# Creational patterns

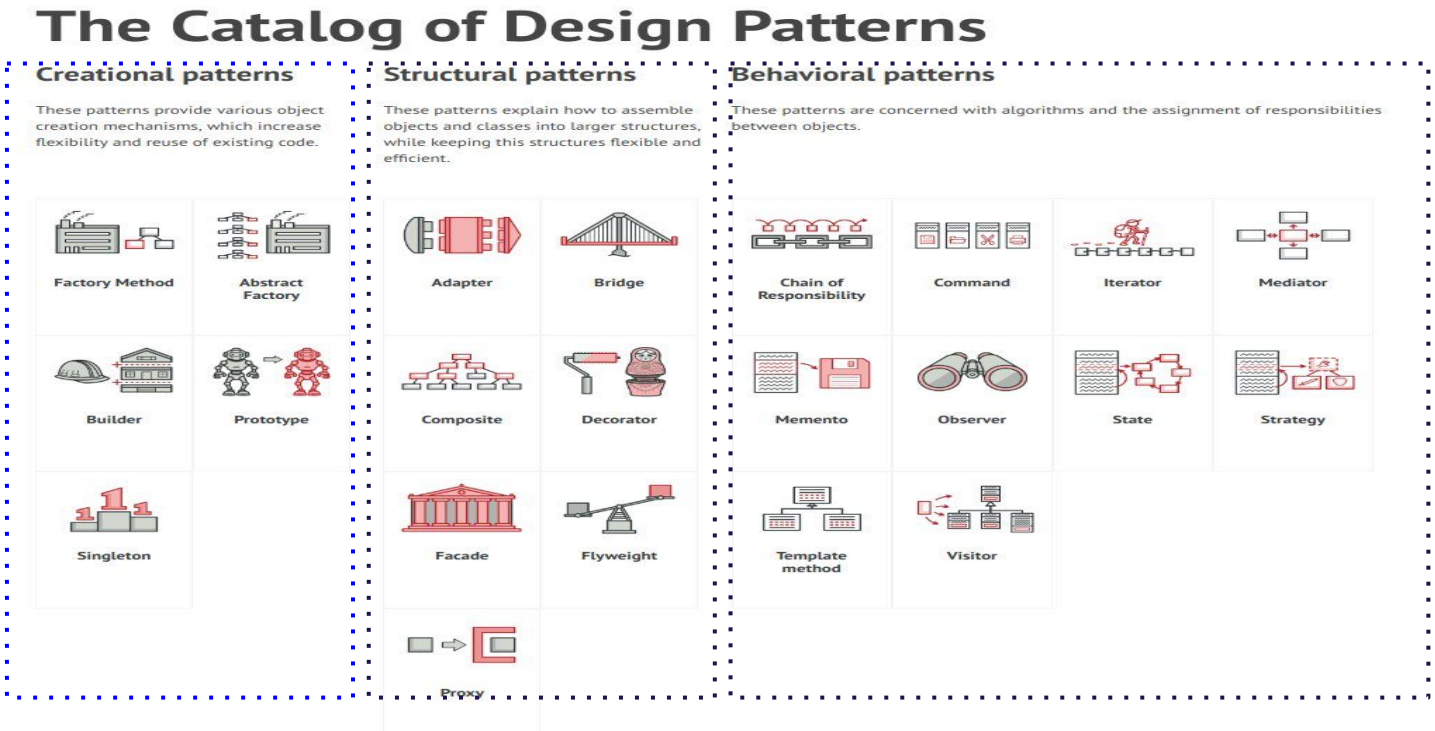
## Factory Method

Upcode Software
Engineer Team

-2023-

# CONTENT

# The Catalog of Design Patterns



# What's a design pattern?

- **Design patterns** are *typical solutions to commonly occurring problems in software design*.
- They are like pre-made **blueprints that you can customize to solve a recurring design problem in your code.**
- Patterns are often confused with algorithms, because both concepts describe typical solutions to some known problems. **While an algorithm always defines a clear set of actions that can achieve some goal, a pattern is a more high-level description of a solution.** The code of the same pattern applied to two different programs may be different.

Software design is **the process of envisioning and defining software solutions to one or more sets of problems**. One of the main components of software design is the software requirements analysis (SRA).
SRA is a part of the software development process that lists specifications used in software engineering.

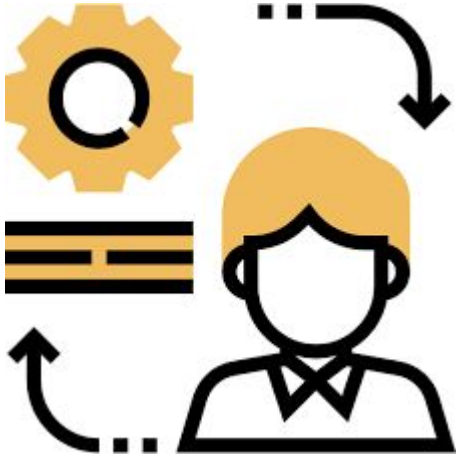
# What does the pattern consist of?

Most patterns are described very formally so people can reproduce them in many contexts. Here are the sections that are usually present in a pattern description:

- **Intent** of the pattern briefly describes both the problem and the solution.
- **Motivation** further explains the problem and the solution the pattern makes possible.
- **Structure** of classes shows each part of the pattern and how they are related.
- **Code example** in one of the popular programming languages makes it easier to grasp the idea behind the pattern.

# Why should I learn patterns?

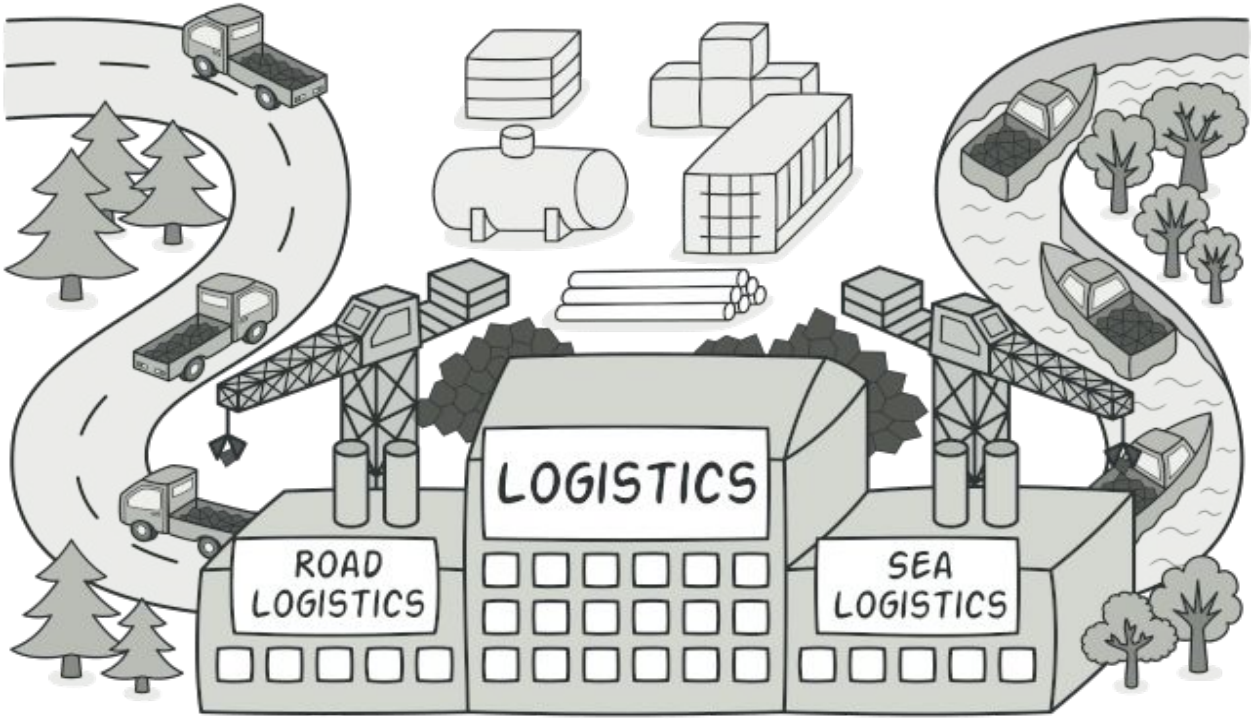
- **The truth** is that you might manage to work as a programmer for many years without knowing about a single pattern.
- A lot of people do just that. Even in that case, though, you might be implementing **some patterns without even knowing it.**
- So **why would you spend time learning them?**



# Why should I learn patterns?

- **Design patterns** are a toolkit of tried and tested solutions to common problems in software design. Even if you never encounter these problems, knowing patterns is still useful because it teaches you **how to solve all sorts of problems using principles of object-oriented design.**
- **Design patterns** define a common language that you and your teammates can use to communicate more efficiently.
    a. You can say, “Oh, just use a Singleton for that,” and everyone will understand the idea behind your suggestion. No need to explain what a singleton is if you know the pattern and its name

# Factory Method



# Factory Method - Definition

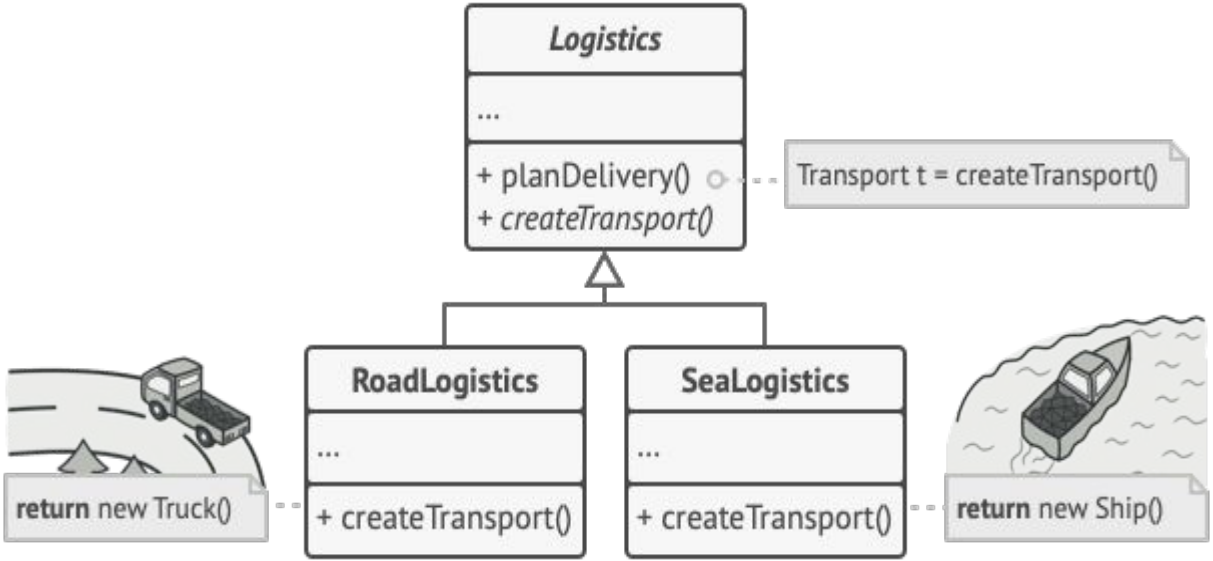
- **Factory Method** is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.

# Factory Method - Problem

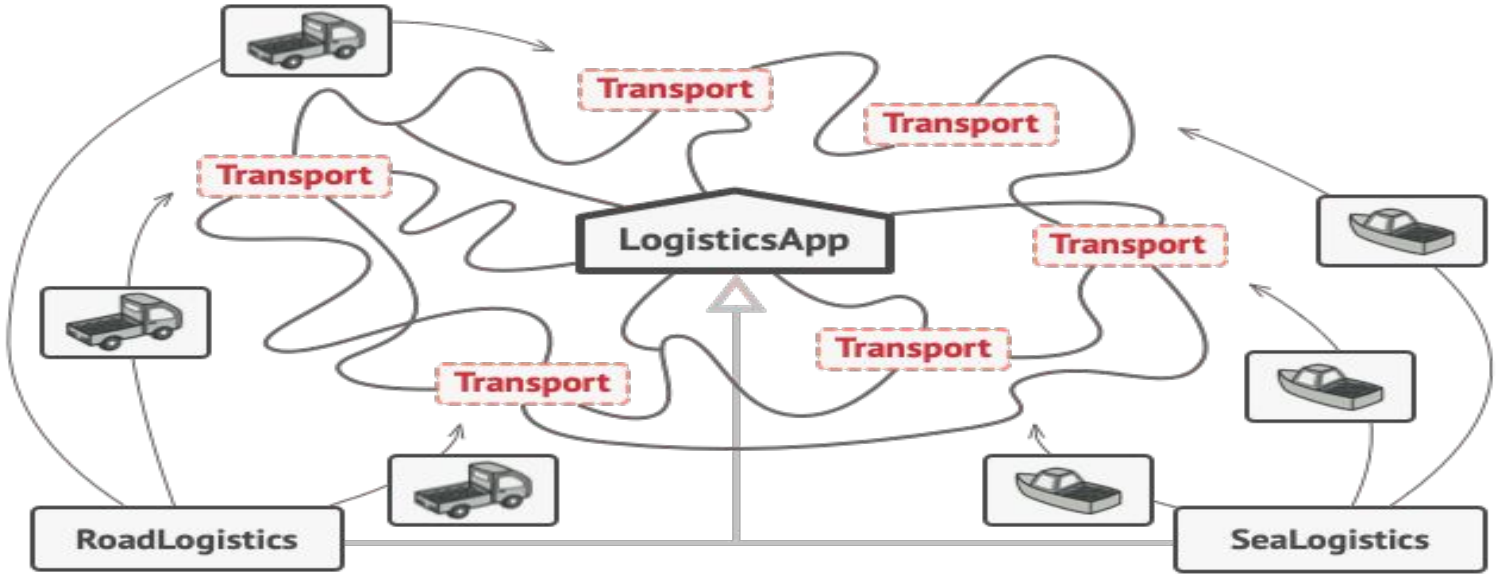
- Imagine that you're creating **a logistics management application**. The first version of your app can only handle transportation by trucks, so the bulk of your code lives inside the `Truck` class.
- After a while, your app becomes pretty popular. **Each day you receive dozens of requests** from sea transportation companies to incorporate sea logistics into the app.

# Factory Method - Solution

- The Factory Method pattern suggests that you replace direct object construction calls (using the new operator) with calls to a special *factory* method.



# Factory Method - Solution 1



# Reference

1. Refactoring GURU
2. Dive into Design Pattern
3. Source Code

# Thank you!

Presented by **Hamdamboy Urunov**

**(hamdamboy.urunov@gmail.com)**