**Отчёт по лабораторной работе: Применение паттерна "Стратегия"**

**1. Описание проблемы вашей предметной области**

Предметная область лабораторной работы связана с моделированием процесса покупки в онлайн-магазине "TSUShop". Пользователь должен иметь возможность выбирать товары (например, книги, канцелярские принадлежности, напитки) из предложенного списка, добавлять их в корзину и завершать покупку, выбирая способ оплаты: по полной цене или с применением скидки (процентной скидки или промокода). Основные требования к системе включают:

* Интерактивный выбор товаров с отображением текущего состояния корзины.
* Поддержка различных видов скидок: без скидки, процентная скидка (15%) и скидка по промокоду (500 рублей).
* Вычисление итоговой суммы с учётом выбранной скидки.

Проблема заключалась в необходимости обеспечить гибкость расчёта скидок, чтобы:

* Разные алгоритмы скидок (без скидки, процентная, по промокоду) могли применяться без изменения основного кода.
* Система оставалась расширяемой для добавления новых видов скидок в будущем (например, сезонные скидки).
* Логика расчёта скидок была отделена от управления корзиной, чтобы упростить поддержку и тестирование.

Без использования паттернов проектирования реализация такой системы могла бы привести к следующим сложностям:

* Жёсткая привязка логики скидок к классу корзины, что затруднило бы добавление новых скидок.
* Дублирование кода для обработки различных типов скидок.
* Усложнение поддержки, так как изменение одного алгоритма скидки могло бы повлиять на всю систему.

Целью было создать приложение, которое решает эти проблемы, оставаясь простым, модульным и полезным для реальных сценариев покупок.

**2. Решение: как используется паттерн "Стратегия" в вашем проекте**

Для решения описанной проблемы был применён паттерн проектирования **"Стратегия"**, который позволяет определять семейство алгоритмов (в данном случае — алгоритмы расчёта скидок), инкапсулировать каждый из них и обеспечивать их взаимозаменяемость. Это позволило отделить логику расчёта скидок от управления корзиной покупок, сделав систему гибкой и расширяемой.

**Структура решения**

Паттерн "Стратегия" реализован следующим образом:

* **Интерфейс IDiscountStrategy**:
  + Определяет контракт для всех алгоритмов скидок.
* **Конкретные стратегии**:
  + NoDiscountStrategy: Не применяет скидку, возвращает 0. Используется, когда пользователь выбирает оплату по полной цене.
  + PercentageDiscountStrategy: Вычисляет скидку как 15% от общей суммы. Подходит для акционных предложений.
  + PromoCodeDiscountStrategy: Применяет фиксированную скидку 500 рублей по промокоду "TSUSHOP2025", но не больше общей суммы. Используется для специальных акций.
* **Контекст (ShoppingCart)**:
  + Класс ShoppingCart управляет корзиной покупок и использует текущую стратегию скидки.
* **Класс Item**:
  + Хранит информацию о товаре: название (Name) и цену (Price).
  + Используется для представления товаров, таких как "Книга 'Лекции по основам программирования'" или "Шоколадка Аленка".
* **Класс Program**:
  + Реализует интерактивный интерфейс:
    - Показывает список товаров, позволяет пользователю добавлять их в корзину.
    - Завершает выбор товаров по вводу "0".
    - Запрашивает выбор скидки (1 — без скидки, 2 — 15%, 3 — промокод).
    - Создаёт соответствующую стратегию и передаёт её в корзину.

**Как работает**

1. Пользователь запускает программу и видит список товаров (например, "Ручка 'Tomsk State University' - 150 рублей").
2. Он вводит номер товара (1–5), чтобы добавить его в корзину, или "0", чтобы завершить выбор. После каждого выбора консоль очищается (Console.Clear()), показывается обновлённая корзина.
3. Если корзина не пуста, пользователю предлагается выбрать способ оплаты:
   * 1: Без скидки.
   * 2: Скидка 15%.
   * 3: Промокод "TSUSHOP2025" на 500 рублей.
4. В зависимости от выбора создаётся объект стратегии (NoDiscountStrategy, PercentageDiscountStrategy или PromoCodeDiscountStrategy), который передаётся в ShoppingCart через SetDiscountStrategy.
5. Метод GetFinalTotal использует стратегию для расчёта скидки, а GetDiscountDescription — для вывода её описания.
6. Итоговая сумма и содержимое корзины отображаются пользователю.

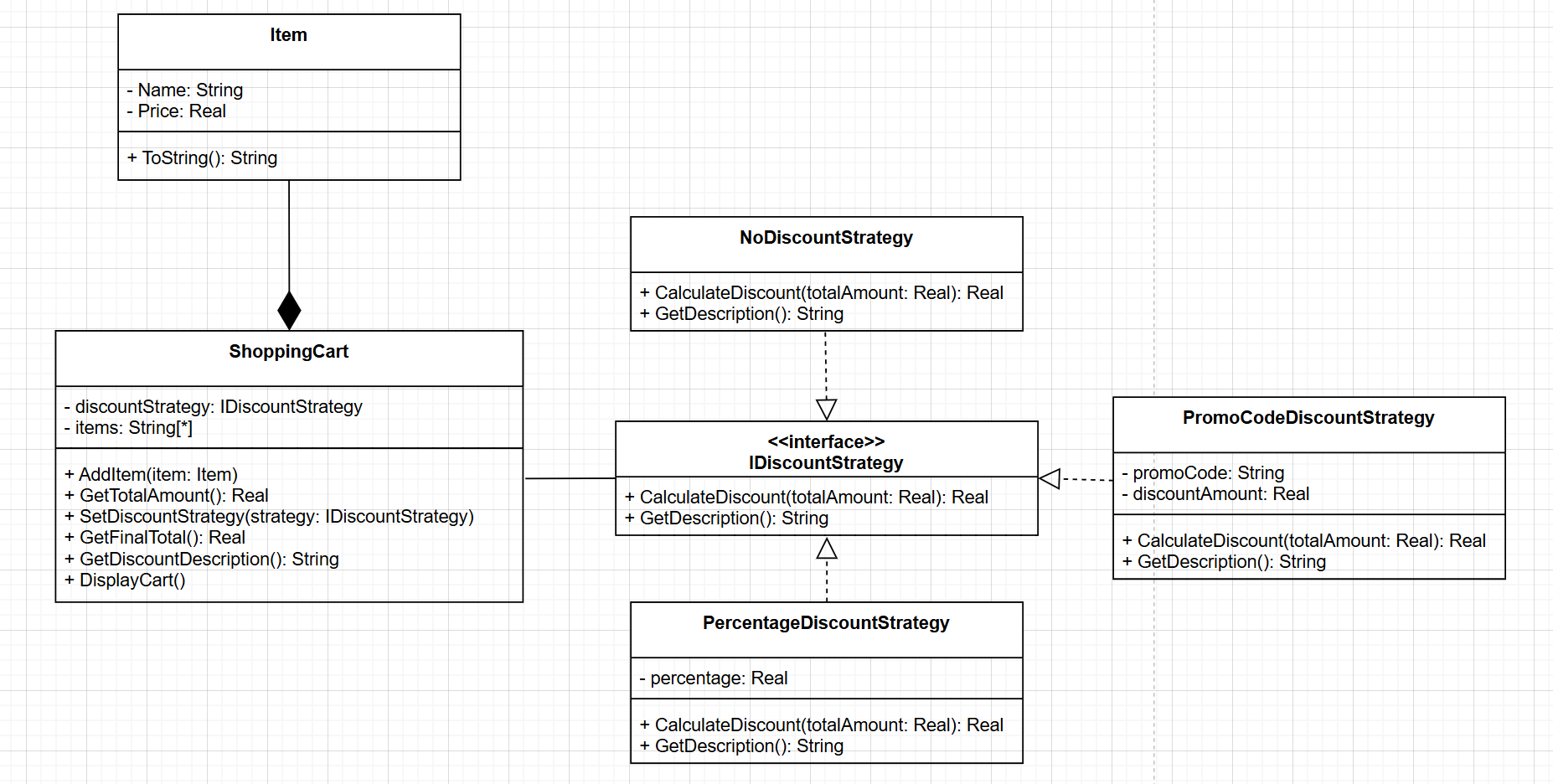
**Преимущества паттерна**

* **Гибкость**: Новая скидка (например, "Скидка 20%") добавляется созданием нового класса, реализующего IDiscountStrategy, без изменения ShoppingCart.
* **Модульность**: Логика скидок отделена от корзины, что упрощает тестирование и поддержку.
* **Переиспользуемость**: Стратегии можно использовать в других частях программы или даже в других проектах.

**3. Диаграмма классов для архитектуры приложения с применением паттерна**

Для наглядного представления архитектуры приложения ниже описана диаграмма классов с использованием паттерна "Стратегия".

**Описание диаграммы классов**

****Рисунок 1 – Архитектура приложения с паттерном "Стратегия"

На рисунке 1 изображена структура классов приложения, где паттерн "Стратегия" используется для разделения логики расчёта скидок от управления корзиной покупок.

**Связи**

* **Композиция**:
  + Между ShoppingCart и Item: Показывает, что корзина содержит множество товаров. На диаграмме — сплошная линия с заполненным ромбом у ShoppingCart, мультипликативность 1 у ShoppingCart и \* у Item.
* **Ассоциация**:
  + Между ShoppingCart и IDiscountStrategy: Указывает, что корзина использует одну стратегию скидки. На диаграмме — сплошная линия, мультипликативность 1 у обоих классов.
* **Реализация**:
  + Между IDiscountStrategy и классами NoDiscountStrategy, PercentageDiscountStrategy, PromoCodeDiscountStrategy: Показывает, что эти классы реализуют интерфейс. На диаграмме — пунктирная линия с пустым треугольником, направленная к IDiscountStrategy.

**4. Вывод: как внедрение паттерна повлияло на работу программы**

Применение паттерна "Стратегия" существенно улучшило архитектуру приложения и его возможности для дальнейшего развития. Основные эффекты:

1. **Гибкость и расширяемость**:
   * Добавление новой скидки (например, сезонной скидки 20%) требует лишь создания нового класса, реализующего IDiscountStrategy, и минимальных изменений в Program для поддержки нового выбора. В версии без паттерна пришлось бы изменять метод GetFinalTotal в ShoppingCart, добавляя новые условия в switch, что нарушает принцип открытости/закрытости.
2. **Модульность и читаемость**:
   * Логика каждой скидки инкапсулирована в отдельном классе (NoDiscountStrategy, PercentageDiscountStrategy, PromoCodeDiscountStrategy), что делает код понятным и удобным для поддержки. В версии без паттерна вся логика скидок сосредоточена в одном методе с условными операторами, что усложняет чтение и увеличивает вероятность ошибок при модификации.
3. **Разделение ответственности**:
   * Класс ShoppingCart отвечает только за управление корзиной и вызов стратегии, а сами алгоритмы скидок реализованы в отдельных классах. Это соответствует принципу единственной ответственности (Single Responsibility Principle). Без паттерна ShoppingCart должен был бы содержать всю логику скидок, что сделало бы его менее специализированным.
4. **Упрощение тестирования**:
   * Каждая стратегия может тестироваться независимо от корзины, что облегчает проверку корректности расчётов. В версии без паттерна тестирование сложнее, так как нужно проверять весь метод GetFinalTotal с разными ветками условий.

**Сравнение с версией без паттерна**

В версии без паттерна "Стратегия" логика расчёта скидок была встроена в класс ShoppingCart с использованием условного оператора switch. Это привело к следующим недостаткам:

* **Жёсткая привязка**: Добавление новой скидки требовало изменения метода GetFinalTotal, что увеличивало риск ошибок и усложняло поддержку.
* **Дублирование логики**: Описание скидок и их расчёт находились в одном классе, что смешивало ответственности.
* **Меньшая модульность**: Код был короче, но менее структурирован, так как вся логика скидок находилась в одном месте.

В версии с паттерном эти проблемы устранены:

* Новая скидка добавляется без изменения существующего кода корзины.
* Каждый алгоритм скидки изолирован в своём классе, что упрощает понимание и модификацию.
* Код более структурирован, что соответствует принципам объектно-ориентированного проектирования.

**Итог**

Паттерн "Стратегия" сделал приложение более гибким, модульным и готовым к расширению. Он позволил отделить алгоритмы расчёта скидок от основной логики корзины, что упростило добавление новых скидок и поддержку кода. В то же время функциональность осталась той же: пользователь может выбирать товары, применять скидки и видеть итоговую сумму, что делает приложение полезным для моделирования реальных покупок в российском онлайн-магазине "TSUShop". Применение паттерна повысило качество кода и его соответствие стандартам проектирования, что важно как для учебных, так и для практических задач.