4. Класификация и приложни рамки

ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ

ГЛ. АС. Д-Р ЕМИЛ ДОЙЧЕВ

GoF класификация на шаблоните за дизайн

✓Според целта (какво прави шаблона):

- Създаващи шаблони (Creational).
  - Отнасят се за процеса на създаване на обекти.
- Структурни шаблони (Structural).
  - Занимават се с композицията на класове и обекти.
- Поведенчески шаблони (Behavioral).
  - Занимават се с взаимодействието между класове и обекти.

GoF класификация на шаблоните за дизайн

✓Според обхвата (върху какво се прилага шаблона):

- Шаблони за класове (Class patterns).
  - Фокусират се на връзките между класовете и техните наследници.
  - Занимават се с повторно използване чрез наследяване
- Шаблони за обекти (Object patterns).
  - Фокусират се на връзките между обектите.
  - Занимават се с повторно използване чрез композиция.

Създаващи шаблони (Creational Patterns)

- ✓ **Абстрактна фабрика (Abstract Factory).** Предоставя интерфейс за създаване на семейства взаимосвързани или зависими обекти, без да се задават конкретните им класове.
- ✓ **Строител (Builder).** Разграничава създаването на сложен обект от представянето му, така че един и същ процес да може да създава обекти с различни представяния.
- ✓ **Метод фабрика (Factory Method).** Дефинира интерфейс за създаване на обект, но позволява на подкласовете да решат кой клас да инстанцират. Метод фабрика дава възможност на класа да преотстъпи процеса на създаване на подкласове.
- ✓ **Прототип (Prototype).** Отделя видовете обекти посредством прототипен представител и създава нови обекти чрез копиране на този прототип.
- ✓ **Сек (Singleton).** Осигурява наличието на само един представител на даден клас и предоставя глобална точка за достъп до него.

Структурни шаблони (Structural Patterns)

- ✓ **Адаптер (Adapter).** Преобразува интерфейса на даден клас в друг такъв, очакван от клиента. Този шаблон дава възможност на класове да работят заедно в случаите, когато по принцип това е невъзможно поради несъответствие в интерфейси.
- ✓ **Мост (Bridge).** Разделя абстракцията от нейната имплементация, така че двете да могат да се променят независимо.
- ✓ **Композиция (Composite).** Композира обекти в дървовидни структури за пренасяне на йерархии от типа „части на цялото”. Този шаблон дава възможност за третиране по един и същ начин както на отделни обекти, така и композиции от обекти.
- ✓ **Декоратор (Decorator).** Допълнително разширява даден обект. Този шаблон дава гъвкава алтернатива на наследяването за разширяване на функционалността.

Структурни шаблони (Structural Patterns) – продължение

- ✓ **Фасада (Façade).** Предоставя унифициран интерфейс към набор от интерфейси в дадена подсистема. Този шаблон дефинира интерфейс от по-високо ниво, който улеснява употребата на подсистемата.

- ✓ **Миниобект (Flyweight).** Използва поделяне за ефективна поддръжка на големи количества малки обекти.

- ✓ **Пълномощно (Proxy).** Предоставя заместител или празна имплементация на друг обект, за да се контролира достъпа до него.

Поведенчески шаблони (Behavioral Patterns)

- ✓ **Верига отговорности (Chain of Responsibility).** Избягва обвързването на изпращача на дадена заявка с получателя и като дава възможност на няколко обекта да обработят заявката. Свързва заедно приемащите обекти и предава заявката по веригата, докато някой от тях я обработи.
- ✓ **Команда (Command).** Капсулира заявките във вид на обекти, така че клиентите да могат да се параметризират с различни заявки, да подреждат в опашката или да документират заявки, или да поддържат операции по отмяна на действие или възстановяването му.
- ✓ **Интерпретатор (Interpreter).** При даден език, дефинира представяне на граматиката му заедно с интерпретатор, използващ това представяне за превод на изречения на този език.
- ✓ **Итератор (Iterator**). Предоставя начин за последователен достъп до елементите на сложен обект, без да се разкрива същинското му представяне.

Поведенчески шаблони (Behavioral Patterns) - продължение

- ✓ **Посредник (Mediator**). Дефинира обект, капсулиращ взаимоотношенията в даден набор от обекти. Посредник съдейства за разхлабването на връзките, като не позволява на обектите да се обръщат изрично един към друг и дава възможност свободно да се променят взаимоотношенията им.

- ✓ **Спомен (Memento).** Без да нарушава капсулирането на данните, запазва вътрешното състояние на даден обект на външен носител, така че той да може да бъде възстановен в това състояние по-късно.

- ✓**Наблюдател (Observer).** Дефинира зависимост от типа „ Едно към много” между обекти, така че когато един обект промени състоянието си, всички зависими от него обекти да бъдат уведомени и обновявани автоматично.

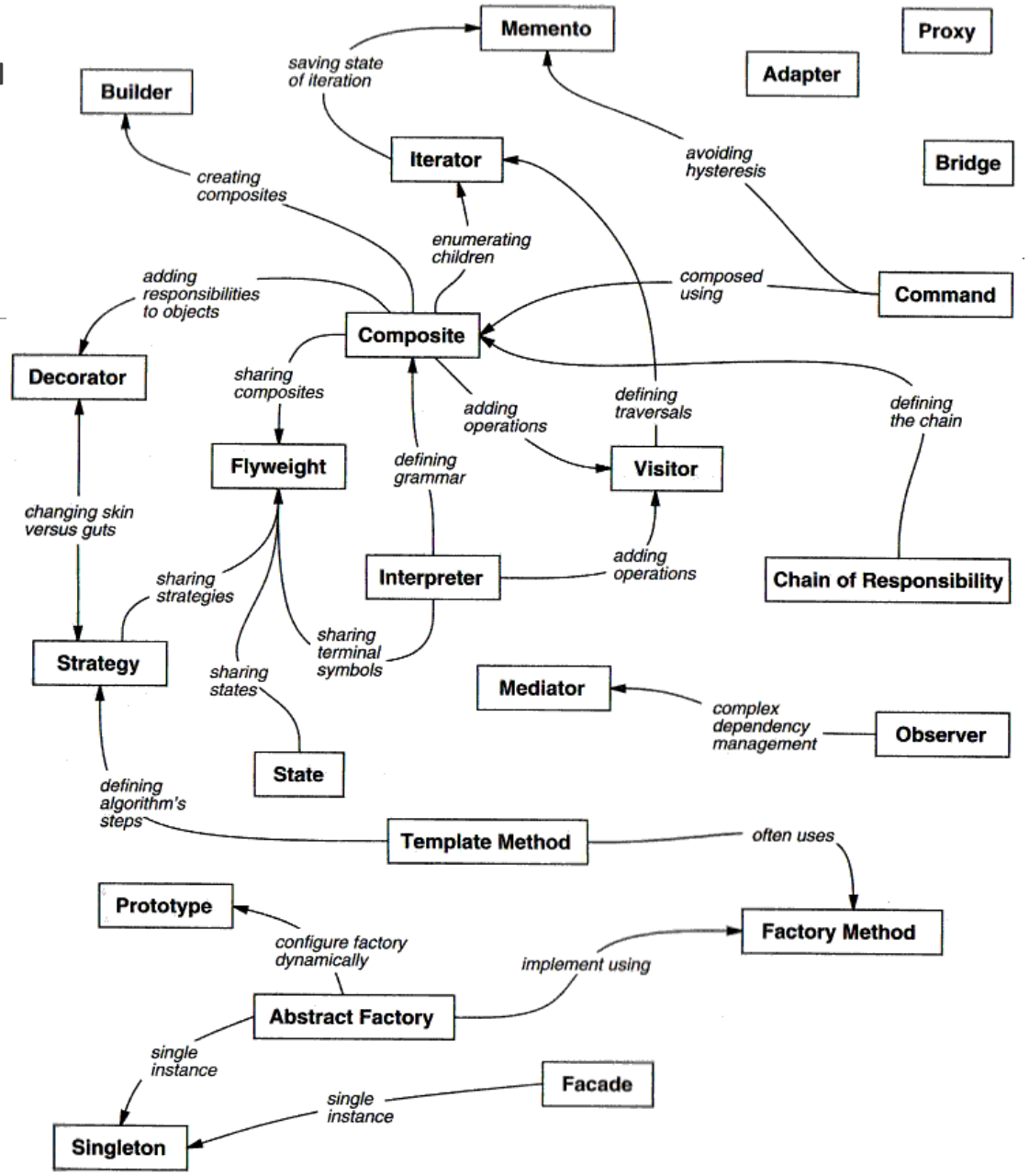
Поведенчески шаблони (Behavioral Patterns) - продължение

- ✓ **Състояние (State).** Позволява на даден обект да промени поведението си , когато вътрешното му състояние се променя. Отвън ще изглежда сякаш обекта се е превърнал в обект от друг клас.
- ✓ **Стратегия (Strategy).** Дефинира семейство алгоритми, капсулира всеки от тях и ги прави взаимозаменяеми. Този шаблон дава възможност на алгоритмите да се променят независимо от клиентите, които ги използват.
- ✓ **Шаблонен метод (Template Method).** Дефинира скелета на алгоритъма в операция, оставяйки някои стъпки на подкласове. Шаблонен метод позволява на подкласовете да предефинират определени стъпки от даден алгоритъм, без да се променя структурата им.
- ✓ **Посетител (Visitor).** Представя операции, които се извършват върху елементите на обектна структура. Този шаблон дава възможност да се дефинира нова операция, без да се променят класовете на елементите, върху които работи тя.

GoF класификация на шаблоните за дизайн

		Цел		
		Създаващи	**Структурни**	**Поведенчески**
Обхват	**Клас**	Метод фабрика	Адаптер (клас)	Интерпретатор
				Шаблонен метод
	Обект	Абстрактна фабрика	Адаптер (обект)	Верига отговорности
		Прототип	Декоратор	Итератор
		Сек	Композиция	Команда
		Строител	Миниобект	Наблюдател
			Мост	Посетител
			Пълномощно	Посредник
			Фасада	Спомен
				Стратегия
				Състояние

Класификация според връзките между шаблоните



Приложни рамки (frameworks)

- ✓ В повечето случаи компонентите се разработват като част от някаква рамка (framework). Рамката помага да се изгради цялостна система от отделните компоненти.
- ✓ Рамките се проектират да решат специфичен набор от проблеми или да направят решаването им по-лесно.
- ✓ Най-често рамките съдържат основен набор от компоненти, който предоставя основни решения в проблемната област.
- ✓ Освен това съдържат и набор от базови класове и интерфейси, които могат да бъдат наследявани/имплементирани за да се постигне специфична функционалност.

Приложни рамки (frameworks)

- ✓ Има няколко вида рамки:
  - **Приложни рамки** (application frameworks) – използват се от различни видове приложения.
  - **Домейн рамки** (domain frameworks) – създадени са за решаване на проблеми в конкретна област.
  - **Помощни рамки** (support frameworks) – предоставят услуги на системно ниво.

Приложни рамки (frameworks)

- ✓ Пример за проста приложна рамка е Java Servlet API. Тя предоставя:
  - Множество от помощни класове, от които рамката създава обекти и ги предоставя за директно използване. Например: `session` и `request` обектите.
  - Множество от класове от високо ниво, които имплементират базова функционалнст. Например: `GenericServlet` и `HttpServlet`.
  - Множество от интерфейси, които специфичните за приложението класове могат да имплементират за да добавят допълнителна функционалност. Например `SingleThreadModel`.
- ✓ Цялата функционалност се имплементира вътрешно от сървлет контейнера.

Приложни рамки (frameworks)

- ✓ Предимствата на разработване базирано на рамки е:
  - Разработване на решения в съответната проблемна област е лесно.
  - Качеството на крайния продукт се повишава.
  - Времето за разработване се намалява.
  - Промените могат да бъдат направени бързо и удобно.
  - Цената за разработване се намалява.
  - Има висока степен на многократна употреба на вече създаден код.

Край: Класификация и приложни рамки

ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ