Padrões de Projeto

Buider e Prototype

Drª. Alana Morais

Aula Passada

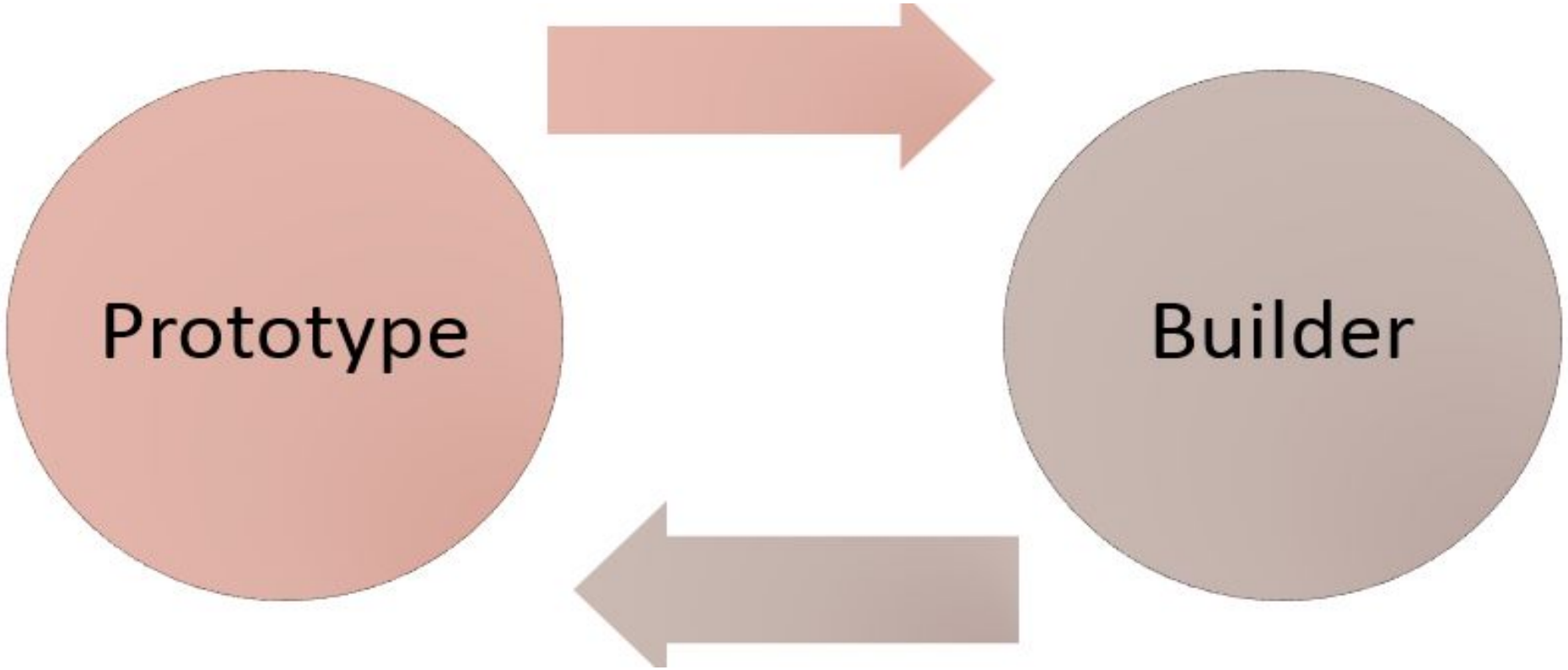
PADRÕES CRIACIONAIS DO GOF

- ABSTRACT FACTORY
- FACTORY METHOD
- SINGLETON

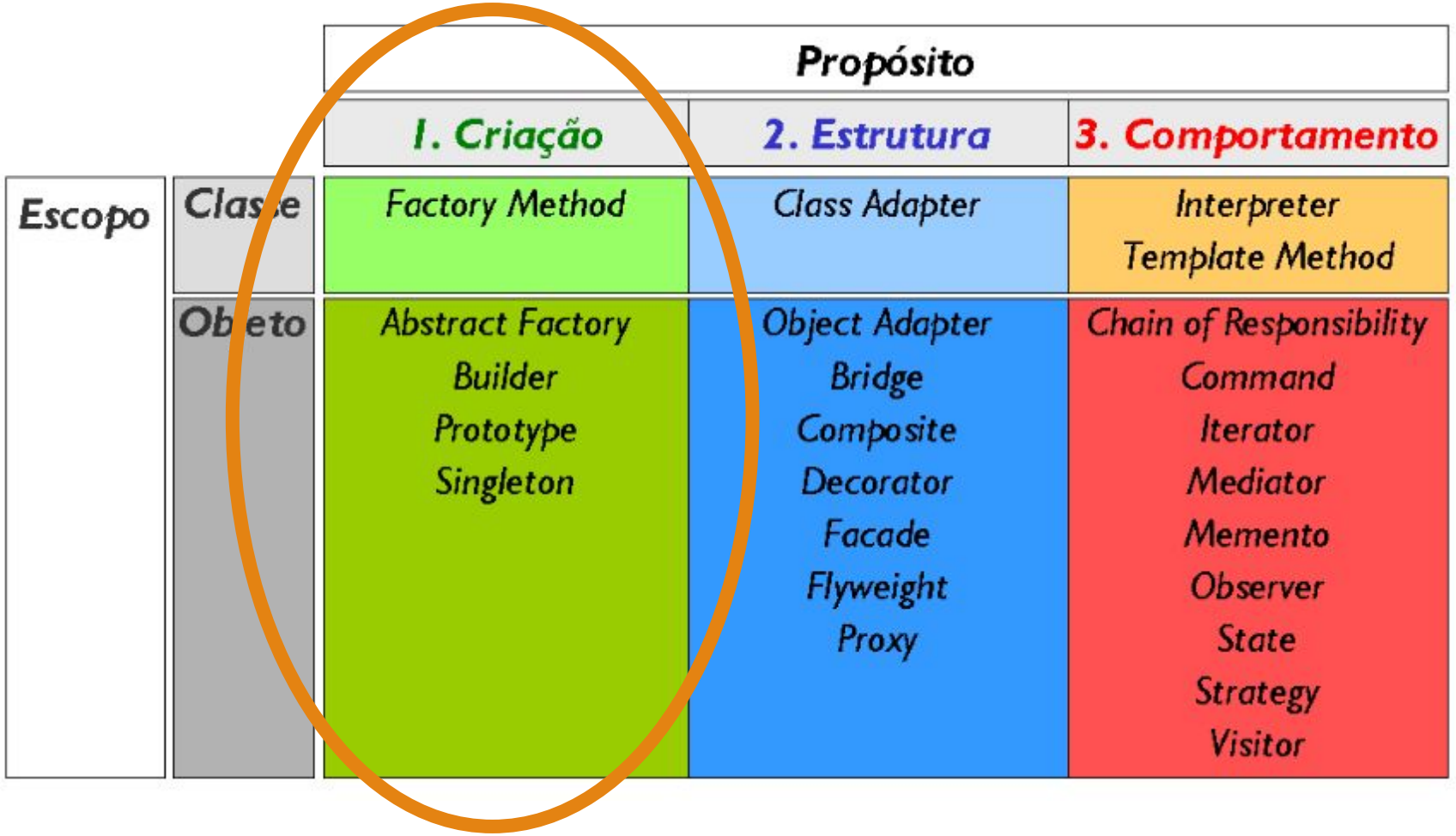
Factory Method x Abstract Method

- Abstract Factory é semelhante ao padrão Factory Method
    - Porém, em vez do cliente chamar um método de criação (Factory Method), ele possui um objeto de criação (Abstract Factory) e o usa para chamar os métodos de criação
- Onde Factory Method quer que você seja diferente (via herança) para criar objetos diferentes, o Abstract Factory quer que você tenha algo diferente (via interface ou classe abstrata)

Aula de Hoje



Padrões GoF – Padrões de Criação



Escopo		Propósito		
		1. Criação	2. Estrutura	3. Comportamento
	Classe	Factory Method	Class Adapter	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Object Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Builder

“Separar a construção de um objeto complexo de sua representação para que o mesmo processo de construção possa criar representações diferentes."

Builder - Problema

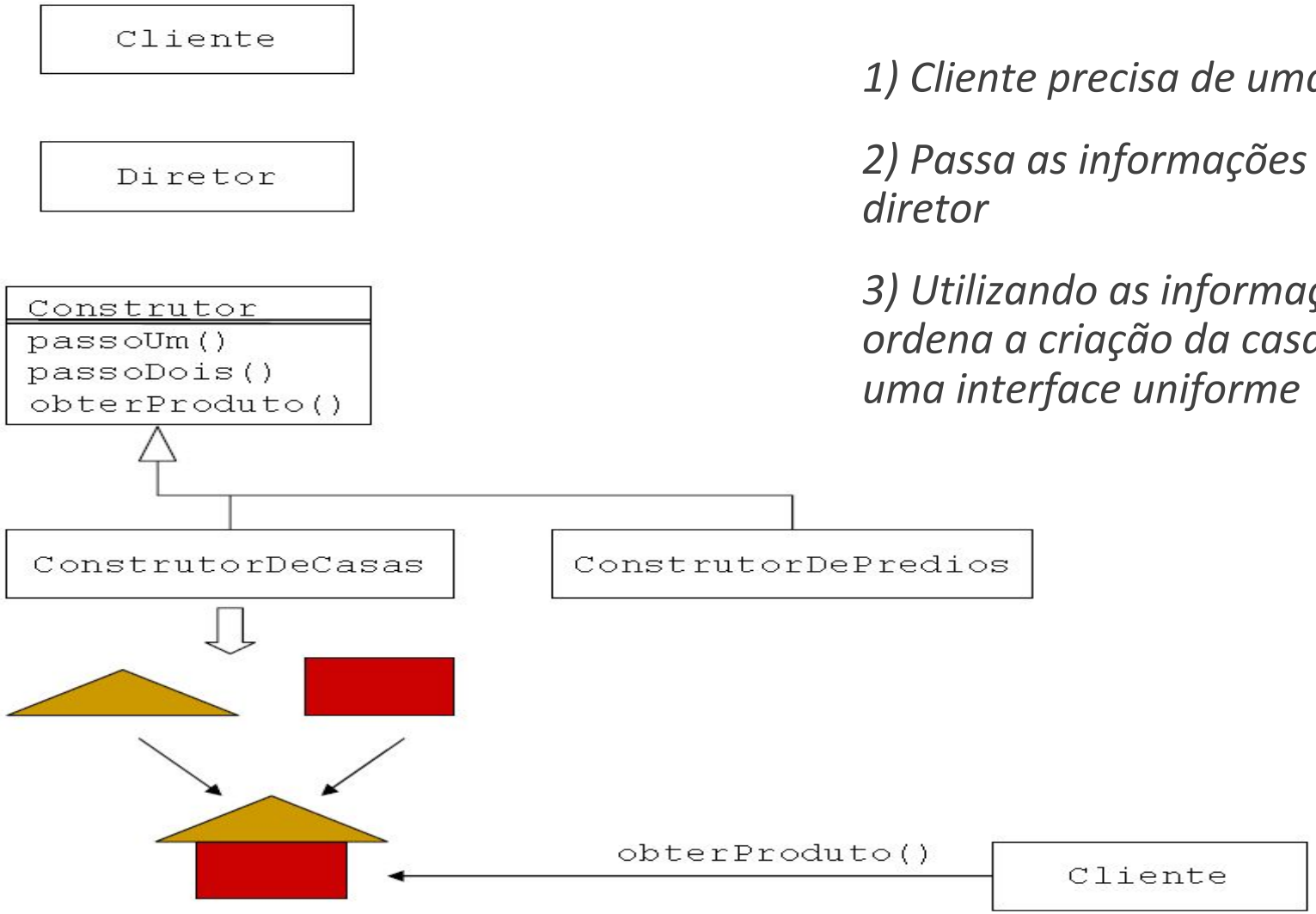
- Uma classe deve se preocupar com apenas uma parte da construção de um objeto.
  - É útil em algoritmos de construção complexos.

- O algoritmo para criar um objeto complexo precisar ser independente das partes que compõem o objeto e da forma como o objeto é construído.

- O projeto deve suportar a substituição dos construtores, permitindo que a mesma interface seja usada para construir representações diferentes dos mesmos dados.

- O processo de construção precisar suportar representações diferentes do objeto que está sendo construído.

Builder - Exemplo

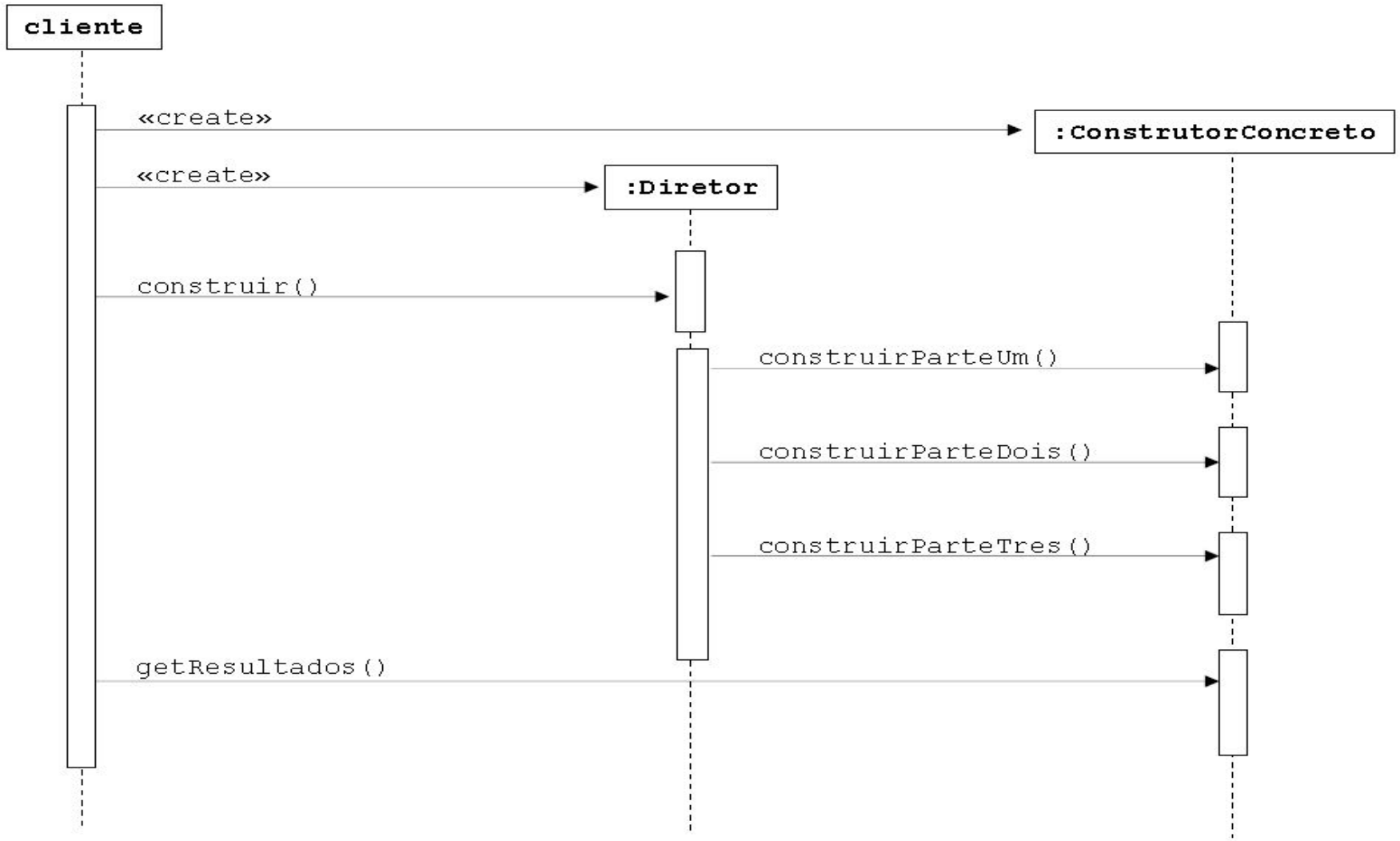


1) Cliente precisa de uma casa.

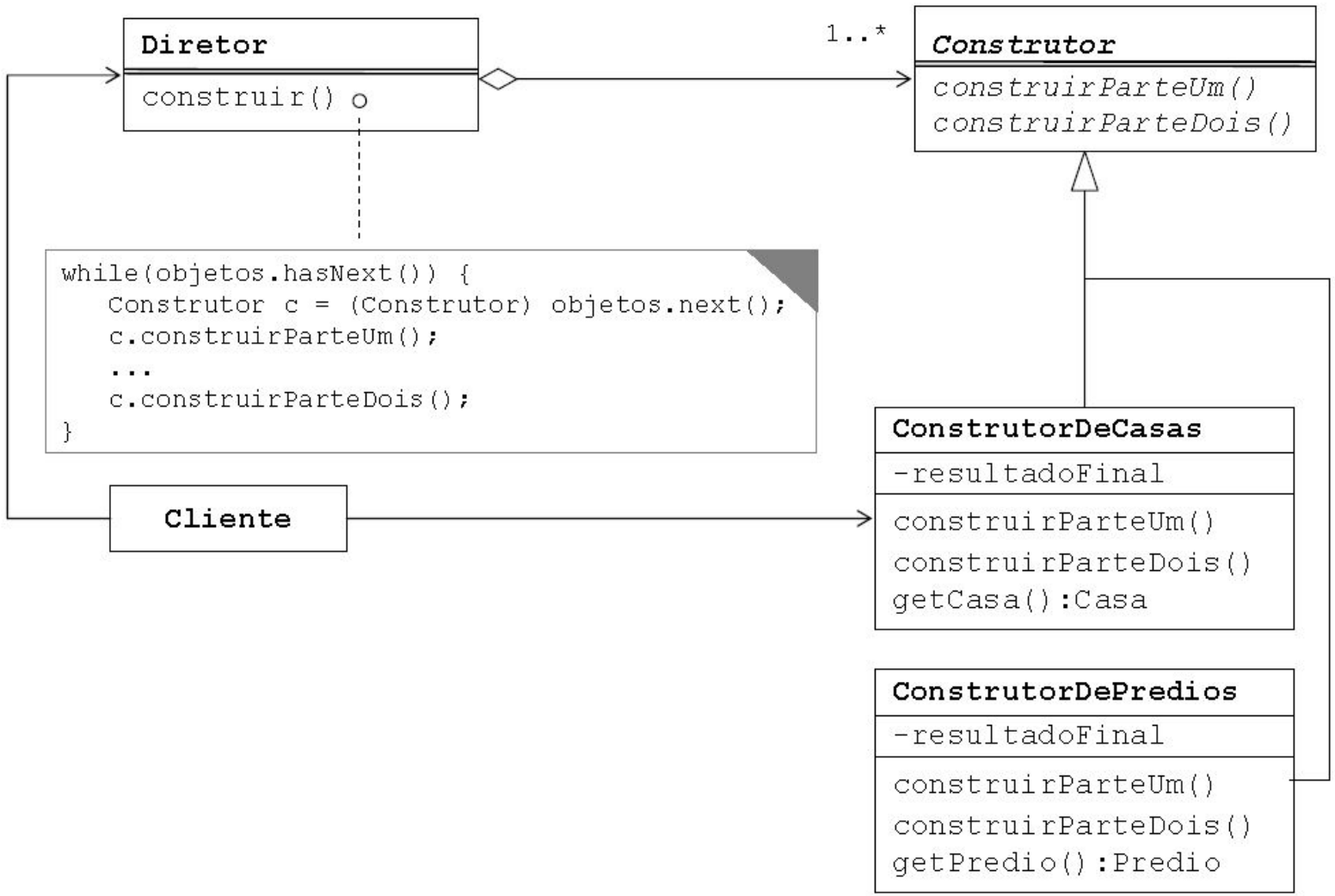
2) Passa as informações necessárias para seu diretor

3) Utilizando as informações passadas pelo cliente, ordena a criação da casa pelo construtor usando uma interface uniforme

Builder - Exemplo



Builder - Exemplo



Builder - Exemplo

Modele um sistema de venda de carros para uma concessionária. Queremos que o sistema seja flexível, para adição de novos carros, e de fácil manutenção. Para isto, utilize o padrão Builder.

Builder - Solução

- Separar a construção de um objeto complexo da sua representação, de forma que o mesmo processo possa criar diferentes tipos de representações
- Use Builder quando:
  - O algoritmo para criar um objeto deve ser independente de suas partes e de como elas são montadas
- Dica: enquanto Abstract Factory e o Factory Method enfatizam famílias de objetos, Builder constrói partes de objetos passo a passo.

Vamos retomar o exemplo do labirinto!

- Implemente uma arquitetura base para a construção do jogo do labirinto encantado da aula passada.
- Desta vez , utilize o padrão Builder para construir os cenários e compare as saídas.

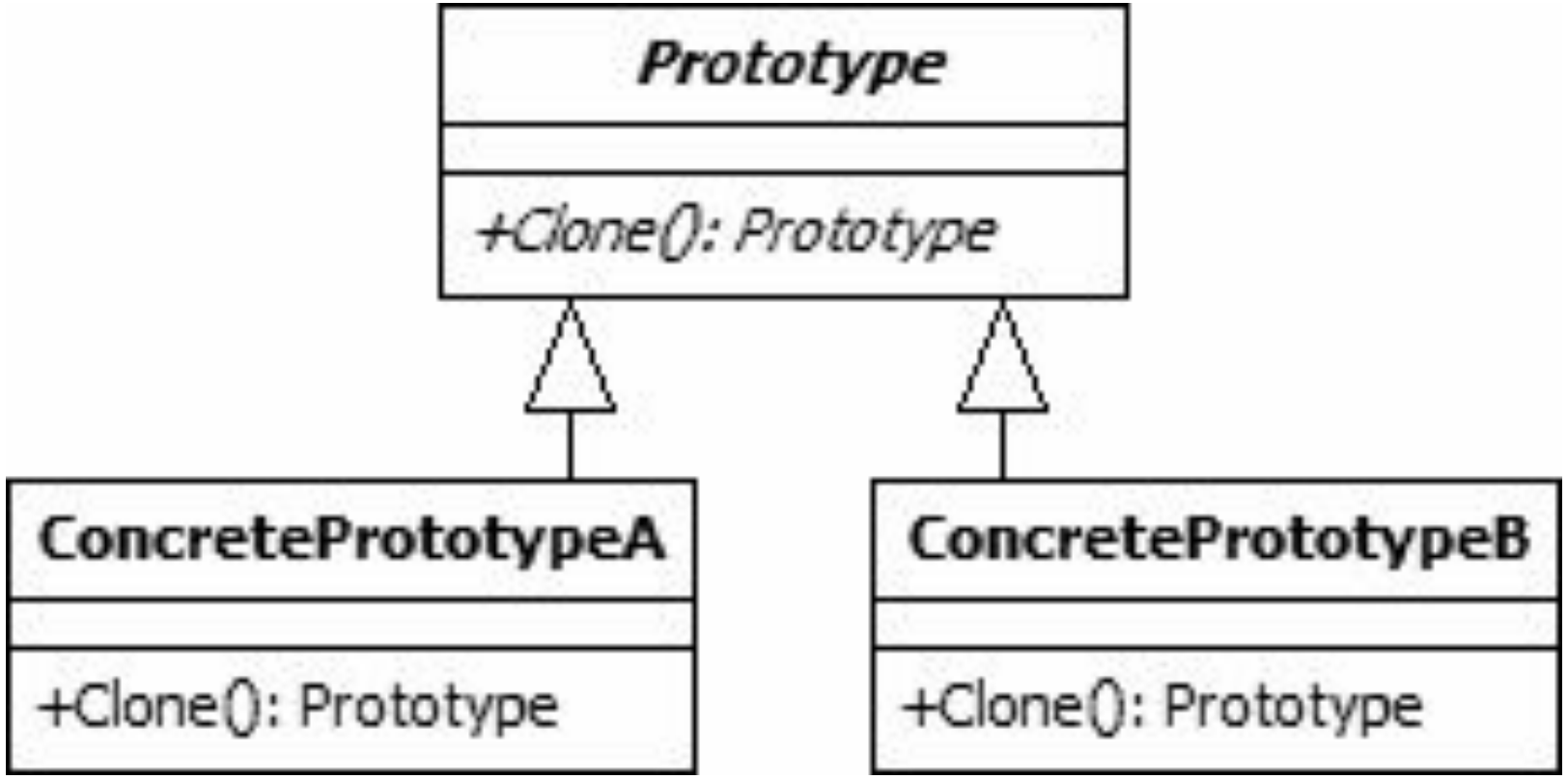
Prototype

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

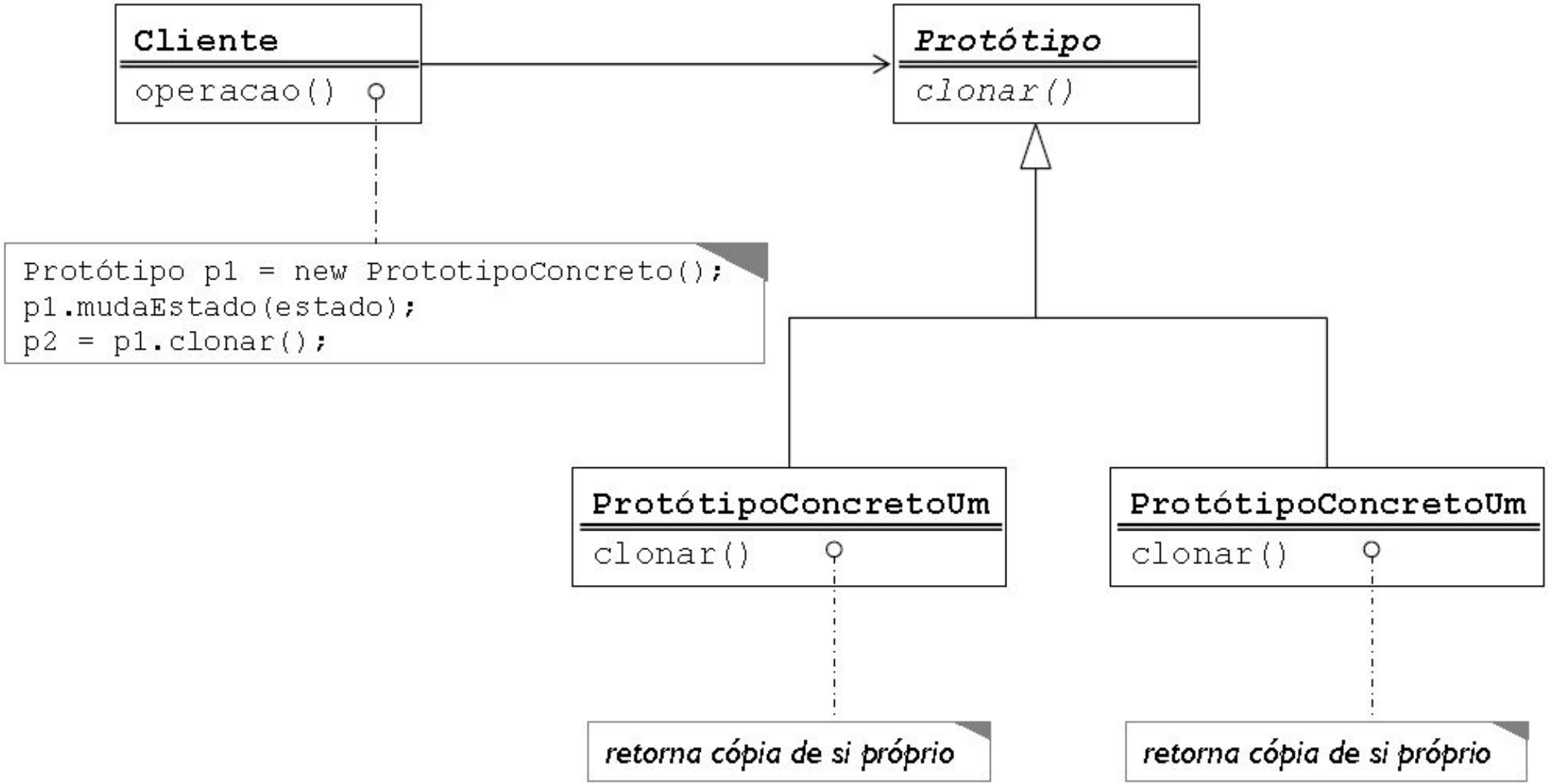
Prototype - Problema

- Precisamos criar novos objetos a partir de uma instância protótipo, que vai realizar uma cópia de si mesmo e retornar para o novo objeto.
- Criar um objeto novo, mas aproveitar o estado previamente existente em outro objeto (Clone).

Prototype - Solução



Prototype - Solução

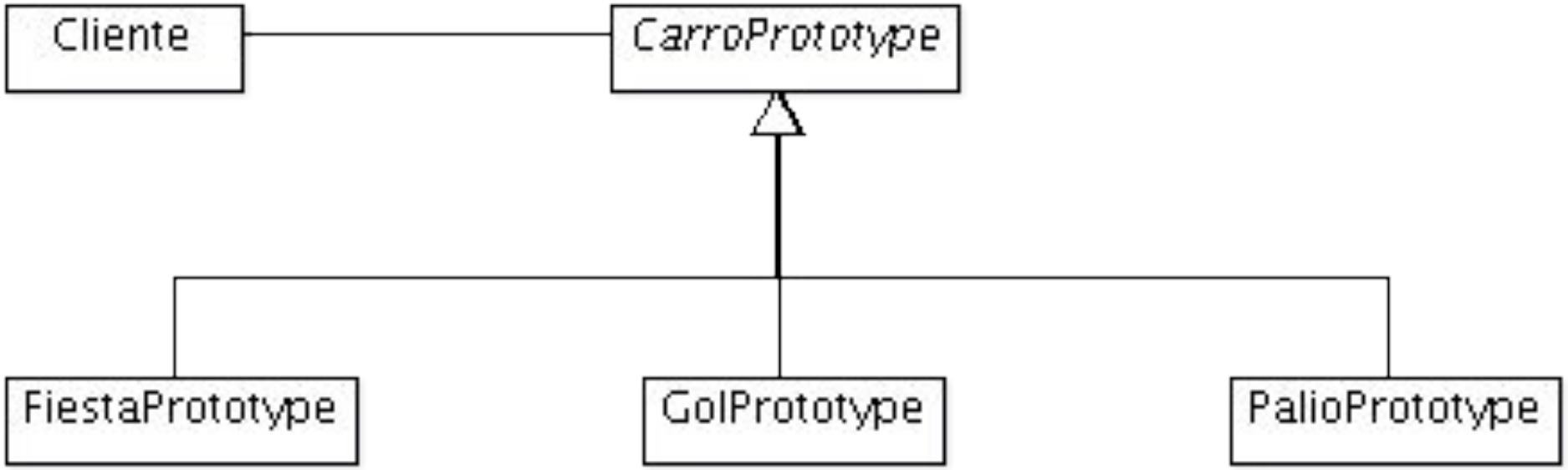


Prototype - Exemplo

- O problema consiste em uma lista de carros que o cliente precisa utilizar, mas que só serão conhecidos em tempo de execução.
- Vamos analisar então como o problema pode ser selecionado utilizando o padrão Prototype.

Prototype - Exemplo

- O problema consiste em uma lista de carros que o cliente precisa utilizar, mas que só serão conhecidos em tempo de execução.
- Vamos analisar então como o problema pode ser selecionado utilizando o padrão Prototype.



Prototype

- Quando um objeto é clonado, o novo objeto é: uma cópia superficial ou profunda.
- Cópia superficial:
  - Duplica todas as propriedades do objeto.
  - De qualquer propriedade contém um tipo de referência, a referência é copiada.
  - Isto significa que alterações no objeto referenciado são visíveis em ambos o clone e o objeto original.
- Cópia em profundidade:
  - Clona o objeto principal e todos os objetos filho.
  - Todas as propriedades de tipos de referência também são clonados, dando uma cópia verdadeiramente independente.

Prototype

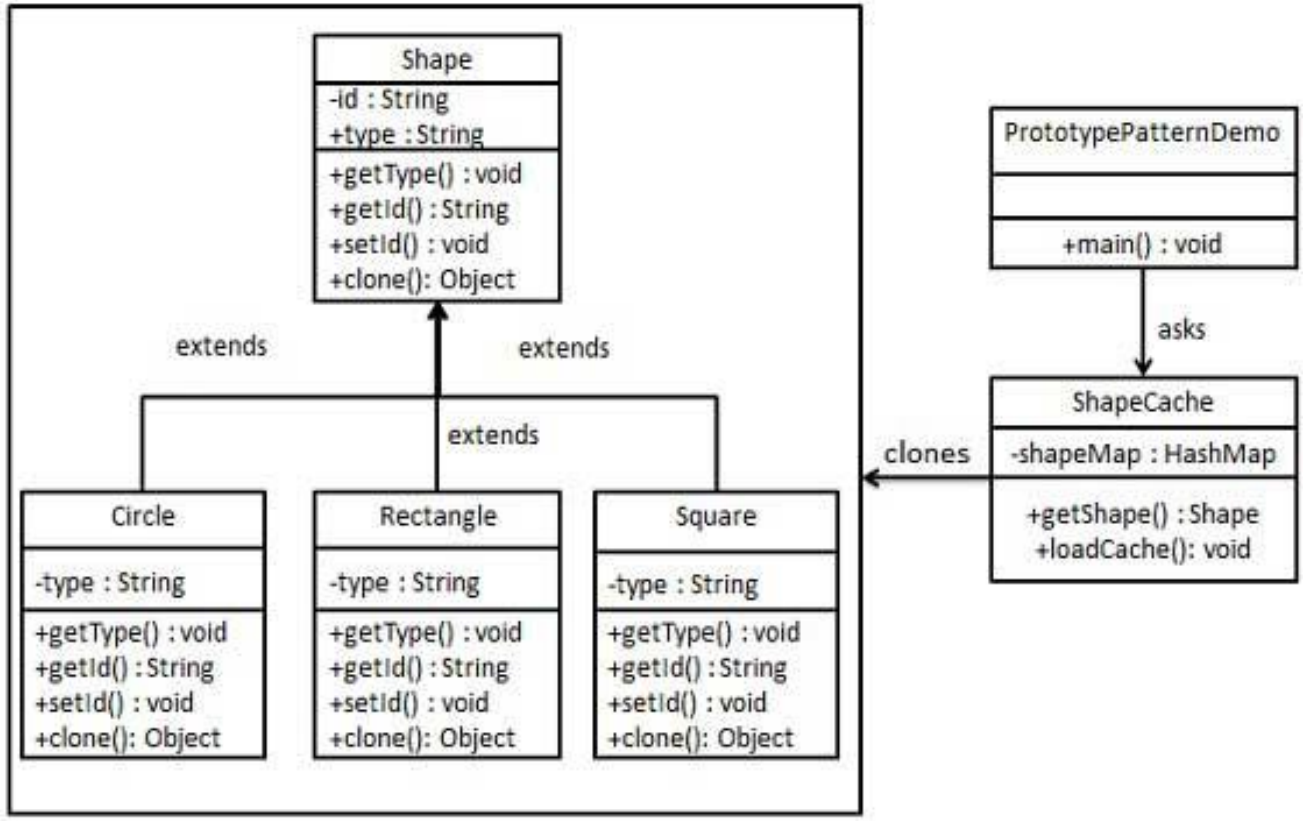
- Quando um objeto é clonado, o novo objeto é: uma cópia superficial ou profunda.
- Cópia superficial:
  - Duplica todas as propriedades do objeto.
  - De qualquer propriedade contém um tipo de referência, a referência é copiada.
  - Isto significa que alterações no objeto referenciado são visíveis em ambos o clone e o objeto original.
- **Cópia em profundidade:**
  - **Clona o objeto principal e todos os objetos filho.**
  - **Todas as propriedades de tipos de referência também são clonados, dando uma cópia verdadeiramente independente.**

Prototype - Vantagem

- O padrão Prototype permite que um cliente crie novos objetos ao copiar objetos existentes
- Uma vantagem de criar objetos deste modo é poder aproveitar o estado existente de um objeto
- Esta prática é particularmente útil quando a qualidade da construção de um novo objeto, utilizando o novo operador, é ineficiente.

Exercício

- Crie uma classe abstrata Shape e classes concretas estendendo a classe Shape. Uma classe ShapeCache deve ser definida como uma próxima etapa que armazena objetos de forma em uma Lista (sugestão HashMap) e retorna seu clone quando solicitado.
- Por fim, implemente a classe PrototypPatternDemo para demonstração, de acordo com o diagrama ao lado.



Dúvidas?

alanamm.prof@gmai.com