- Introdução
- Agregação
- Composição
- Dependência

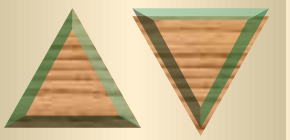
Agregar
Compor
Depender

Prof.Dr. Enzo Seraphim

Associação

Relacionamento de Associação pode ser aprimoradas em:

- **Agregação ↓** expande a fronteira de encapsulamento de atributos para conter uma ou mais classes
- **Composição ↓** é um aprimoramento da agregação que impõe instanciação
- **Dependência ↓** impõe existência entre classes



- Introdução
- Agregação
- Composição
- Dependência

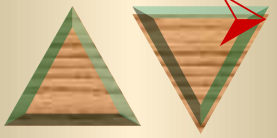
Agregar
Compor
Depender

Prof.Dr. Enzo Seraphim

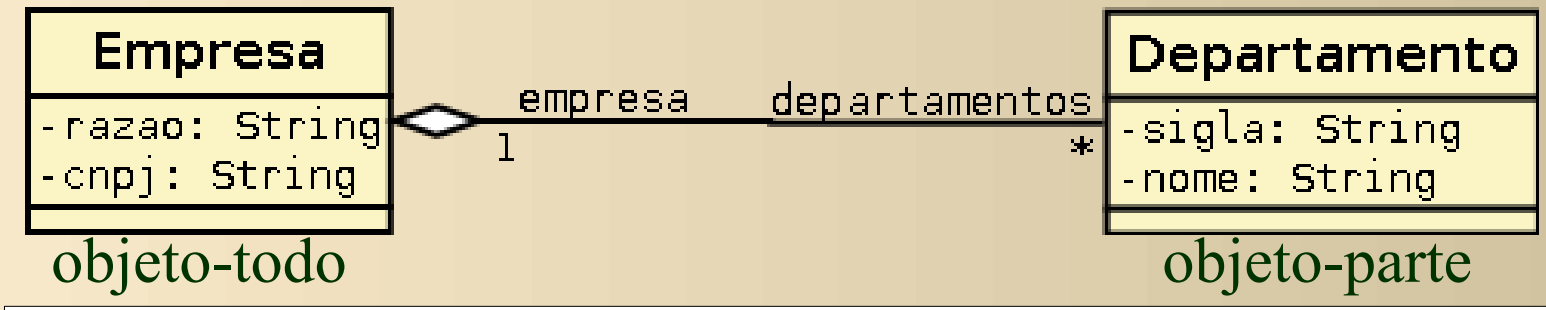
Agregação

Agregação

- Palavras chaves usadas para identifica-la:
  - "consiste em", "contém", "é parte de"
- Conhecida como relação de conteúdo.
  - Um todo é relacionado com suas partes.
- Demonstra que as informações de um objeto-todo precisam ser complementadas pelas informações contidas em um(ou mais) objetos-parte.
- Linha de associação com diamante branco junto à classe agregadora
- Multiplicidade, direcionamento e implementação são as mesmas da associação



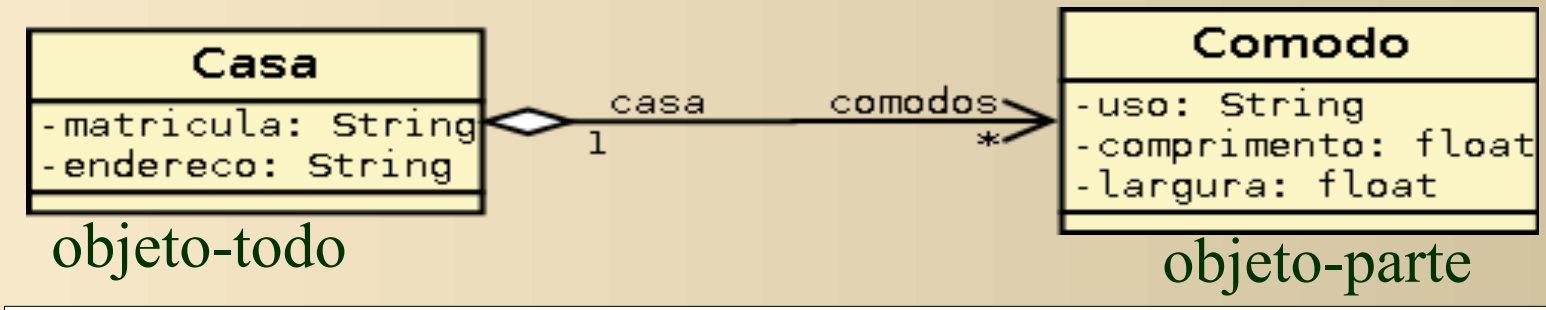
Agregação Bidirecional



Agregação

```
public class Empresa {
    private String razao;
    private String cnpj;
    private List<Departamento> departamentos=
            new ArrayList<Departamento>();
    //gets e sets
}
public class Departamento {
    private String sigla;
    private String nome;
    private Empresa empresa;
    // gets e sets    }
```

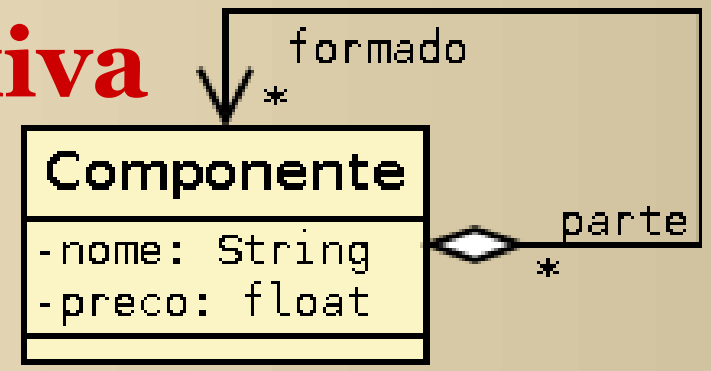
Agregação Unidirecional



Agregação

```
public class Casa {
   private String matricula;
   private String endereco;
   private List<Comodo> comodos =
         new ArrayList<Comodo>();
   //gets e sets
}
public class Comodo {
   private String uso;
   private float comprimento;
   private float largura;
   //gets e sets   }
```

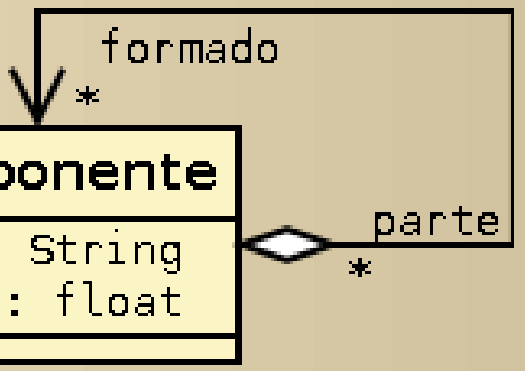
Agregação Reflexiva



Agregação

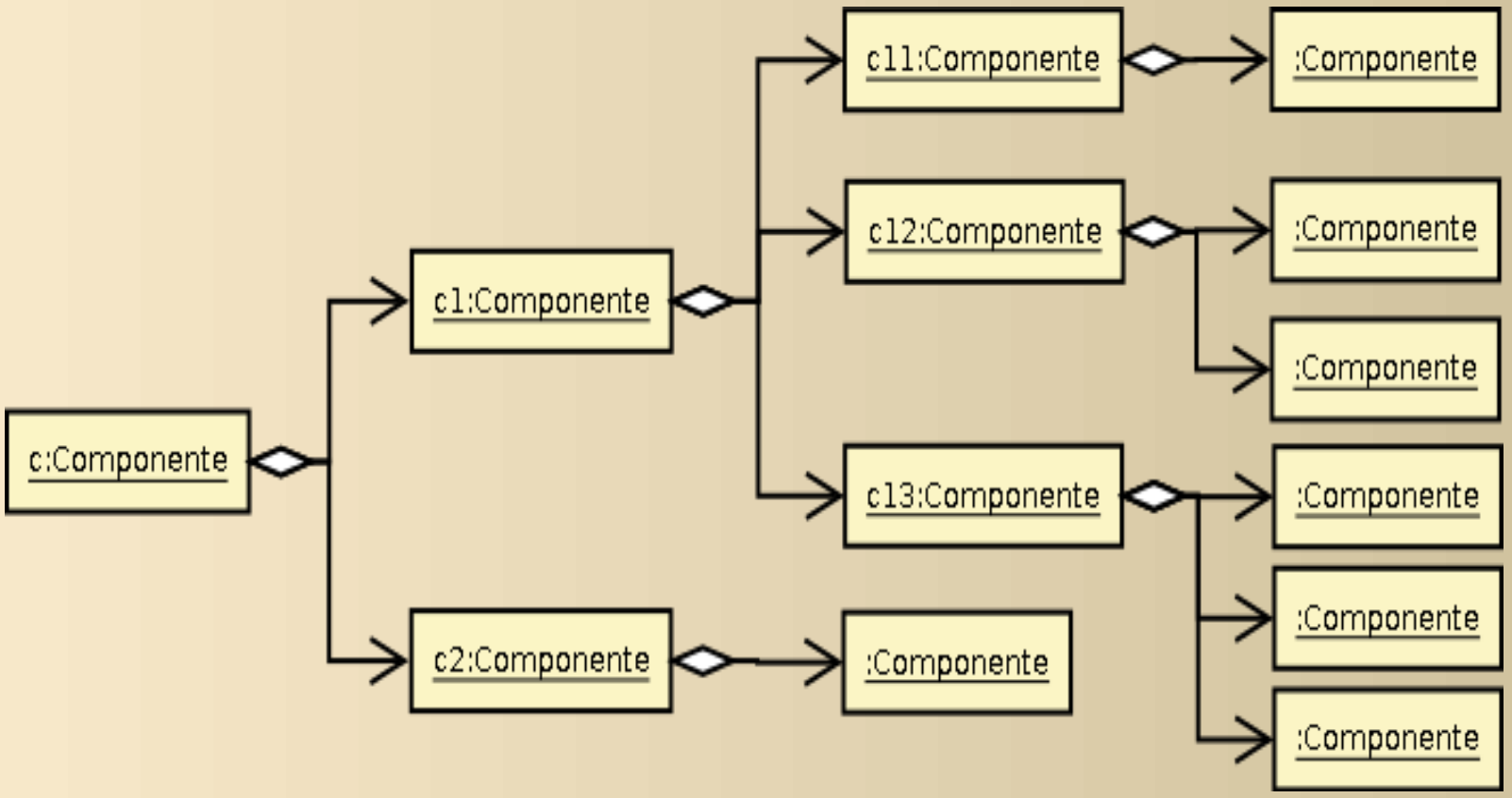
```
public class Componente {
    private String nome;
    private float preco;
    private List<Componente> componentes=
            new ArrayList<Componente>();
    //gets e sets
}
```

Agregação Reflexiva



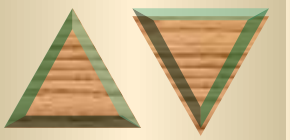
Agregação

Exemplo com diagrama objetos



Agregação

```
public static void main(String[] args) {
   Componente c11 = new Componente();
   c11.getComodos().add(new Componente());
   Componente c12 = new Componente();
   c12.getComodos().add(new Componente());
   c12.getComodos().add(new Componente());
   Componente c13 = new Componente();
   c13.getComodos().add(new Componente());
   c13.getComodos().add(new Componente());
   c13.getComodos().add(new Componente());
   Componente c1 = new Componente();
   c1.getComodos().add(c11);
   c1.getComodos().add(c12);
   c1.getComodos().add(c13);
   Componente c2 = new Componente();
   c2.getComodos().add(new Componente());
   Componente c = new Componente();
   c.getComodos().add(c1);
   c.getComodos().add(c2);   }
```



- Introdução
- Agregação
- Composição
- Dependência

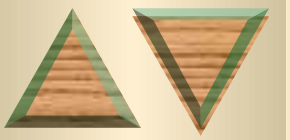
Agregar
Compor
Depender

Prof.Dr. Enzo Seraphim

Composição

Composição

- Uma agregação mais forte
- Impõe a criação do Objeto-Parte para TODA criação de um Objeto-Todo
- Linha de associação com diamante negro junto à classe agregadora
- Multiplicidade, direcionamento são as mesmas da associação
- Implementação usa de instanciação para impor criação da parte



Composição Bidirecional

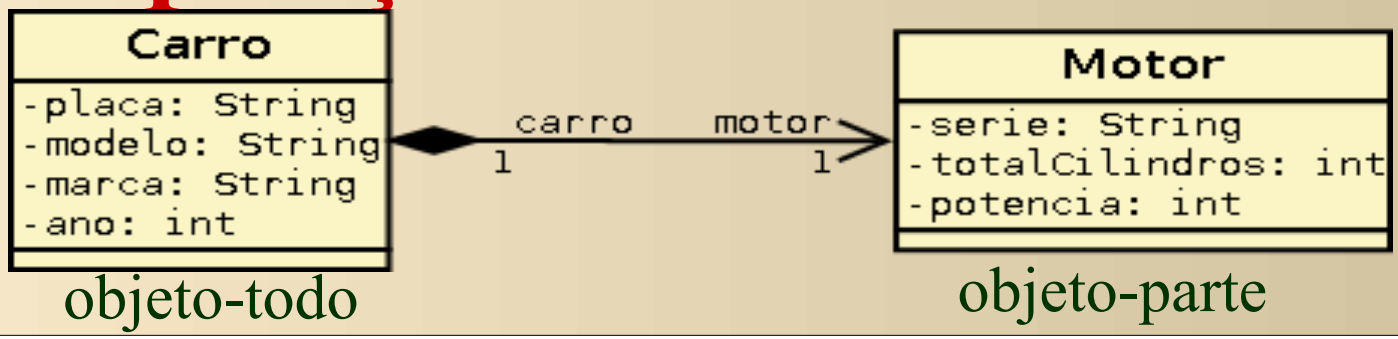
objeto-todo

objeto-parte

Composição

```
public class Mao {
   private int base;
   private int altura;
   private Dedo dedos[] = new Dedo[5];
   public Mao() {
      for(int i=0;i<5;i++) {
         dedos[i]=new Dedo();
      }   } //gets e sets   }
public class Dedo {
   private String nome;
   private int comprimento;
   private Mao mao;   //gets e sets   }
```

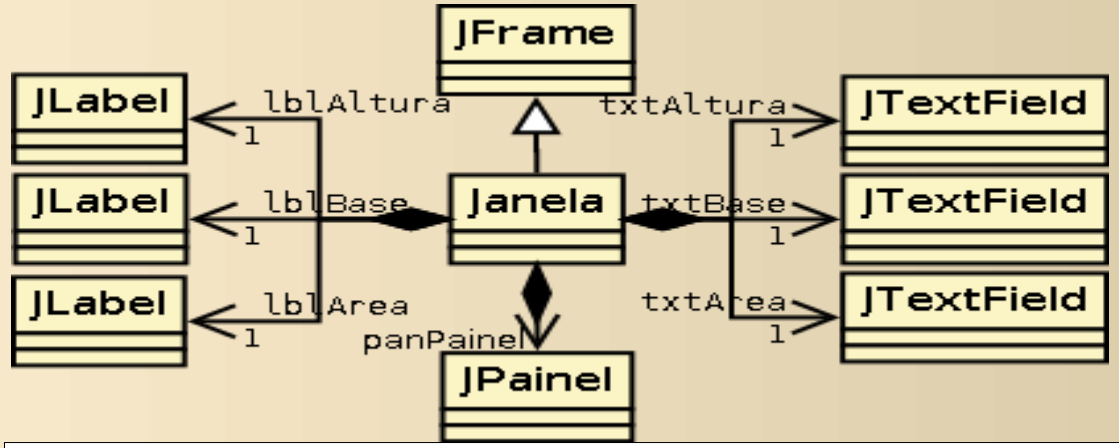
Composição Unidirecional

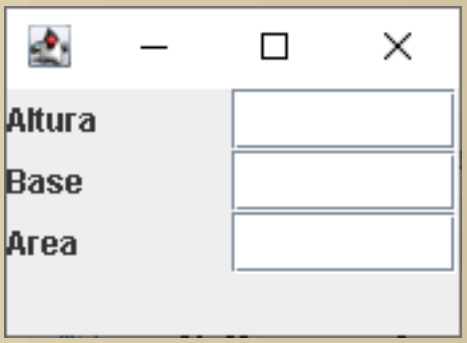


Composição

```
public class Carro {
    private String placa;
    private String modelo;
    private String marca;
    private int ano;
    private Motor motor = new Motor();
    //gets e sets    }
public class Motor {
    private String serie;
    private int totalCilindros;
    private int potencia;
    //gets e sets    }
```

Composição






```
public class Janela extends JFrame {
private JLabel lblAltura = new JLabel("Altura");
private JLabel lblBase = new JLabel("Base");
private JLabel lblArea = new JLabel("Area");
private JTextField txtAltura = new JTextField();
private JTextField txtBase = new JTextField();
private JTextField txtArea = new JTextField();
private JPanel panPainel = new Jpanel();
public Janela() {
 panPainel.setLayout(new GridLayout(4, 2));
 setContentPane(panPainel);
 panPainel.add(lblAltura);
 panPainel.add(txtAltura);
 panPainel.add(lblBase);
 panPainel.add(txtBase);
 panPainel.add(lblArea);
 panPainel.add(txtArea);   }   }
```

- Introdução
- Agregação
- Composição
- Dependência

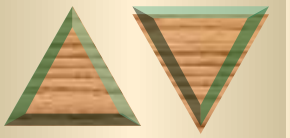
Agregar
Compor
Depender

Prof.Dr. Enzo Seraphim

Relação de dependência

Dependência

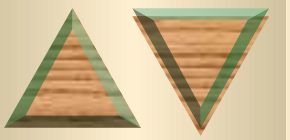
- Impõe a existência do Objeto-Completo para TODA criação de um Objeto-Dependente
- Linha tracejada com seta para Objeto-Completo
- Geralmente multiplicidade tem valor 1
- Sempre são de unidirecionamento
- Implementação usa de parametrização no construtor da classe que tem a dependência



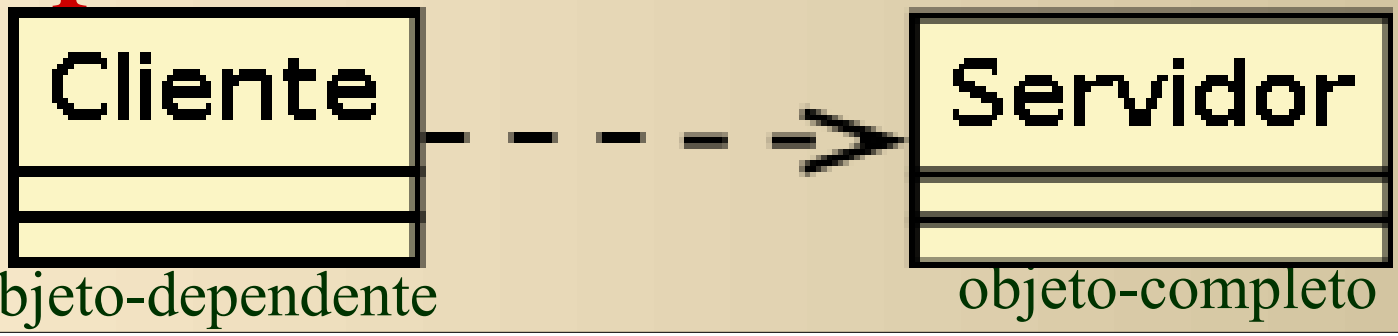
Construtor

Dependência

- Método chamado assim que uma nova instância do objeto é criada
- Caso não esteja declarada, toda linguagem declara automaticamente o construtor padrão (sem parâmetros)
- Qualquer declaração de construtor para classe suprime a declação automática
- Pode haver mais de um construtor na classe variando-se parâmetros



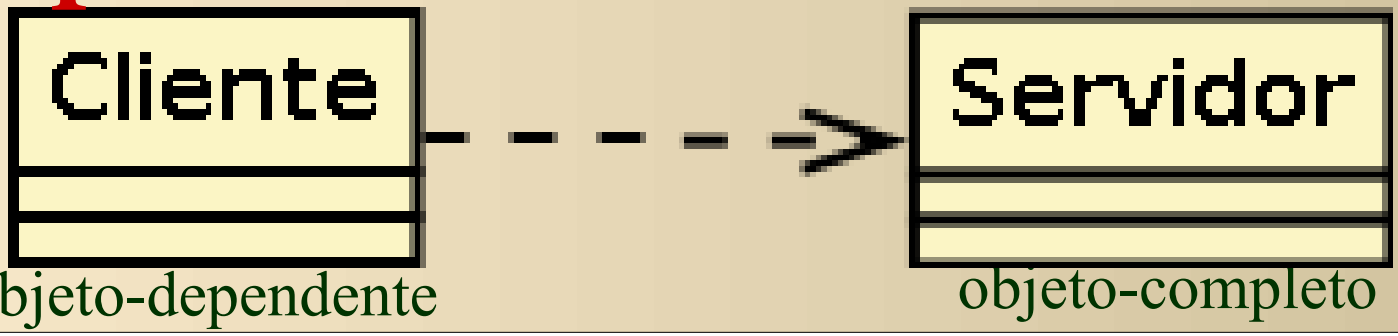
Dependência



Dependência

```
public class Cliente {
    private Servidor servidor;
    public Cliente(Servidor servidor) {
        this.servidor = servidor;
    }
    //gets e sets
}
public class Servidor {
}
```

Dependência

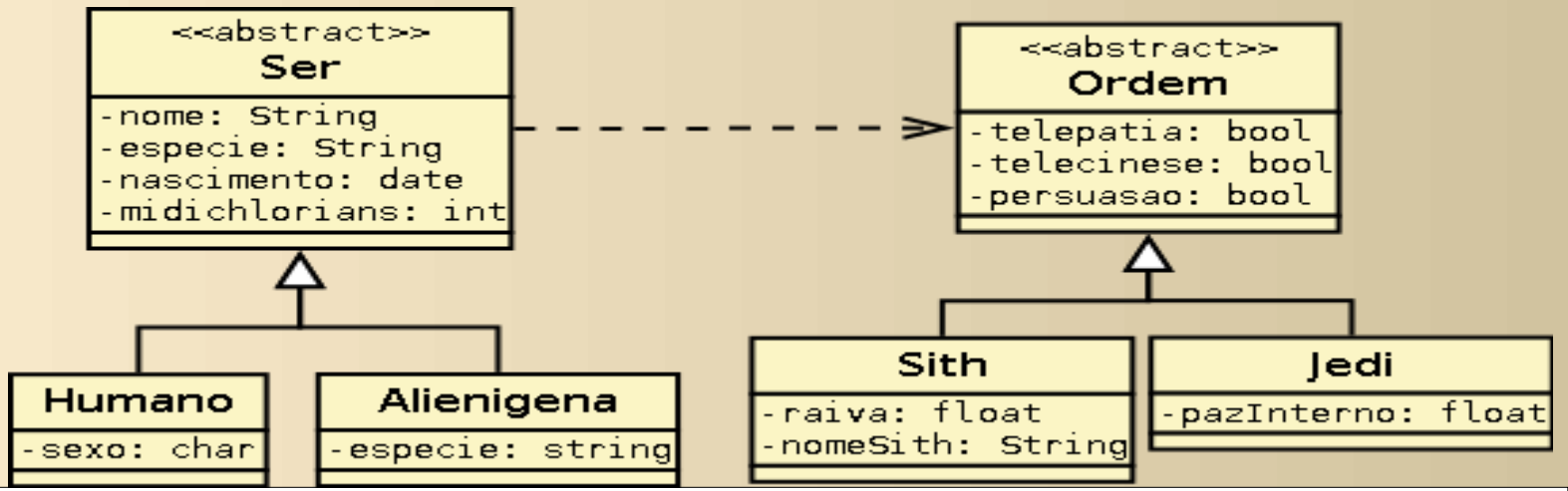


Dependência

```
public class App2 {
 public static void main(String[] args) {
   Servidor s = new Servidor();
   // não compila linha abaixo
   // Cliente c = new Cliente();
   Cliente c = new Cliente(s);
 }
}
```

Dependência em herança

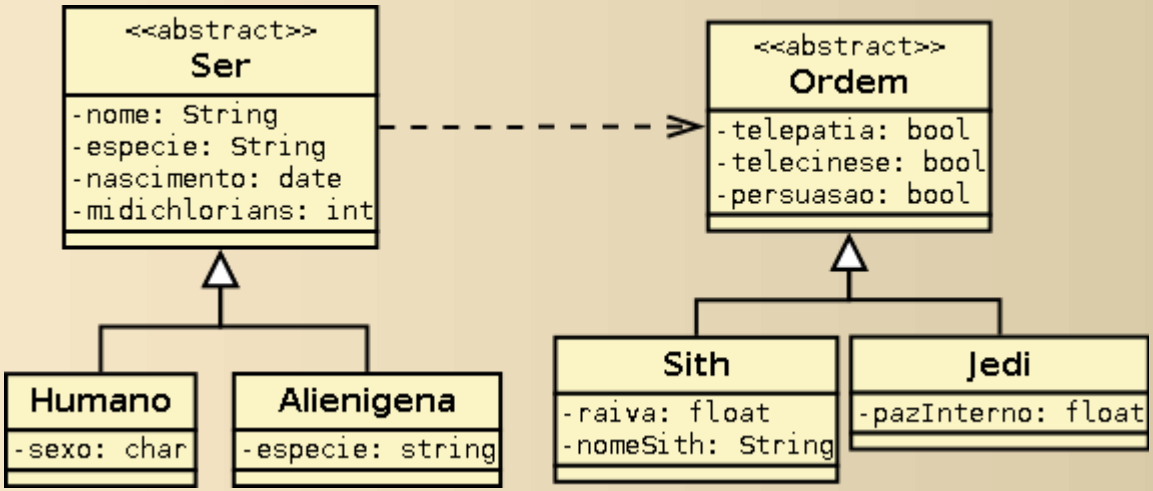
Dependência




```
public class Ser {
 //demais atributos
 private Ordem ordem;
 public Ordem(Ordem ordem){this.ordem=ordem;}}
public class Humano {
 public Humano(Ordem ordem){
  this.super(ordem);}}
public class Alienigena {
 public Alienigena(Ordem ordem){
  this.super(ordem);}}
```

Dependência em hernaça

Dependência



Prof.Dr.
Enzo Seraphim
seraphim@unifei.edu.br
IESTI/UNIFEI

Agregar
Compor
Depender