Diagrama de Seqüência

Características Básicas

- É um **diagrama de interação** e considera **aspectos dinâmicos**;
- Define como os objetos **colaboram**;
- Ênfase na **ordenação temporal**;
- Captura o comportamento de um **cenário único**;
- Mostra **objetos** e as **mensagens** trocadas entre eles, em um caso de uso;
- São usados durante a análise do **problema** (alto nível, com mensagens de negócio) e/ou durante o desenho da **solução** (foco na implementação, com mensagens descritas na forma de métodos que são implementados pelas classes.

Categorias de Objetos

- Objetos participantes (atores);
- Objetos de interface com os atores (telas, etc);
- Objetos de controle e eventos (controladores);
- Objetos de entidades e de domínio (negócio e integração).

Objetos Participantes (Atores)

- **Pessoa** ou outro **sistema** que interage com o sistema enviando e recebendo dados e controles;
- Demostram os contratos com o sistema;
- As interações são capturadas nos Casos de Uso;

Objetos de Interface

- Objetos que fazem a **comunicação** entre o sistema e os objetos participantes (**atores**), para exibição ou coleta de dados;
- Em um sistema típico, é a interface gráfica com o usuário – GUI;
- Para aplicações Web baseadas em JEE, podem ser páginas JSP (usando ou não um framework) ;
- Para aplicações desktop em Java, podem ser instâncias de classes que usam Swing;
- São classes com estereotipo <<boundary>>.

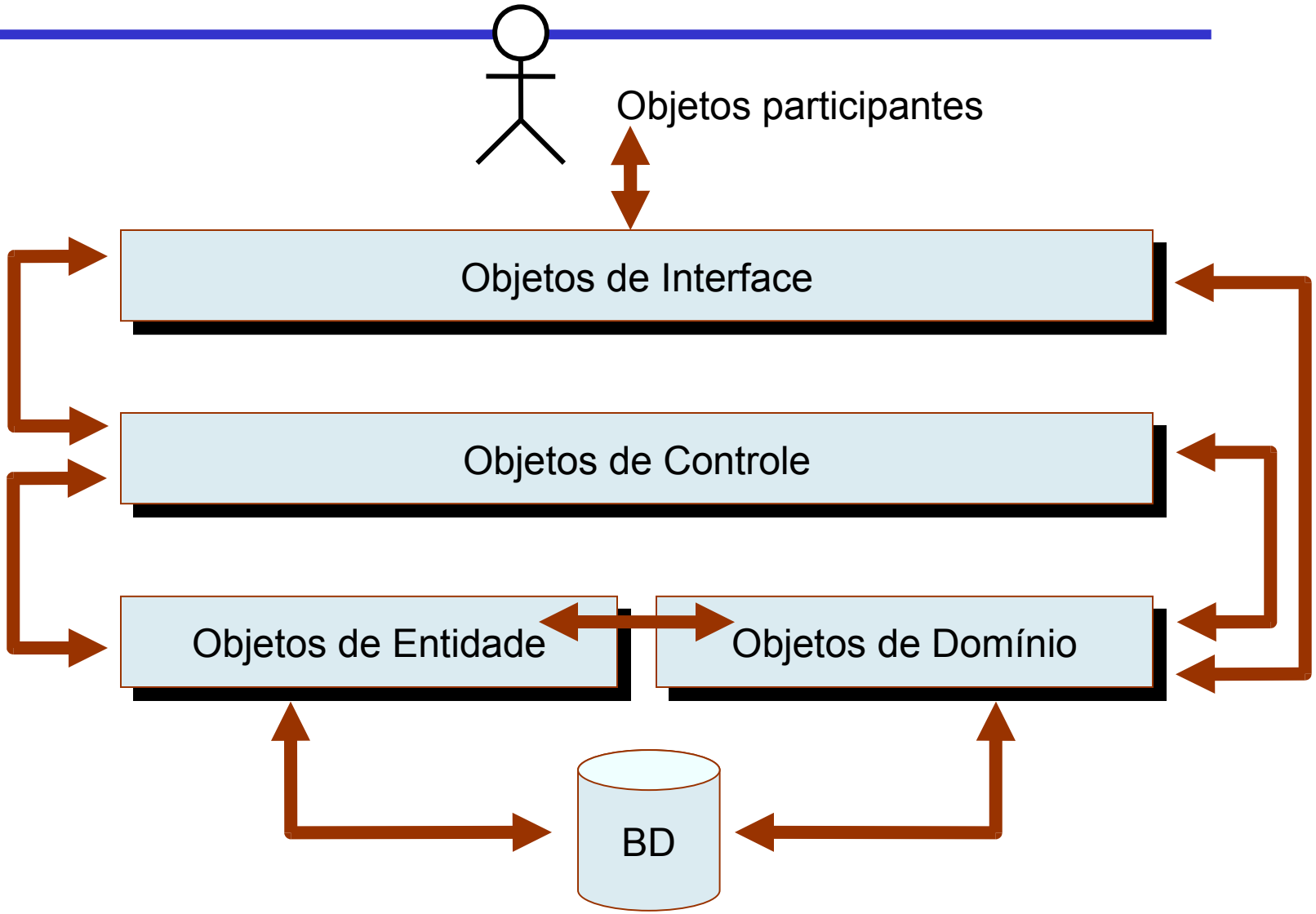
Objetos de Controle

- **Coordenam** as solicitações dos usuários, vindas da camada de apresentação, e as operações de responsabilidade dos objetos de entidade e de domínio;
- São classes com estereótipo <<control>>.

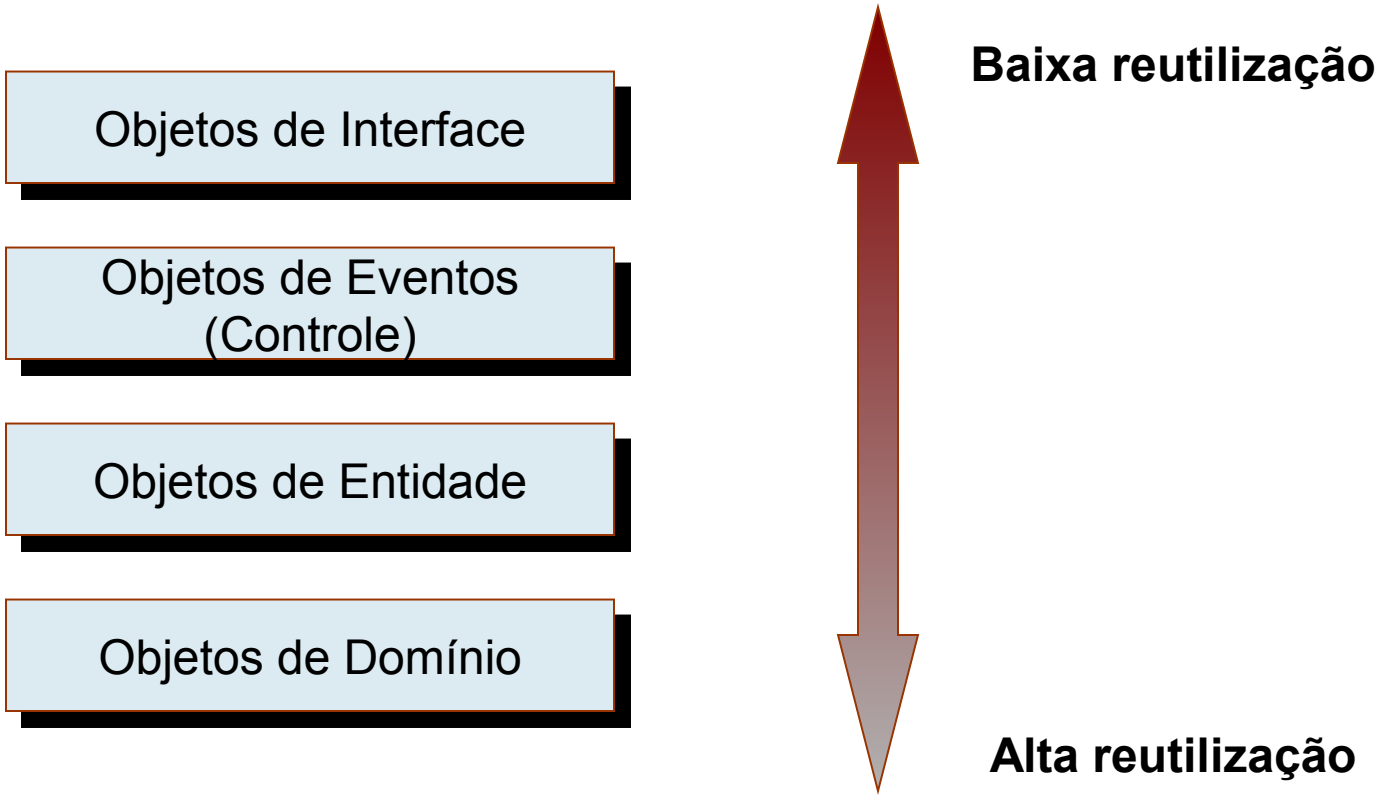
Objetos de Entidades e de Domínio

- **Armazenam** e **manipulam dados** dos objetos do **negócio** e simulam o seu **comportamento**;
- São as classes de negócio da aplicação, levantadas de acordo com o domínio e vocabulário do problema;
- Definem os valores que um atributo pode assumir e as operações válidas para estes valores;
- Classes da arquitetura de persistência em banco de dados;
- Integram as bibliotecas das linguagens ou *frameworks* de persistência;
- Podem ser tipos de atributos ou classes de comunicação com o mecanismo de persistência (em uma aplicação Java típica, podem ser os Data Access Object – DAO).
- Classes com estereótipo <<entity>>

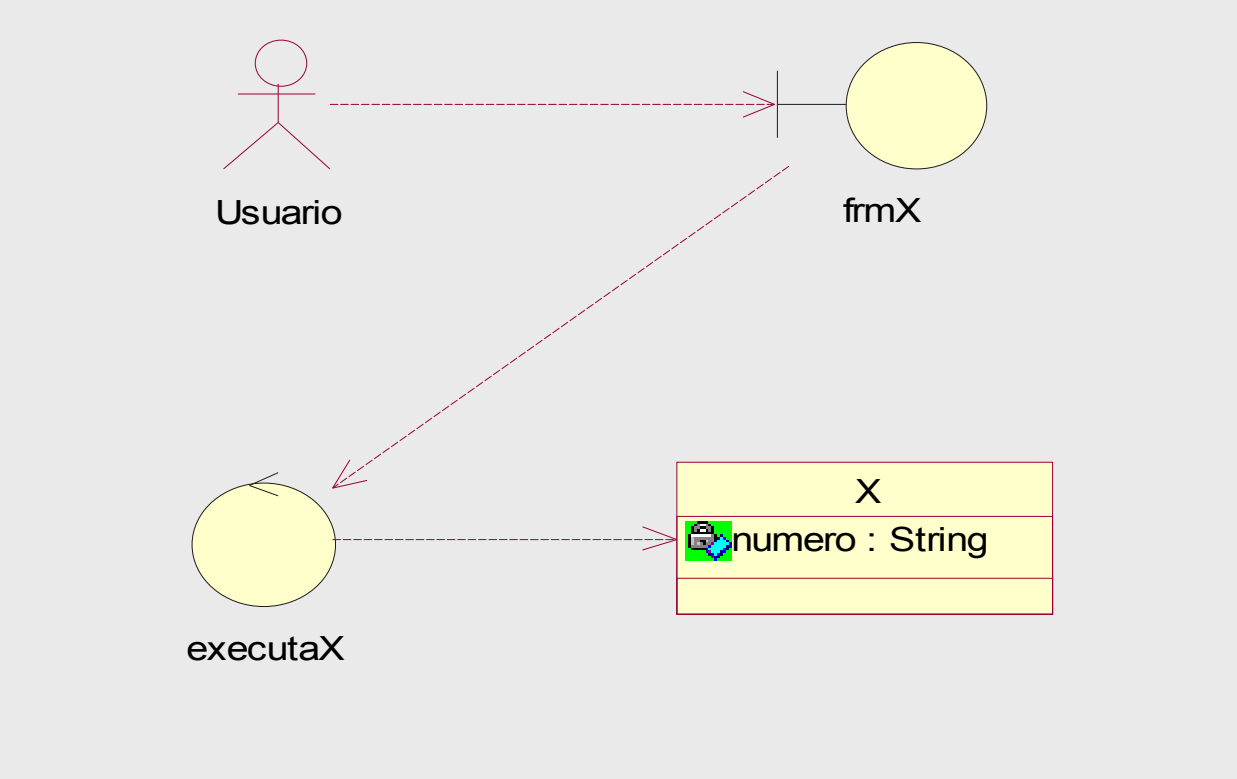
Visão das categorias



Grau de reutilização



Representação dos Objetos



Realização de Casos de Uso

- Os Casos de Uso capturam “**o que**” deve ser feito, as realizações agrupam o “**como**” será feito;
- São instâncias do caso de uso que podem ser representadas por diagramas que mostram a realização;
- Os diagramas usados para representar a realização de um caso de uso são os diagramas de interação.

Diagrama de Seqüência

- Descreve o que ocorre nos objetos participantes, em termos de chamadas, e como os objetos se comunicam através de mensagens;
- As mensagens representam as interações entre as instâncias de objetos e entre estas e os atores;
- Normalmente utilizado para modelar a realização dos casos de uso;
- Contém atores, objetos e mensagens;
- As notas oferecem uma extensão semântica.

Diagrama de Seqüência - Semântica

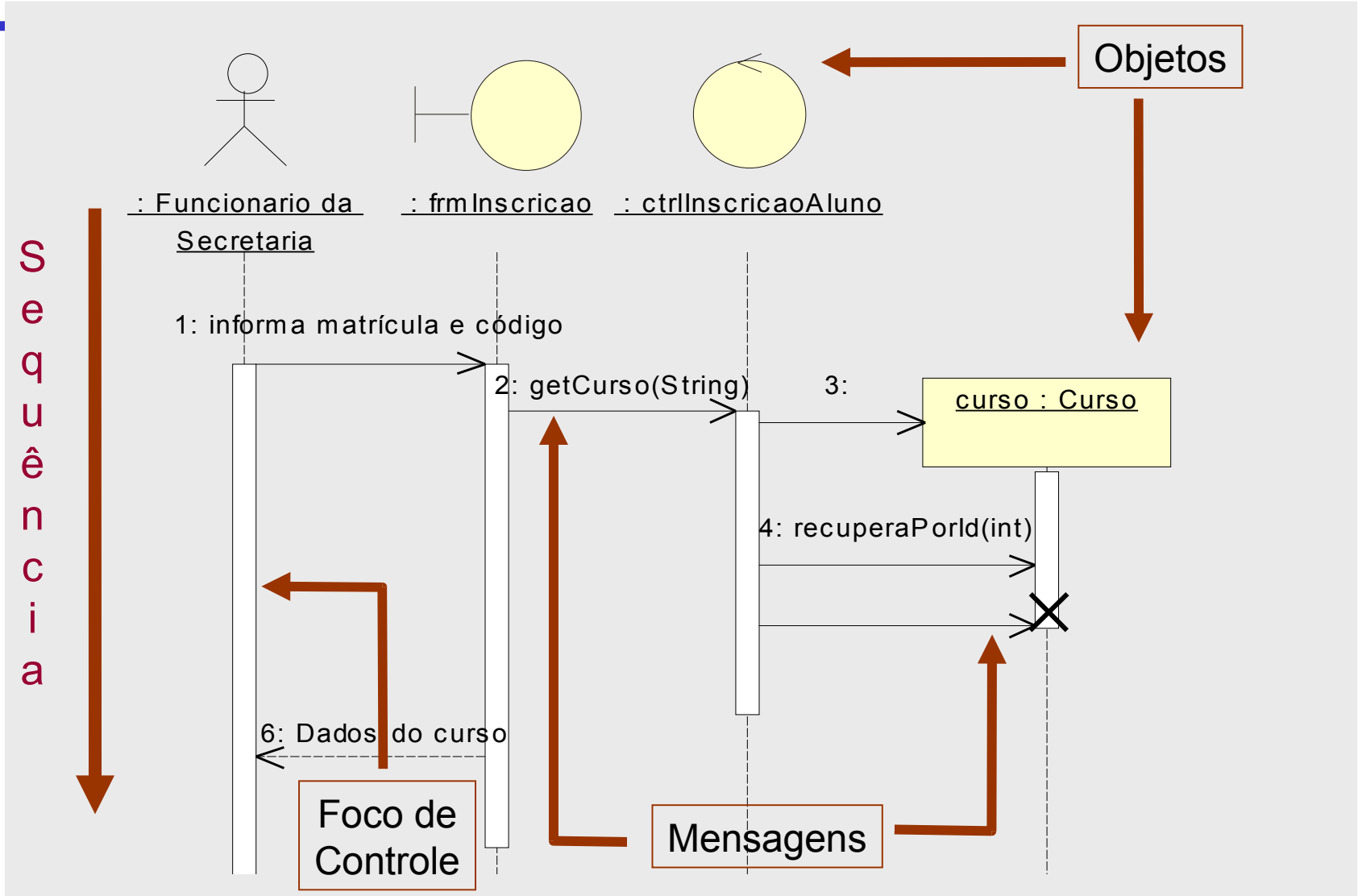


Diagrama de Seqüência e Realização de Casos de Uso

- **Organização:**
  - **Um diagrama para o fluxo básico;**
  - **Um diagrama para cada fluxo alternativo.**
- **Para os analistas e usuários, provê uma visão de alto nível sobre como se dará a interação dentro de um cenário de caso de uso;**
- **Para os projetista, provê as entradas básicas que auxiliarão na determinação das classes (e nestas quais os métodos), responsabilidade e interfaces.**

Diagrama de Seqüência - Tipos de Diagrama

- O diagrama de seqüência poderá ser feito em vários momentos do sistema, onde o formalismo dos objetos e mensagens poderá variar.
- Momentos:
  - Análise;
  - Projeto.

Diagrama de Seqüência - Análise

- Captura a visão do analista sobre como o caso de uso vai ser realizado;
- Direciona a solução do projetista.
- Contém:
  - boundary, controler, entity;
  - Mensagens informais.

Diagrama de Seqüência - Análise

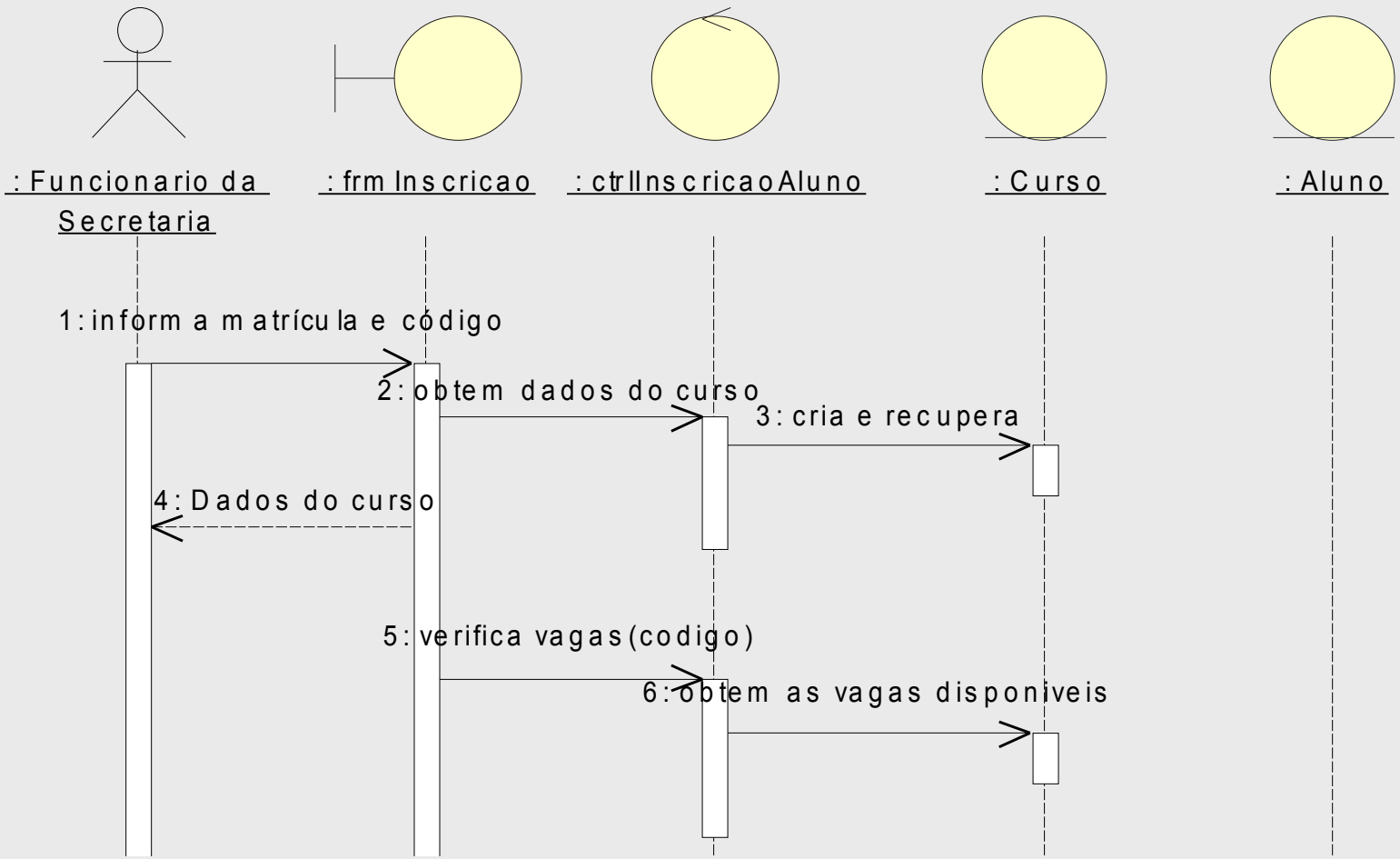
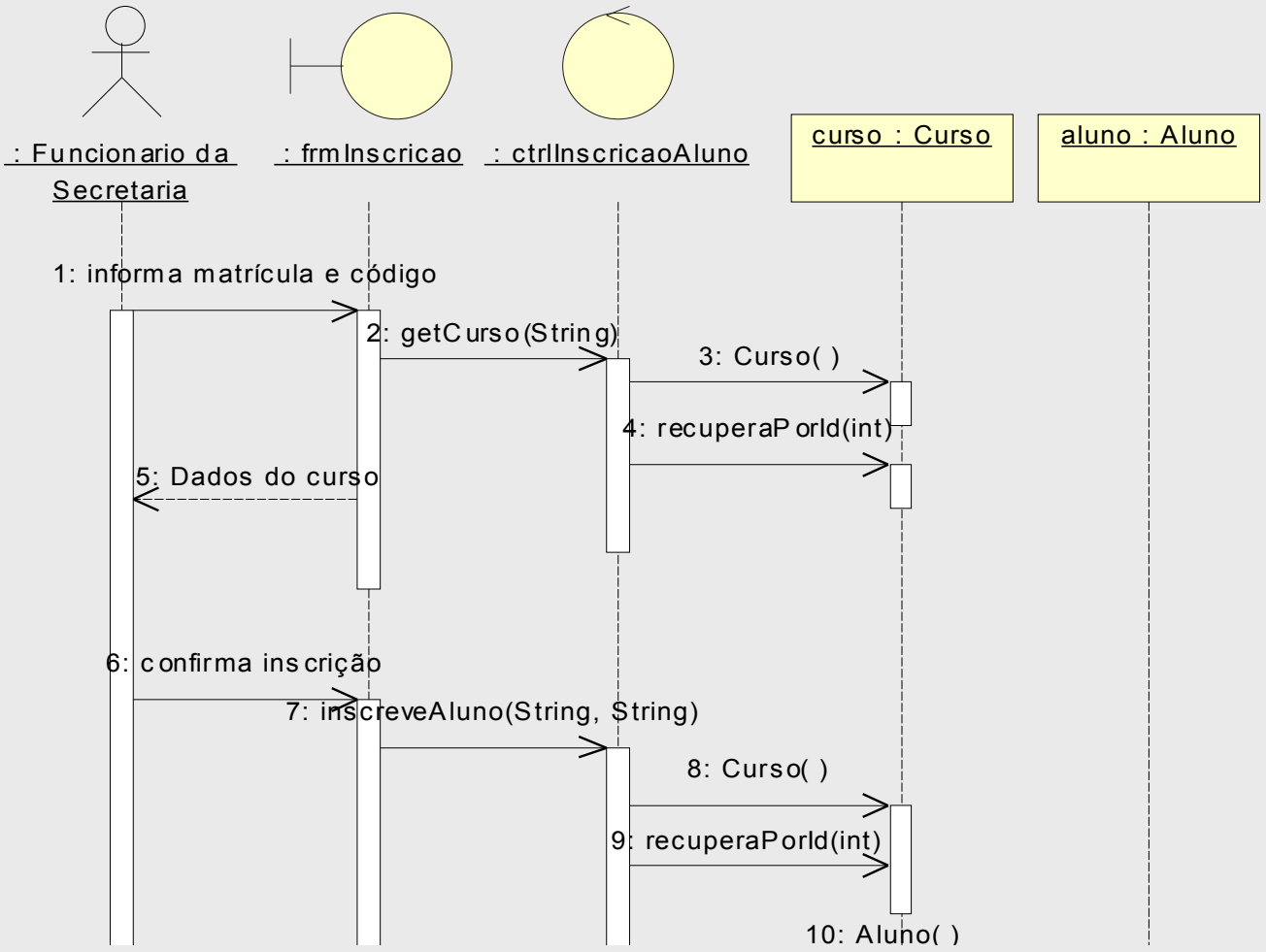


Diagrama de Seqüência - Projeto

- É a visão final da realização do caso de uso;
- Incorpora elementos da linguagem e da arquitetura;
- É o roteiro para a escrita do programa.
- Contém:
  - Atores, estereótipos de camadas (interface, servlet), objetos de domínio (além dos demais);
  - Nomes e mensagens formais (métodos existentes nas classes).

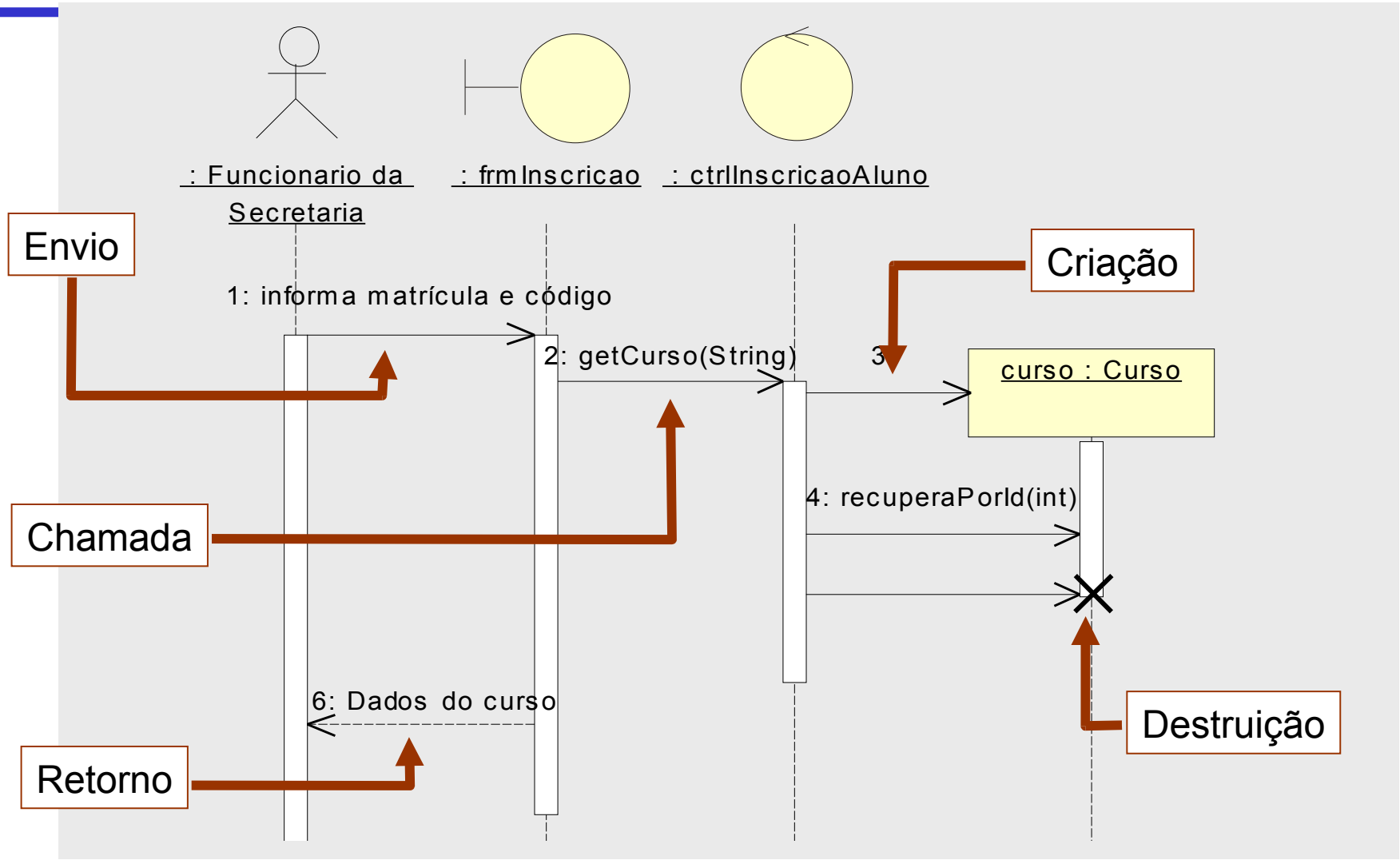
Diagrama de Seqüência - Projeto



Tipos de mensagem

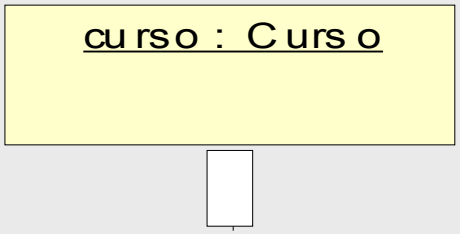
- **Chamada:**
  - Um objeto invoca uma operação em outro objeto.
- **Auto-chamada:**
  - Um objeto invoca uma operação nele mesmo.
- **Envio:**
  - Um objeto envia um sinal a outro.
- **Retorno:**
  - Retorna um valor para o objeto solicitante.
- **Criação:**
  - Cria um objeto.
- **Destruição:**
  - Destrói um objeto.

Tipos de mensagem



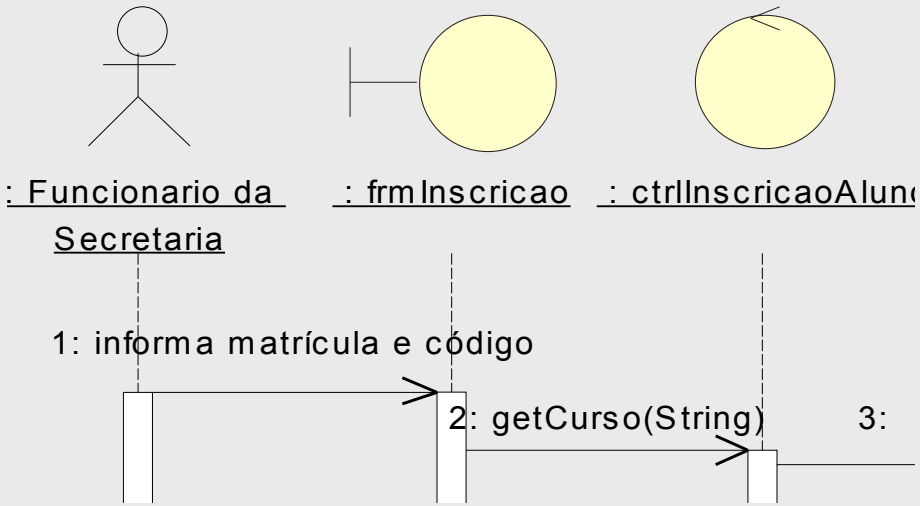
Representando os objetos

- **Nos diagramas de seqüência, geralmente os objetos representados são instâncias reais da classe;**
- **Os nomes serão os mesmos a serem escritos nos programas.**



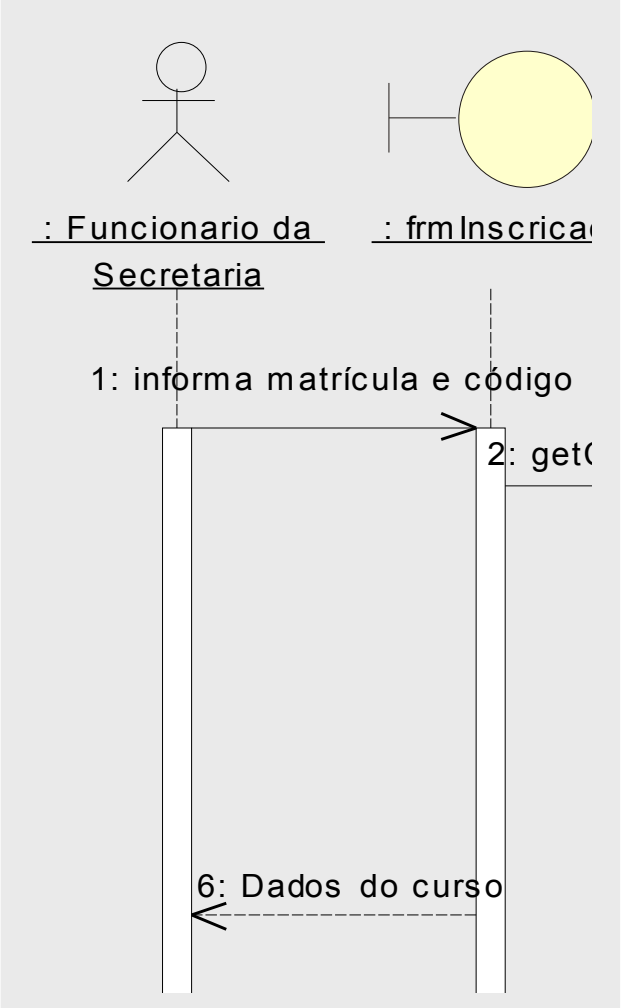
Mensagens de Chamada

- **Mensagens normais que invocam métodos em outros objetos;**
- **Os parâmetros devem ser representados pelos nomes e os tipos reais.**



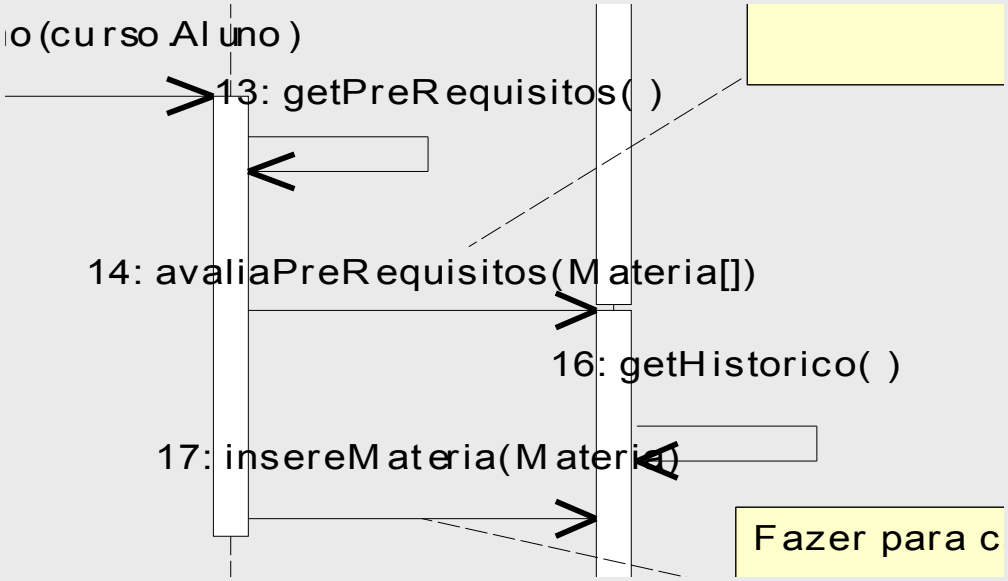
Mensagens de Retorno

- **Mensagens de retorno devem ser representadas apenas quando não forem óbvias para não poluir o diagrama.**



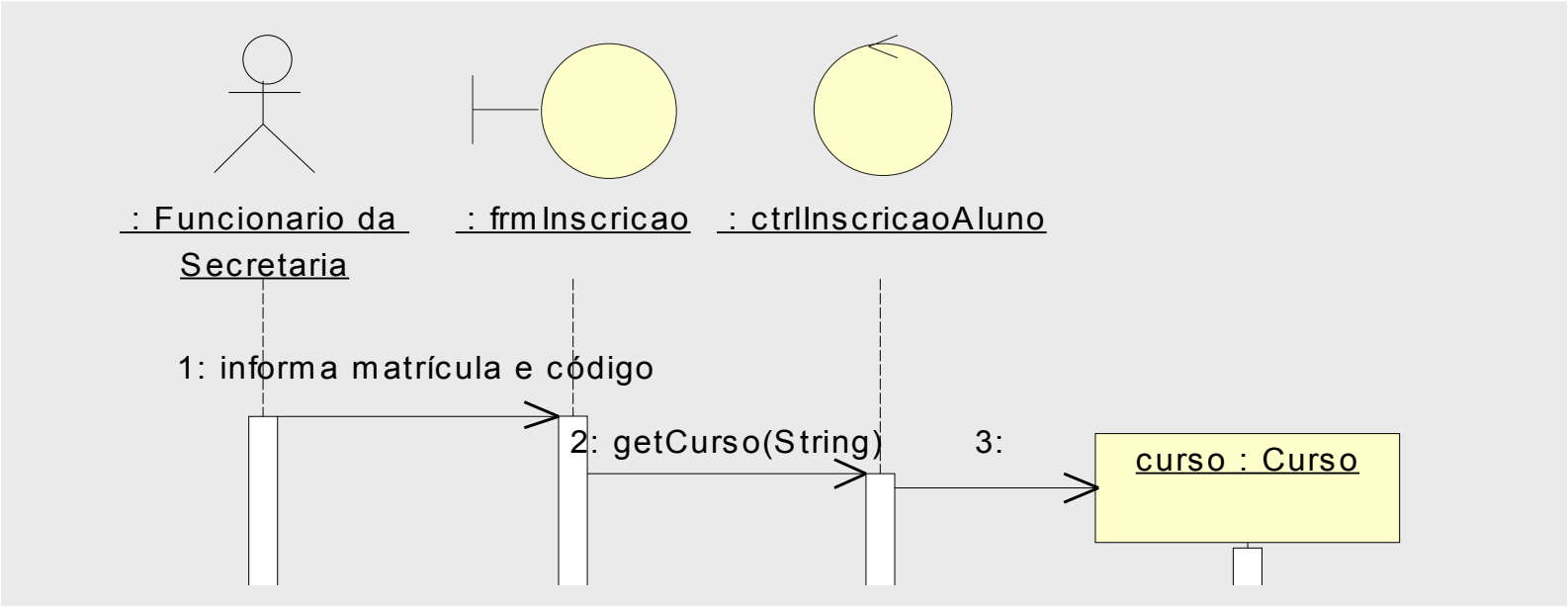
Mensagem *self*

- Mensagens de que o objeto envia para ele próprio.



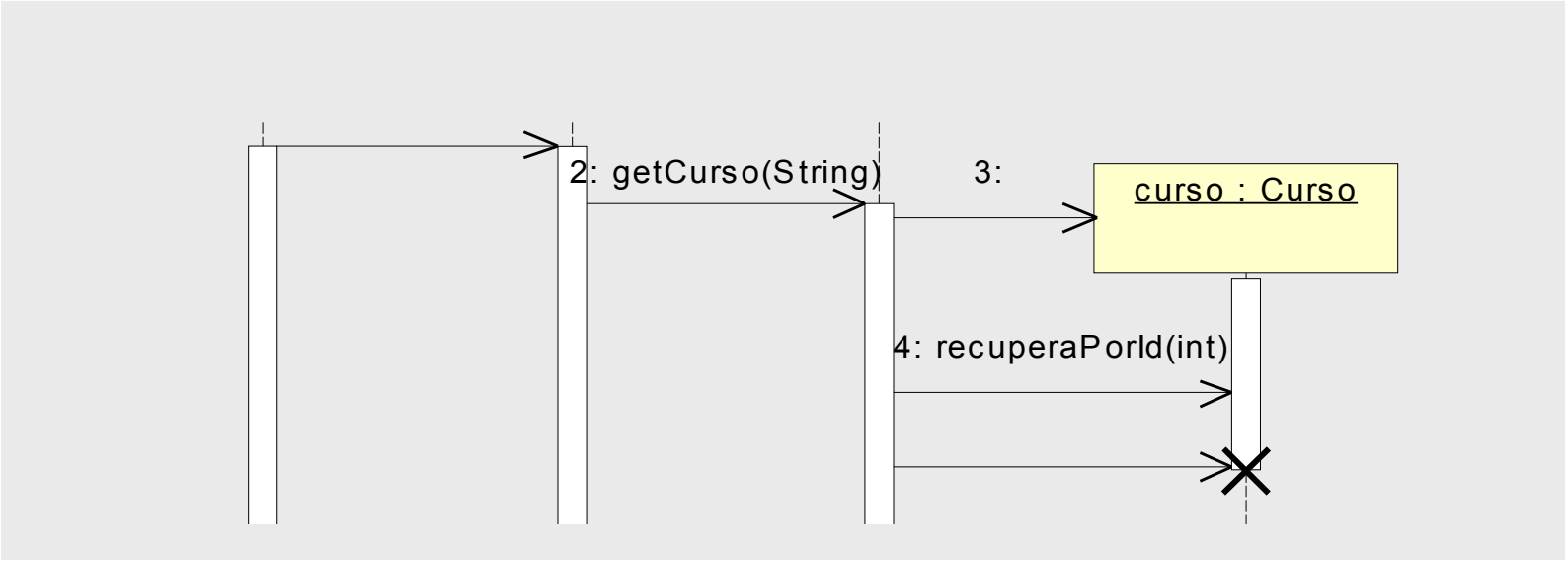
Mensagens de Criação

- **Representam o momento em que o objeto será criado;**
- **Se o construtor *default* for utilizado, não é preciso representá-lo.**

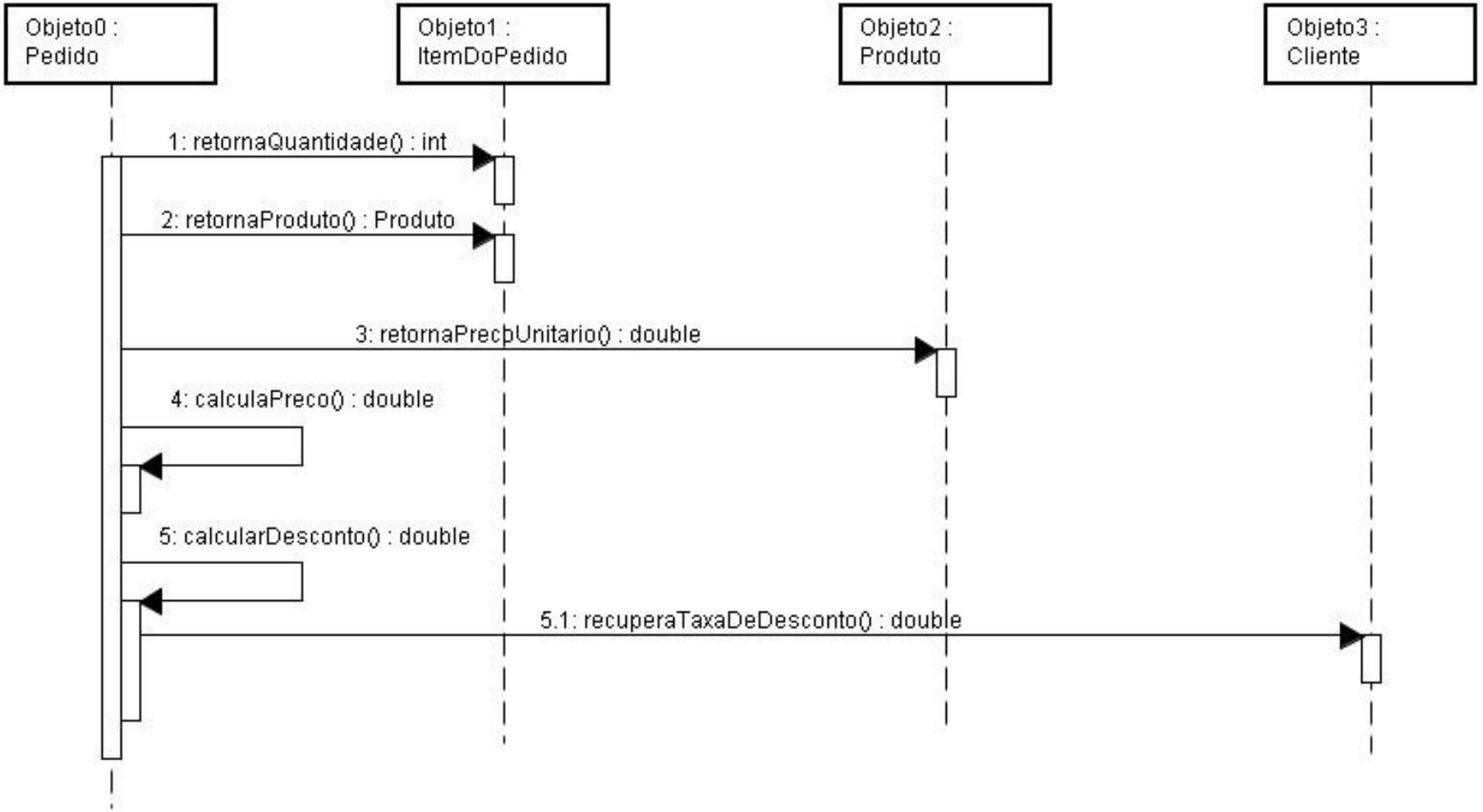


Mensagens de Destruição

- Indica a mensagem de destruição do objeto.
- Não é necessária a representação para a maioria das linguagens de programação.



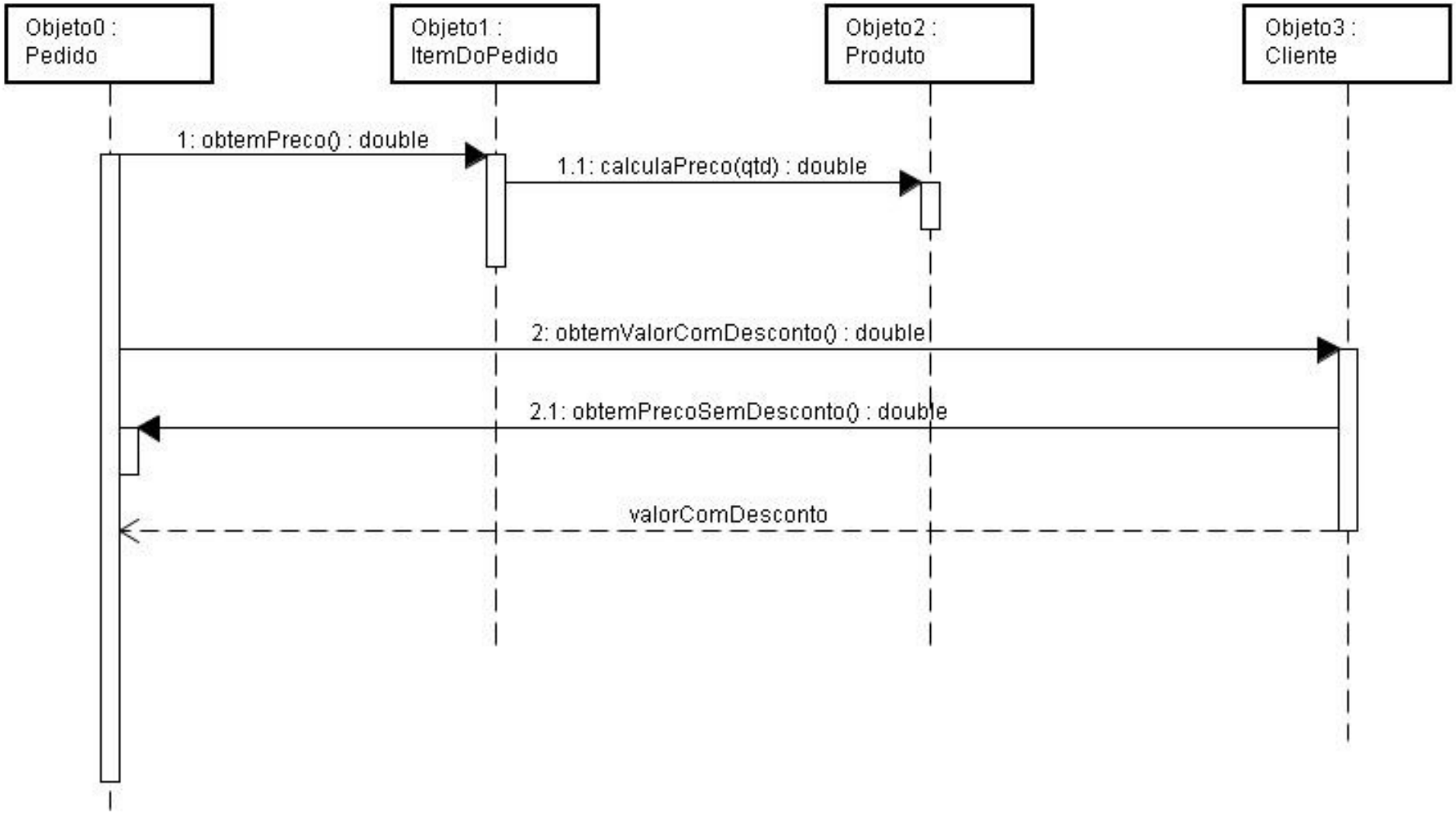
Exemplos


Objeto0 : Pedido
Objeto1 : ItemDoPedido
Objeto2 : Produto
Objeto3 : Cliente
1: retornaQuantidade() : int
2: retornaProduto() : Produto
3: retornaPrecoUnitario() : double
4: calculaPreco() : double
5: calcularDesconto() : double
5.1: recuperaTaxaDeDesconto() : double

Exemplos

```
public class Pedido {

    private ItemDoPedido[] itensDoPedido;
    private Cliente cliente;

    public double calculaPrecoFinalItem() {
     int qtd = ItemDoPedido[0].retornaQuantidade();
     Produto produto = ItemDoPedido[0].retornaProduto();
     double precoProduto = produto.retornaPrecoUnitario();
     double precoBaseItem = this.calculaPreco(precoProduto, qtd);
     double taxaDesconto = Cliente.recuperaTaxaDesconto();
     double precoFinalItem = this.calculaPrecoFinal(precoBaseItem, taxaDesconto);
     return precoFinalItem;
    }

    public double calculaPreco(double precoProduto, int qtd) {
     return precoProduto * qtd;
    }

    public double calculaPrecoFinal(double precoBaseItem, double taxaDesconto) {
     return precoBaseItem * (1 - taxaDesconto);
    }

}
```

Exemplos


Objeto0 : Pedido
Objeto1 : ItemDoPedido
Objeto2 : Produto
Objeto3 : Cliente
1: obtemPreco() : double
1.1: calculaPreco(qtd) : double
2: obtemValorComDesconto() : double
2.1: obtemPrecoSemDesconto() : double
valorComDesconto

Exemplos

```
public class Pedido {

    private ItemDoPedido[] itensDoPedido;
    private Cliente cliente;

    public double calculaPrecoFinalItem() {

     double precoBaseItem = ItemDoPedido[0].obtemPreco();
     double precoFinalItem = Cliente.obtemValorComDesconto(precoBaseItem);

     return precoFinalItem;

    }

}
```