

Programação Orientada a Objetos

Fausto Maranhão Ayres

10

Relacionamento entre objetos (Bidirecional)

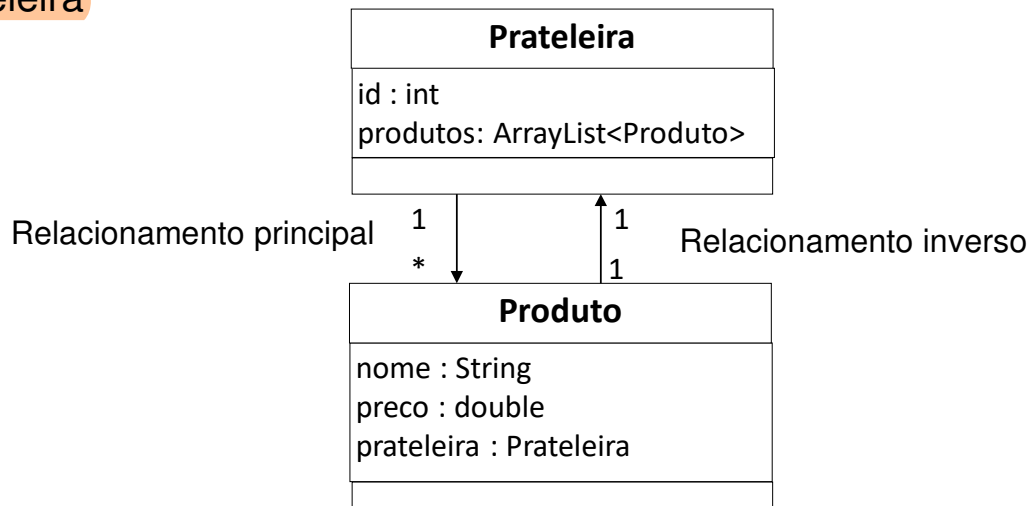
Implementação do relacionamento bidirecional um-para-muitos (1:*)

Relacionamento bidirecional 1:*

(1:*) bidirecional combina (1:*) unidirecional com (1:1) unidirecional invertido

Ex: Sistema Almoxarifado

Uma prateleira tem vários produtos e cada produto tem apenas uma prateleira



fausto.ayres@ifpb.edu.br

3

Classe Prateleira

```
public class Prateleira {
    private int id;
    private int tamanho;
    private ArrayList<Produto> produtos = new ArrayList<>();

    public void adicionar(Produto p) {
        produtos.add(p);
    }

    public void remover(Produto p) {
        produtos.remove(p);
    }

    public Produto localizar(String nome) {
        for (Produto p: produtos) {
            if (p.getNome().equals(nome))
                return p;
        }
        return null;
    }
}
```

Diagram annotations:

- relacionamento 1:*
- vazio
- Busca pelo nome
- retorna o produto encontrado
- No caso de não encontrar

fausto.ayres@ifpb.edu.br

4

Classe Produto

```
public class Produto{
    private String nome;
    private double preco;
    private Prateleira prateleira;

    public void setPrateleira(Prateleira p) {
        prateleira = p;
    }
    public Prateleira getPrateleira() {
        return prateleira;
    }
    ...
}
```

relacionamento 1:1
inverso

fausto.ayres@ifpb.edu.br

5

Criar relacionamentos

```
Produto arroz, feijao, carne, leite;
arroz = new Produto("arroz", 5.0) ;
feijao = new Produto("feijao", 8.0) ;
carne = new Produto("carne", 40.0) ;
leite = new Produto("leite", 5.0) ;
Prateleira prat1 = new Prateleira(1, 10);
Prateleira prat2 = new Prateleira(2, 20) ;

prat1.adicionar(arroz);
prat1.adicionar(feijao);
prat1.adicionar(carne);
prat2.adicionar(leite);
arroz.setPrateleira(prat1);
feijao.setPrateleira(prat1);
carne.setPrateleira(prat1);
leite.setPrateleira(prat2);
...
```

Relacionar as prateleiras com
seus produtos e vice-versa

fausto.ayres@ifpb.edu.br

6

Cont.

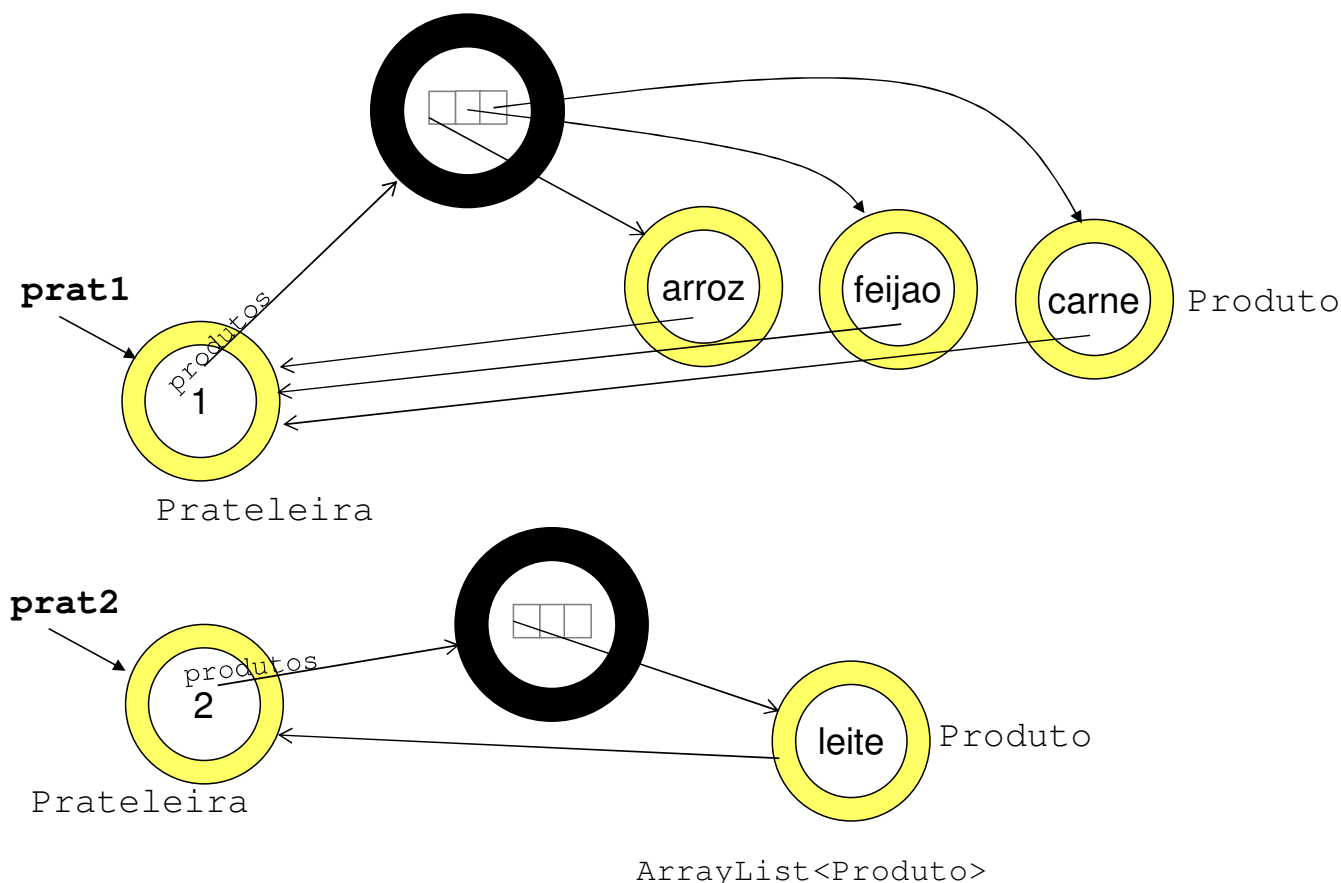
```
...  
//exibir os objetos relacionados  
System.out.println(prat1);  
System.out.println(prat2);  
  
System.out.println(arroz);  
System.out.println(feijao);  
System.out.println(carne);  
System.out.println(leite);  
...
```

Relacionar as prateleiras com
seus produtos e vice-versa

fausto.ayres@ifpb.edu.br

7

Grafo de objetos resultante



8

Cont.

```
//localizar leite na prateleira 2
Produto aux = prat2.localizar("leite");

if(aux == null)
    System.out.println("não localizou");
else
    System.out.println("localizou:" + aux);
```

Cont.

```
//transferir arroz para prateleira 2
prat1.remove(arroz);
prat2.adicionar(arroz);
arroz.setPrateleira(prat2);
System.out.println("transferiu arroz para prateleira 2");
```

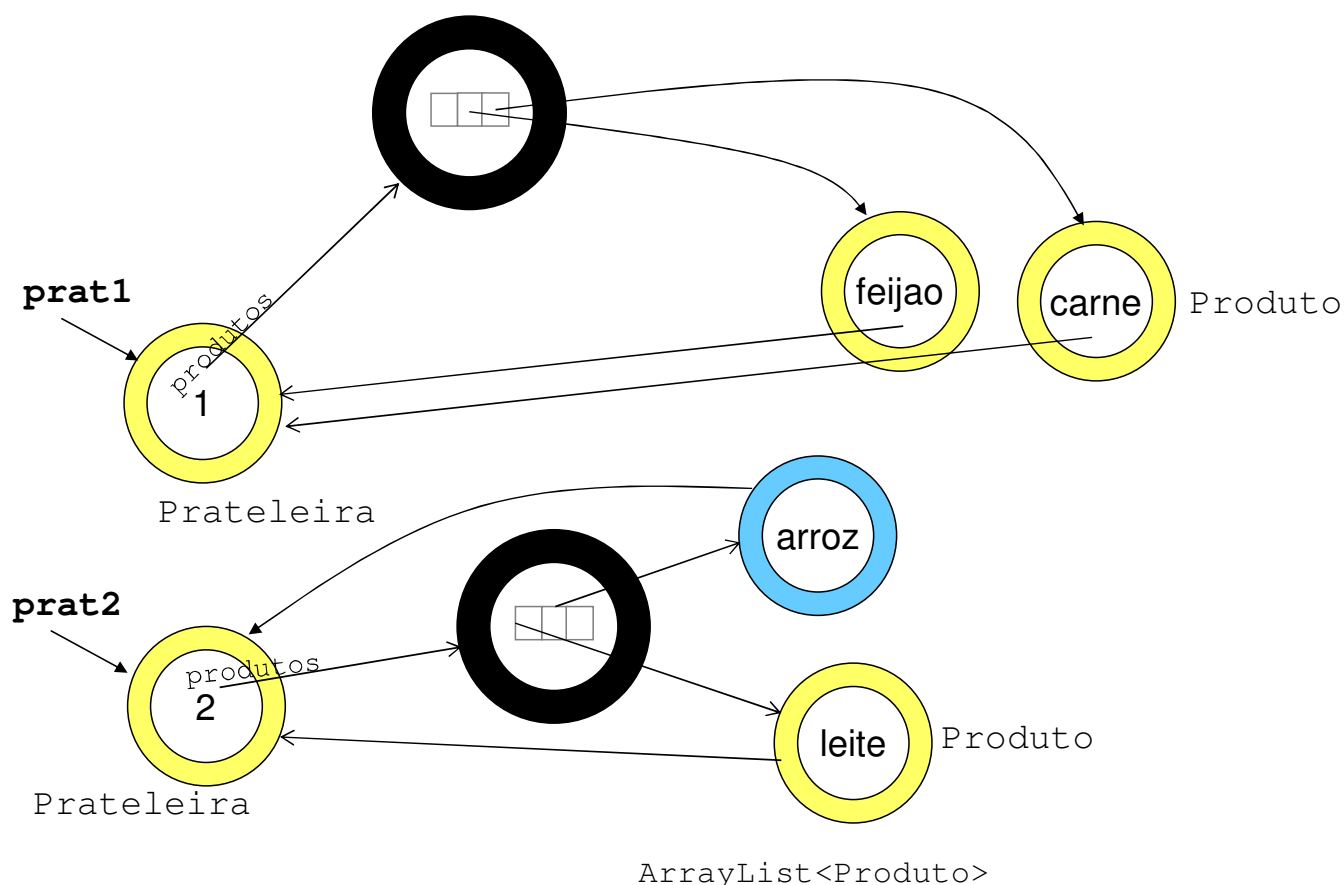
Cont.

```
...  
//exibir os objetos relacionados (final)  
System.out.println(prat1);  
System.out.println(prat2);  
  
System.out.println(arroz);  
System.out.println(feijao);  
System.out.println(carne);  
System.out.println(leite);  
...
```

fausto.ayres@ifpb.edu.br

11

Grafo de objetos resultante



12

Otimizar a implementação

Otimizar adicionar() e remover()

- adicionar os 2 relacionamentos invertidos no mesmo momento
- remover os 2 relacionamentos invertidos no mesmo momento

```
public class Prateleira{...
```

```
    public void adicionar(Produto p) {
```

relacionamento
principal

```
        produtos.add(p);
```

```
        p.setPrateleira(this);
```

relacionamento
inverso

```
    }
```

```
    public void remover(Produto p) {
```

relacionamento
principal

```
        produtos.remove(p);
```

```
        p.setPrateleira(null);
```

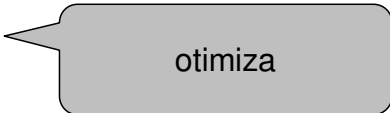
relacionamento
inverso

```
    }
```

Teste com otimização

```
Produto arroz, feijao, carne, leite;  
arroz = new Produto("arroz", 5.0) ;  
feijao = new Produto("feijao", 8.0) ;  
carne = new Produto("carne", 40.0) ;  
leite = new Produto("leite", 5.0) ;  
Prateleira prat1 = new Prateleira(1, 10);  
Prateleira prat2 = new Prateleira(2, 20) ;
```

```
prat1.adicionar(arroz);  
prat1.adicionar(feijao);  
prat1.adicionar(carne);  
prat2.adicionar(leite);  
arroz.setPrateleira(prat1);  
feijao.setPrateleira(prat1);  
carne.setPrateleira(prat1);  
leite.setPrateleira(prat2);  
...
```



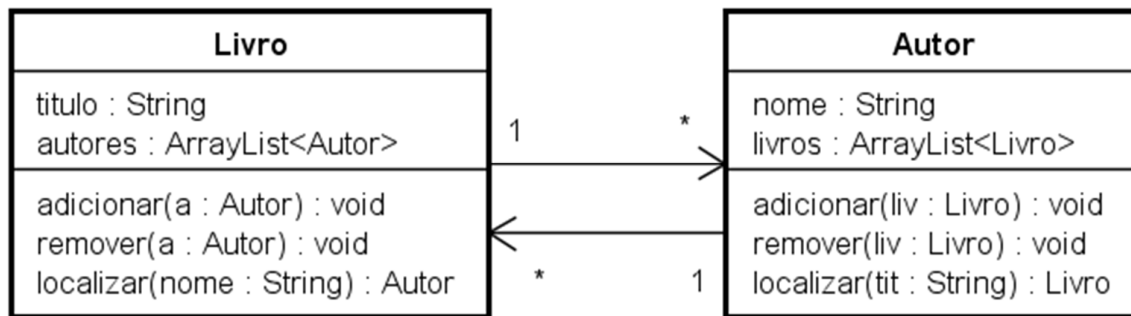
otimiza

**Implementação do
relacionamento bidirecional
muitos-para-muitos (*:*)**

Relacionamento bidirecional *:*

Um relacionamento (*:*) combina 2 relacionamentos unidirecionais (1:*) invertidos

Ex: um livro tem vários autores e um autor tem vários livros



Classe Livro

```
public class Livro {
    private String titulo;
    private ArrayList<Autor> autores = new ArrayList<>();

    public void adicionar(Autor aut) {
        autores.add(aut);
    }
    public void remover(Autor aut) {
        autores.remove(aut);
    }
    public Autor localizar(String nome) {
        for(Autor aut: autores)
            if(aut.getNome().equals(nome))
                return aut;

        return null;
    }
}
```

relacionamento 1:*

vazio

Busca pelo nome do autor

retorna o autor encontrado

Em caso de não encontrar

Classe Autor

```
public class Autor {  
    private String nome;  
    private ArrayList<Livro> livros = new ArrayList<>();  
  
    public void adicionar(Livro liv) {  
        livros.add(liv);  
    }  
    public void remover(Livro liv) {  
        livros.remove(liv);  
    }  
    public Livro localizar(String titulo) {  
        for(Livro liv: livros)  
            if(liv.getTitulo().equals(titulo))  
                return liv;  
  
        return null;  
    }  
}
```

relacionamento 1:*

vazio

Busca pelo
titulo do livro

retorna o livro encontrado

Em caso de não encontrar

fausto.ayres@ifpb.edu.br

19

Criar objetos e relacionamentos

```
Livro java = new Livro("java", "ciência moderna",...);  
Livro php = new Livro("php", "pratica",...);  
Autor joao = new Autor("joao");  
Autor maria = new Autor("maria");  
  
java.adicionar(joao);    //rel Livro -> Autor  
java.adicionar(maria);  
php.adicionar(maria);  
joao.adicionar(java);    //rel Autor -> Livro  
maria.adicionar(java);  
maria.adicionar(php);  
  
//transferir autor "joao"  
Autor a = java.localizar("joao");  
if(a!=null){  
    java.remover(a);    php.adicionar(a);  
    a.remover(java);    a.adicionar(php);  
}
```

Obs: pode-se otimizar adicionar/remover

fausto.ayres@ifpb.edu.br

20

Otimizar adicionar() e remover()

```
public class Livro {  
    private String titulo;  
    private ArrayList<Autor> autores = new ArrayList<>();  
  
    public void adicionar(Autor aut) {  
        autores.add(aut);  
        aut.add(this);  
    }  
    public void remover(Autor aut) {  
        autores.remove(aut);  
        aut.remove(this);  
    }  
    ...  
}
```

relacionamento 1:*

fausto.ayres@ifpb.edu.br

21

Criar objetos e relacionamentos

```
Livro java = new Livro("java", "ciência moderna",...);  
Livro php = new Livro("php", "pratica",...);  
Autor joao = new Autor("joao");  
Autor maria = new Autor("maria");
```

```
java.adicionar(joao);  
java.adicionar(maria);  
php.adicionar(maria);
```

otimizado

```
joao.adicionar(java);  
maria.adicionar(java);  
maria.adicionar(php);
```

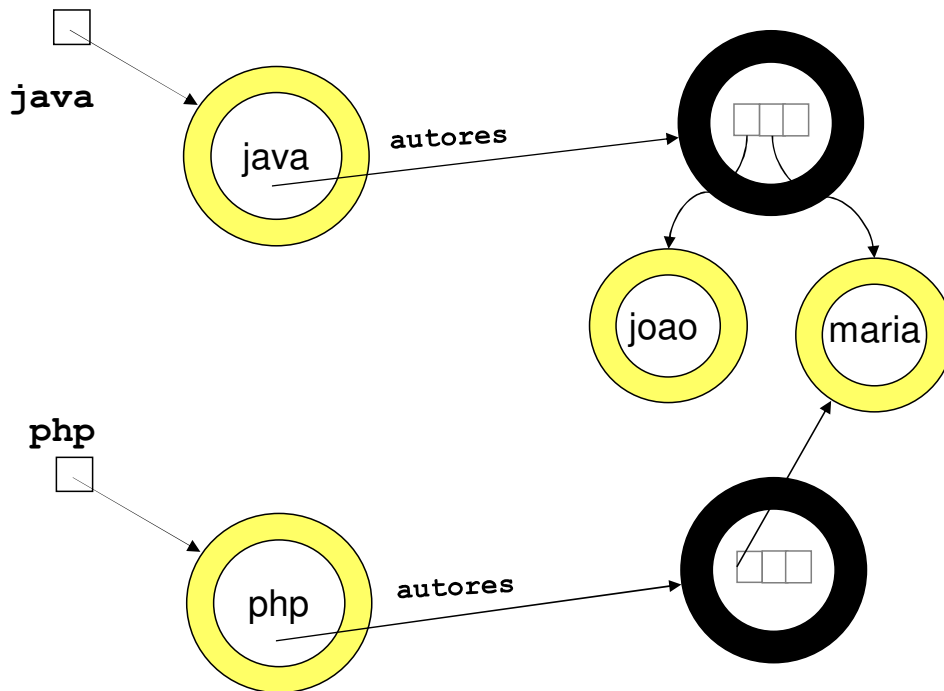
eliminado

fausto.ayres@ifpb.edu.br

22

Grafo de objetos

■ Relacionamentos Livro → Autor

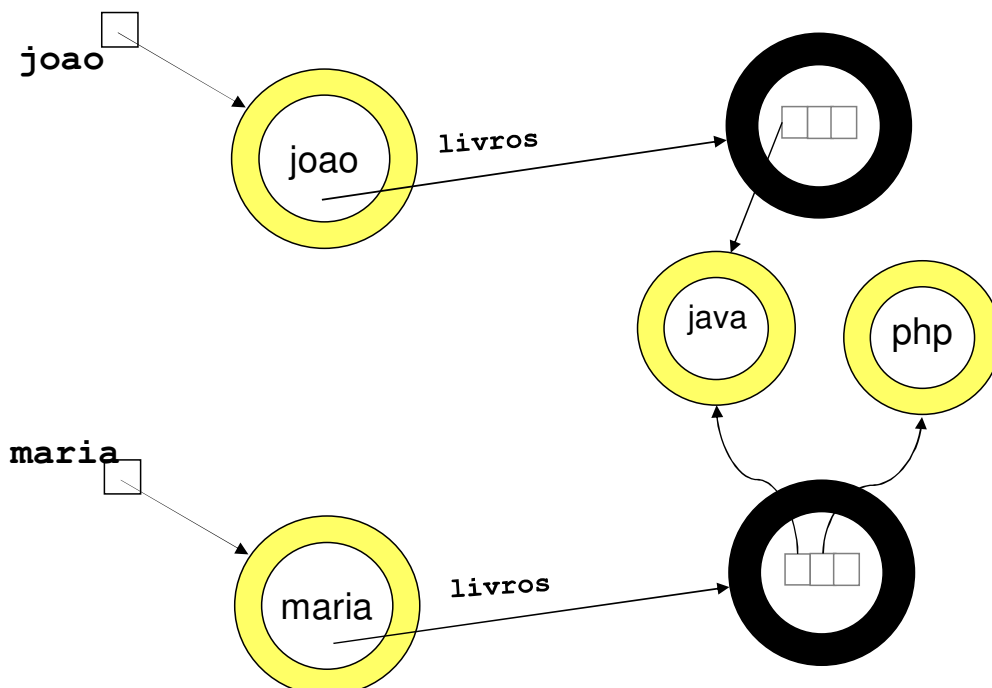


fausto.ayres@ifpb.edu.br

23

Cont.

■ Relacionamentos Autor → Livro

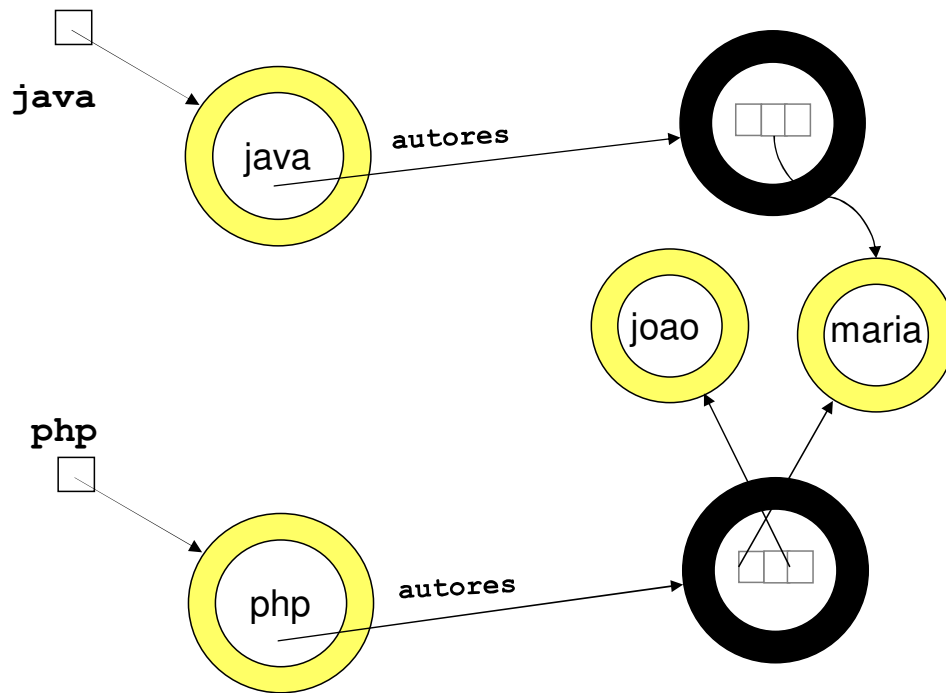


fausto.ayres@ifpb.edu.br

24

Alterar relacionamento

```
//transferir autor joao do livro java para livro php  
java.remove(joao);      //otimizado  
php.adicionar(joao);    //otimizado
```



fausto.ayres@ifpb.edu.br

25

Consultar

```
System.out.println("quantos livros maria tem?");  
System.out.println(maria.getLivros().size());
```

fausto.ayres@ifpb.edu.br

26