**Примена на “Template” Метод**

Template Методот е behavioral (однесувачки) шаблон за дизајн, корисен кога имаме заедничка структура на алгоритам во неколку класи, но одредени чекори се разликуваат меѓу нив. Со сместување на фиксниот редослед на чекорите во еден главен метод (наречен „template method“), им дозволуваме на подкласите да ги променат само деталите кои се разликуваат.

**Како функционира:**

Креиравме апстрактна базична класа (во овој случај, BaseFilter), која содржи метода run() за целокупниот процес:

* + setup()
  + scrape\_data()
  + parse\_data()
  + save\_data()
  + call\_next\_filter()

1. Оваа метода run() е “template”т.е. шаблон — го определува редоследот на чекорите, но не ги конкретизира сите детали.
2. Подкласите (како Filter1, Filter2, Filter3) ги преформулираат (override) апстрактните или hook методи (scrape\_data(), parse\_data(), итн.) за да го извршат нивното специфично решение.
3. Овој template осигурува дека сите филтри следат ист општ тек, што го намалува повторувањето на кодот и придонесува за конзистентност.

**Зошто избравме Template Method:**

* Во нашиот проект, Filter1, Filter2 и Filter3 имаа слична постапка (превземи → парсирај → зачувај → премини на следен филтер), но со различни детали за scraping и saving. Со Template Method го обединивме процесот во BaseFilter.run(), а секој филтер го заменува само својот дел за scrape\_data() или parse\_data().
* Како последица имаме: помалку повторување на кодот, поедноставно одржување и појасен пристап при промена на редоследот на чекорите.

**Предности:**

* Намалено повторување: заедничките чекори се напишани само еднаш во базичната класа.
* Јасна структура: главната логика е во run()
* Лесно додавање нови филтри: само се наследува од BaseFilter и се прави override за потребните чекори.

**Заклучок:**

Template Method овозможува ред и одржливост кај „филтрите“, издвојувајќи **што** (алгоритамскиот тек) од **како** (конкретната имплементација) во секој чекор.