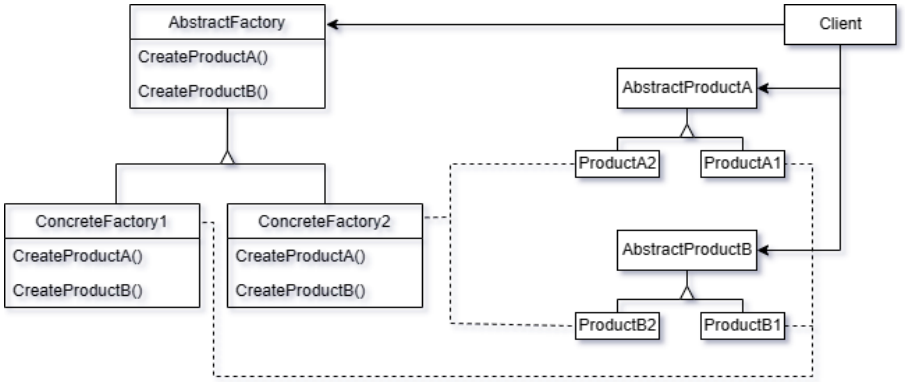
Abstract Factory

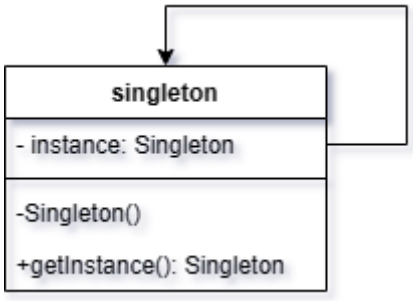


O padrão Abstract Factory permite a criação de famílias relacionadas ou dependentes de uma classe abstrata ou uma interface e sem que a classe concreta seja explicita. Conhecido como Kit. Seu objetivo é isolar a criação de objetos de seu uso e criar famílias de objetos relacionados sem depender de classes concretas. Permitindo novos tipos derivados sem qualquer alteração ao código que usa a classe base. Sua aplicabilidade é em cenários onde produtos ou classes de famílias precisam ser instanciados, sem dependência de classes concretas.

O mesmo facilita bastante quando trabalhamos com aplicações que devem funcionar com ambientes diferentes, como interfaces gráficas, sistemas de banco de dados diferentes, mas onde temos bases comuns, apenas com comportamentos específicos definidos para cada “ambiente”.

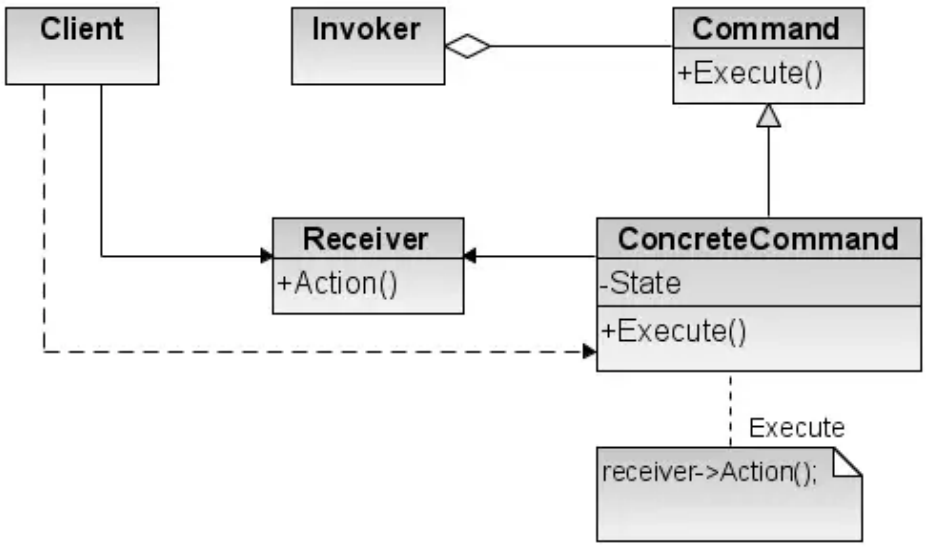
Singleton

Singleton é um dos padrões mais simples, porém responsável por garantir que exista apenas uma instância de classe, único e global de acesso ao objeto. Alguns projetos precisam dessa caracteristica , por exemplo uma infraestrutura de log de dados, desta maneira existe apenas um objeto responsável pelo log em toda aplicação, acessível unicamente através da classe singleton.



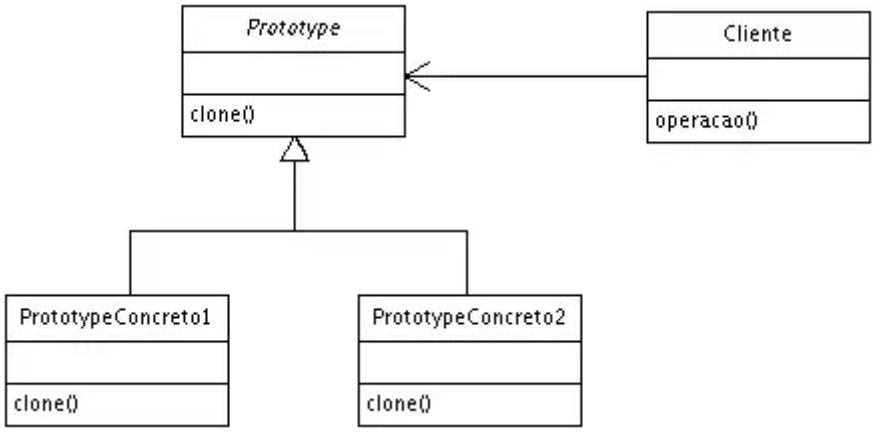
Command

Este padrão faz parte do grupo comportamentais. Tem como responsabilidade encapsular uma solicitação como um objeto, o que lhe permite parametrizar com outros objetos com diferentes solicitações, enfileirar ou realizar solicitações e implementar o cancelamento de operações. Nome do método, objeto que o método pertence e valores dos parâmetros do método.



Sua aplicação se baseia no uso de uma classe abstrata ou interface, a qual declara um contrato para execução de operações. Na sua forma mais simples, esta interface inclui uma operação abstrata execute. As subclasses concretas de Command especificam um par receptor-ação através do armazenamento do receptor como uma variável de instância e pela implementação de execute para invocar a solicitação. Command faz com que seja mais fácil construir componentes que delegam, enfileiram ou executam métodos em um momento de sua escolha, sem a necessidade de conhecer a classe do método.

ProtoType



O padrão de projeto ProtoType faz parte dos padrões criacionais, que são aqueles que permite a criação de novos objetos a partir de um modelo ou protótipo. Sua utilização é aplicada quando há necessidade de clonar objetos, funciona muito bem com o padrão Abstract Factory, para criação (clonagem) de objetos, com o prototype hierarquia pode ser mais simples de ser resolvida nesses casos, por realizar uma mesma tarefa com um número reduzido de classes.

MVC

É um padrão de arquitetura de software responsável por contribuir na otimização da velocidade entre as requisições feitas pelo comando dos usuários. a arquitetura MVC é dividida em três componentes essenciais: Model, Controller e View.

Model: Sua responsabilidade é gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas.

View: Essa camada é responsável por apresentar as informações de forma visual ao usuário.

Controller: A camada de controle é responsável por intermediar as requisições enviadas pelo View com as respostas fornecidas pelo Model, processando os dados que o usuário informou e repassando para outras camadas.

Sua implementação pode ser feita por Web, Mobile ou Desktop, ela pode ser implementada através de diversos frameworks de Java como o Spring MVC ou Play Framework ou também em frameworks mais modernos de Ruby como Ruby on Rails.