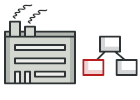
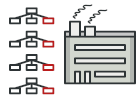
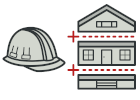
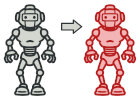
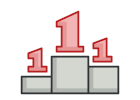
**Padrões criacionais**

Estes padrões fornecem vários mecanismos de criação de objetos, que aumentam a flexibilidade e reutilização de código já existente.

[**Factory Method**](https://refactoring.guru/pt-br/design-patterns/factory-method)[**Abstract Factory**](https://refactoring.guru/pt-br/design-patterns/abstract-factory)[**Builder**](https://refactoring.guru/pt-br/design-patterns/builder)[**Prototype**](https://refactoring.guru/pt-br/design-patterns/prototype)[](https://refactoring.guru/pt-br/design-patterns/singleton)

O **Factory Method** é um padrão criacional de projeto que fornece uma interface para criar objetos em uma superclasse, mas permite que as subclasses alterem o tipo de objetos que serão criados.

O **Abstract Factory** é um padrão de projeto criacional que permite que você produza famílias de objetos relacionados sem ter que especificar suas classes concretas.

O **Builder** é um padrão de projeto criacional que permite a você construir objetos complexos passo a passo. O padrão permite que você produza diferentes tipos e representações de um objeto usando o mesmo código de construção.

O **Prototype** é um padrão de projeto criacional que permite copiar objetos existentes sem fazer seu código ficar dependente de suas classes.

O **Singleton** é um padrão de projeto criacional que permite a você garantir que uma classe tenha apenas uma instância, enquanto provê um ponto de acesso global para essa instância.