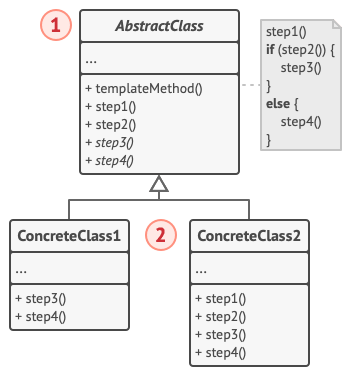
**TEMPLATE**

Metoda template este un design patter comportamental ce definește algoritmul ca un schelet de operații și lasă detaliile să fie implementate de clasele copil, structura generală și secvența algoritmului fiind păstrate în clasa părinte.

 Această metodă sugerează să împărțim algoritmul într-o serie de pași, transformarea pașilor în metode și să punem o serie de apeluri către aceste metode înăuntrul unei singure metode template. Etapele pot fi fie abstracte, fie pot avea o implementare implicită. Pentru a utiliza algoritmul, clientul trebuie să furnizeze propria sa subclasă, să implementeze toate etapele abstracte și să suprascrie unele dintre cele opționale, dacă este necesar (dar nu și metoda șablon în sine).

**Clasa abstractă** declară metode care acționează ca etape ale unui algoritm, precum și metoda șablon care apelează aceste metode într-o anumită ordine. Etapele pot fi fie declarate abstracte, fie pot avea o implementare implicită.

**Clasele concrete** pot suprascrie toate etapele, dar nu și metoda șablon în sine.

**Aplicabilitate:**

* Folosiți șablonul de proiectare a metodei template atunci când doriți să permiteți clienților să extindă doar anumite etape ale unui algoritm, dar nu întregul algoritm sau structura sa.
* Metoda template vă permite să transformați un algoritm monolitic (bine închegat) într-o serie de etape individuale care pot fi ușor extinse de subclase, menținând în același timp structura definită într-o superclasă.
* Când transformați un astfel de algoritm într-o metodă template, puteți muta etapele cu implementări similare într-o superclasă, eliminând duplicarea codului. Codul care variază între subclase poate rămâne în subclase.

Pro și contra:

* Putem lăsa clienții să suprascrie doar anumite părți dintr-un algoritm mai mare.
* Putem reduce duplicarea codului prin mutarea acestuia într-o superclasă.
* Anumiți clienți pot fi limitați de scheletul algoritmului furnizat.
* Metodele template cu cât au mai mulți pași cu atât tind să fie mai greu de întreținut.

Template se bazează pe moștenire, permite modificarea părților unui algoritm prin extinderea acestor părți în subclase, este static fiindcă funcționează la nivelul clasei. Acesta este întâlnit în viața de zi cu zi de exemplu în industria alimentară unde o rețetă de prăjitură poate fi văzută ca o clasă de bază abstractă în timp clasele concrete pot reprezenta diferite tipuri de prăjituri cum ar fi prăjitura cu fructe sau cu ciocolată.

**BIBLIOGRAFIE**

[**https://refactoring.guru/design-patterns/template-method**](https://refactoring.guru/design-patterns/template-method)

[**https://www.geeksforgeeks.org/template-method-design-pattern/**](https://www.geeksforgeeks.org/template-method-design-pattern/)