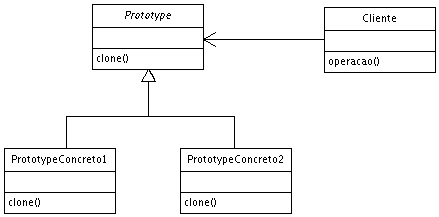
Prototype

# Finalidade:

"Especificar os tipos de objetos a serem criados usando uma instância como protótipo e criar novos objetos ao copiar este protótipo." [GoF]

Criar um objeto novo, mas aproveitar o estado previamente existente em outro objeto.

# Diagrama UML:



# Consequências:

Vantagens:

* Poder aproveitar o estado existente de um objeto
* Permite que um cliente crie novos objetos ao copiar objetos existentes
* Em determinadas circunstâncias, copiar um objeto pode ser mais eficaz que criar um novo

Desvantagens:

* Se torna complexo se for considerado a possibilidade de existirem referências circulares nos atributos de um objeto
* Object.clone() pode ser usado como implementação do Prototype pattern em Java mas é preciso lembrar que ele só faz cópias rasas: é preciso copiar também cada objeto membro e seus campos recursivamente.

Abstract Factory

# Finalidade:

“Abstract Factory - Fornece uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.”[GoF].

# Diagrama UML:

# https://lh4.googleusercontent.com/22LUkuhp8ouHRwexSTDa0An55BfgQJgM_bFqZSSZX2nzejfkDAWVTBmsLZnZGDGdGonkqHu0duKtXVFtqPOK3fkhJBkEtVoQjjBfZJ3xZeNuze68-5SGLzr3pj7csMdVHu-n0QlRE1Y

# Consequências:

Vantagens:

* O padrão isola classes concretas. A factory encapsula a responsabilidade e o processo de criação de objetos, isolando clientes das classes de implementação.
* Produtos de uma determinada família devem funcionar conjuntamente e não misturados com os de outra família.

Desvantagens:

* Dar suporte a muitos produtos força mudanças na Fábrica Abstrata.