PYTHON329 HW №29
Домашнее задание

Краткое Содержание Задания

В этом домашнем задании вы разработаете систему для заказа пиццы, используя паттерны проектирования "Абстрактная Фабрика" и "Строитель". Задача включает создание классов для управления ингредиентами и размерами пиццы, а также логики сборки самой пиццы. Основная цель - показать, как можно эффективно использовать данные паттерны для создания гибкой и расширяемой системы.

Структура

Абстрактный Класс `IngredientFactory`

- **Назначение**: Определяет интерфейс для создания ингредиентов пиццы.
- **Методы**:
  - `create_cheese()`: Абстрактный метод, должен быть переопределен для возврата типа сыра.
  - `create_sauce()`: Абстрактный метод, должен быть переопределен для возврата типа соуса.

Класс `DodoIngredientFactory`

- **Назначение**: Реализует `IngredientFactory`, предоставляя конкретные ингредиенты, используемые Додо Пиццей.
- **Методы**:
  - `create_cheese()`: Возвращает тип сыра, используемый Додо Пиццей.
  - `create_sauce()`: Возвращает тип соуса, используемый Додо Пиццей.

Класс `SizeFactory`

- **Назначение**: Управляет созданием размеров пиццы.
- **Методы**:
  - `create_size(size: str)`: Принимает название размера и возвращает его описание.

Класс `PizzaBuilder`

- **Назначение**: Собирает пиццу, используя ингредиенты и размеры.
- **Атрибуты**:
  - `ingredient_factory`: Экземпляр `IngredientFactory`, используется для получения ингредиентов.
  - `size_factory`: Экземпляр `SizeFactory`, используется для определения размера пиццы.
  - `pizza_type`: Тип пиццы.
  - `size`: Размер пиццы.
- **Методы**:
  - `__init__(...)`: Инициализирует экземпляры фабрик и тип пиццы.
  - `set_size(size)`: Устанавливает размер пиццы.
  - `build()`: Собирает и возвращает описание пиццы.

Функция `create_pizza()`

- **Назначение**: Создаёт заказ пиццы.
- **Возвращает**: Описание заказа пиццы.

Функция `main()`

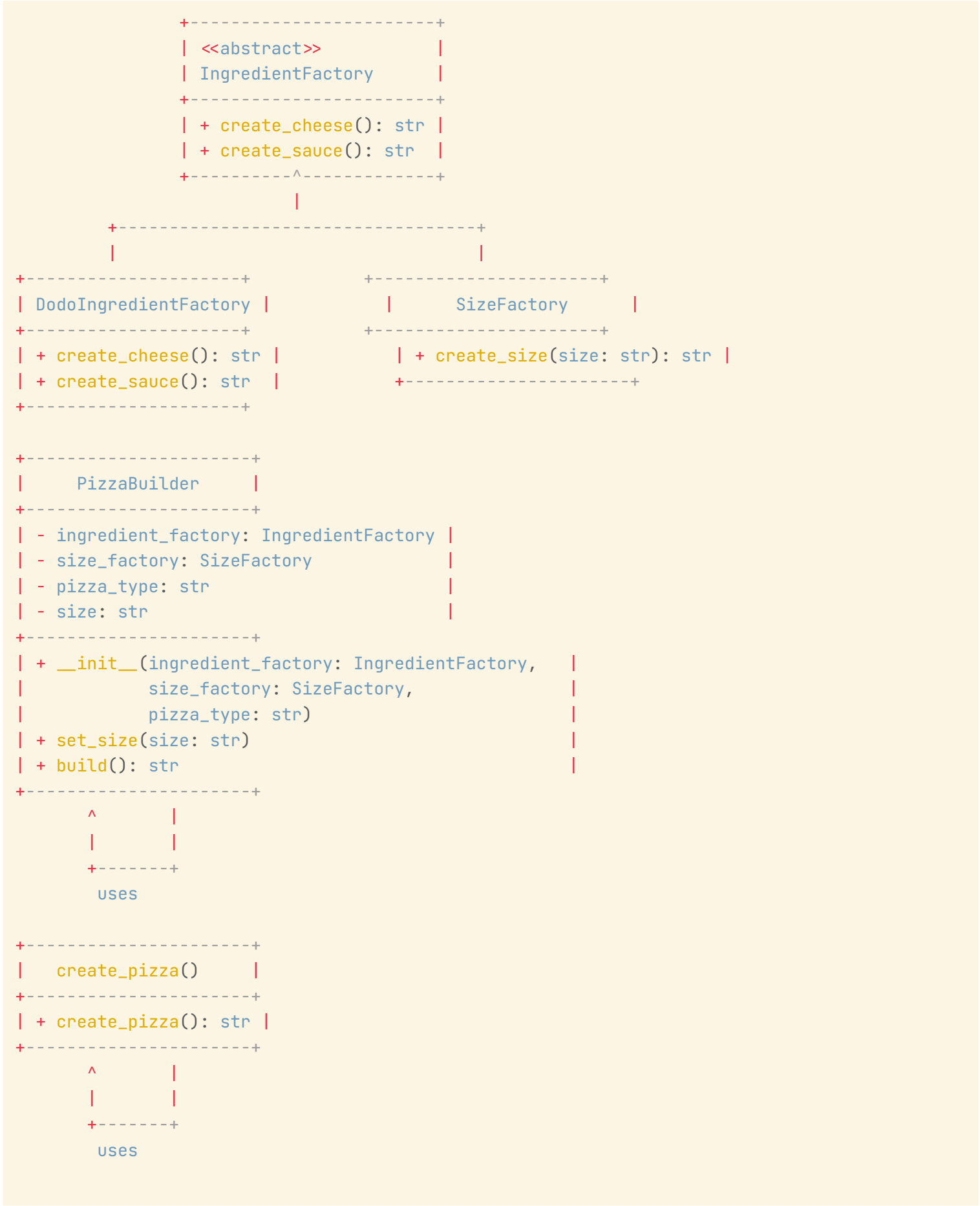
- **Назначение**: Точка входа в программу.
- **Действия**: Запускает процесс создания пиццы и выводит описание заказа.

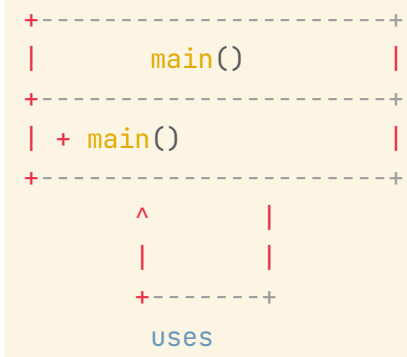
Отношения и Взаимодействия

- **Наследование**: `DodoIngredientFactory` является конкретной реализацией `IngredientFactory`.
- **Использование**: `PizzaBuilder` использует `DodoIngredientFactory` и `SizeFactory` для создания пиццы. Функции `create_pizza()` и `main()` используют `PizzaBuilder` для выполнения своих задач.

Это описание предоставляет более ясное понимание взаимосвязей между классами и их функциями в контексте системы заказа пиццы.

Вот текстовая UML диаграмма для вашего кода:





Объяснение Диаграммы

- **Классы и Наследование**: `IngredientFactory` является абстрактным классом, от которого наследуется `DodoIngredientFactory`. Это показано стрелкой с пустым концом, указывающей на базовый класс.
- **Методы**: В каждом классе указаны методы. Для абстрактных методов в `IngredientFactory` используется `+ create_cheese(): str` и `+ create_sauce(): str`.
- **Ассоциации и Зависимости**: Класс `PizzaBuilder` использует (`uses`) классы `DodoIngredientFactory` и `SizeFactory`. Это обозначено стрелками.
- **Функции**: `create_pizza()` и `main()` не являются классами, но включены в диаграмму, чтобы показать, как они используют классы.

Эта UML диаграмма предоставляет визуальное представление структуры и отношений между классами и функциями в вашей программе.

Критерии проверки 👌

1. **Корректность Реализации Паттернов**: Проверьте, правильно ли реализованы паттерны "Абстрактная Фабрика" и "Строитель". Абстрактные методы должны быть определены в `IngredientFactory`, а их конкретные реализации - в `DodoIngredientFactory`. `PizzaBuilder` должен корректно использовать эти фабрики для создания объектов.
2. **Гибкость и Масштабируемость**: Оцените, насколько легко в систему можно добавить новые виды пицц и ингредиентов. Хорошая реализация позволит это делать с минимальными изменениями в существующем коде.
3. **Качество Кода**: Код должен быть чистым, хорошо организованным и легко читаемым. Комментарии и документация к коду приветствуются.
4. **Работоспособность**: Приложение должно корректно принимать ввод пользователя и выводить ожидаемый результат - описание собранной пиццы.
5. **Обработка Ошибок**: Проверьте, как программа реагирует на некорректный ввод пользователя (например, неправильный размер пиццы).

Это задание поможет студентам глубже понять принципы проектирования и реализации паттернов "Абстрактная Фабрика" и "Строитель", а также научит их создавать гибкие и масштабируемые приложения.