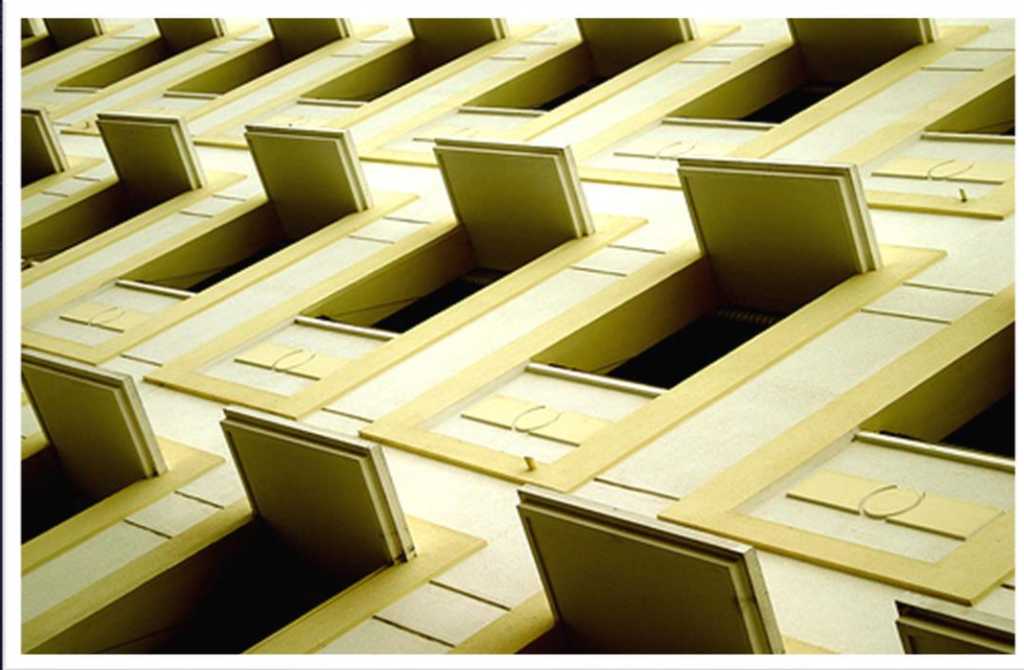





Design Patterns

Patterns



Pattern photo by Vinoth Chandar from [flickr](#) used under [CC BY](#)



*Design patterns, are
general, reusable solutions
to a commonly occurring
problem within a given
context in software design.*

Design Patterns

Unlike algorithms, design patterns are not clear instructions that can automatically be transferred to your system.

Design Patterns

Design patterns are more like templates that describe the general concept to solve the problem.

It doesn't contain implementation details; it contains structural blueprints.



History of Design Patterns

Design patterns are not novel and sophisticated discoveries, they are instead, typical solutions to common problems.



History of Design Patterns

The pattern of these solutions become so ubiquitous that it becomes worthwhile to put a name to it.

The background of the slide is a dark, atmospheric photograph of a forest. In the lower-left, a small, rustic hut with a thatched roof is nestled among trees. To the right, a waterfall cascades down a rocky ledge. The overall scene is dimly lit, with a blue and purple color palette, creating a serene and somewhat mysterious mood.

History of Design Patterns

Design patterns in software engineering are just borrowed concepts from architecture/design.

History of Design Patterns

The concept of design patterns is often attributed to Christopher Alexander, from his book, *A Pattern Language: Towns, Buildings, Construction*

History of Design Patterns

Four software engineers, Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm, used this as an inspiration to publish the famous book, *Design Patterns: Elements of Reusable Object-Oriented Software*.

History of Design Patterns

The four became collectively known as the "Gang of Four". And their book became known as the GoF book.

It contains a catalog of 23 design patterns solving various problems of OOP design.



Why
patterns?

The answer to this problem
is similar to the reason as to
why you don't "reinvent the
wheel"

Why patterns?

Design patterns are tried and tested solutions, knowing these patterns give programmers a toolset to solve a variety of problems in software design.

Why patterns?

Design patterns also help with communication.

A team of software engineers well versed in design patterns wouldn't need to explain to each other what exactly must be done to use an "Adapter pattern"



Why not
patterns?

Design patterns are
sometimes used to simulate
features that the
programming language
doesn't have



Why not
patterns?

If you use a powerful enough language you wouldn't need the pattern at all.

Example of this is how the Strategy pattern can be replaced by lambdas

Why not patterns?

Patterns are not end-all be-all solutions to any design problem out there. At the end of the day context matters the most.

An inexperienced programmer will implement a problem to the dot, instead of adapting the pattern for the context

The background of the slide is a dark, atmospheric photograph of a forest. In the lower-left background, a small, light-colored tent is visible among the trees. The overall scene is dimly lit, with the trees' silhouettes and foliage creating a textured, layered appearance. The colors are muted, with various shades of dark green, blue, and brown.

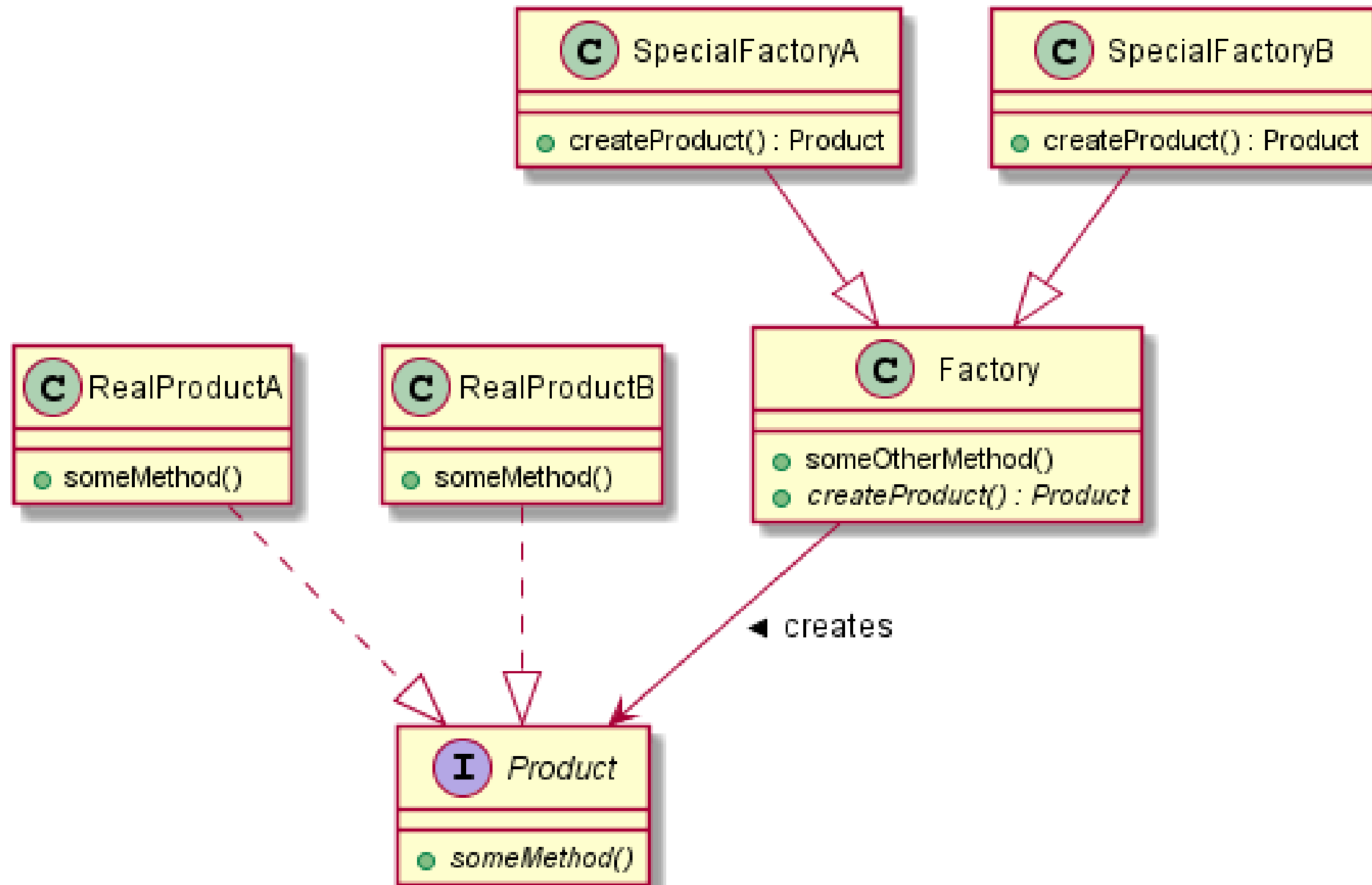
Why not
patterns?

Sometimes, you don't need
a pattern at all.

A simple problem solved
using a complicated
solution is inelegant

Classifications of Patterns

- Creational patterns
- Structural patterns
- Behavioral patterns



Factory method pattern

References

Pattern photo by Vinoth Chandar from [flickr](#) used under [CC BY](#)



Alexander (1977). *A Pattern Language: Towns, Buildings, Construction*.

Gamma, Vlissides, Johnson, and Helm (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*