Користени ШАБЛОНИ

main.py

Во кодот е искористен шаблонот за дизајн “Abstract Factory Pattern”, заедно со шаблонот "Strategy Pattern”. Еве како и зошто се применети овие шаблони:

1. Abstract Factory Pattern

Шаблонот се користи за динамичко инстанцирање на различни класи кои следат заеднички интерфејс. Во овој случај:

- Апстрактната класа DataFilter:
  - Дефинира заеднички интерфејс за сите филтри (execute методот).
  - Овозможува креирање на различни филтри без директно зависење од нивните конкретни имплементации.
- Клиентот (главниот дел од кодот):
  - Во главната функција __main__, се користи листа на филтри, кои динамички се инстанцираат и извршуваат преку повик на нивниот унифициран execute метод.
  - Ова ја олеснува замената или додавањето на нови филтри без промена на логиката во главната програма.

Придобивки:

- Флексибилност: Лесно додавање нови филтри без модификација на постоечкиот код.
- Скалабилност: Секој нов филтер само треба да ја наследи апстрактната класа DataFilter и да го имплементира методот execute.

2. Стратегија (Strategy Pattern)

Шаблонот "Стратегија" се користи за дефинирање на група алгоритми кои можат да се заменуваат едни со други за време на извршување. Во овој случај:

- Класите CompanyCodeFetcher и HistoricalDataFetcher:
  - Ги имплементираат различните стратегии за обработка на податоците.
  - Секоја класа претставува независен алгоритам за обработка (повлекување на кодови или историјат на податоци).
- Главниот дел од кодот (__main__):
  - Ги користи овие стратегии преку унифицираниот интерфејс DataFilter.

Придобивки:

- Разделување на одговорностите: Секој алгоритам е во своја класа, што го олеснува одржувањето и тестирањето.
- Динамична селекција: Филтрите се селектираат и извршуваат динамички, без да се менува главниот код.

Овој пристап комбинира два шаблона (Abstract factory и Strategy) за да постигне флексибилност, модуларност, и повторна употреба на кодот, што е одлично решение за скалабилни системи каде што алгоритмите за обработка на податоци може да се менуваат или додаваат во иднина.

---

app.py

Во кодот се користат неколку шаблони за дизајн (design patterns) за да се постигне модуларност, повторна употребливост и добра организација. Еве ги клучните шаблони кои се забележуваат и нивните причини за користење:

1. Factory Method Pattern

Каде се користи:

- Во класата DataLoader за методите load_company_codes и load_historical_data.

Објаснување:

- Шаблонот "Factory Method" се користи за создавање објекти или за иницијализација на ресурси, така што точната имплементација на креирањето е скриена од корисникот.
- Во овој случај, DataLoader овозможува лесно вчитување на податоците од CSV-датотеки без да се грижите за деталите како отварање на датотеките или ракување со грешки.
- Ова ја подобрува повторната употребливост и го намалува дуплирањето на код.

2. Singleton Pattern

Овој шаблон е индиректно имплементиран:

- Во класата PredictionModel, која управува со модели за предвидување.

Објаснување:

- Иако не е експлицитно имплементиран како "Singleton", оваа класа е дизајнирана така што сите предвидувачки модели (models) се зачувуваат во еден објект.
- Ова обезбедува дека инстанцата на моделите е централно управувана, што е корисно за избегнување конфликт помеѓу различни предвидувања.

3. MVC Pattern

Каде се користи:

- Во структурата на Flask апликацијата.

Објаснување:

- Flask имплементира основна форма на MVC:
  - Модел: Логиката за податоци и обработка е во класите како DataLoader, TechnicalAnalysis и PredictionModel.
  - Поглед (View): HTML шаблоните (на пример, index.html, company_data.html) претставуваат кориснички интерфејс.
  - Контролер: Рутите на Flask (/, /company, /technical-analysis, и др.) го поврзуваат моделот и погледот, управувајќи со барањата и одговорите.
- Овој шаблон го олеснува одвојувањето на логиката од презентацијата, што резултира со подобро организиран и лесно одржлив код.

4. Strategy Pattern

Каде е имплицитно имплементиран:

- Во техничката анализа и предвидувачките модели.

Објаснување:

- Различни стратегии за обработка на податоци (на пример, пресметување индикатори во TechnicalAnalysis или тренирање модели за 1 ден, 1 недела и 1 месец) се имплементирани како одделни функционалности.
- Ова овозможува динамичка селекција на методи за анализа или предвидување без промена на основната структура на кодот.

Овие шаблони за дизајн го прават кодот поорганизиран, лесен за разбирање и проширување. Тие помагаат во одвојувањето на одговорностите, овозможувајќи повторна употреба на функционалности и подобра поддршка за идни надградби.