

Creational patterns

Singleton



Upcode Software
Engineer Team

-2023-



CONTENT

- 1.The Catalog of Design Patterns.
- 2.What is Singleton ?
- 3.Why needs to Singleton ?
- 4.How to use Singleton ?
- 5.Where to use Singleton ?
- 6.SOURCE code.

The Catalog of Design Patterns

The Catalog of Design Patterns

Creational patterns

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



Factory Method



Abstract Factory



Builder



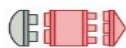
Prototype



Singleton

Structural patterns

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



Adapter



Bridge



Composite



Decorator



Facade



Flyweight



Proxy

Behavioral patterns

These patterns are concerned with algorithms and the assignment of responsibilities between objects.



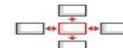
Chain of Responsibility



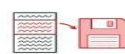
Command



Iterator



Mediator



Memento



Observer



State



Strategy



Template method



Visitor



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.



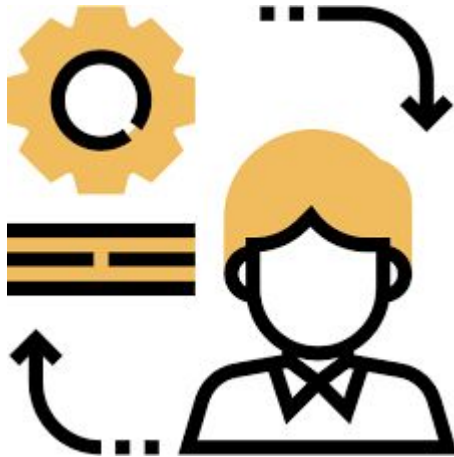
What does the pattern consist of?

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?

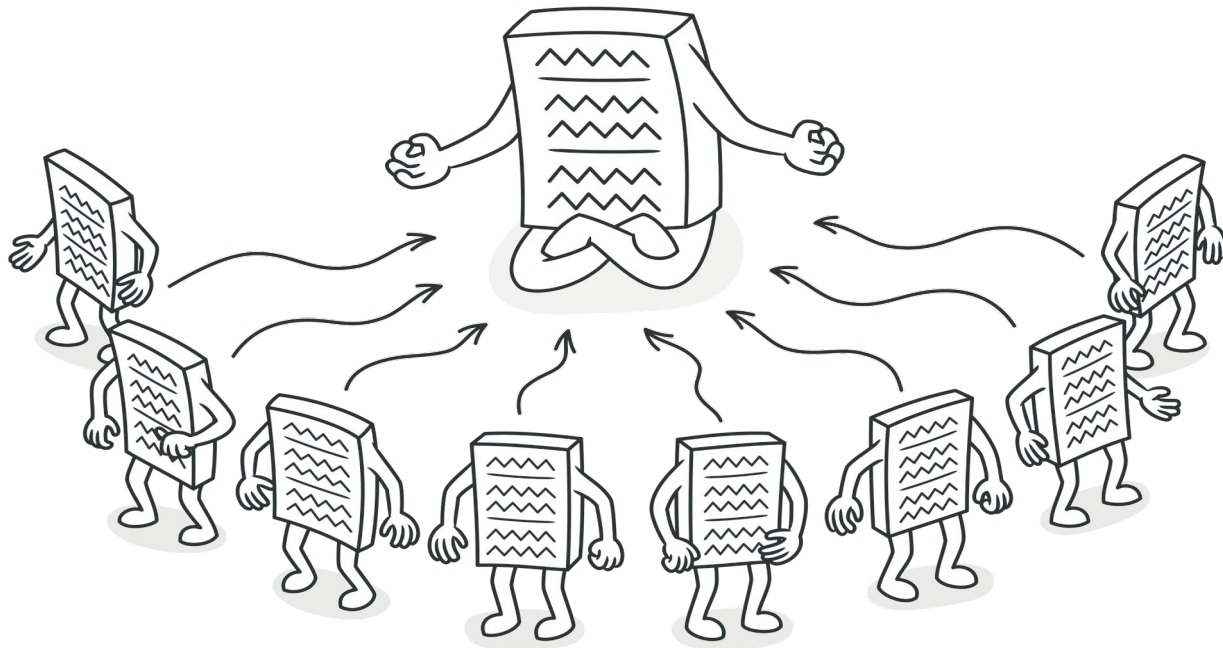




Why should I learn patterns?

- **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

1. Singleton



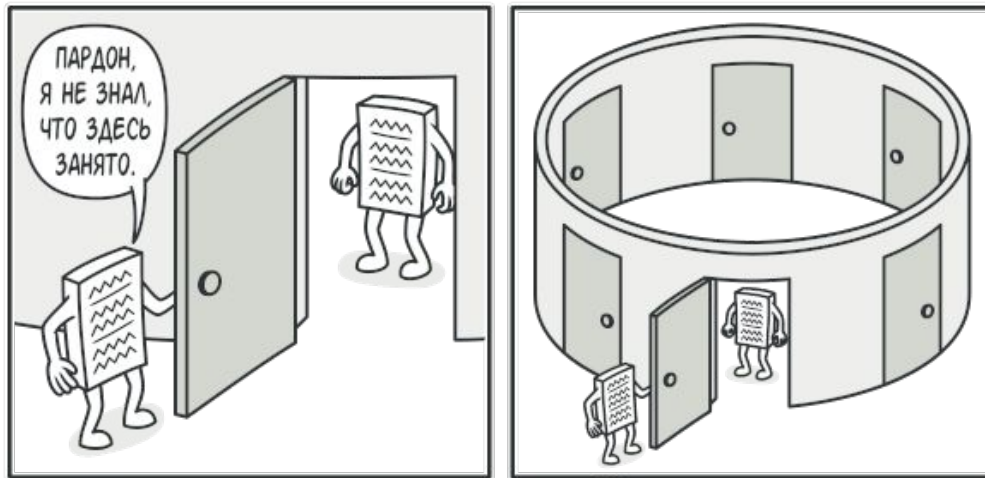


Singleton - Definition

- Одиночка (Singleton, Синглтон) - порождающий паттерн, который **гарантирует, что для определенного класса будет создан только один объект**, а также предоставит к этому объекту точку доступа. Синглтон позволяет создать объект только при его необходимости. Если объект не нужен, то он не будет создан.
- Singleton (Singleton, Singleton) - a generating pattern that **guarantees that one object will be created for a certain class**, as well as only access to this object with an access point. Singleton allows you to create an object only when it is needed. If the object is not needed, then it will not be created.

Singleton - Problem

- Нарушение принципа единственной ответственности.
- Затруднения в тестировании.
- Потокобезопасность.
- Проблемы с сериализацией.
- Сложности в расширении.



Singleton - Solution

- Все реализации одиночки сводятся к тому, чтобы скрыть конструктор по умолчанию и создать публичный статический метод, который и будет контролировать жизненный цикл объекта-одиночки.



Non singleton class

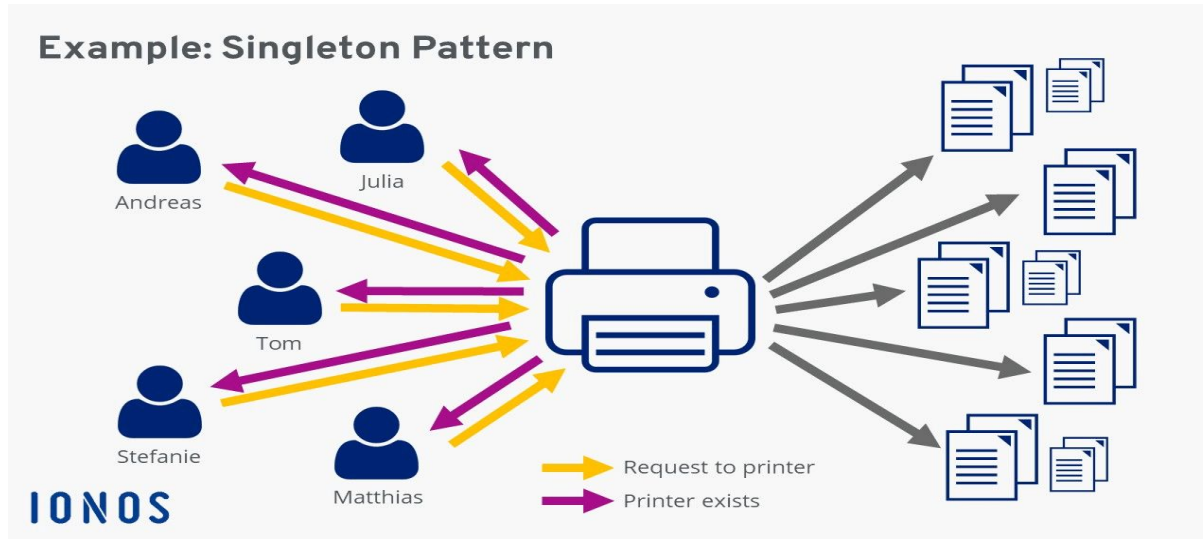
JavaGoal.com



Singleton class

Singleton - Solution 1

- Если у вас есть доступ к классу одиночки, значит, будет доступ и к этому статическому методу. Из какой точки кода вы бы его ни вызвали, он всегда будет отдавать один и тот же объект.





Reference

1. Refactoring [GURU](#)
2. Dive into Design Pattern



Thank you!

Presented by Tursunov Nodirkhuja

(ameriqano@gmail.com)