Класификация на шаблоните за дизайн

[ **КЛАС** | **ОБЕКТ** ] - спрямо обхват

[ **СЪЗДАВАЩИ** | **СТРУКТУРНИ** | **ПОВЕДЕНЧЕСКИ** ] – спрямо цел

***Създаващи:***

- Фабрика (Factory, *aka* Виртуален конструктор [Virtual Constructor])  
 - Абстрактна фабрика (Abstract Factory, *aka* Комплект [Kit])  
 - Прототип (Prototype)  
 - Сек (Singleton)  
 - Строител (Builder)  
***Структурни:***  
 - Адаптер (Adapter, *aka* Обвивка [Wrapper])  
 - Декоратор (Decorator, *aka* Обвивка [Wrapper])  
 - Композиция (Composite)  
 - Миниобект (Flyweight)  
 - Мост (Bridge)  
 - Пълномощно (Proxy, *aka* Заместник, *aka* Сурогат [Surrogate])  
 - Фасада (Facade)  
***Поведенчески:***  
 - Интерпретатор (Interpreter)  
 - Шаблонен метод (Template Method)  
 - Верига отговорности (Chain of Responsibility)  
 - Итератор (Iterator, *aka* Курсор [Cursor])  
 - Команда (Command, *aka* Действие [Action], *aka* Транзакция [Transaction])  
 - Наблюдател (Observer, *aka* Зависимост [Dependents], *aka* Publish-Subscribe)  
 - Посетител (Visitor)  
 - Посредник (Mediator)  
 - Спомен (Memento)  
 - Стратегия (Strategy, *aka* Policy)  
 - Състояние (State)

В много обектно-ориентирани системи има повтарящи се примери за класове и комуникиращи по между си обекти – това са т.н. *шаблони*, които разрешават специфични проектантски проблеми, правят обектно-ориентирания проект: по-гъвкав, по-елегантен и с повече възможности за повторна (многократна) употреба.

*Шаблонът за дизайн* (проектиране) систематично именува, разяснява и оценява важен и повтарящ се дизайн в обектно-ориентираните системи.

Улесняват многократната употреба на успешни дизайни и архитектури.

Документирането им ги прави по-достъпни за разработчиците на нови системи.

Спомагат за правилния избор на дизайнерски алтернативи, които правят една система годна за многократно използване, и отхвърлят онези, които пречат на тази цел.

Подобряват документацията и поддръжката на системата, като въвеждат изрична спецификация на взаимодействията между класове и обекти.

Шаблоните документират често срещат проблем и неговото решение в определен контекст.

Шаблоните правят връзката между проблеми и решения.

Дефиниция на термините:

- *Контекст* – среда, обкръжение, ситуация или взаимосвързани условия, при които нещо съществува.

- *Проблем* – отворен въпрос; нещо, което трябва да бъде изследвано и решение; обикновено проблемът е ограничен в рамките на контекста.  
 - *Решение* – отговор на проблема, в рамките на контекста, който помага за разрешаването му.

Дефиниции за шаблон:

„Всеки шаблон представлява правило, състоящо се от три части, които дават връзката между някакъв контекст, проблем и решение“  
 - Кристофър Александър

„Всеки шаблон представлява правило, състоящо се от три части, които дават връзката между някакъв контекст, някаква система от условия, които се появяват често в този контекст, и някаква софтуерна конфигурация, която позволява тези условия да бъдат изпълнени.“  
 - Ричард Гейбриъл

„Шаблонът е идея, която е била полезна в един конкретен контекст и вероятно може да бъде използвана и в други.“  
 - Мартин Фаулър

Най-често използваните категории шаблони са:

* + Шаблони за проектиране (Design patterns)
  + Шаблони за архитектура (Architectural patterns)
  + Шаблони за анализ (Analysis patterns)
  + Създаващи шаблони (Creational patterns)
  + Структурни шаблони (Structural patterns)
  + Поведенчески шаблони (Behavioral patterns)

Според целта (какво прави шаблона):

* + Създаващи шаблони (Creational) – занимават се с процеса по 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.

Според обхвата (върху какво се прилага шаблона):

- Шаблони за класове (Class patterns):  
 \* фокусират се върху връзките между класовете и техните наследници  
 \* занимават се с повторно използване чрез наследяване  
 - Шаблони за обекти (Class patterns):  
 \* фокусират се върху връзките между обектите  
 \* занимават се с повторно използване чрез композиция

**НАКРАТКО**

***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 класовете 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 тупа „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.

\* Фасада (Facade) – предоставя стандартизиран 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 от 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 така дава възможност на няколко обекта да обработят заявката. Свързва заедно приемащите обекти и предава заявката по веригата, докато някой от тях не 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.  
  
\* 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.  
  
\* 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 взаимнозаменяеми. 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.

Разлика между шаблоните ***Factory*** и ***Abstract Factory***:

Шаблонът ***Factory*** използва наследяване и разчита на подкласовете да управляват изисканата инстанция на обект, докато при ***Abstract Factory*** класът отстъпва отговорността на инстанцирането на обект на друг обект посредством композиция.