рациона в другой части задачи. Необходимо обеспечить возможность расширения по получению данных и расчётам других микро- и макроэлементов. Для поддержания требуемого функционала подходит паттерн **Фабричный метод** — порождающий паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов. Для того, чтобы гарантировать совместимость данных используем паттерн **Адаптер** — это структурный паттерн проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе.

Цель курсовой работы – разработать требуемую модель программы, научиться вести разработку согласно принципам ООП, пользоваться паттернами программирования, а также языком программирования Python.

1. **Постановка задачи (Вариант 16)**

Разработайте объектно-ориентированную модель для конструктора расчётов рациона (количества белков, жиров, углеводов).

- Система должна быть расширяема по характеристикам человека, по режимам физической деятельности и ожидаемым результатам.
- Система должна поддерживать справочник продуктов растительного и животного происхождения.
- Также необходимо обеспечить возможность настройки комбинированных режимов питания с циклами различной длительности.

Вследствие разделения реализации задачи на две части задача может быть представлена в виде:

Разработайте объектно-ориентированную модель для сбора пользовательской информации и её интерпретации для расчётов необходимого дневного уровня потребления белков, жиров, углеводов и пр.

- Система должна быть расширяема по характеристикам человека, по режимам физической деятельности и ожидаемым результатам.

2. **Анализ предметной области**

А) Текстовое описание сущностей предметной области с указанием их свойств и функций.

Ввод данных осуществляется через консоль. Для выведения словаря, содержащего показатели необходимых питательных элементов и ключей к ним, используем Фабричный метод, позволяющий гибко настраивать способы расчётов питательных веществ и Адаптер, позволяющий неизменно получать данные в корректном виде. Данные расчётов собираются в словарь в классе NeededConstructor.

Классы:

Input, Adapter, Calories, CalculateMethodsFactory, NeededConstructor

Fibre_method, Protein_method, Fat_method, Carbohydrate_method – это классы продукты.
Интерфейсы:
Calculate_methods – интерфейс продуктов – способов вычисления.
Конкретный класс фабрики:
CalculateMethodsFactoryExample - это класс, реализующий интерфейс фабрики и содержащий непосредственно код.

Б) Текстовое описание декомпозиции или интеграции сущностей, делегирование функций одними сущностями другим.
Получая вводные данные, программа составляет некоторое количество переменных(полей), которые впоследствии будут интегрированы в один словарь, сохраняющий уникальные атрибуты переменных.

В) Текстовое описание ассоциаций (агрегаций, композиций) и обобщений.
Агрегация (агрегирование по ссылке) — отношение «часть-целое» между двумя равноправными объектами, когда один объект (контейнер) имеет ссылку на другой объект. Оба объекта могут существовать независимо: если контейнер будет уничтожен, то его содержимое — нет.
Композиция (агрегирование по значению) — более строгий вариант агрегирования, когда включаемый объект может существовать только как часть контейнера. Если контейнер будет уничтожен, то и включённый объект тоже будет уничтожен.
Обобщение и наследование позволяют выявить аналогии между различными классами объектов, определяют многоуровневую классификацию объектов.

Описание связей в своем приложении считаю логичным начать «сверху вниз»

Класс NeededConstructor пользуется функционалом Фабричного метода. Ассоциация кратности {1;1}

Фабричный метод обращается к интерфейсу продуктов. Ассоциация кратности {1;1}

Фабричный метод создаёт продукты, реализующие интерфейс Calculate_methods. Реализация.

Классы-продукты используют поля класса Calories. Ассоциация агрегации {1;1}

Класс Calories обращается к паттерну Адаптер, получая совместимые для расчётов данные. Ассоциация кратности {1,1}.

Класс(паттерн) Adapter обращается к классу Input, получая введённые пользователем данные. Ассоциация кратности {1,1}.

3. Разработка объектно-ориентированной модели предметной области.

3.1 Описание структуры классов (POJO, Java Bean или другая).

JavaBean — это объект Java, который удовлетворяет определенным соглашениям программирования:

- класс JavaBean должен реализовывать либо Serializable, либо Externalizable;

- класс JavaBean должен иметь общедоступный конструктор без аргументов;
- все свойства JavaBean должны иметь общедоступные методы установки и получения (при необходимости);
- все переменные экземпляра JavaBean должны быть закрытыми.

Класс POJO - это обычный класс без каких-либо специальностей, класс, полностью слабо связанный с технологией/фреймворком. Класс не реализуется из технологии/фреймворка и не расширяется из API-интерфейса технологии/фреймворка.

Многие классы в будущем приложении расширяются из интерфейсов, а также удовлетворяют соглашениям JavaBean, следовательно, структура классов – JavaBean.

Data transfer object (DTO) — это объект, предназначенный только для транспортировки данных т.е. в нем нет никакой логики. У этого паттерна есть много областей применения, но т.к. в Python явная типизация опциональна, то DTO в Python приобретают новое назначение: **DTO помогает явно указать контракт между компонентами.**

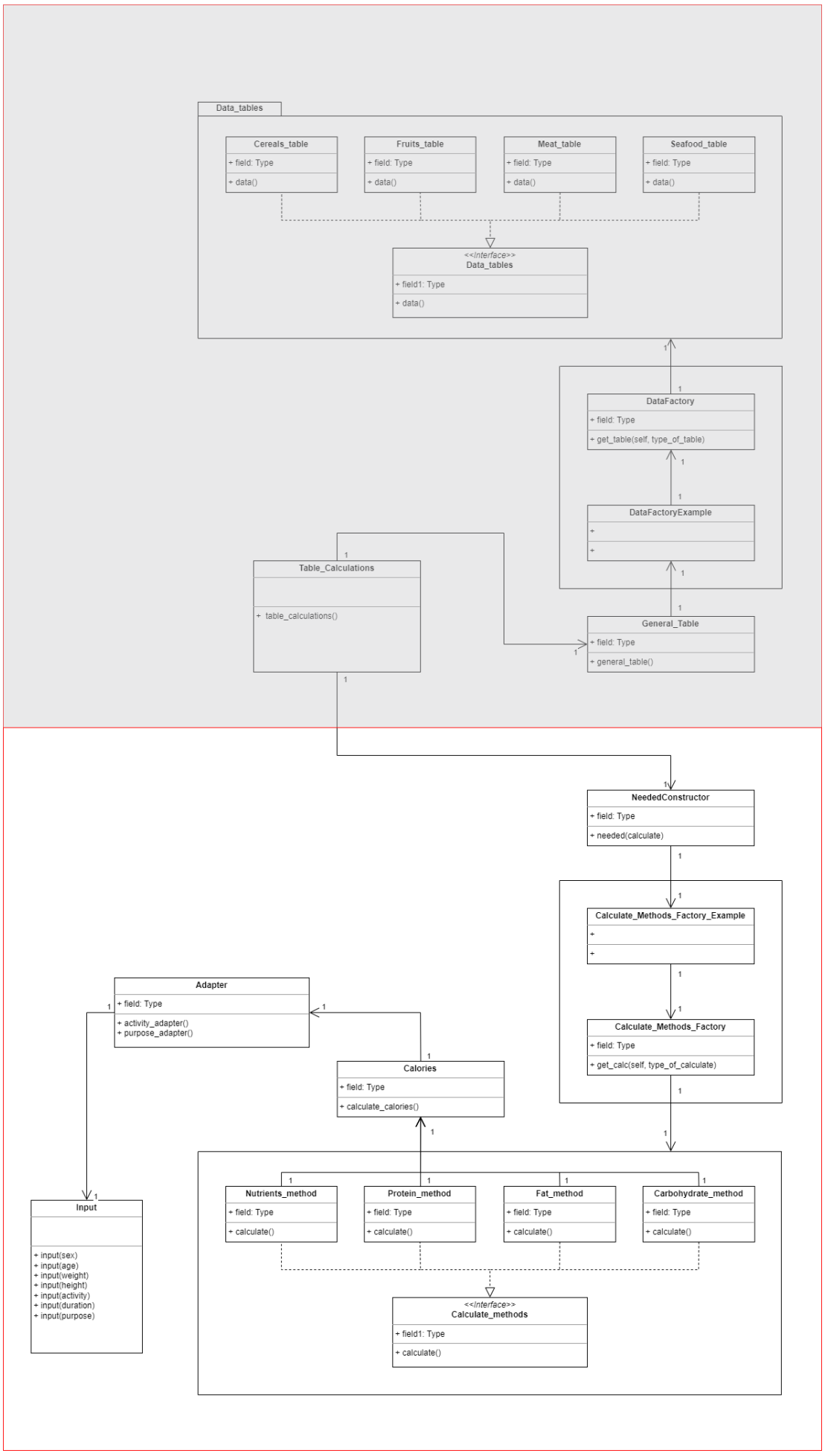
3.2 Описание подсистемы управления созданием объектов, описание применения порождающих шаблонов проектирования.

Программа получает данные, обрабатывает их, для обработки и получения совместимых для расчётов данных используется паттерн Адаптер. Для выбора наиболее эффективной методики расчётов используется Фабричный метод – с его помощью создаются отдельные методы расчётов, соответствующие определенным питательным веществам.

3.3 Разработка объектно-ориентированной модели предметной области (с построением диаграммы классов). Обоснование слабой межмодульной связанности и расширяемости системы.

Программный код достаточно четко разбит на модули: продукты, фабрики, «полезные функции» и интерфейсы

Продукты отвечают за хранение, прием и отдачу полей. Это дает четкое понимание откуда нужно взять поля для создания нового объекта, анализа и т.д.


Data_tables
Cereals_table
+ field: Type
+ data()
Fruits_table
+ field: Type
+ data()
Meat_table
+ field: Type
+ data()
Seafood_table
+ field: Type
+ data()
<<Interface>>
Data_tables
+ field1: Type
+ data()
DataFactory
+ field: Type
+ get_table(self, type_of_table)
DataFactoryExample
+
+
Table_Calculations
+ table_calculations()
General_Table
+ field: Type
+ general_table()
NeededConstructor
+ field: Type
+ needed(calculate)
Calculate_Methods_Factory_Example
+
+
Calculate_Methods_Factory
+ field: Type
+ get_calc(self, type_of_calculate)
Adapter
+ field: Type
+ activity_adapter()
+ purpose_adapter()
Calories
+ field: Type
+ calculate_calories()
Input
+ input(sex)
+ input(age)
+ input(weight)
+ input(height)
+ input(activity)
+ input(duration)
+ input(purpose)
Nutrients_method
+ field: Type
+ calculate()
Protein_method
+ field: Type
+ calculate()
Fat_method
+ field: Type
+ calculate()
Carbohydrate_method
+ field: Type
+ calculate()
<<Interface>>
Calculate_methods
+ field1: Type
+ calculate()

4. Реализация объектно-ориентированной библиотеки и приложения:

4.1 Аналитический обзор языков программирования (привести краткую характеристику 2-3 варианта). Описать преимущества и недостатки каждого инструмента, обосновать выбор языка программирования для решения задачи по варианту.

В качестве методологии программирования было выбрано объектно-ориентированное программирование. Самыми популярными и востребованными языками ООП являются: Java, Python, C++, C#, JavaScript, Ruby.

Java — язык программирования общего назначения. Относится к объектно-ориентированным языкам программирования, к языкам с сильной типизацией. Создатели реализовали принцип WORA: write once, run anywhere или «пиши один раз, запускай везде». Это значит, что написанное на Java приложение можно запустить на любой платформе, если на ней установлена среда исполнения Java (JRE, Java Runtime Environment).

Плюсы:

1) Концепция ООП (объектно-ориентированного программирования). Это значит, что программист сам определяет тип данных, его структуру и набор применяемых к нему функций.

2) Это язык высокого уровня, т.е., он больше похож на человеческую речь, а не на машинный код. Следовательно, у него сравнительно простой синтаксис, что делает его быстрым для освоения и удобным для написания кода, его чтения и обслуживания. Есть и более простые варианты (например, Python), однако у человека с базовым пониманием основ программирования здесь не должно возникнуть сложностей;

3) Безопасность. У Java есть несколько функций, которые ликвидируют часто встречающиеся уязвимости. В частности, это Security Manager – создаваемая для каждого приложения политика безопасности, в которой можно прописать правила доступа;

4) Удобство для распределённого программирования. Этот язык изначально создавался для совместной работы (в том числе удалённой), поэтому он позволяет совместно использовать данные и программы несколькими компьютерами одновременно;

5) Принцип «написать один раз и использовать везде» — написанное на Java приложение можно запустить на любой поддерживающей его платформе;

6) Стабильное и постоянно развивающееся сообщество. По многочисленности и активности с ним мало кто может соперничать. В Сети есть масса ресурсов, где на любой вопрос по этой теме либо уже есть ответ, либо найдётся кто-нибудь, кто его подскажет, равно как и сотни курсов, семинаров и обучающих программ, как платных, так и бесплатных.

Минусы:

1) Низкая скорость. Все высокоуровневые языки приходится компилировать с помощью виртуальной машины, что плохо сказывается на их производительности. Java – не исключение, кроме того, у него есть и некоторые собственные особенности, вызывающие дополнительные проблемы с производительностью;

2) Многословие (verbosity). Сходство с естественными языками делает Java проще для изучения и понимания, но также ведёт и к тому, что он содержит много лишней информации и довольно громоздок;

Python — универсален и используется во многих местах. Однако Python прочно обосновался в машинном обучении и науке о данных. Это один из предпочтительных языков для этой новой и постоянно растущей области. Он имеет четко структурированное семантическое ядро и достаточно простой синтаксис. Все, что пишется на этом языке, всегда легко читаемо. В случае необходимости передать аргументы язык использует функцию call-by-sharing. Набор операторов в языке вполне стандартен. Удобная особенность синтаксиса – это форматирование текста кода при помощи разбивки их на блоки с помощью отступов, которые создают нажатием клавиш «Space» и «Tab». В синтаксисе отсутствуют фигурные или операторные скобки, обозначающие начало и конец блока. Такое решение заметно сокращает количество строк тела программы и приучает программиста соблюдать хороший стиль и аккуратность при написании кода.

Плюсы Python:

1) Понятность кода. Синтаксическая особенность Python — выделение блоков кода отступами, что значительно упрощает зрительное восприятие программ, написанных на этом языке.

2) Интерпретируемость. Программы, написанные на языке программирования Python, не переводятся в машинный код, а сразу выполняются программой-интерпретатором. Это позволяет запускать код на любой платформе с установленным заранее интерпретатором.

3) Объектно ориентированность. Python — это язык, созданный согласно парадигме объектно ориентированного программирования (ООП). В ней основными являются понятия объекта и класса. Классы — это специальные типы данных, объекты — экземпляры классов. То есть любое значение является объектом конкретного класса. В Python вы можете не только использовать уже существующие классы, но и создавать свои собственные.

4) Динамическая типизация. В отличие от C-подобных языков программирования, в Python переменные связываются с типом в момент присваивания в них конкретных значений.

Минусы:

1) Python является одним из самых медленных языков программирования.

2) Python не подходит для задач, которые требуют большого объема памяти.

C ++ — обладает скоростью C с функциональностью классов и объектно-ориентированной парадигмой. Это скомпилированный, надежный и мощный язык. Фактически, он даже используется для создания компиляторов и интерпретаторов для других языков. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Плюсы

1) Поддержка объектно-ориентированного программирования (ООП). ООП помогает сделать код проще, и его быстрее писать. Большой цикл статей про ООП.

2) Высокая скорость.

3) Возможности для работы с данными на низком уровне — то есть на уровне, близком к аппаратному. Благодаря этому на C++ можно писать драйвера, микроконтроллеры.

4) Популярность:

Для C++ создано много библиотек и компиляторов.

C++ используется практически везде (несколько примеров мы уже привели выше).

5) Синтаксис C++ похож на синтаксис C, C# и Java, так что переключаться между этими языками достаточно легко.

6) Совместимость с C благодаря тому, что C++ создавался на его основе.

Минусы

1) Небезопасность: C++ даёт большую свободу действий, но и не удержит вас от ошибок. А лёгкий доступ к памяти делает его уязвимым не только во время хакерских атак, но и при неосторожной работе.

2) Зависимость от платформы: написать на C++ портативный код (такой, который бы работал на разных платформах) очень сложно.

3) Синтаксис строгий и «многословный»: код читается хуже, чем в некоторых других языках

4) Сложность: у C++ сложный синтаксис и маленькая стандартная библиотека, а ещё надо разбираться в указателях и работе с памятью, поэтому учить его нелегко, особенно с нуля.

Сравнение Python и Java (Python — это интерпретируемый язык с динамической типизацией. Java же — компилируемый язык со статической типизацией. Эти различия делают Python и Java полными противоположностями друг друга в плане скорости запуска и выполнения программ. Код, написанный на Python, быстрее запускается и дольше выполняется. В то время как программы на Java медленнее запускаются, но гораздо быстрее выполняются. С помощью Java можно разрабатывать кроссплатформенные приложения, но и Python также совместим со многими операционными системами. Также с помощью этих языков программирования разработчики могут создавать сетевые приложения. Если говорить о сложности этих двух языков, то Java безусловно уступает Python в простоте изучения.)

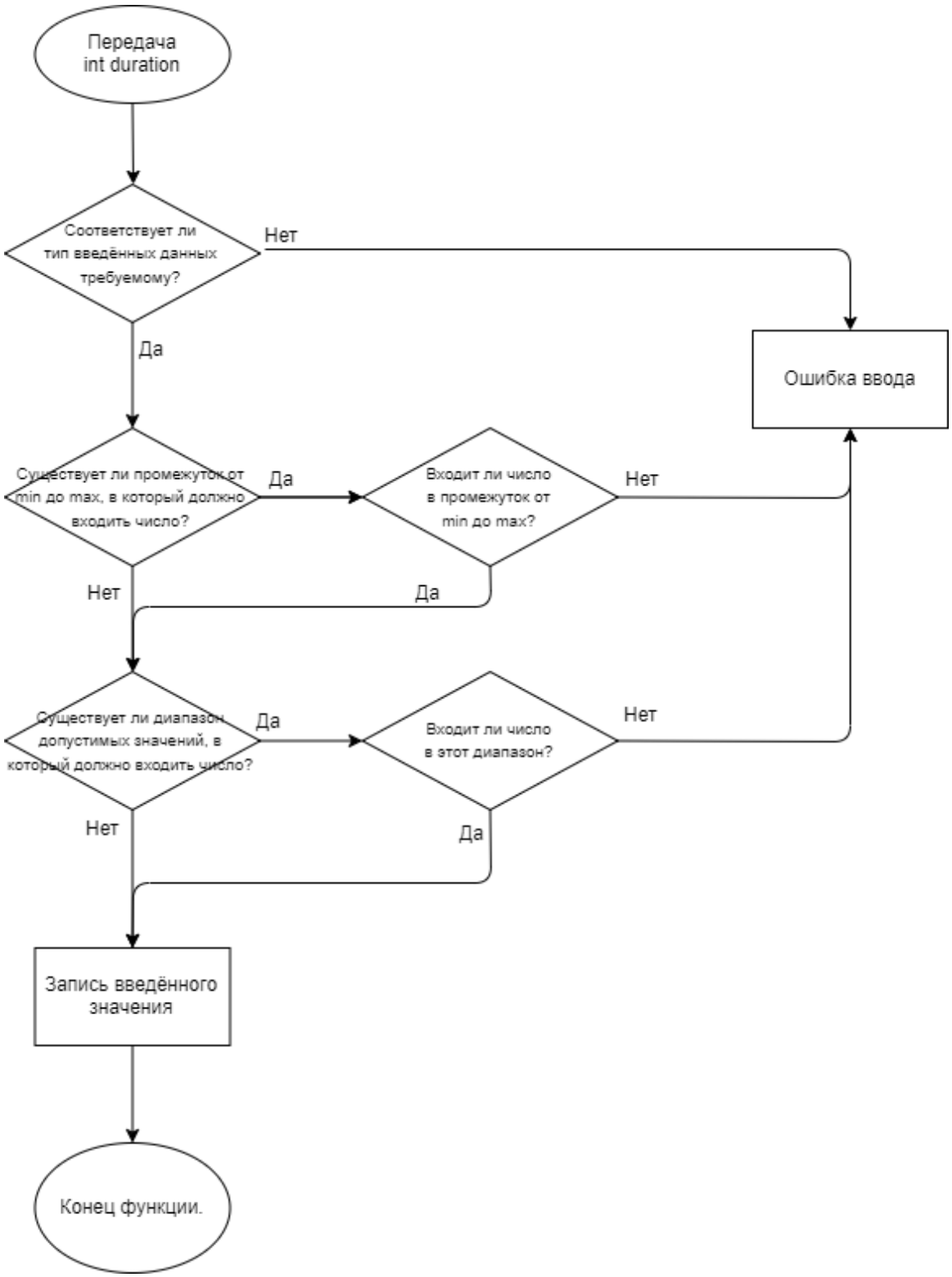
Сравнение C++ и Java (Зависимость от платформы: при компиляции программа на Java сначала переводится в особый байт-код с помощью интерпретатора. Благодаря этому код в Java можно запустить на любой платформе. C++ при компиляции переводится в машинный код, поэтому это платформо-зависимый язык. Указатели: огромная часть работы в C++ — работа с указателями, а в Java их нельзя использовать. В Java нет многих комплексных функций, которые есть в C++, — например структур, указателей и объединений. Многопоточность поддерживается в Java, но не в C++. Уровень языка: и Java, и C++ — высокоуровневые языки, но на Java работать на низком уровне не получится, в отличие от C++.)

Сравнение Python и C++ (Одно из главных достоинств Python — простой и понятный синтаксис. Программистам с C++ он будет понятен почти сразу, пусть изначально может и не хватать скобок и точек с запятой. У Python огромная стандартная библиотека с ридерами/райтерами для CSV, ZIP и других форматов, XML-парсеры, инструменты для работы с сетью и так далее. Язык подходит для создания веб-приложений. Лучше всего подходит для машинного обучения. Главное преимущество C++ — производительность. Его скорость работы намного выше в сравнении с Python. C++ подходит почти для всех платформ, а также для встроенных систем, в то время как Python работает только на отдельных платформах, поддерживающих высокоуровневые языки. C++ более предсказуем благодаря статической типизации. Это же влияет и на производительность. При работе с C++ можно изучать низкоуровневое программирование, ведь язык близок к железу. В случае с Python это не сработает.)

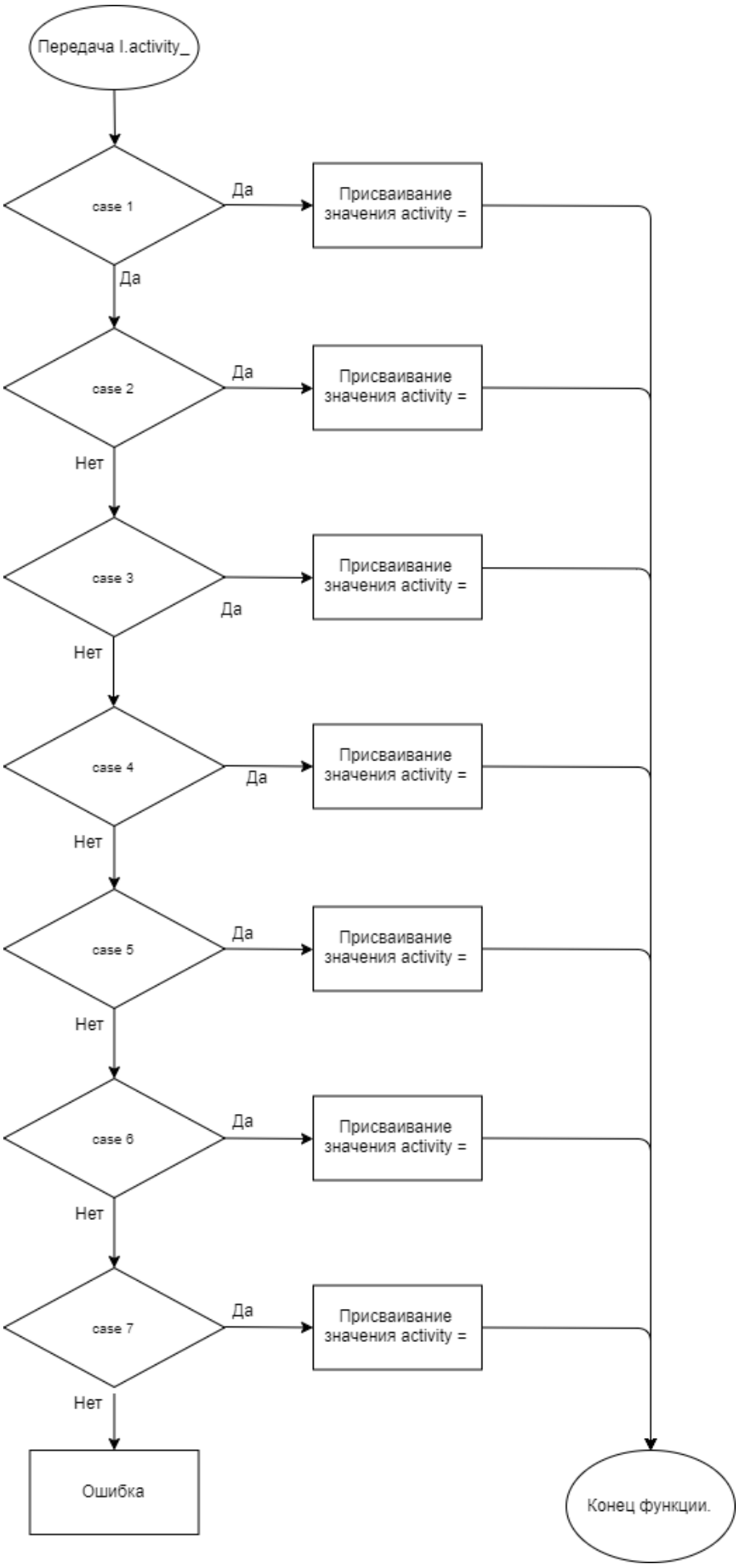
По итогу основным языком разработки был выбран язык Python. Он является универсальным языком программирования. Для решения задачи по варианту он подходит больше, т.к. имеет ряд преимуществ. Он позволяет многократно использовать одни и те же объекты в разных программах, более удобен для организации структуры программ, помогает избегать ошибок и упрощает поддержку и модернизацию старого кода, наряду с чем позволяет реализовать математическую модель расчётов, предоставляя обширные возможности математических расчётов с помощью библиотеки PULP.

4.2 Описание алгоритмов, реализуемых программно (привести блок-схемы хотя бы двух алгоритмов). Описание реализации методов.

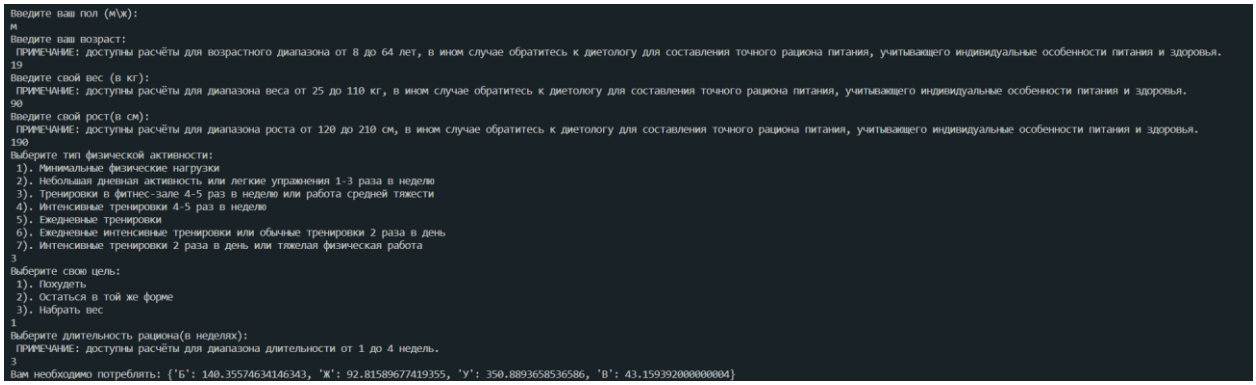
Блок-схема метода Input_ из класса Input. Данный алгоритм осуществляет проверку данных на вводе, возвращая значение типа, заранее заданного для проверки.



Блок-схема метода activity_adapter из класса Adapter. Данный алгоритм реализует конвертацию данных из логического формата, где пользователь вводил int значение, соответствующее определённой логике, в формат данных для реализации математической модели расчётов.


Передача I.activity_
case 1
Да
Присваивание значения activity =
Да
case 2
Да
Присваивание значения activity =
Нет
case 3
Да
Присваивание значения activity =
Нет
case 4
Да
Присваивание значения activity =
Нет
case 5
Да
Присваивание значения activity =
Нет
case 6
Да
Присваивание значения activity =
Нет
case 7
Да
Присваивание значения activity =
Нет
Ошибка
Конец функции.

4.3 Описание тестового примера: входные, выходные и промежуточные данные для каждого теста, привести скриншоты тестов, из которых очевидна корректность работы программы (при необходимости добавьте пояснения к скриншотам).



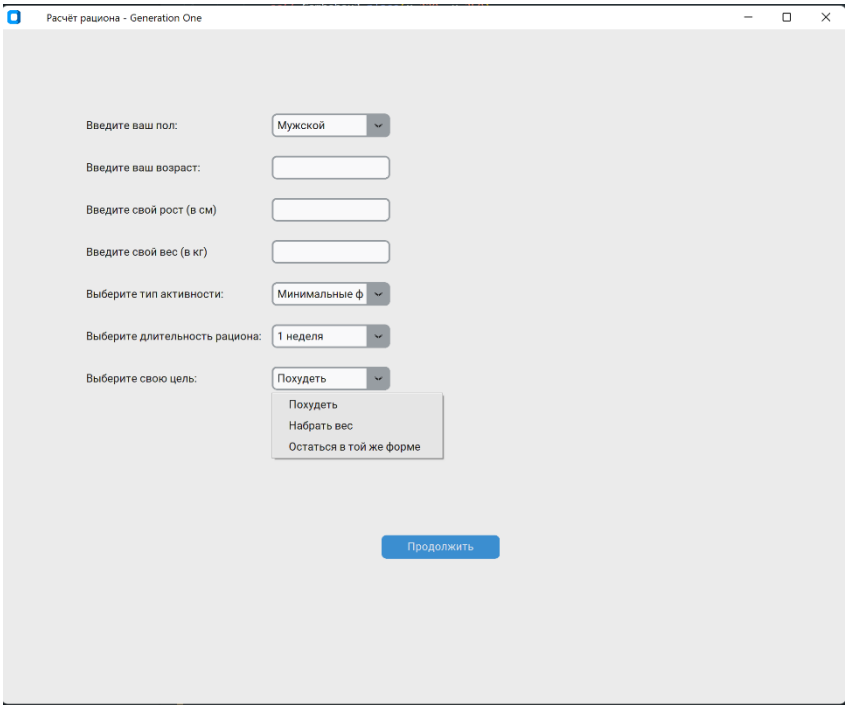
В конце работы программы выдаётся готовый словарь необходимых значений, который интерпретируется в другой части программы для совершения всех расчётов.

Заключение

Разработано консольное приложение, способное принимать данные, конвертируя их в совместимый для последующий расчётов формат. Данное приложение имеет некоторый функционал, позволяющий на основе введённых характеристик рассчитывать необходимый уровень потребления основных питательных веществ.

Преимуществами программы является простота использования, строгая структура, открывающая возможность расширения.

Недостатками являются недостаточная защищённость данных, недостаточное ослабление связей между классами. Нет классов или интерфейсов, которые бы улучшили структуру взаимодействия классов, осуществляющих вывод данных и непосредственно рассчитывающих эти данные.



В перспективах: добавление графического пользовательского интерфейса, составление словаря необходимых питательных веществ на основе усовершенствованных методик расчёта(расчёт витаминов и микроэлементов, использование нескольких формул расчёта калорийности рациона и получение усреднённого значения и тд.)

Список литературы

1. Погружение в паттерны проектирования / Александр Швец изд. 2018

2. JavaBeans / https://ru.wikipedia.org/wiki/JavaBeans

3. Языки объектно-ориентированного программирования / https://bestprogrammer.ru/programmirovanie-i-razrabotka/yazyki-obektno-orientirovannogo-programmirovaniya

4. Python краткий обзор языка / https://techrocks.ru/2019/01/21/about-python-briefly/

5. Различия Python и C++ / https://pythonru.com/baza-znanij/python-ili-c

6.Что нужно знать о C++ / https://skillbox.ru/media/code/vybiraem_yazyk_programmirovaniya_chto_nuzhno_znat_o_s_

7. Плюсы и минусы Java https://www.cischool.ru/plyusy-i-minusy-java/

8. Плюсы и минусы Python https://skysmart.ru/articles/programming/preimushestva-i-nedostatki-python

9. https://codelab.ru/cat/patterns

10. http://book.uml3.ru/sec_3_3

11. https://github.com/blinky-z/OOP-Tutorial

12. https://docs.python.org/3/reference/executionmodel.html

13.Как устроен фабричный метод Python https://webdevblog.ru/shablon-fabrichnogo-metoda-i-ego-realizaciya-v-python/

Приложение 1. Программный код

Input.py

```
class Input:
    def input_(prompt, type=None, min=None, max=None, range=None):
        if min is not None and max is not None and max < min:
            raise ValueError("Минимальное значение должно быть меньше или равно максимальному.")
        while True:
            ui = input(prompt)
            if type is not None:
                try:
                    ui = type(ui)
                except ValueError:
```

```
                    print("Тип введённых данных должен быть:
{0}.".format(type.__name__))
                    continue
            if max is not None and ui > max:
                print("Введённые данные должны быть меньше либо равны:
{0}.".format(max))
            elif min is not None and ui < min:
                print("Введённые данные должны быть больше либо равны:
{0}.".format(min))
            elif range is not None and ui not in range:
                if isinstance(range, range):
                    template = "Введённые данные должны быть в диапазоне от
{0.start} до {0.stop}."
                    print(template.format(range))
                else:
                    template = "Введённые данные должны соответствовать: {0}."
                    if len(range) == 1:
                        print(template.format(*range))
                    else:
                        expected = " или ".join((
                            ", ".join(str(x) for x in range[:-1]),
                            str(range[-1])
                        ))
                        print(template.format(expected))
            else:
                return ui

sex = Input.input_("Введите ваш пол (м\ж): \n", str.lower, range=('м','ж'))
age = Input.input_("Введите ваш возраст: \n ПРИМЕЧАНИЕ: доступны расчёты для
возрастного диапазона от 8 до 64 лет, в ином случае обратитесь к диетологу для
составления точного рациона питания, учитывающего индивидуальные особенности
питания и здоровья. \n", int, 8, 64)

weight = Input.input_("Введите свой вес (в кг): \n ПРИМЕЧАНИЕ: доступны расчёты
для диапазона веса от 25 до 110 кг, в ином случае обратитесь к диетологу для
составления точного рациона питания, учитывающего индивидуальные особенности
питания и здоровья. \n", int, 40, 110)

height = Input.input_ ("Введите свой рост(в см): \n ПРИМЕЧАНИЕ: доступны расчёты
для диапазона роста от 120 до 210 см, в ином случае обратитесь к диетологу для
составления точного рациона питания, учитывающего индивидуальные особенности
питания и здоровья. \n", int, 120, 210)

activity_ = Input.input_ ("Выберите тип физической активности: \n 1). Минимальные
физические нагрузки \n 2). Небольшая дневная активность или легкие упражнения 1-3
раза в неделю \n 3). Тренировки в фитнес-зале 4-5 раз в неделю или работа средней
тяжести \n 4). Интенсивные тренировки 4-5 раз в неделю \n 5). Ежедневные
тренировки \n 6). Ежедневные интенсивные тренировки или обычные тренировки 2 раза
в день \n 7). Интенсивные тренировки 2 раза в день или тяжелая физическая работа
\n", int, range=(1,2,3,4,5,6,7))
```

```
purpose_ = Input.input_("Выберите свою цель: \n 1). Похудеть \n 2). Остаться в
той же форме \n 3). Набрать вес \n", int, range=(1,2,3))
duration = Input.input_("Выберите длительность рациона(в неделях): \n ПРИМЕЧАНИЕ:
доступны расчёты для диапазона длительности от 1 до 4 недель. \n", int, 1, 4)
```

Adapter.py

```
import Input as I
class Adapter:

    def activity_adapter():
        match I.activity_:
            case 1:
                activity = 1.2
            case 2:
                activity = 1.4
            case 3:
                activity = 1.6
            case 4:
                activity = 1.8
            case 5:
                activity = 2
            case 6:
                activity = 2.2
            case 7:
                activity = 2.4
        return activity


    def purpose_adapter():

        match I.purpose_:
            case 1:
                purpose = -0.1
            case 2:
                purpose = 0.15
            case 3:
                purpose = 0
        return purpose









    sex = I.sex
    age = I.age
    weight = I.weight
    height = I.height
    duration = I.duration
```

CalcculateMethodsFactory.py

```
from abc import ABC, abstractmethod
import Adapter as A

class Calories():
    def calculate_calories():
        match A.sex:
            case "Ж":
                pre_calc_calories = 9.99 * A.weight + 6.25 * A.height - 4.92 *
A.age - 161
            case "М":
                pre_calc_calories = 9.99 * A.weight + 6.25 * A.height - 4.92 *
A.age + 5

        global cal
        cal = (pre_calc_calories + (A.purpose_adapter() * pre_calc_calories)) *
A.activity_adapter()

class Calculate_methods(ABC):
    @abstractmethod
    def calculate():
        pass

class Protein_method(Calculate_methods):
    def calculate():
        protein = (cal*0.2)/4.1
        return protein

class Fat_method(Calculate_methods):
    def calculate():
        fat = (cal*0.3)/9.3
        return fat

class Carbohydrate_method(Calculate_methods):
    def calculate():
        carbohydrate = (cal*0.5)/4.1
        return carbohydrate

class Fibre_method(Calculate_methods):

    def calculate():
        fibres = (cal*0.015)
        return fibres

class Calculate_Methods_Factory:
    def get_calc(self, type_of_calculate):
        Calories.calculate_calories()
        match type_of_calculate:
```

```
            case "Protein":
                return Protein_method.calculate()
            case "Fat":
                return Fat_method.calculate()
            case "Carbohydrate":
                return Carbohydrate_method.calculate()
            case "Fibre":
                return Fibre_method.calculate()
```

ClculateMethodsFactoryExample.py

```
from CalculateMethodsFactory import Calculate_Methods_Factory


factory = Calculate_Methods_Factory()
protein = factory.get_calc("Protein")
fat = factory.get_calc("Fat")
carbohydrate = factory.get_calc("Carbohydrate")
fibre = factory.get_calc("Fibre")
```

NeededConstructor.py

```
import CalculateMethodsFactoryExample as C

class NeededConstructor:
    def needed():
        protein = C.protein
        fat = C.fat
        carbohydrate = C.carbohydrate
        fibre = C.fibre

        needed = {'Б': protein, 'Ж': fat, 'У': carbohydrate, 'В': fibre}

        print("Вам необходимо потреблять:", needed)
```

Приложение 2. Руководство пользователя консольного приложения