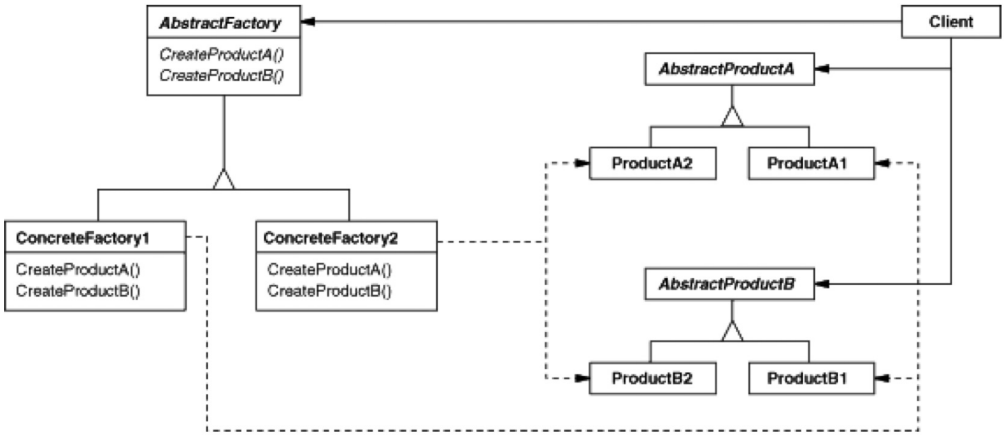
Designs Patterns

Abstract Factory:

Used to separate the family object building from the actual implementation of the programn, making it transparent and remaining the same.

Usually the object implementation is at the start of the programa

▼ Structure



Possible use cases:

- Separate Two or more families of UI Elements, depending of certain conditions.
- The paid and free version separation, of an application.
- Distinguish between certain format options

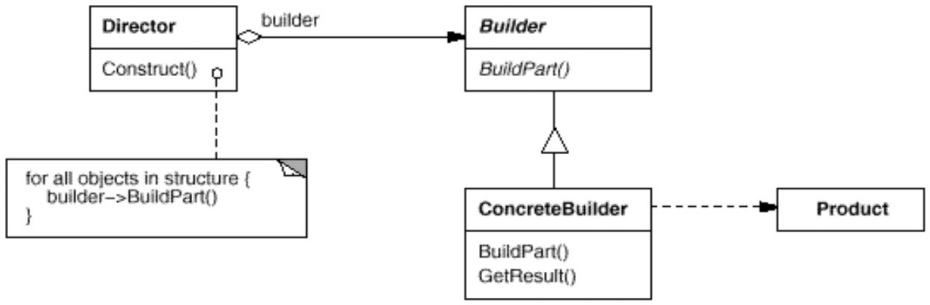
Implementation:

https://github.com/alexsotocx/design_patterns/tree/master/AbstractFactory

Builder Pattern:

Used to create complex objects in a structured way and produce a result. It separates the construction from its representation

▼ Structure



112

It garantees the construction of a valid and usable object.

Used to:

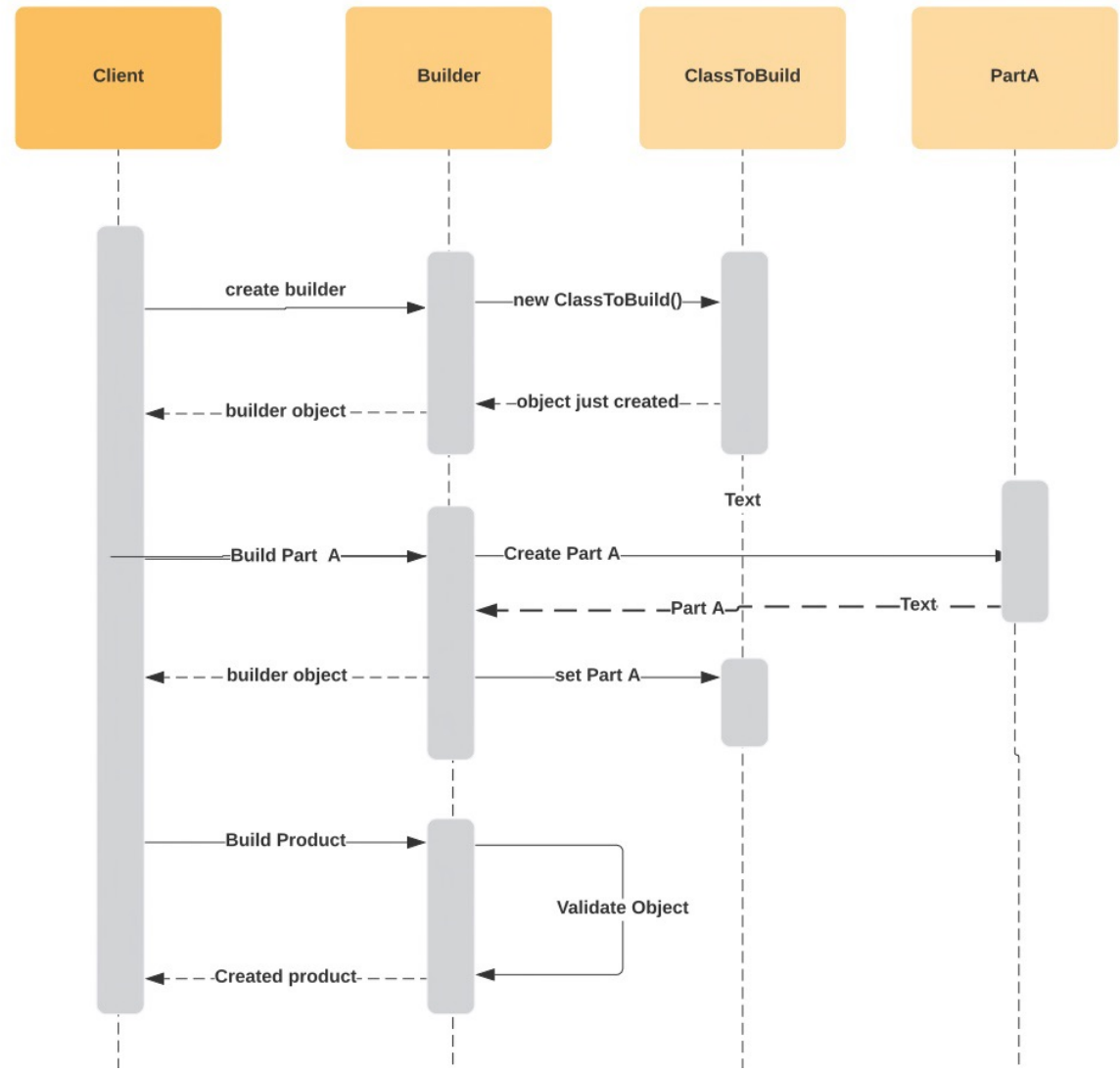
- Allow multiple representation from the same data using the same algorith.
- Make the algorithm independent from the actual result creation

Possible usages:

- Signal conversions (Format conversion in general).
- Hide constructor of a class in a library

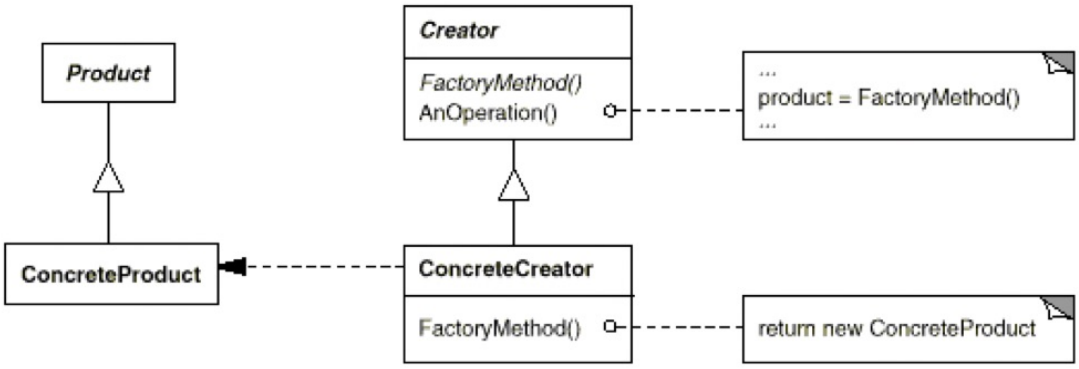
Interactions:

The client is in charge of creating the builder, with it, the client will select dinamically or statically the parts (components) to build, while the builder class will keep the context and it will build the target class to build.



Factory pattern:

- Used to hide a complex object creation or dynamically create objects of certain type.
- A Factory Function decides to create a new object, return an already created one or clone a certain prototype and return it.
- Encapsulates and provides a polymorphism so the code is not tied to something concrete.
- Structure:



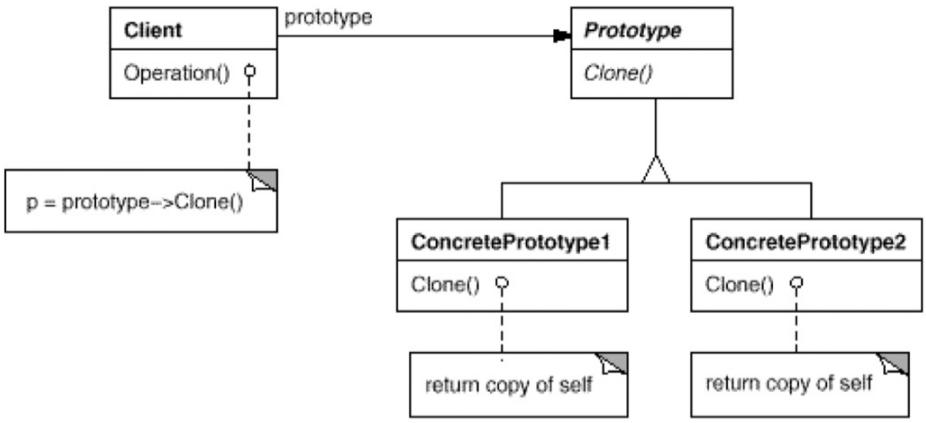
Use it when:

- Dynamic object creation. Makes the program decide which instances to create on realtime.
- Create singleton objects.

Prototype Pattern

* Used to decuple the object creation from its representation
* Reduce the number of subclases by cloning already predefined ones.
* Enable "dynamic" creation of "clases" (composition) for later reuse

▼ **Structure**



- The prototype works by cloning predefined classes, modifying its state, helping in that way the creation of new object with new behaviours.
- Can be used with the composition pattern, but the class needs to asure the deep clone.

Singleton pattern

Used when an object of an specific class needs to exist just one time during the runtime,

the singleton pattern in itself, is an application of the Factory method. Where the creator method checks if an object already exist or create it and returning it.

- Allows subclassing,
- More usable than class static methods, because it can have state
- "Don't" popula

▼ Structure