



Encapsulamento

Alberto Costa Neto
DComp - UFS



Motivação

- O que aprendemos até agora?
- O que diferencia a POO da Programação Imperativa?
- Estas informações são suficientes para construir um projeto OO eficiente?



Roteiro

- Pilares da Programação Orientada a Objetos
- Entendo sobre Encapsulamento
- Exemplo
- Como ocultar?
- Por que Encapsular?
- O que Encapsular?



Pilares da POO

PROGRAMAÇÃO ORIENTADA A OBJETOS

ENCAPSULAMENTO

HERANÇA

POLIMORFISMO



Pilares da POO

PROGRAMAÇÃO ORIENTADA A OBJETOS

ENCAPSULAMENTO

HERANÇA

POLIMORFISMO



“Esconda seu jogo!”



Pilares da POO

PROGRAMAÇÃO ORIENTADA A OBJETOS

ENCAPSULAMENTO

HERANÇA

POLIMORFISMO



“Filho de peixe, peixinho é!”



Pilares da POO

PROGRAMAÇÃO ORIENTADA A OBJETOS

ENCAPSULAMENTO

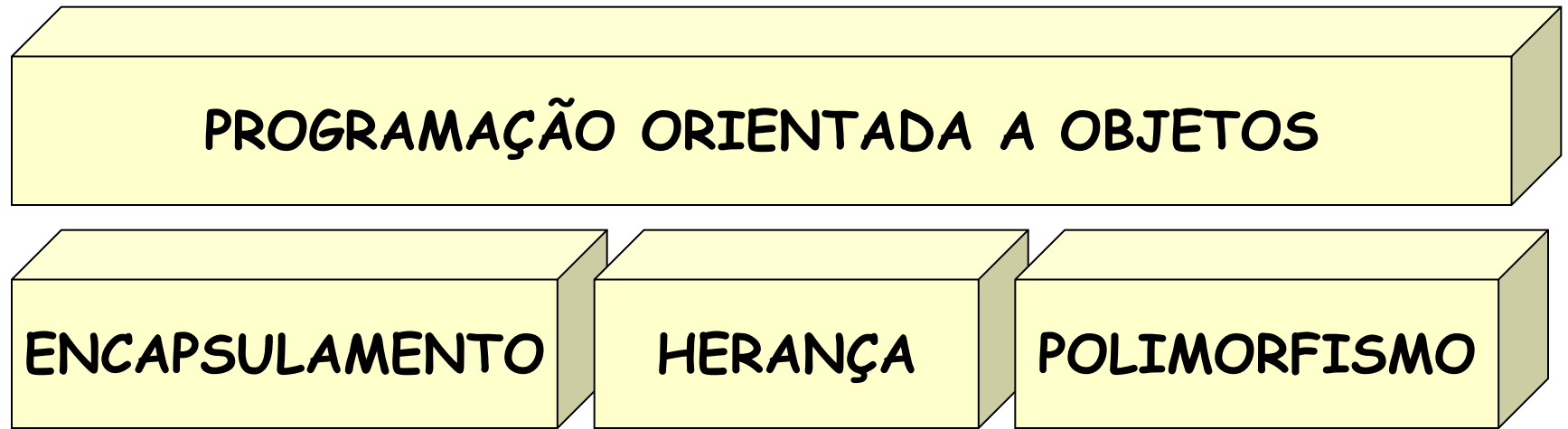
HERANÇA

POLIMORFISMO



"Sou camaleão!"

Pilares da POO



Devem estar presentes em uma linguagem para que a mesma seja considerada OO



Pilares da POO

- O que vamos estudar hoje...

PROGRAMAÇÃO ORIENTADA A OBJETOS

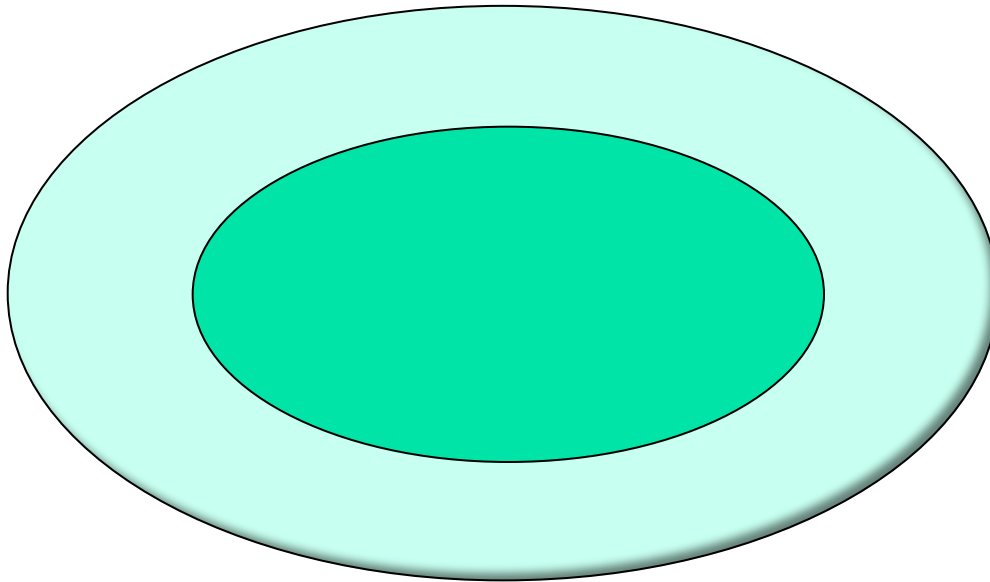
ENCAPSULAMENTO

HERANÇA

POLIMORFISMO

Entendo sobre Encapsulamento

- O que é encapsular?

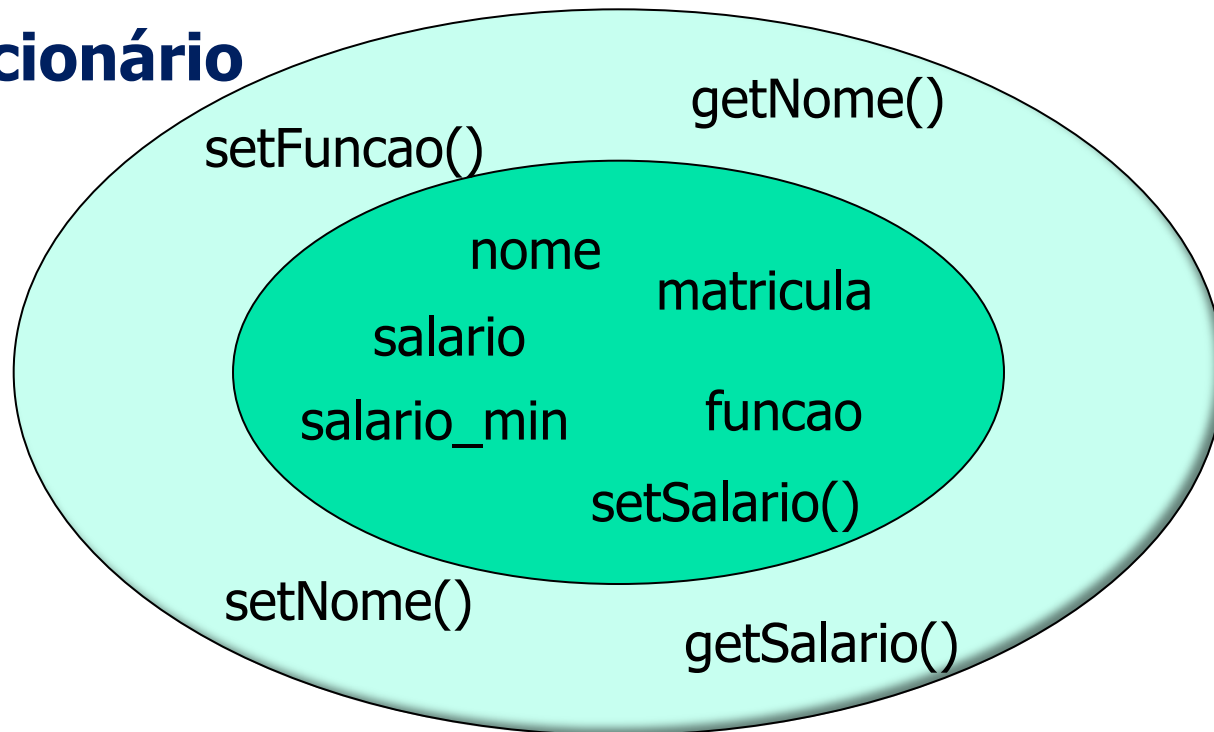


- Por que encapsular?

Entendo sobre Encapsulamento

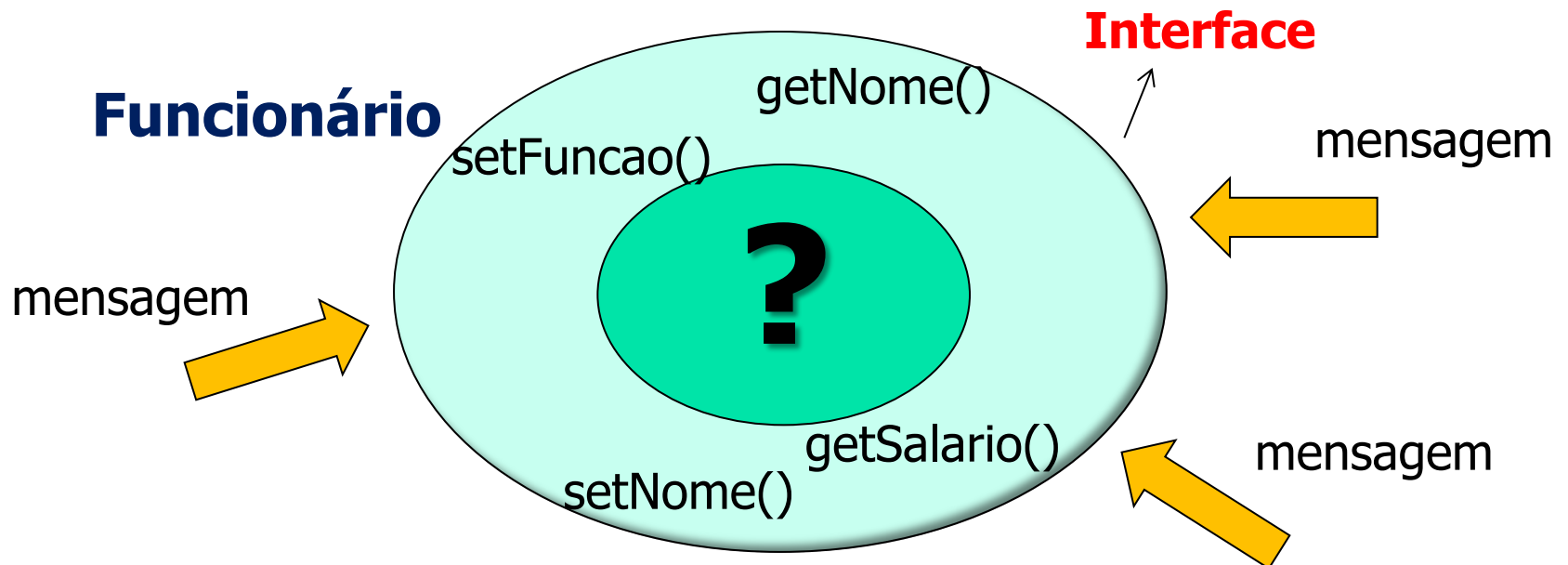
- O que é encapsulamento na POO?

Funcionário



Entendo sobre Encapsulamento

- O que é encapsulamento na POO?
 - Ocultamento de detalhes de implementação do mundo exterior



```
public class Funcionario {
```

```
    private String nome;  
    private int matricula;  
    private float salario;  
    private static float salario_min;  
    private char funcao;
```

```
    private float setSalario() {...}
```

```
    ...
```

```
    public String getNome() {...}  
    public float getSalario() {...}  
    public setNome (String n) {...}  
    public setFuncao(Char f) {...}
```

```
    ...
```

```
}
```

Implementação oculta
(Information Hiding)

Interface visível



Entendo sobre Encapsulamento

- A interface do componente...
 - Define o que uma entidade externa pode fazer com o componente
 - Lista os serviços fornecidos
 - Interface = painel de controle



Exemplo

- Implementando a classe Funcionário...

```
public class Funcionario {  
  
    public Funcionario (String nm, int mat, char funcao) {  
        this.nome = nm;  
        this.matricula = mat;  
        this.funcao = funcao;  
        this.setSalario();  
    }  
  
    ...  
    public void setFuncao (char func) {  
        if (this.funcao != func) {  
            this.funcao = func;  
            this.setSalario();  
        }  
    }  
    ...  
}
```



```
public class Funcionario {  
    ...  
    private void setSalario(){  
        switch (this.funcao) {  
            case 'G':  
                this.salario = 20 * salario_min;  
                break;  
            case 'T':  
                this.salario = 10 * salario_min ;  
                break;  
            case 'O':  
                this.salario = 4 * salario_min ;  
                break;  
            default:  
                this.salario = salario_min ;  
        }  
    }  
    ...  
}
```

```
public class Funcionario {  
    ...  
    public static void setSalarioMin (float min){  
        salario_min = min;  
    }  
  
    public String getNome() {  
        return this.nome;  
    }  
  
    public float getSalario() {  
        return this.salario;  
    }  
  
    public void setNome (String n) {  
        this.nome = n;  
    }  
    ...  
}
```



Exemplo

■ O código compila?

```
public class Exemplo1Encap {  
    public static void main (String[] teste) {  
        Funcionario.setSalarioMin (510f);  
        Funcionario func = new Funcionario ("Debora", 12, 'G');  
        System.out.println ("Funcionario : " + func.nome);  
        System.out.print (" salario : " + func.salario);  
    }  
}
```



Dever de sala

1) A partir do código contido nos slides anteriores:

- Criar a classe Funcionário
- O método `setSalario()` é visível?
- Crie a Classe `Exemplo1Encap`, fazendo com que ela funcione



Dever de sala

- 2) Crie a classe Empregado
- 3) Crie a classe Exemplo2Encap (contida no próximo slide)

```
public class Exemplo2Encap {  
    public static void main (String[] teste) {  
        Empregado.setSalarioMin (510f);  
  
        Empregado emp = new Empregado ("Paulo", 15, 'O');  
  
        System.out.println ("Funcionario : " + emp.nome);  
  
        System.out.print (" salario : " + emp.salario);  
  
        emp.salario = 25000f;  
  
        System.out.println ("Funcionario : " + emp.nome);  
  
        System.out.print (" salario : " + emp.salario);  
    }  
}
```



Dever de sala

- O código anterior compilou?
- Qual classe foi mais fácil usar?
- Quais os problemas no uso da classe Empregado?
- O que acontece se houver mudança na regra de definição do salário?



Como ocultar?

- De quem ocultar?
 - Objetos de classes pertencentes a outros pacotes
 - Objetos de classes pertencentes ao mesmo pacote
 - Objetos que herdam características da classe sendo definida



Como ocultar?

- Tipos de Visibilidade
 - Pública (*public*)
 - Privada (*private*)
 - Protegida (*protected*)
 - Pacote (*package*) → nem todas as linguagens

Por que Encapsular?

Encapsulamento e Ocultamento da informação



Comunicação através da interface do módulo



Independência da implementação



Facilita a manutenção



O que Encapsular?

- Usualmente
 - Estado do objeto
 - ➔ Exceto constantes
- Especificamente
 - Algum comportamento necessário



Referências

- Slides “Pilares da POO - Encapsulamento”. Prof^a. Débora. UFS. 2010.
- Slides “OO-Encapsulamento” Prof. Marcos Dósea. UFS. 2010.
- Aprenda Programação Orientada a Objetos em 21 dias.
 - Dia 2