

DESIGN PATTERNS

27.10.2010 – ESE Exercise

WHAT ARE DESIGN PATTERNS

- A description or template for how to solve a problem that can be used in many different situations.
- Typically describe relationships and interactions of classes or objects
- More than just a class diagram, depends on context, force, etc.
 - «A solution to a problem in a context»
 - Context refers to a set of situations in which the pattern applies.
 - Problem refers to a set of forces - goals and constraints - that occur in this context.
 - Solution refers to a design form or design rule that someone can apply to resolve these forces.

WHAT DESIGN PATTERNS ARE NOT

- An actual implementation in a specific programming language
- An idiom, an algorithm, a UML Diagram, a use case, etc.
 - A design pattern may describe how and why in a certain context one of the above is an appropriate solution

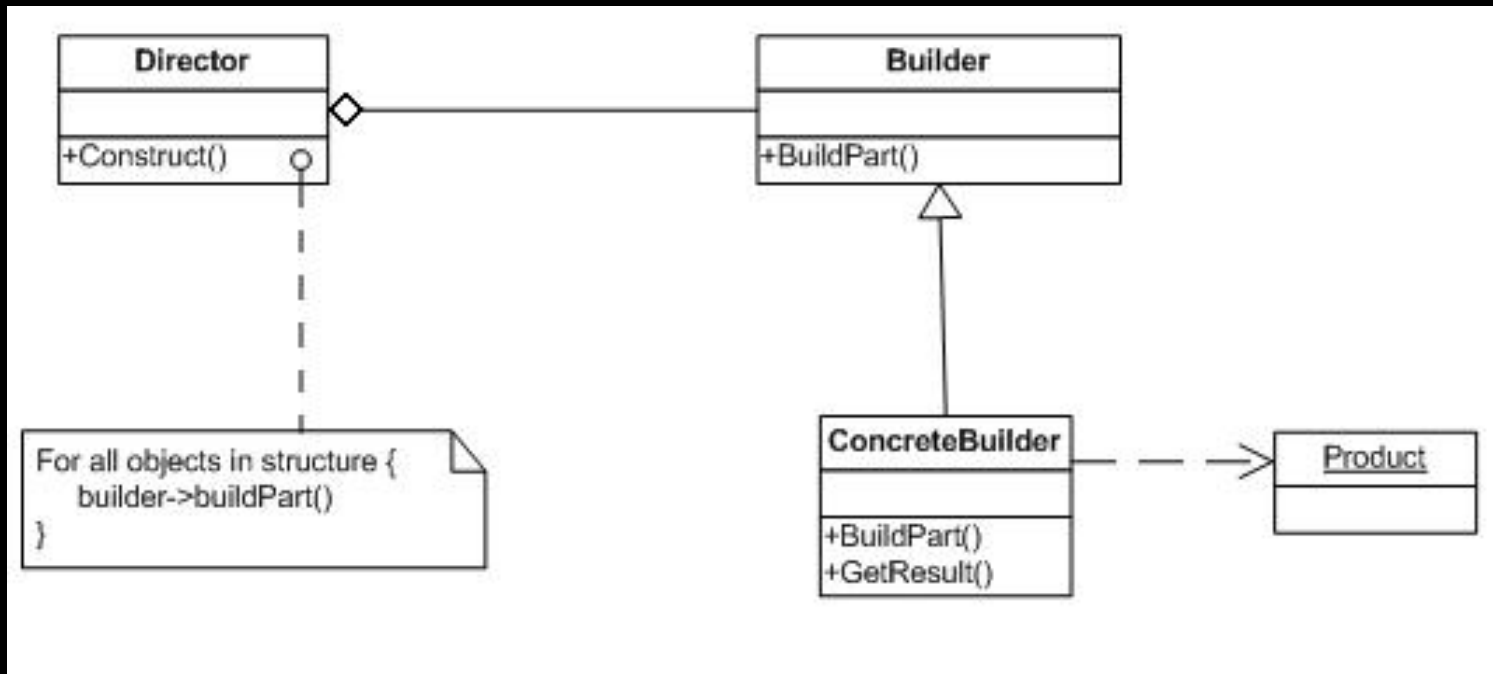
SOME DESIGN PATTERNS

- Factory / Builder
- Singleton / Multiton
- Object Pool
- Adapter / Wrapper
- Composite
- Proxy
- Iterator
- Observer
- Null Object
- Visitor
- Command

BUILDER / FACTORY PATTERN

- When to use it:
 - Complex construction of objects
 - Different implementation can create different objects

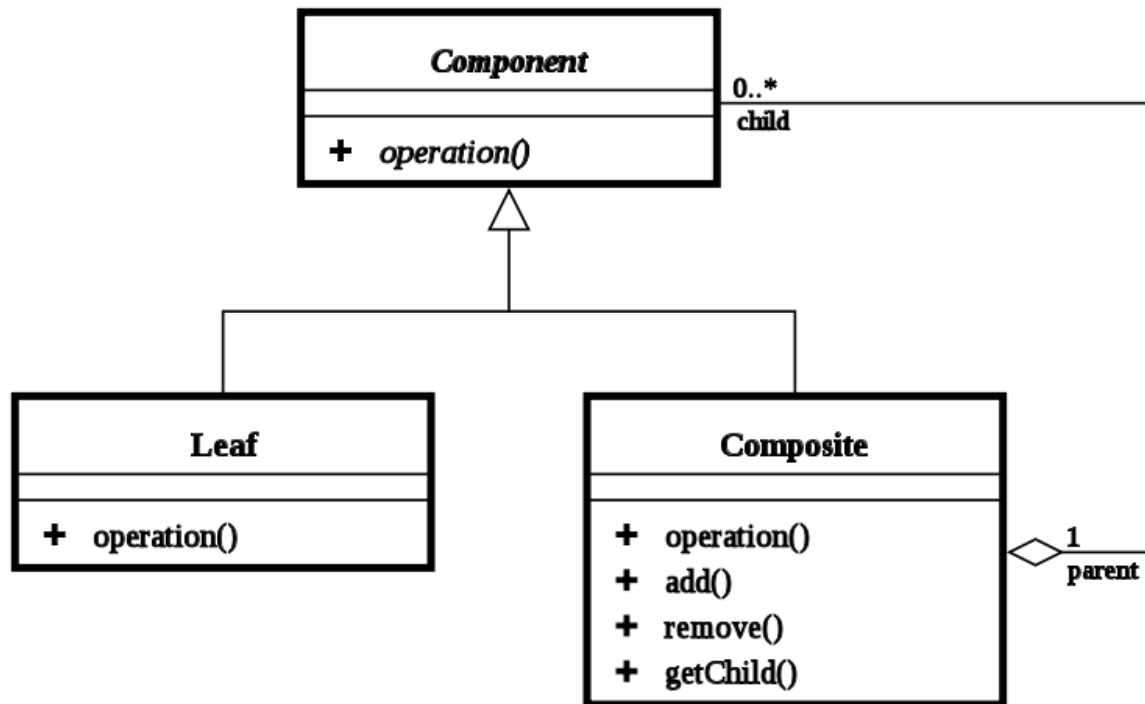
BUILDER / FACTORY PATTERN



COMPOSITE PATTERN

- When to use it:
 - When client should ignore the difference between one object and a composition of objects
 - If the same operations are performed on multiple instances the same way with nearly identical code
 - A composite pattern makes it less complex to treat single objects and composites the same way.
-

COMPOSITE PATTERN



INTERACTIVE DISCUSSION

- Did you use design patterns in your project so far?
 - If yes, why? If no, why not?
 - What are the (dis)advantages of using these patterns
- Should you design and implement code by reusing existing patterns? Or should you write the code and then identify pattern you've used "spontaneously" as a mean to communicate about the code? Or is it a mix of both?

QUESTIONS