Desenvolvimento de Aplicações CORBA
(Invocação Estática)

Engenharia de Computação - IFCE

Cidcley Teixeira de Souza
cidcley@ifce.edu.br

A Aplicação

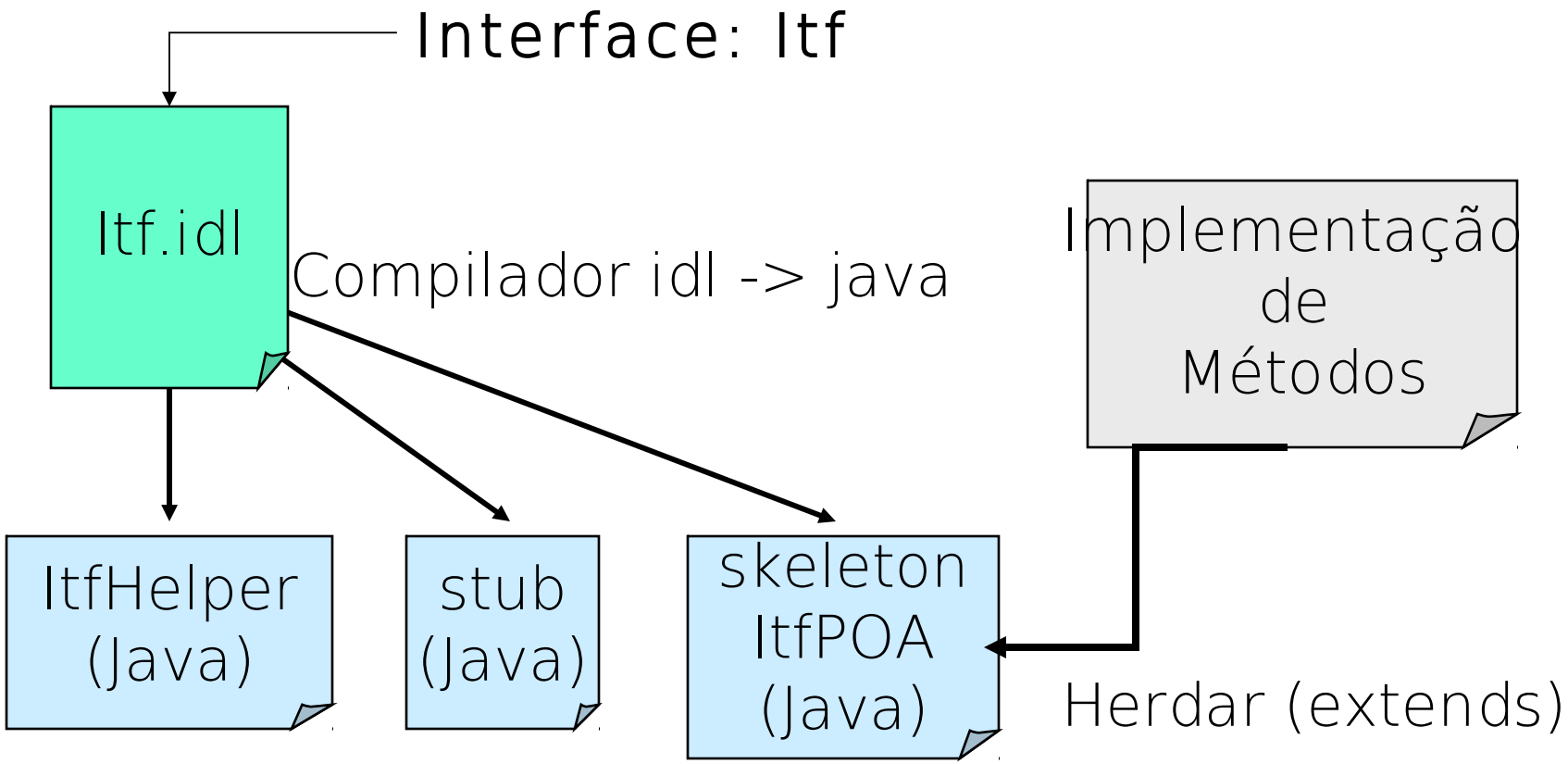
- Um Servidor oferecendo uma operação de soma e de divisão, onde deve receber dois argumentos reais e retornar o valor da operação;
- Um Cliente envia pedidos para as duas operações do servidor;

Software Usado

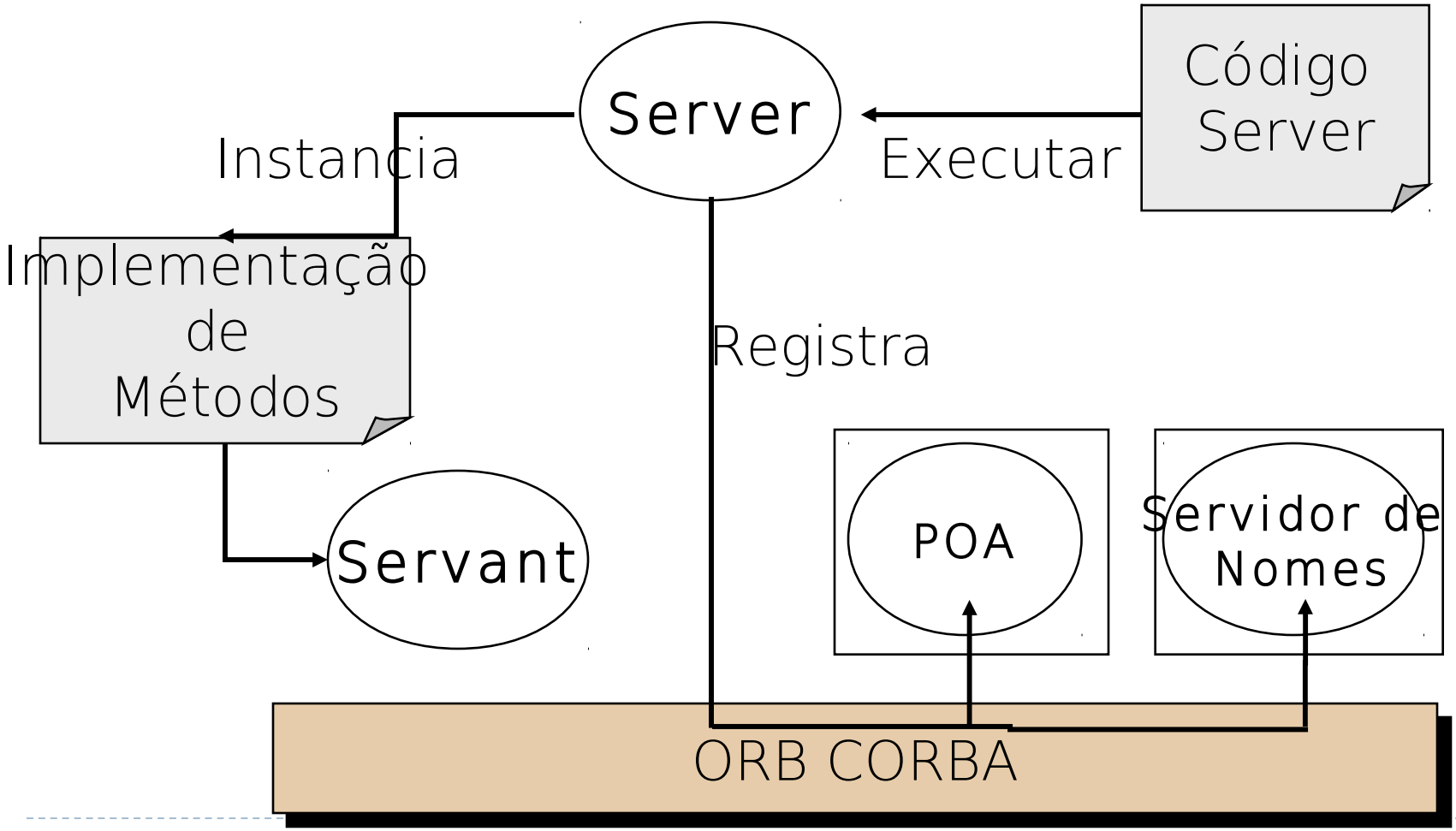
- Linguagem de Programação
  - Java

- ORB CORBA Adotado
  - Java IDL (j2sdk1.4 ou superior)

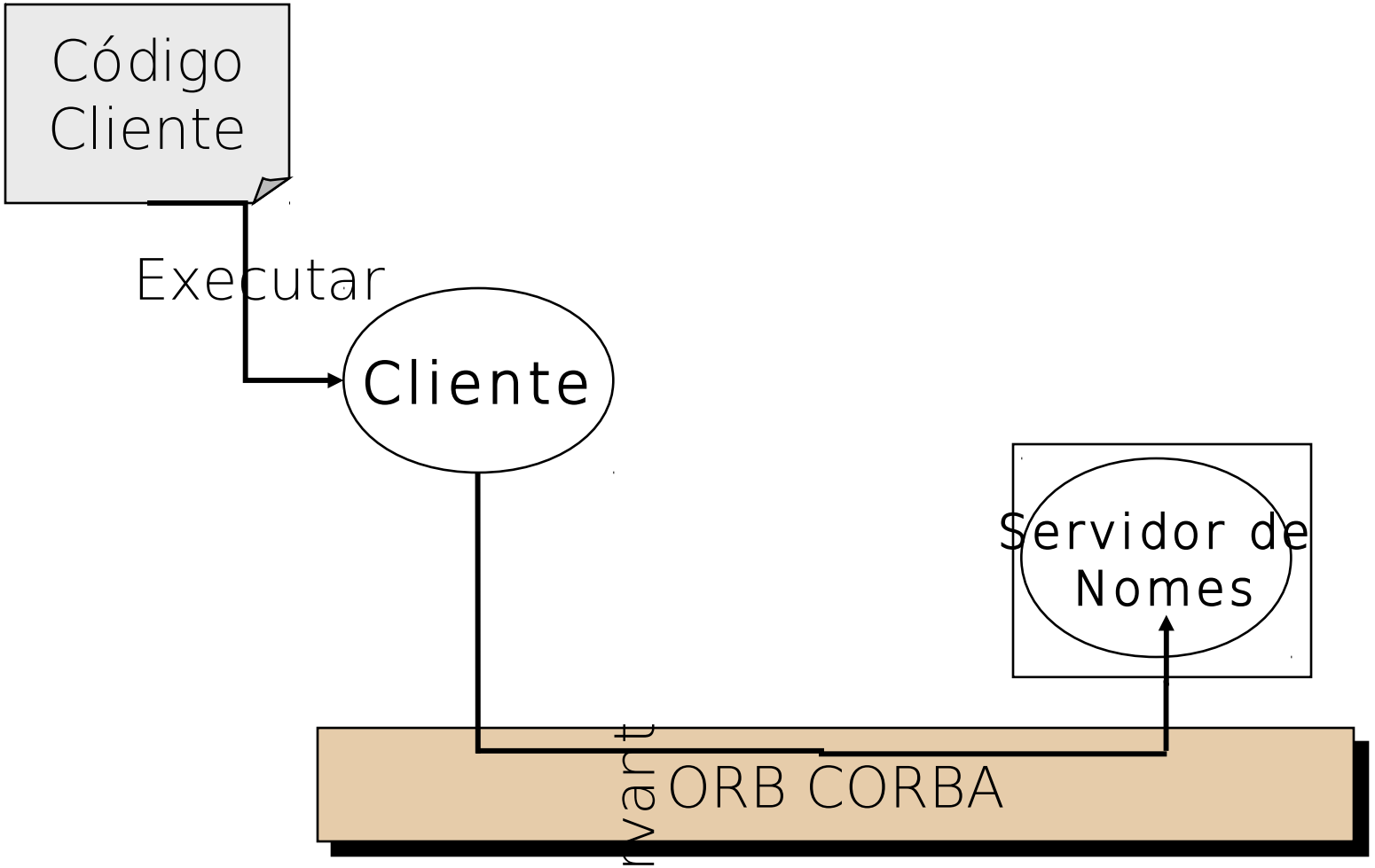
Visão Geral do Processo
(Lado Servidor I)



Visão Geral do Processo
(Lado Servidor II)



Visão Geral do Processo
(Lado Cliente)



Implementação

- Lado Servidor
  - CalcIDL.idl (Arquivo IDL)
  - CalculadoraImpl.java (Implementação da Interface)
  - Servidor.java (Classe servidora)
- Lado Cliente
  - Cliente.java (Classe Cliente)

Passos de Criação da Aplicação

- Passo 1 – Criar IDL
- Passo 2 – Compilar IDL
- Passo 3 – Implementar Interface
- Passo 4 – Implementar Servidor
- Passo 5 – Implementar Cliente
- Passo 6 – Executar Aplicação

Passo 1 - Criar IDL (CalcIDL.idl)

```
module Matematica{
  exception DivisaoPorZero{
       float arg1;
       float arg2;
  };
  interface Calculadora{
       float soma (in float arg1, in float arg2);
       float divisao (in float arg1, in float arg2)
            raises (DivisaoPorZero);
  };
};
```

Passo 2 – Compilar IDL

- Compilando a Interface

```
% idlj -fall CalcIDL.idl
```

  - Obs: por default apenas stubs clientes são gerados. A opção –fall garante a geração de skeletons também;

Passo 2 – Compilar IDL

- Arquivos gerados pelo compilador IDL:
  - CalculadoraPOA.java
    - Skeleton Baseado no POA
  - _CalculadoraStub.java
    - Stub Cliente
  - Calculadora.java
    - Versão Java da Interface IDL
  - CalculadoraHelper.java
    - Operações auxiliares (ex.: narrow())

Passo 2 – Compilar IDL

- Arquivos gerados pelo compilador IDL:
  - CalculadoraHolder.java
    - Objeto que trata das funções e leitura e envio de mensagens;
  - CalculadoraOperations.java
    - Esboço para implementação das operações definidas em IDL
  - DivisaoPorZeroHelper.java
    - Operações auxiliares das exceções
  - DivisaoPorZeroHolder.java
    - Empacotamente e desempacotamento de dados nas exceções;

Passo 3 - Implementar Interface

```
import Matematica.*;
public class CalculadoraImpl extends CalculadoraPOA{
  public float soma(float arg1, float arg2){
      System.out.println("Soma = "+arg1+" + "+arg2);
      return arg1 + arg2;
  }
  public float divisao(float arg1, float arg2)
      throws DivisaoPorZero{

  System.out.println("Divisao="+arg1+"/"+arg2);
        if (arg2 == 0)  throw new

  DivisaoPorZero(arg1,arg2);
           return arg1 / arg2;
   }
}
```

Passo 4 - Implementar o Servidor (I)

```
import Matematica.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;

public class Servidor{
  public static void main(String args[]){
  try{
        ORB orb = ORB.init(args, null);
```

Passo 4 - Implementar o Servidor (II)

```
org.omg.CORBA.Object objPoa =
        orb.resolve_initial_references("RootPOA");

POA rootPOA = POAHelper.narrow(objPoa);


org.omg.CORBA.Object obj =

  orb.resolve_initial_references("NameService")
  ;

NamingContext naming =
        NamingContextHelper.narrow(obj);
```

Passo 4 - Implementar o Servidor (III)

```
CalculadoraImpl calc = new CalculadoraImpl();

org.omg.CORBA.Object   objRef =
                rootPOA.servant_to_reference(calc);

NameComponent[] name = {new

  NameComponent("Calculadora","Exemplo")};

  naming.rebind(name,objRef);
```

Passo 4 - Implementar o Servidor (IV)

```
  rootPOA.the_POAManager().activate();

  System.out.println("Servidor Pronto ...");
      orb.run();
   }catch (Exception ex){
        System.out.println("Erro");
        ex.printStackTrace();}
  }
}
```

Passo 4 - Implementar o Servidor

- Compilação do Servidor

```
% javac Servidor.java
```

Passo 5 - Implementar o Cliente (I)

```
import Matematica.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;

public class Cliente{

  public static void main(String args[]){
  try {

      ORB orb = ORB.init(args,null);
```

Passo 5 - Implementar o Cliente (II)

```
org.omg.CORBA.Object obj =
  orb.resolve_initial_references("NameService")
  ;

NamingContext naming =
        NamingContextHelper.narrow(obj);

NameComponent[] name = {new
  NameComponent("Calculadora","Exemplo")};

org.omg.CORBA.Object objRef =
  naming.resolve(name);
```

Passo 5 - Implementar o Cliente (III)

```
Calculadora calc =
    CalculadoraHelper.narrow(objRef);
 try{

  System.out.println("5+3="+ calc.soma(5,3));
  System.out.println("5/0="+ calc.divisao(5,0));

 }catch (DivisaoPorZero ex){

    System.out.println("Divisao Por Zero");
    System.out.println("A Diviao foi "+ex.arg1 +"
    / "+ex.arg2);
  }
```

Passo 5 - Implementar o Cliente (IV)

```
        }catch (Exception e) {
          System.out.println("ERROR : " + e) ;
          e.printStackTrace(System.out);
   }
    }
}
```

Passo 5 - Implementar o Cliente

- Compilação do Cliente

```
% javac Cliente.java
```

Passo 6 – Executar Aplicação

- Inicializar o Servidor de Nomes
  - `% tnameserv`
- Inicializar o Servidor
  - `% java Servidor [-ORBInitialHost Host]`
- Inicializar o Cliente
  - `% java Cliente [-ORBInitialHost Host]`

Passo 6 – Executar Aplicação

- Resultado

```
% 5 + 3 = 8.0
% Divisao Por Zero
% A Divisao foi 5.0 / 0.0
```

Fim