

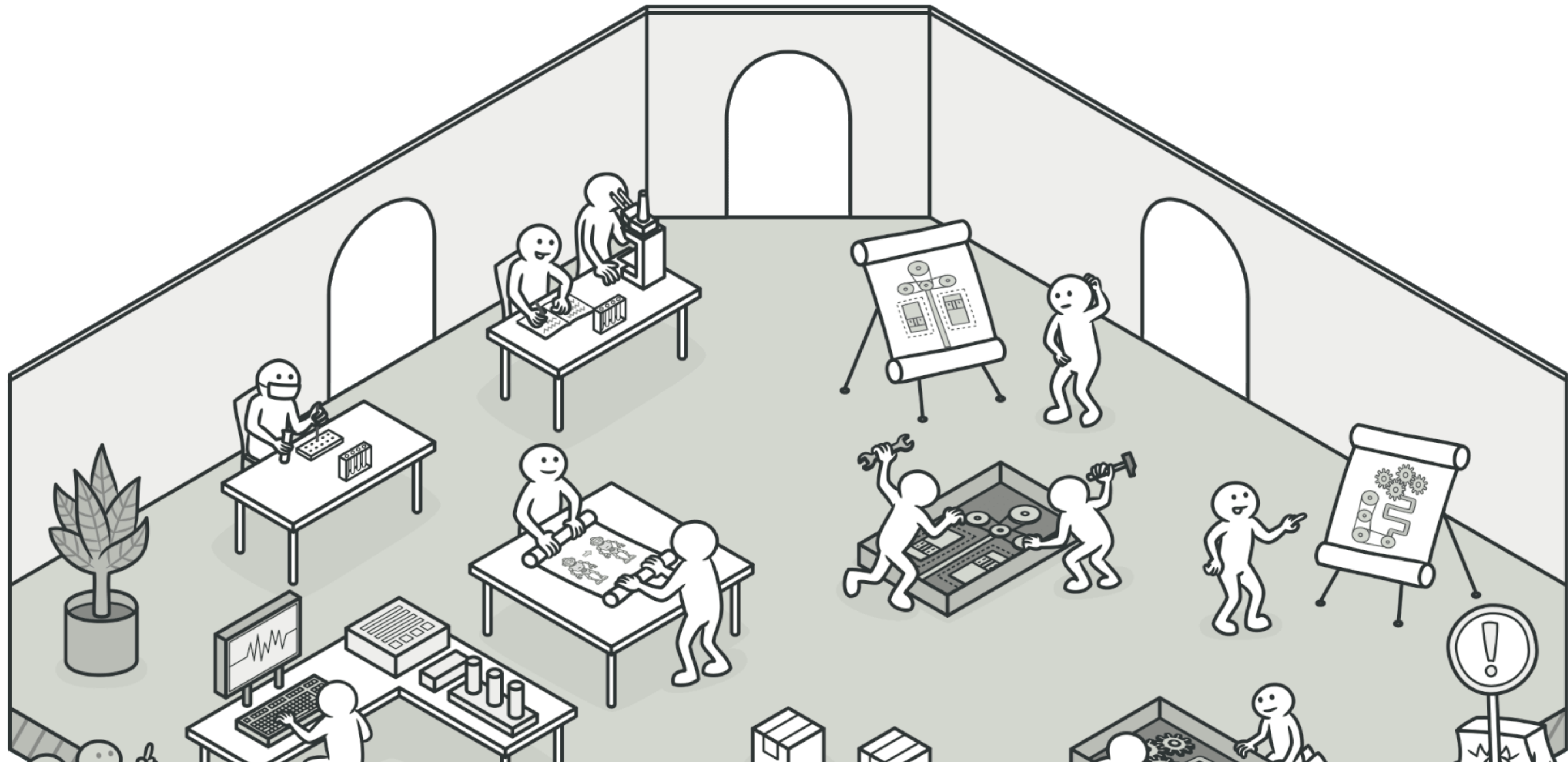
Design Patterns

p.1

by Yelyzaveta Koliechkina

for Analytics & AI, Schneider Electric

A design pattern is a general repeatable solution to a common problem in software design.



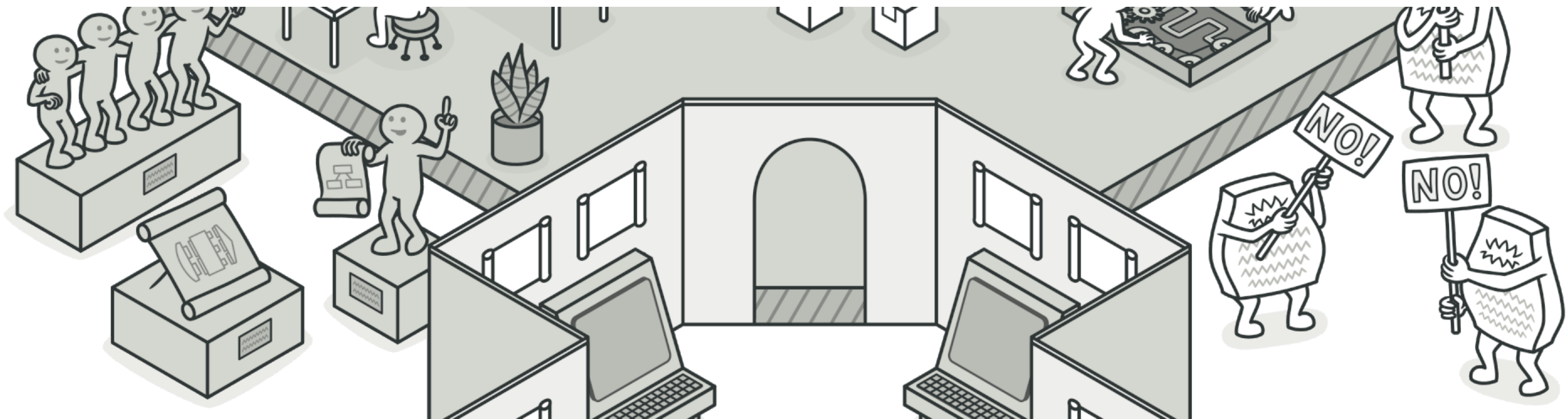
In 1994, the first book describing software design patterns was published by four authors: *Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm*.

"Design Patterns: Elements of Reusable Object-Oriented Software"

or the book by the Gang of Four, "the GoF book"
described **23 patterns** solving problems of object-oriented design

Why to use design patterns?

- speed up the development process
- use tested, proven development paradigms
- improve code readability



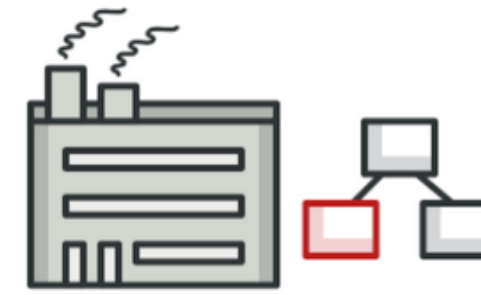
Design patterns differ by their complexity, level of detail and scale of applicability to the entire system being designed.

In addition, all patterns can be categorized by their intent, or purpose.

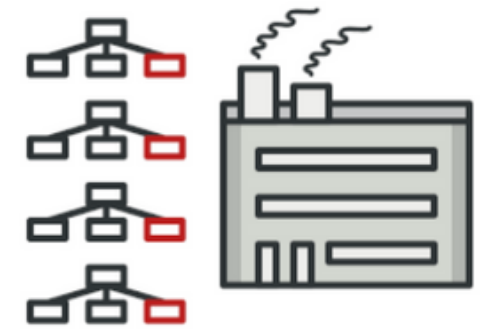
- Creational patterns
- Structural patterns
- Behavioral patterns
- Concurrency patterns
- Architectural patterns

Creational patterns

provide various object creation mechanisms, which increase flexibility and reuse of existing code.



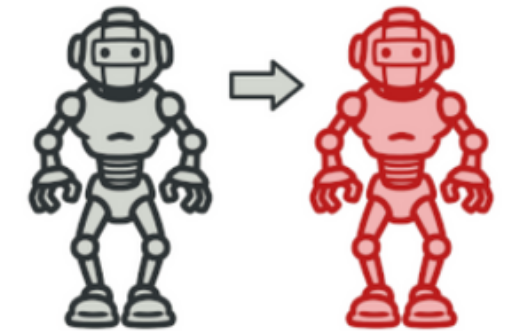
Factory Method



Abstract Factory



Builder



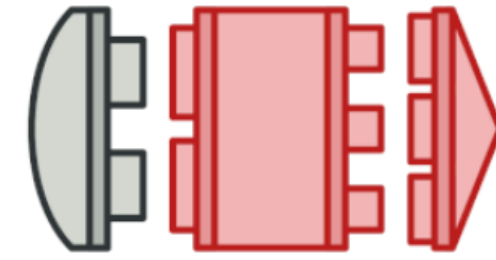
Prototype



Singleton

Structural patterns

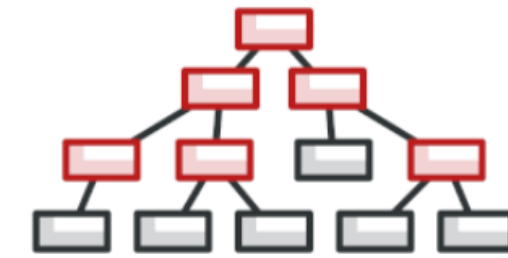
explain how to assemble
objects and classes into larger
structures, while keeping
these structures flexible and
efficient.



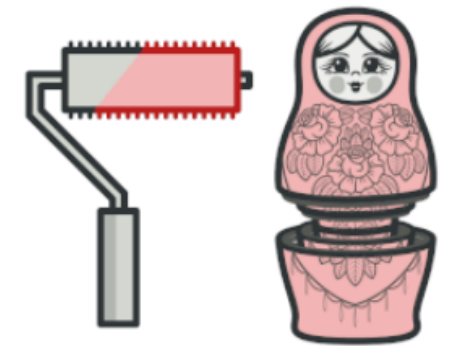
Adapter



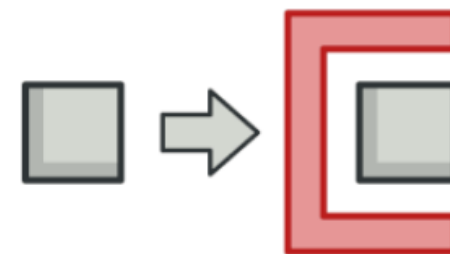
Bridge



Composite



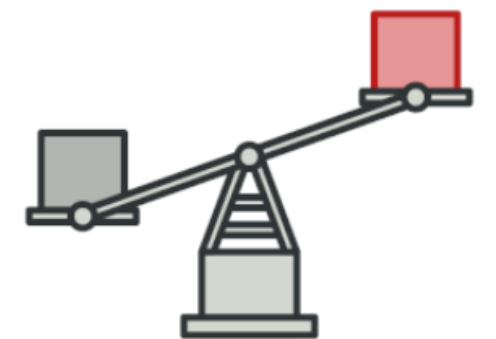
Decorator



Proxy



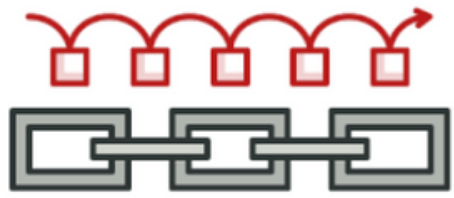
Facade



Flyweight

Behavioral patterns

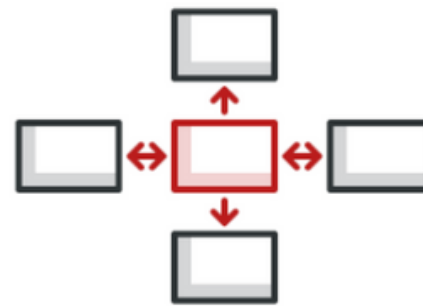
take care of effective communication and the assignment of responsibilities between objects.



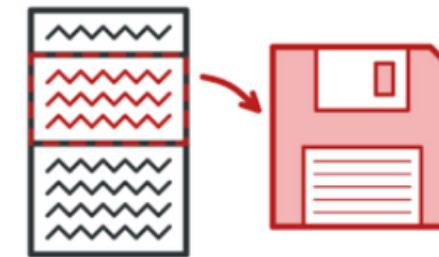
Chain of Responsibility



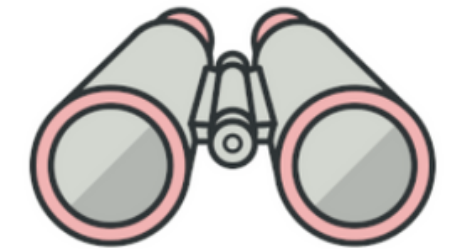
Iterator



Mediator



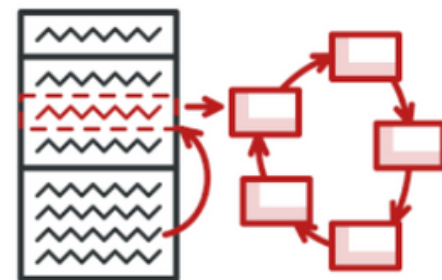
Memento



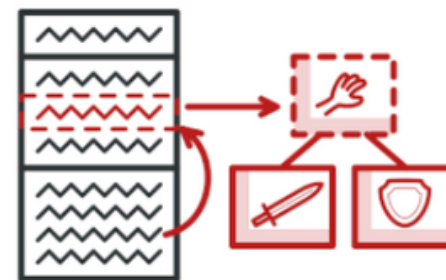
Observer



Command



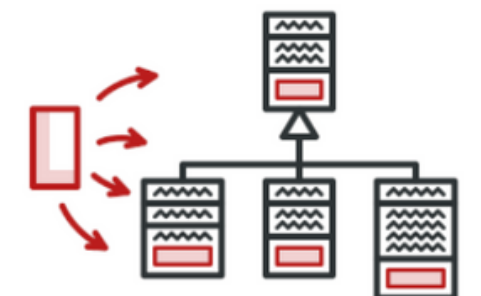
State



Strategy

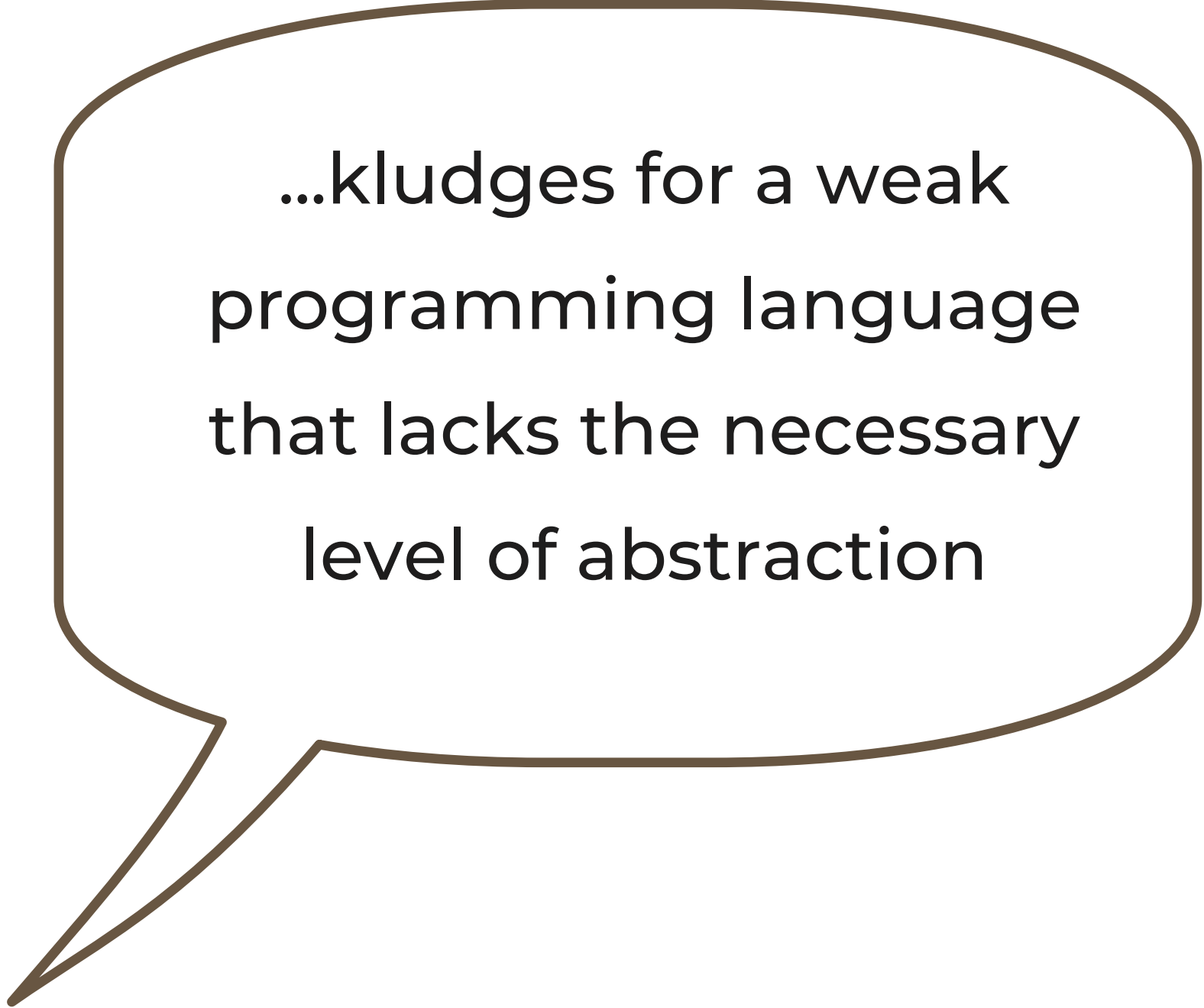


Template Method



Visitor

Are patterns actually good?



...kludges for a weak
programming language
that lacks the necessary
level of abstraction

A kludge or kluge (/klʌdʒ, kluːdʒ/) is a workaround or quick-and-dirty solution that is clumsy, inelegant, inefficient, difficult to extend and hard to maintain.

Are patterns actually good?

Unjustified use

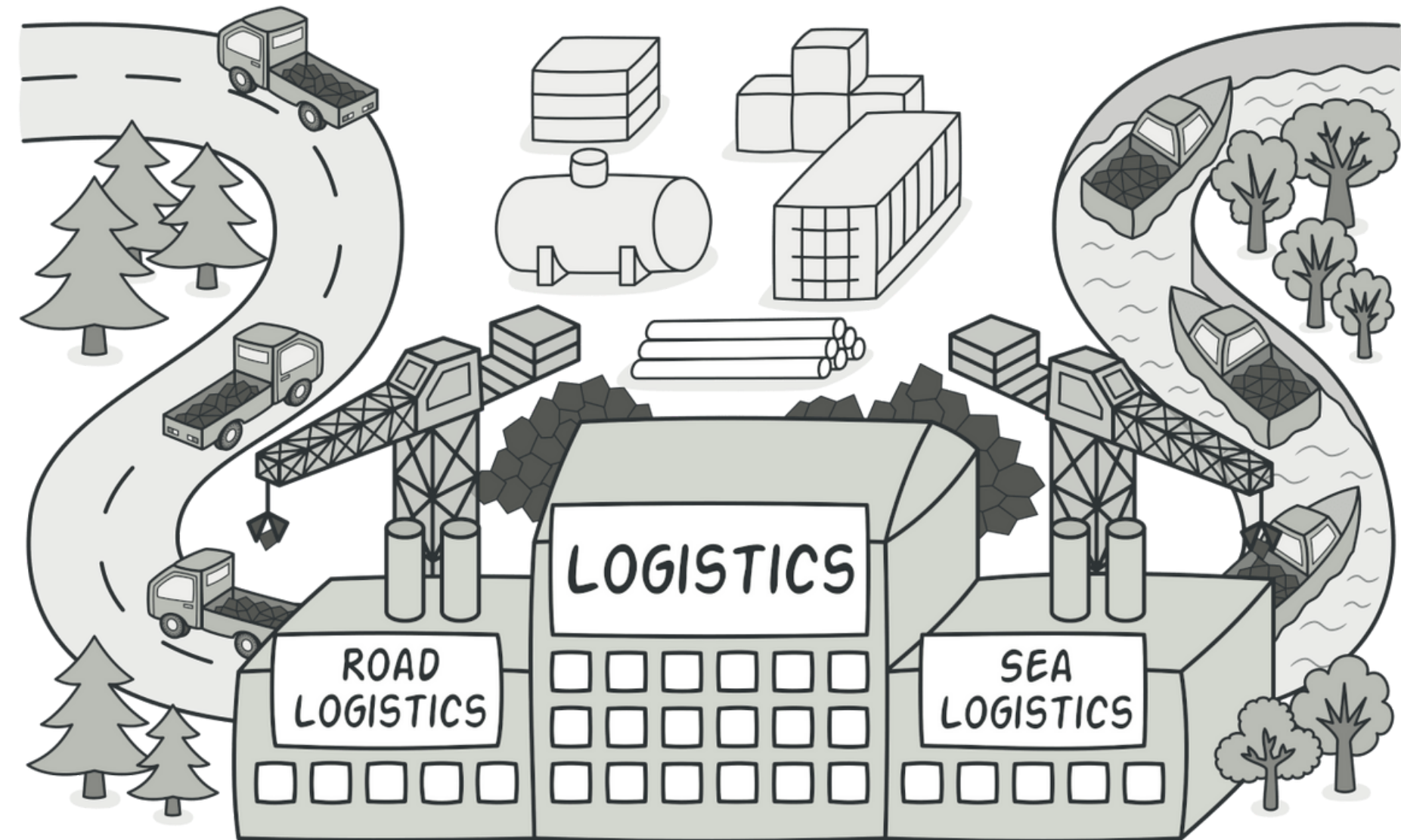
"If all you have is a hammer, everything looks like a nail"

Inefficient solutions,
systematize
approaches that are
already widely used.

Factory Method

Also known as Virtual Constructor

a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created



Problem

you don't know beforehand the exact types and dependencies of the objects your code should work with
(example - app for Logistics Department)



Solution

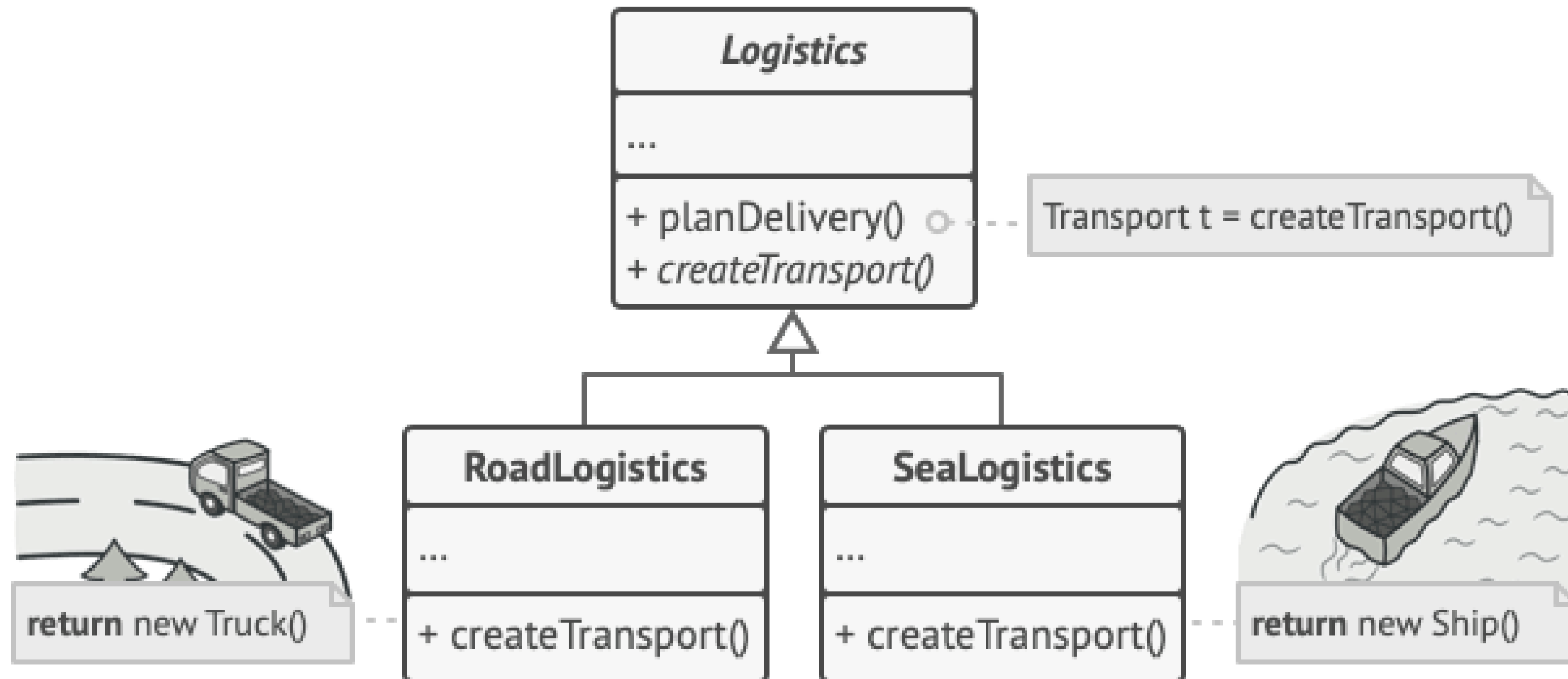
The Factory Method pattern suggests that you replace direct object construction calls with calls to a special factory method.

The objects are still created via the `new` operator, but it's being called from within the factory method.

Objects returned by a factory method are often referred to as products.

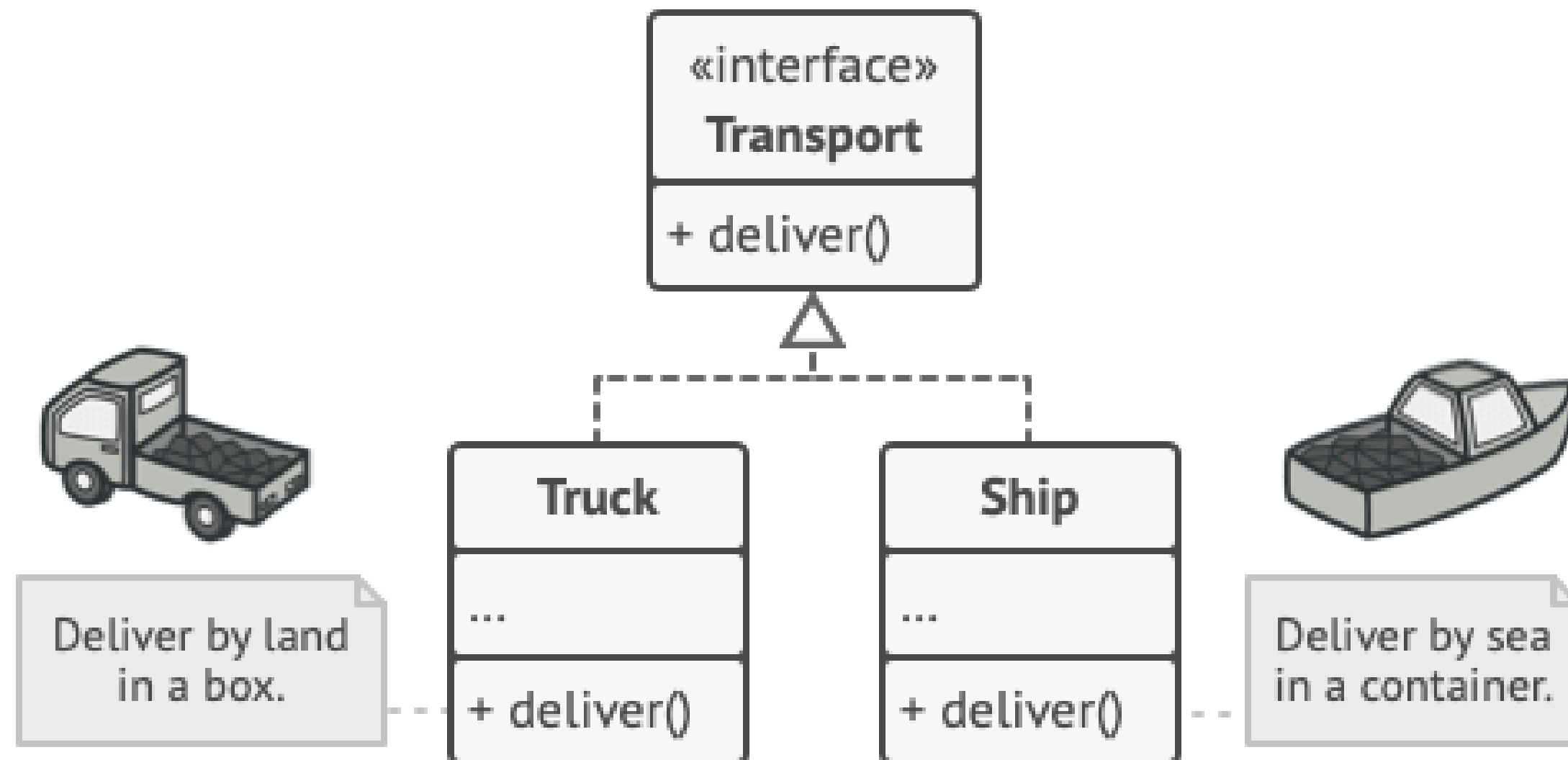
Solution

Now you can override the factory method in a subclass and change the class of products being created by the method.



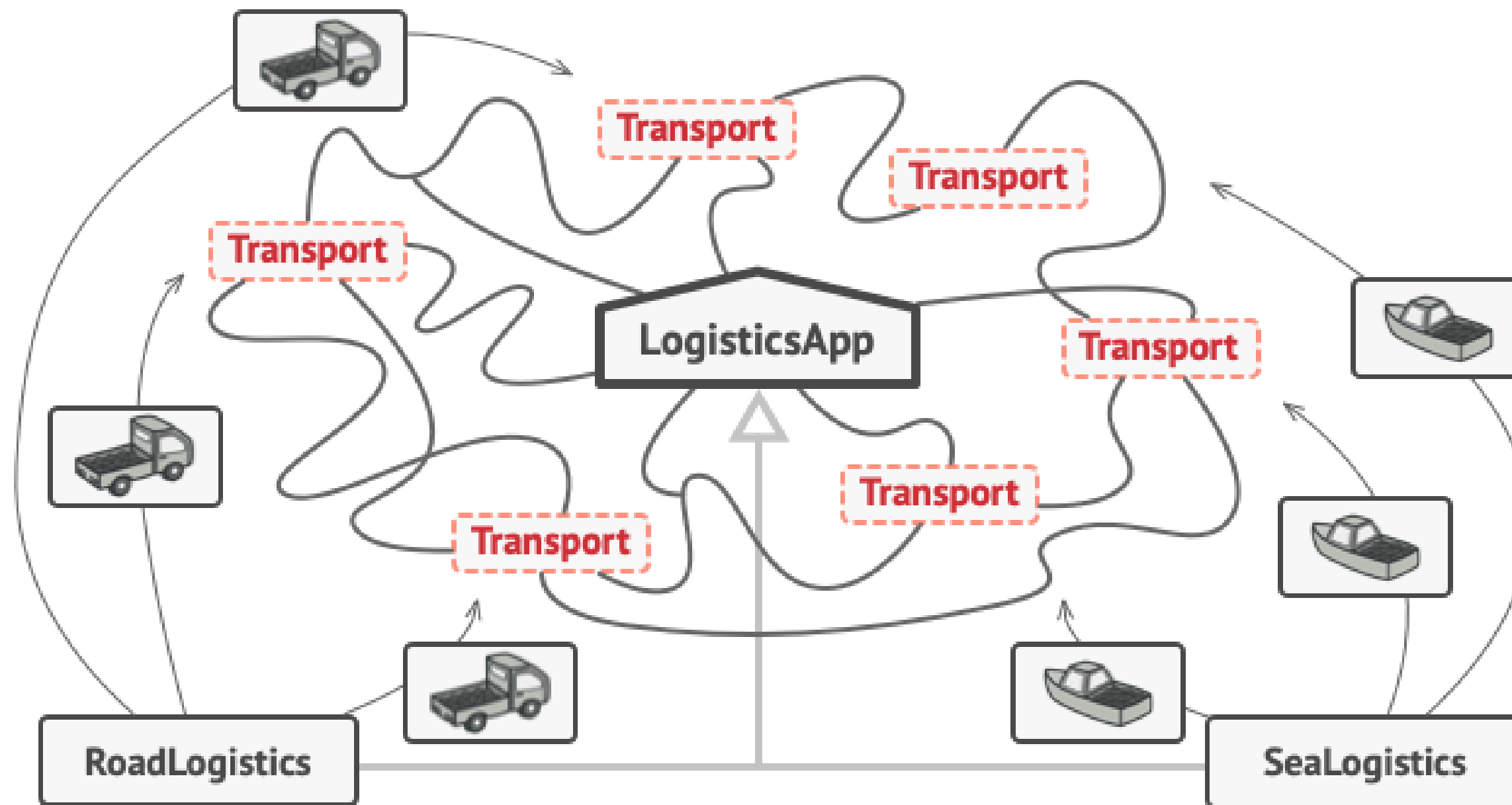
Solution

subclasses may return different types of products only if these products have a common base class or interface



Solution

The code that uses the factory method (often called the client code) doesn't see a difference between the actual products returned by various subclasses.



- Use the Factory Method when you don't know beforehand the exact types and dependencies of the objects your code should work with
- The Factory Method separates product construction code from the code that actually uses the product. Therefore it's easier to extend the product construction code independently from the rest of the code.
- Many designs start by using Factory Method (less complicated and more customizable via subclasses) and evolve toward Abstract Factory, Prototype, or Builder (more flexible, but more complicated).

Sources



Design Patterns and Refactoring

Design Patterns and Refactoring articles and guides. Design Patterns video tutorials for newbies. Simple descriptions and full source code examples in Java, C++, C#, PHP and Delphi.

sourcemaking.com

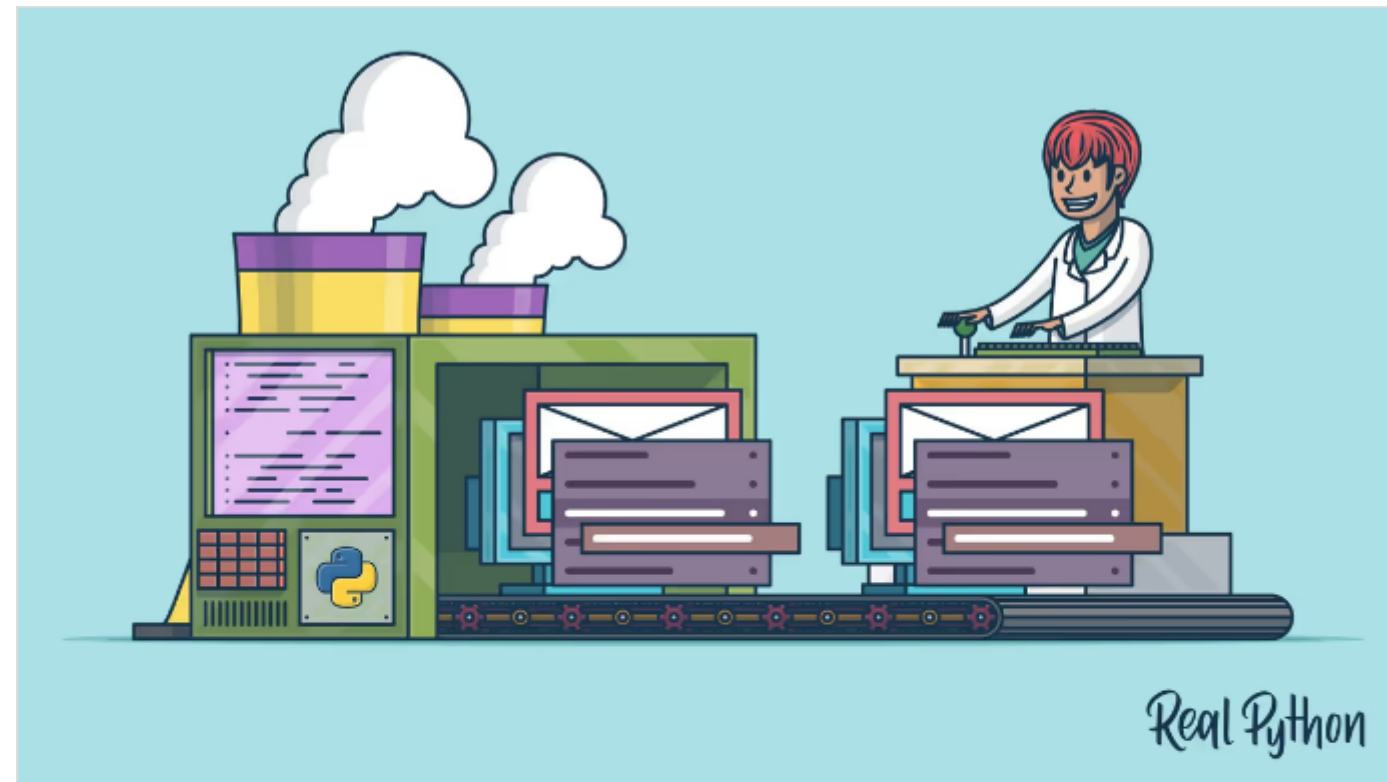


REFACTORING
• GURU •

The Catalog of Design Patterns

The catalog of design patterns grouped by intent, complexity, and popularity. The catalog contains all classic design patterns and several architectural patterns.

 refactoring.guru



Real Python

The Factory Method Pattern and Its Implementation in Python

In this Python tutorial, you'll learn about the Factory Method design pattern and its implementation. You'll understand the components of Factory Method, when to use it, and how to modify existing code to leverage it. You'll also see a general purpose...

 [realpython /](http://realpython/)