UNIVERSIDADE VILA VELHA-ES

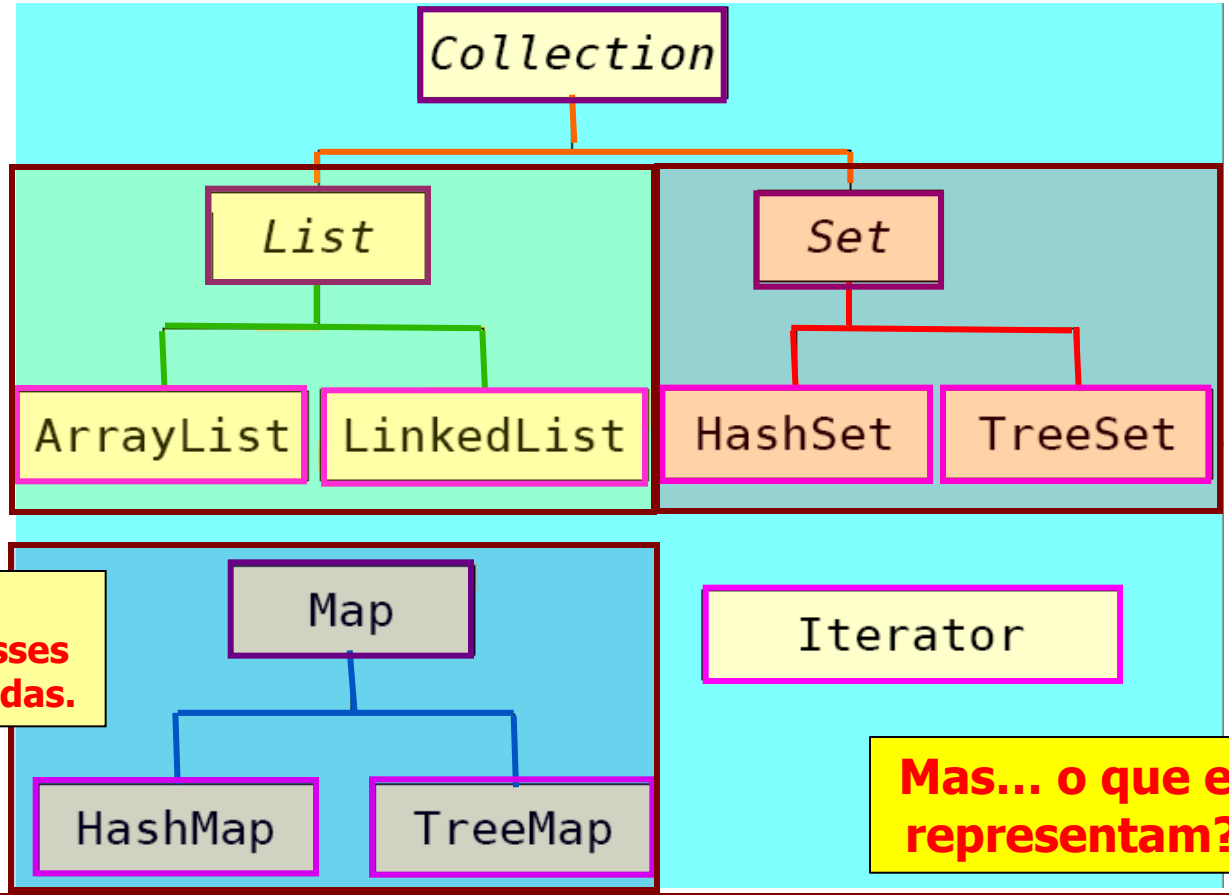
Programação Java IV

Prof. Vinicius Rosalen

Java Collection Framework

Hierarquia (vamos falar mais sobre isso logo logo..)

- Os elementos que compreendem a estrutura de coleções estão no pacote java.util.

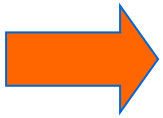


Implementação de relações – N-N e Coleções

- Blz,
  - Relembrado esses elementos, podemos conversar sobre outro tipo de relacionamento que poderemos encontrar nas modelagens…

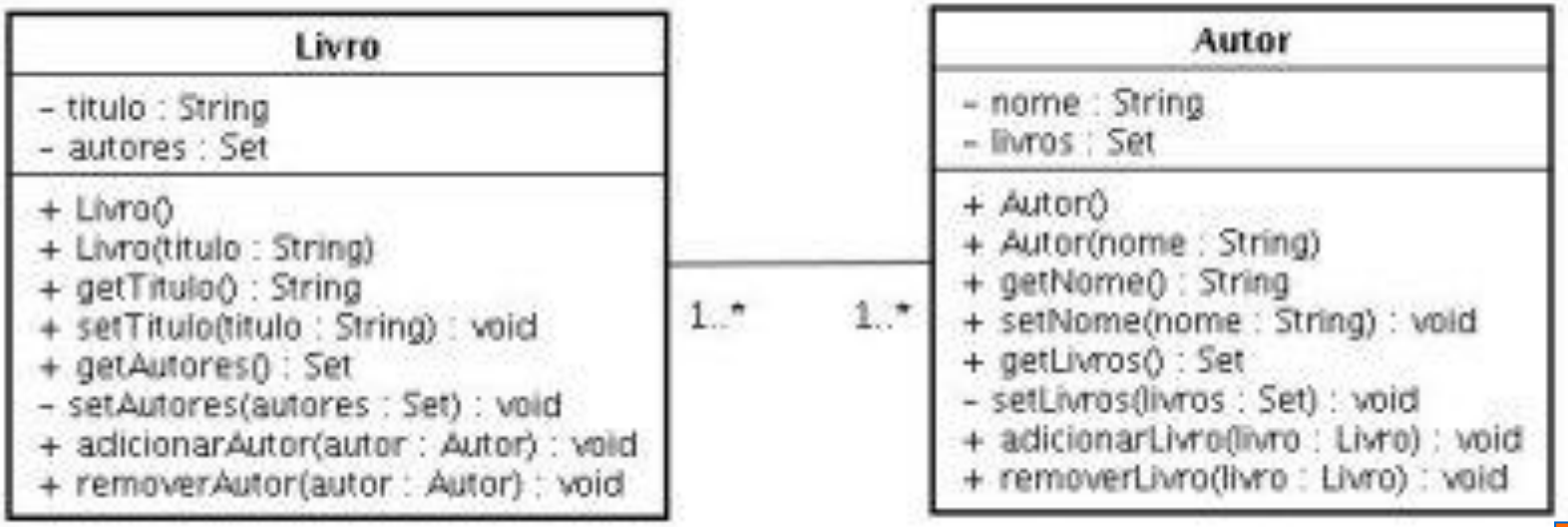
  - São os relacionamento *muitos-para-muitos* ou *N-N*

    - Vamos conversar um pouco sobre eles…



Implementação de relações – N-N e Coleções

- Um exemplo de relacionamento muitos-para-muitos é aquele que se estabelece entre as classes Livro e Autor:
  - Um livro pode ser escrito por um ou muitos autores;
  - Um autor pode escrever um ou muitos livros.



Relações de n para n bidirecional

Implementação de relações – N-N e Coleções

Para implementar relacionamentos desse tipo, muitos-para-muitos, podemos...

- Inserir, em ambas as classes, um atributo do tipo coleção (Set, por exemplo);
- Inserir os métodos get e set correspondentes ao atributo coleção;
- Inserir métodos para adicionar e remover objetos na coleção

- **E mais um detalhe importante……**

Implementação de relações – N-N e Coleções

Insira nos métodos adicionar e remover chamadas para os métodos correspondentes da outra classe.

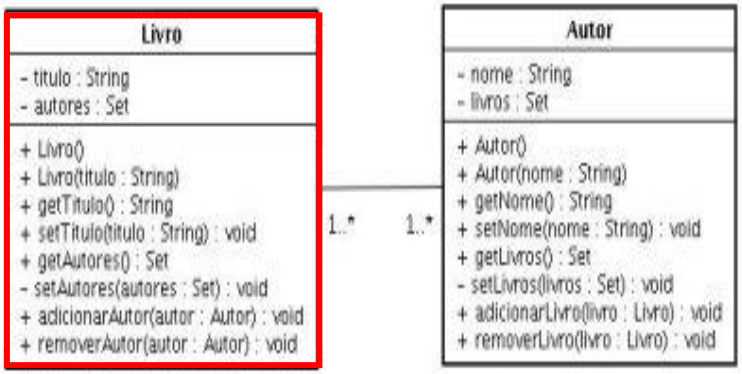
- Por exemplo:

No método adicionarAutor, da classe Livro, chame o método adicionarLivro da classe Autor.

Já no método adicionarLivro da classe Autor, chame o método adicionarAutor da classe Livro.

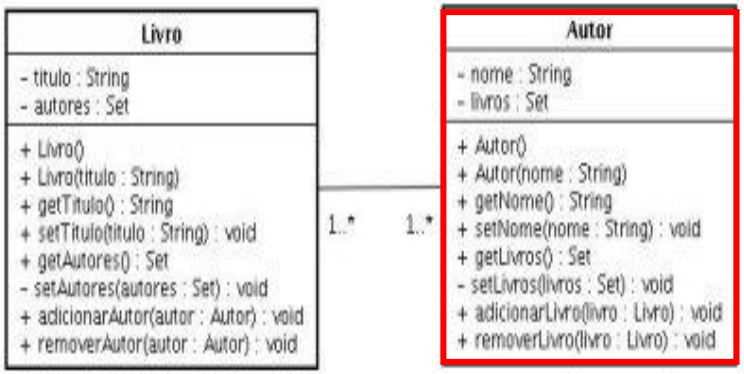
Para evitar um loop infinito, teste antes se o objeto já não foi incluído ou removido da coleção.

Implementação de relações – N-N e Coleções




```
1  import java.util.HashSet;
2  import java.util.Set;
3
4  public class LivroMuitosParaMuitos {
5      private String titulo;
6      private Set<AutorMuitosParaMuitos> autores;
7
8      public LivroMuitosParaMuitos() {
9          super();
10         this.autores = new HashSet<AutorMuitosParaMuitos>();
11     }
12     public LivroMuitosParaMuitos(String titulo)...
16     public void setTitulo(String titulo)...
19     public String getTitulo()...
22     private void setAutores(Set<AutorMuitosParaMuitos> autores)...
25     public Set<AutorMuitosParaMuitos> getAutores()...
28
29         public void adicionarAutor(AutorMuitosParaMuitos autor){
30         if (!this.getAutores().contains(autor)){
31             this.getAutores().add(autor);
32             autor.adicionarLivro(this);
33         }
34     }
35     public void removerAutor(AutorMuitosParaMuitos autor){
36         if ( this.getAutores().contains(autor)){
37             this.getAutores().remove(autor);
38             autor.removerLivro(this);
39         }
40     }
41
42     @Override
43     public boolean equals(Object obj)...
56     @Override
57     public int hashCode()...
62 }
```

Implementação de relações – N-N e Coleções




```
1  import java.util.HashSet;
2  import java.util.Set;
3
4  public class AutorMuitosParaMuitos {
5      private String nome;
6      private Set<LivroMuitosParaMuitos> livros;
7
8      public AutorMuitosParaMuitos() {
9          super();
10         this.livros = new HashSet<LivroMuitosParaMuitos>();
11     }
12     public AutorMuitosParaMuitos(String nome)...
16     public void setNome(String nome)...
19     public String getNome()...
22     private void setLivros(Set<LivroMuitosParaMuitos> livros)...
25     public Set<LivroMuitosParaMuitos> getLivros()...
28
29     public void adicionarLivro(LivroMuitosParaMuitos livro){
30         if (!this.getLivros().contains(livro)){
31             this.getLivros().add(livro);
32             livro.adicionarAutor(this);
33         }
34     }
35     public void removerLivro(LivroMuitosParaMuitos livro){
36         if ( this.getLivros().contains(livro)){
37             this.getLivros().remove(livro);
38             livro.removerAutor(this);
39         }
40     }
41
42     @Override
43     public boolean equals(Object obj)...
56     @Override
57     public int hashCode()...
62 }
```

Implementação de relações – N-N e Coleções

Qual a saída impressa?
E se descomentar a linha 16??

```
1  import java.util.Iterator;
2
3  public class AppMuitosParaMuitosLivroAutor {
4      public static void main(String[] args) {
5
6          LivroMuitosParaMuitos livro01 = new LivroMuitosParaMuitos("Livro 01");
7          LivroMuitosParaMuitos livro02 = new LivroMuitosParaMuitos("Livro 02");
8          LivroMuitosParaMuitos livro03 = new LivroMuitosParaMuitos("Livro 03");
9
10         AutorMuitosParaMuitos autor01 = new AutorMuitosParaMuitos("Autor 01");
11         AutorMuitosParaMuitos autor02 = new AutorMuitosParaMuitos("Autor 02");
12         AutorMuitosParaMuitos autor03 = new AutorMuitosParaMuitos("Autor 03");
13
14         livro02.adicionarAutor(autor01);
15         livro02.adicionarAutor(autor02);
16                 //livro02.adicionarAutor(autor01);
17
18         autor03.adicionarLivro(livro02);
19         autor03.adicionarLivro(livro03);
20
21         System.out.println("============================");
22         System.out.println("Autores do Livro 02:");
23         for(AutorMuitosParaMuitos autor: livro02.getAutores()){
24             System.out.println(autor.getNome());
25         }
26
27         System.out.println("============================");
28         System.out.println("Livros do Autor 03");
29         for(LivroMuitosParaMuitos livro: autor03.getLivros()){
30             System.out.println(livro.getTitulo());
31         }
32     }
33 }
```

Implementação de relações – N-N e Coleções

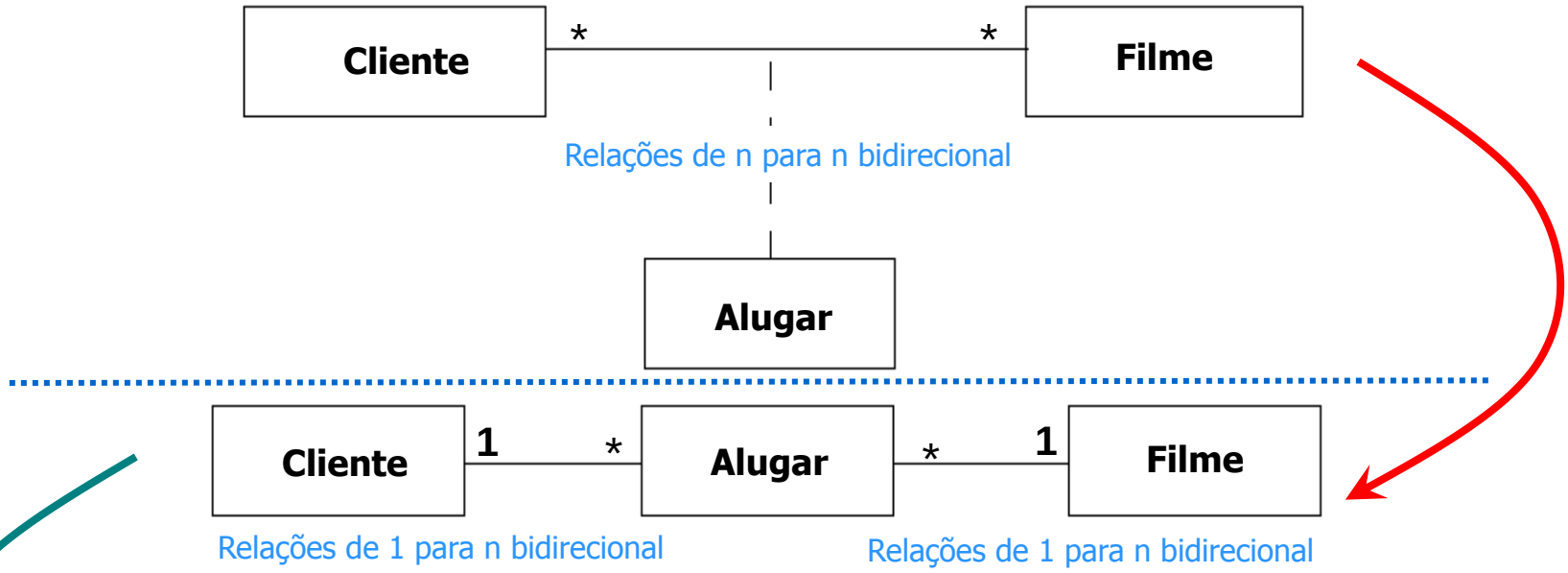
- Em algumas relações poderá haver vantagem na inclusão de uma Classe Intermediária
  - Principalmente se existem informações que necessitam ser armazenadas devido a relação entre as classes

- Consideremos, como exemplo, a relação entre as classes Cliente e Filme.
  - Um Cliente pode *Alugar* vários Filmes e
  - Um Filme pode ser *Alugado* por mais de um Cliente.
  - E, eu desejaria também armazenar o valor e a data do Aluguel

- Poderíamos mentalizar uma associação entre Cliente e Filme sendo que a classe Alugar armazene dados dessa associação.

Implementação de relações – N-N e Coleções

Nossa modelagem ficaria assim…




```
class Cliente {

private Set Alugar;
...
}
```

```
class Alugar {

private Cliente cliente;
private Filme filme;
private float valor;
private String data;
...
}
```

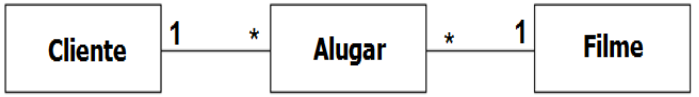
```
class Filme {

private Set Alugar;
...
}
```

Implementação de relações – N-N e Coleções

```
class Cliente {

private Set Alugar;
...
}
```

```
class Alugar {

private Cliente cliente;
private Filme   filme;
private float   valor;
private String  data;
...
}
```

```
class Filme {

private Set Alugar;
...
}
```



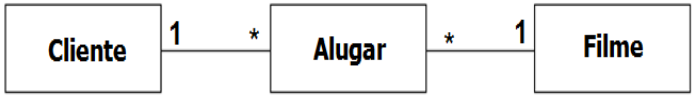

```
1  import java.util.HashSet;
2  import java.util.Set;
3
4  public class ClienteMuitosParaMuitos {
5
6      private String nome;
7      private Set alugar; // Um cliente pode fazer vários aluguéis!
8
9      public ClienteMuitosParaMuitos() {
10         this.alugar = new HashSet<AlugarMuitosParaMuitos>();
11     }
12
13     public ClienteMuitosParaMuitos(String nome)..
17     public Set<AlugarMuitosParaMuitos> getAlugar()..
20     public void setAlugar(Set<AlugarMuitosParaMuitos> alugar)..
23     public String getNome()..
26     public void setNome(String nome)..
29
30     public void adicionarAlugar(AlugarMuitosParaMuitos aluguel){
31         this.getAlugar().add(aluguel);
32     }
33     public void removerAlugar(AlugarMuitosParaMuitos aluguel){
34         this.getAlugar().remove(aluguel);
35     }
36     @Override
37     public String toString()..
40 }
```

Implementação de relações – N-N e Coleções

```
class Cliente {

private Set Alugar;
...
}
```

```
class Alugar {

private Cliente cliente;
private Filme   filme;
private float   valor;
private String  data;
...
}
```

```
class Filme {

private Set Alugar;
...
}
```




```
1  import java.util.HashSet;
2  import java.util.Set;
3
4
5  public class FilmesMuitosParaMuitos {
6      private String nome;
7      private String genero;
8      private Set alugar; // Um filme pode ser alugado várias vezes
9
10     public FilmesMuitosParaMuitos() {
11         this.alugar = new HashSet<AlugarMuitosParaMuitos>();
12     }
13
14     public FilmesMuitosParaMuitos(String nome, String genero)...
19     public Set<AlugarMuitosParaMuitos> getAlugar()...
22     public void setAlugar(Set<AlugarMuitosParaMuitos> alugar)...
25     public String getGenero()...
28     public void setGenero(String genero)...
31     public String getNome()...
34     public void setNome(String nome)...
37
38     public void adicionarAlugar(AlugarMuitosParaMuitos aluguel){
39             this.getAlugar().add(aluguel);
40     }
41     public void removerAlugar(AlugarMuitosParaMuitos aluguel){
42         this.getAlugar().remove(aluguel);
43     }
44
45     @Override
46     public String toString()...
49
50 }
```

Implementação de relações – N-N e Coleções

```
class Cliente {

private Set Alugar;
...
}
```

```
class Alugar {

private Cliente cliente;
private Filme   filme;
private float   valor;
private String  data;
...
}
```

```
class Filme {

private Set Alugar;
...
}
```


```
public class AlugarMuitosParaMuitos {
    private ClienteMuitosParaMuitos cliente;
    private FilmesMuitosParaMuitos filme;
    private Float valor;
    private String data;

    public AlugarMuitosParaMuitos()...
    public AlugarMuitosParaMuitos(ClienteMuitosParaMuitos cliente, FilmesMuitosParaMuitos filme
        this.cliente = cliente;
        this.filme = filme;
        this.valor = valor;
        this.data = data;
        this.setCliente(cliente);
        this.setFilme(filme);
    }
    public ClienteMuitosParaMuitos getCliente()...
    public FilmesMuitosParaMuitos getFilme()...
    public String getData()...
    public void setData(String data)...
    public Float getValor()...
    public void setValor(Float valor)...

    public void setCliente(ClienteMuitosParaMuitos cliente) {
        this.cliente = cliente;
        if (this.cliente != null) {
            cliente.adicionarAlugar(this);
        }
    }
    public void setFilme(FilmesMuitosParaMuitos filme) {
        this.filme = filme;
        if (this.filme != null) {
            filme.adicionarAlugar(this);
        }
    }

    @Override
    public boolean equals(Object obj)...
    @Override
    public int hashCode()...
    @Override
    public String toString()...
}
```

Implementação de relações – N-N e Coleções

```
public class AppMuitosParaMuitosClienteAlugarFilmes {
    public static void main(String[] args) {

        ClienteMuitosParaMuitos cliente01 = new ClienteMuitosParaMuitos("Vinicius");
        ClienteMuitosParaMuitos cliente02 = new ClienteMuitosParaMuitos("Rosalen");
        ClienteMuitosParaMuitos cliente03 = new ClienteMuitosParaMuitos("Silva");

        FilmesMuitosParaMuitos filme01 = new FilmesMuitosParaMuitos("Filme 01", "Suspense");
        FilmesMuitosParaMuitos filme02 = new FilmesMuitosParaMuitos("Filme 02", "Romance");
        FilmesMuitosParaMuitos filme03 = new FilmesMuitosParaMuitos("Filme 03", "Aventura");
        FilmesMuitosParaMuitos filme04 = new FilmesMuitosParaMuitos("Filme 04", "Terror");

            AlugarMuitosParaMuitos alugel01 = new AlugarMuitosParaMuitos();
            alugel01.setCliente(cliente01);
            alugel01.setFilme(filme01);
            alugel01.setValor(new Float(100));
            alugel01.setData("11/11/1111");

            AlugarMuitosParaMuitos alugel02 = new AlugarMuitosParaMuitos(
                                  cliente02, filme02, new Float(200), "22/22/2222");

            AlugarMuitosParaMuitos alugel021 = new AlugarMuitosParaMuitos(
                                  cliente02, filme03, new Float(300), "33/33/3333");

            AlugarMuitosParaMuitos alugel022 = new AlugarMuitosParaMuitos(
                                  cliente02, filme04, new Float(400), "44/44/4444");

            AlugarMuitosParaMuitos alugel011 = new AlugarMuitosParaMuitos(
                                  cliente01, filme04, new Float(400), "44/44/4444");

System.out.println("-----------------------------------------------------------------");
        System.out.println("A partir do Filme: quem foi/foram o(s) cliente(s) que o alugaram:");
        for(AlugarMuitosParaMuitos aluguel: filme04.getAlugar()){
            System.out.println("filme04: " + aluguel.getCliente());
        }

    System.out.println("-----------------------------------------------------------------");
            System.out.println("A partir do Cliente: qual/quais foi/foram os filme(s) que ele alugou:");
            for(AlugarMuitosParaMuitos aluguel: cliente02.getAlugar()){
                System.out.println("cliente02: " + aluguel.getFilme());
            }
```

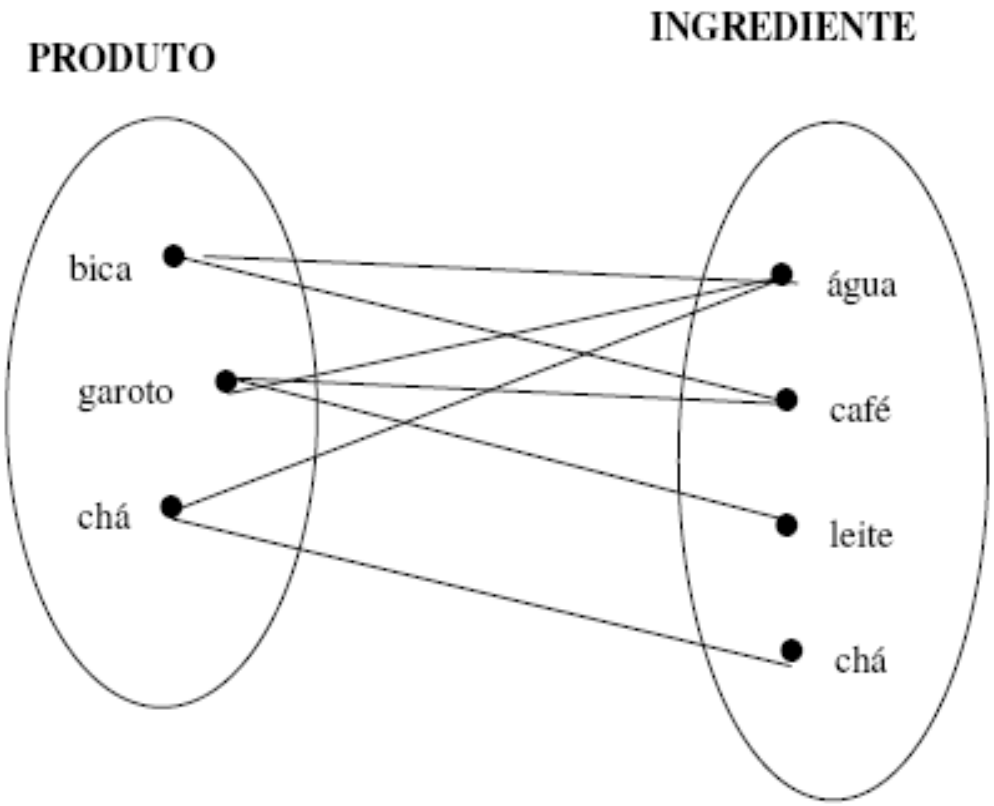
UNIVERSIDADE VILA VELHA-ES

Exercício Junto....

- Blz, entendemos muita coisa hoje.....
  - ⇨ Vamos agora construir juntos um exemplo que tenha:
    - Relacionamento N – N
    - Coleção Genérica (tipada)
  - – Baseado no seguinte diagrama.... ⇨

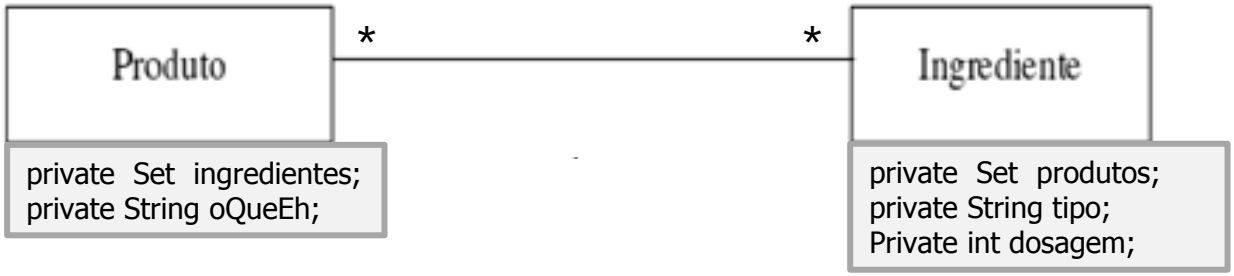
Exercício Junto....

- Consideremos, como exemplo, a relação entre as classes Produto e Ingrediente.



Exercício Junto....

- Fazer um exemplo de N-N junto…



- Criar as classses Produto e Ingredientes com os atributos básicos.
- Gerar os gets e set para todos atributos e construtores padrão (com o new Set).
  - Colcara tipagem e Criar o equals e hashCode nas classes
- Criar na classe Produto e ingredientes (as que tem a coleção) os métodos para adicionar e remover objetos na coleção.
- Agora podemos criar classe Principal, criando alguns Produtos e Ingredientes, chamando o método para adicionar seus respectivos e depois 1: Listar quais são os Ingredientes do Produto; 2: Listar qual é ou quais são o(s) Produto(s) associados ao Ingrediente

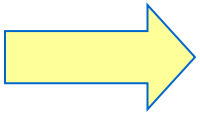
UNIVERSIDADE VILA VELHA-ES

Exercícios

- Blz... Agora é hora de exercitar.....

- Tente resolver os seguintes problemas...
  - Em dupla
  - Apresentar ao professor no final da aula
  - Pontuação em Atividades em sala de aula...

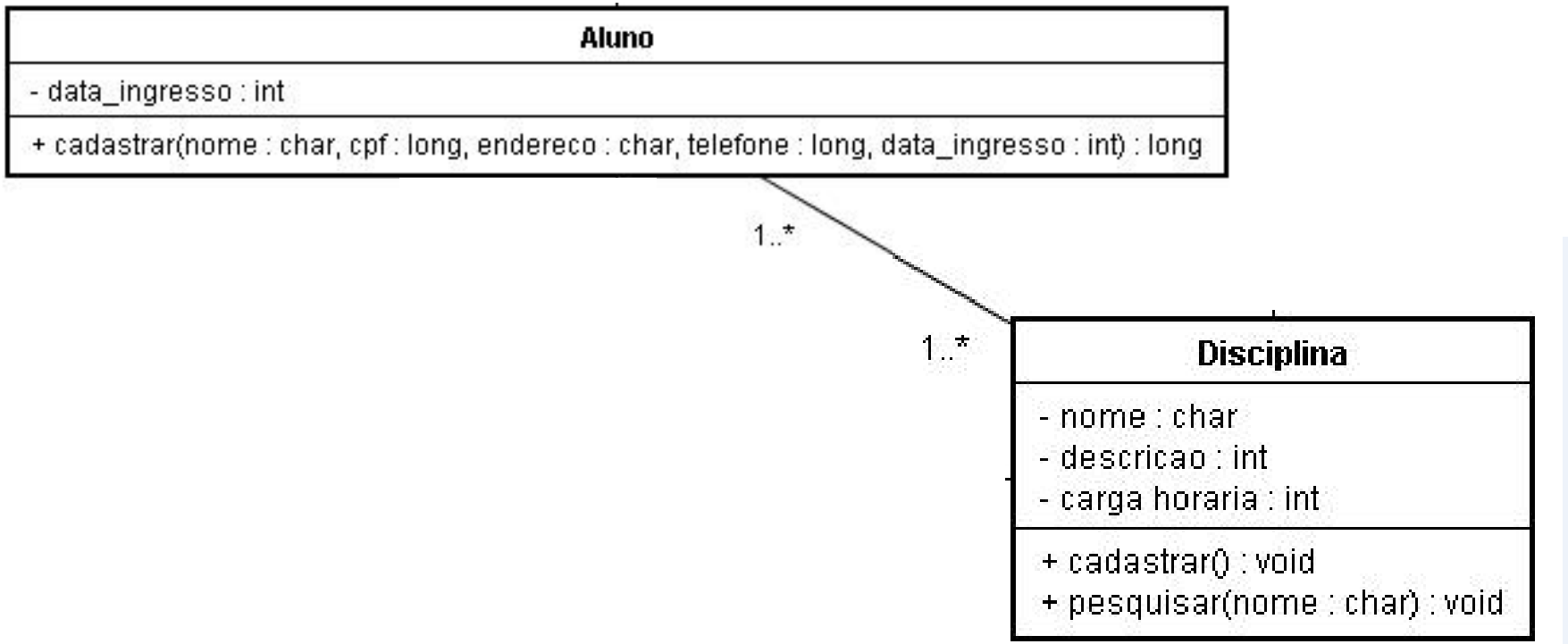
  - Faça o JAVADOC de todos os exercícios!!!



UNIVERSIDADE VILA VELHA-ES

Exercício

- Identifique alguns atributos e comportamentos simples para as classes abaixo e implemente.



UNIVERSIDADE VILA VELHA-ES

Exercício

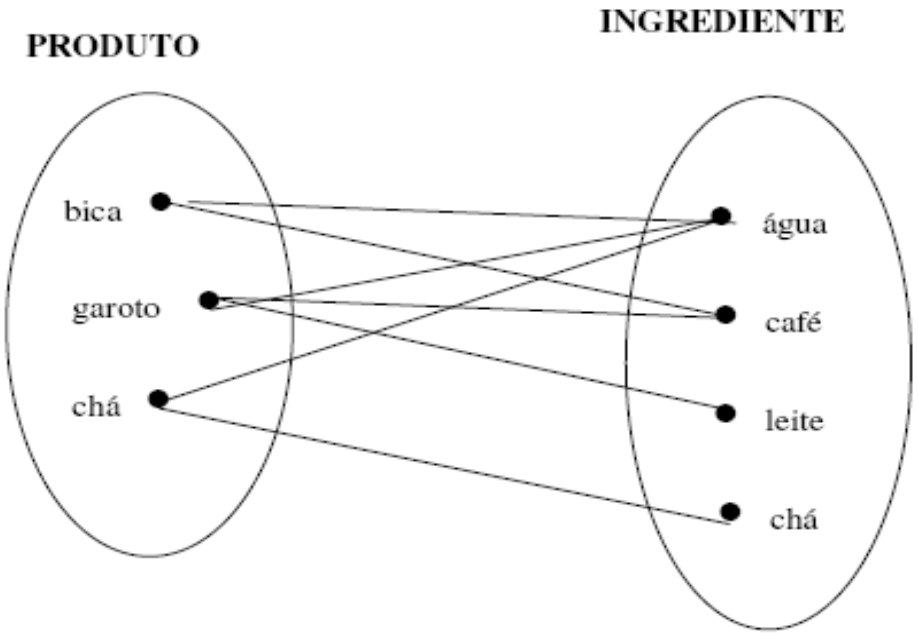
- Identifique alguns atributos e comportamentos simples para as classes abaixo e implemente.

UNIVERSIDADE VILA VELHA-ES

Exercício

- Se existem informações que necessitam ser armazenadas devido a relação entre as classes poderá haver vantagem na inclusão de uma Classe Intermediária

- Consideremos, como exemplo, a relação entre as classes Produto e Ingrediente.



UNIVERSIDADE VILA VELHA-ES

Exercício

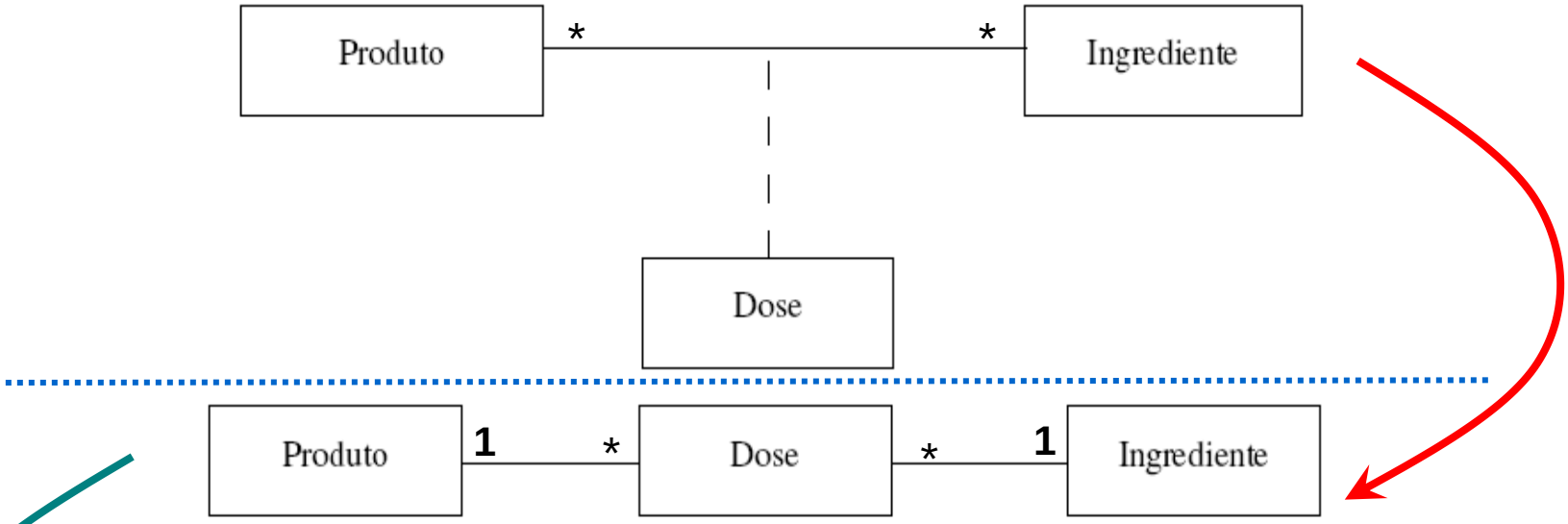
- Um Produto é criado a partir de um ou mais Ingredientes segundo um determinado conjunto de doses.
  - O conceito de Dose poderá ser implementado em nível dos métodos da classe Produto.

- Outra solução, talvez mais flexível, será a inclusão de uma classe de nome Dose que ajuda a estabelecer a relação entre as classes Produto e Ingrediente:
  - Diz-se, nesse caso, que Dose é uma classe de objectificação da relação.

UNIVERSIDADE VILA VELHA-ES

Relações N para N

Nossa modelagem ficaria assim…




```
class Produto{
 private Set doses;
....
}
```

```
class Dose{
private Produto prod;
private Ingrediente ing;
private int quantidade;
...
}
```

```
class Ingrediente{
 private Set doses;
...
}
```

UNIVERSIDADE VILA VELHA-ES

Exercício

- Identifique alguns atributos e comportamentos simples para as classes abaixo e implemente.