The *GoF Patterns* can be presented using real-life examples. This is useful for educational purposes and might even evoke a geeky chuckle as well.

A personal assistant is a *Proxy* and a pair of glasses is a *Decorator*. Use real-world metaphors to ramp up on your design pattern education.

### Creational Patterns

#### Abstract Factory (87)

- Imagine a manufacturer producing interior design parts — lights, cabinets, tables. The company produces each part in two styles and has a factory for each style: a Traditional Asian Factory (*ConcreteFactory1*) and a Contemporary European Factory (*ConcreteFactory2*). Both provide exactly the same part types (e.g., a Door; *AbstractProductA*), but using completely different methods to arrive at completely different styles (asked for a "Door", the Traditional Asian Factory produces an oak door, *ProductA1*, while the Contemporary European factory produces a steel door, *ProductA2*).

At any time, a *Client* will either be designing a house in one of the two styles, so the client will simply choose one factory and all requests will go there. The expectation and requests will be exactly the same whether the client has chosen Asian or European, with style-independent requests like "I'll have three Desks and four Doors for each".

#### Builder (97)

- In manufacturing, a *Director* manages the overall process (e.g., "get parts, run machine, test, deliver, bill") and delegates product-specific detail to *Builders*. Detailed Builder Example online.

#### Prototype (117)

- Dolly the Sheep, amoeba, and other clones. The researcher (*Client*) "requested" Dolly's mum (*ConcretePrototype1*) to produce a clone of herself.
- In a word processor, cut-and-paste some text. You now have a separate, identical, copy, which you can modify.

#### Singleton (127)

- A monotheistic viewpoint argues God is a *Singleton* - to quote the Highlander movie about a completely underlated topic, "There can be only One".
- Scientists also have their version of *Singleton*: this universe and its universal laws.
- Postmodern family structures aside, a child may view *Father* as a singleton and *Mother* as another *Singleton* (one of each, even though both belong to the same superclass).

### Structural Patterns

#### Adapter (139)

- The name is a giveaway: a power adapter. A machine (*Target*) has a slot to provide power. There is a power point (*Adaptee*). You know the power point does what the machine needs, but they don't "know" about each other, i.e., their creators did not directly design with the other in mind. Furthermore, you can't change either of them. The solution? Stick an Adapter in between them.
- Someone who translates between Japanese and English is performing an *Adapter* role (I won't call it an Interpreter, which is another pattern). When the Emperor of Japan gets together with the President of the USA, neither has the skill to speak in the others' language, and you can't change that! So you stick an Interpreter (*Adaptor*) in between them to pass messages between the two in terms they understand.

#### Bridge (151)

- Utility industries such as telecommunications and power often follow the *Bridge* pattern. A regulator sets standards to allow interoperability and ensure quality-of-service (*Abstraction*), while companies (*Implementors*) are flexible to use whatever means to satisfy this standard "interface".
Composite (163)

Note: Most recursive structures are examples of Composite.

- Many organisations structure their personnel around Composite patterns. When a manager is requested to "report progress", he "loops" through each worker, requesting them to "report progress", and combines the results.

Decorator (175)

- Many humans wrap Decorators around themselves to enhance their input. Put on a pair of glasses, for example, to intercept incoming light and provide modified light to the eyes. The eyes still get the same kind of stuff - i.e. a stream of light into the eyes - only the content of this stuff has changed. A Decorator decorates - or transforms - information, but it always outputs the same type as came in. A pair of glasses would not be a Decorator if it converted light to sound or smell. The eyes take in light, and will perform the same procedures with light whether or not the Decorator is present. This "transparent" property of Decorator is what allows Decorators to be chained together.
- Terminator 2 demonstrates how we might one day enhance our vision even further by slapping on a set of augmented reality glasses. The guy in front of you - is he a predatory robot from the future that you have been sent to destroy? What's his model number? Any chance he could one day become a state governor? The augmented overlay answers all this and more.
- You can decorate output too. Singers without musical ability, such as boy bands and female "performers", use Decorators to filter their voices, leading to something more closely resembling music. The interface, i.e. an audio stream, remains the same, but the content undergoes a radical improvement. It is common with musical instruments to apply a chain of Decorators (low-pass filters, high-pass filters, etc.) which the sound passes through.

Facade (185)


Flyweight (195)

- Some companies maintain a pool (Flyweight Factory) of flexible software developers (Flyweights), instead of keeping developers always assigned to certain projects. When a project needs work, one of the developers (ConcreteFlyweight) is "awoken", briefed, and set to work for a while, before being returned to the pool. The benefit is the programmers can be shared across different projects; the downside is a more complex process, as well as the overhead of frequent briefing sessions. (Comment: this mode of working would fit better in Low Context societies.)

Proxy (207)

- A CEO's personal assistant (PA) acts as a Proxy when negotiating an appointment time. The PA provides the same interface as the CEO would, (e.g. discussing the time on the telephone), and may or may not delegate some questions to the CEO. But anyone seeking an appointment may only talk to the PA and never to the CEO.

Behavioral Patterns

Chain of Responsibility (223)

- Helpdesks use several lines of support. An internet user calls her ISP to complain about the slow connection. The computerised answering system offers some generic advice, and, since that doesn't work, she holds on and a helpdesk operator in a faraway country eventually picks up and tries to solve the problem by asking her to reboot. Unable to suggest anything else useful, the operator delegates responsibility to his boss. The boss asks if she has recently changed modem, but unable to suggest anything else useful, delegates responsibility to the system administrator in the user's local call centre. The system administrator explains to her that the lines have been slow in her area due to an accident. The issue is closed, and the chain of responsibility need proceed no further.
- The chain can also work from senior to junior. When someone knocks on the door during a Young Ones episode, the task of answering is delegated from Mike, the ringleader, down to whipping boy Neil:
  MIKE: "There's someone at the door, Rick!"
  RICK: "There's someone at the door, Vyvyan!"
  VYVYAN: "There's someone at the door, Neil!"

  (In passing, I should note my pleasure in finally being able to reference the "Bachelor Boys" in a software engineering context.)
Command (233)

- A General writes down a "Strike at Dawn" directive (Command) for his Lieutenant, with an "Instructions" section (execute() method). To streamline the process, all directives are standardised, with a single Instructions section, so it can be assumed that the Lieutenant can execute any Command he is supplied.

Interpreter (257)

- Human languages can be represented in the same way as the computer languages that this pattern helps to parse. For instance, the English sentence, "John stubbornly coded." is a NonterminalExpression consisting of one TerminalExpression, "John", and one NonTerminalExpression, "stubbornly coded".

Iterator (257)

- You sit down at a full dinner table and the host wants to introduce you to all the guests - what order does he use? In some circumstances, he will simply go around the table; but if the Overlord Dictator of the Central Galaxies Concern happens to be sitting somewhere near the middle, perhaps he will prefer to introduce in order of importance. The collection of people has a fixed order, but the iteration strategy is flexible.

Mediator (273)

- A stock exchange acts as a Mediator. It mediates among public companies, investors, government regulators, tax agencies, and others (Colleagues). This limits (and sometimes eliminates), direct interaction between these parties, so they are not very tightly coupled to each other.

Memento (283)

- You create a simple calculator with Undo capability. It is not enough to reverse operations, because some cause loss of information. For example, the user types "6" then multiplies by "0". The current value is now "0" and the previous operation was "multiplyBy(0)". Not enough information to get our "6" back if the user wants to Undo. The solution is to retain a Memento for unreversible (and maybe more) operations.
- You (Caretaker) are about to head abroad and move into your wealthy parents-in-law's vacation home for six months (Originator state). You don't really care what it's like now - all that matters is that it must be restored to the same state when you leave. Before arriving, you arrange for the house butler (Originator behaviour) to take a photograph (Memento) so the place can be restored later, and to send them to you for safekeeping. You retain the photos and upon leaving, give them to the butler so he can restore the state of the place. Note that you procured and stored the Memento, but never had to look at, or use, its contents.

Observer (293)

- People (Observers) register to discover when a website (Subject) has been updated. A change to the website triggers a brief email to all people who have registered simply telling them the website has updated. If they want, they can see how it changed by surfing to the website and inspecting its contents. This is a Pull model because only a change notification is sent; the observer still has to inspect the subject to discover its new state.
- People register themselves as magazine subscribers (Observer). Then, each magazine (Subject) is sent to anyone who has registered as a subscriber. This is a Push model because the entire magazine is sent; no further inspection is required.

State (305)

- People's behaviour depends heavily on their emotional state (State), even when all other factors are the same. If an angry driver is overtaken, he might treat it as an invitation for a spontaneous game of Grand Prix. If the same driver is in a good mood, he might chuckle and ask why some people are in such a hurry. The stimulus (input data) is the same in both cases, i.e. how do you respond to a driver overtaking you? It is your current State that dictates the response.

Strategy (315)

- A recipe is the archetypal Strategy. Whether for Mississippi Mud Cake or Sauteed Giraffe Steak, the structure and "interface" remains the same - you can query a menu for ingredients, required appliances, and the steps to make it.
Template Method (325)

- You're sick of programming and decide to become a lion-tamer. You must first sit an "animal-tamer" course, where you study alongside would-be wolf-tamers and elephant-tamers. You each learn the generic technique for associating an action with a command: (1) wait for the animal to do something interesting, (2) shout the corresponding command, (3) reward the animal. Another generic technique involves building a rapport with the animal: (1) remain at a distance, (2) wait for the animal to approach.

When you finish the course, you are an "abstract animal-tamer". You have lots of generic techniques but you must plug in the gaps with lion-specific information. For instance, how do "reward the animal?". For a lion, you reward it by offering a nice animal to eat. For an elephant, the answer will be completely different - who knows, maybe you give the elephant an end-of-year bonus? In any event, "reward the animal" is a template method. It is referenced and relied upon by the high-level animal taming process, but cannot actually be stated at that generic level. It can only be stated in a manner specialised for each individual animal to be tamed.

Visitor (331)

- Imagine you manage a ThemePark (ObjectStructure) with plenty of Attractions (Elements) - Rides, VideoGames, Restaraunts (ConcreteElements). Sometimes, you need to arrange a visit to all Attractions.

For example, one type of "Visit Everywhere" would be to collect cash once every two hours - performed by a CashCollectingVisitor (ConcreteVisitor). Another type would be to observe staff behaviour - performed by a covert StaffObservingVisitor (a second ConcreteVisitor), maybe every few months. These Visitors perform different tasks, but share a common problem: how to traverse the park to visit all Attractions. Thus, each Visitor is imbued with a mechanism for visiting all Attractions, such as an ordered list or a set of heuristics.

Each Visitor then differs in how it approaches the various Attractions - so there is a concrete procedure for each combination of (Visitor, Attraction). For example, what happens when CashCollectingVisitor visits a VideoGame (CashCollectingVisitor.visitVideoGame())? The concrete procedure is to open the game with a key and collect the cash. What does StaffObservingVisitor do in the same situation (StaffObservingVisitor.visitVideoGame())? In this case, the concrete procedure is to do absolutely nothing. But on visiting a Restaurant (StaffObservingVisitor.visitRestaraunt()), she will be very busy, as the procedure here is to surreptitiously make an order, sit down and make notes while sipping a cafe latte.

Background - About the Pattern Metaphors

The GoF Patterns are at once utterly powerful and absurdly simple. Yet, my experience indicates many pattern novices grasp neither their power nor their simplicity. A likely reason is lack of experience with object-oriented architectures. Many pattern learners are students who have not felt the pain of a complex system that has been mis-architected. Many others have sadly worked on systems so mis-architected that the confused tangle of code offered no affordances as to where patterns might prove useful.

It's useful to complement software design patterns with stories from the real-world. Many of the GoF patterns do apply to real-world structures - structures of human organisations, work practices, buildings, towns, and so on. People learning patterns can relate to these things, so why not exploit the associations to improve learning? Amazing how even expert software developers are prone to discuss ideas in terms of real-life metaphors. In fact, some of the best architects I know frequently anthropomorphise their software when explaining how it works: "So this guy sends the message to him, but he's busy so he says 'send it later, or send it to one of my delegates now if you want'."

Here, I present examples of real-world metaphors based on the GoF patterns. A good exercise for learners is to think of your own. In a learning context, the patterns can even be acted out. It might seem a bit funny, but activities like that are examples of elaborate learning, meaning you remember them.

For ease of reference, the patterns are listed in the same order as the text. However, if you are learning the patterns, I recommend learning the patterns in this more effective order described at http://mahemoff.com/paper/software/learningGoFPatterns/