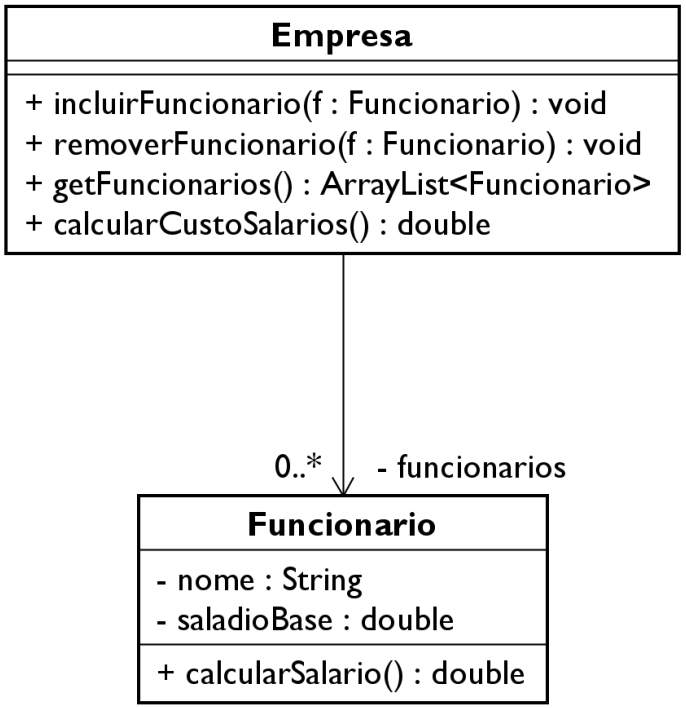
Polimorfismo

Motivação

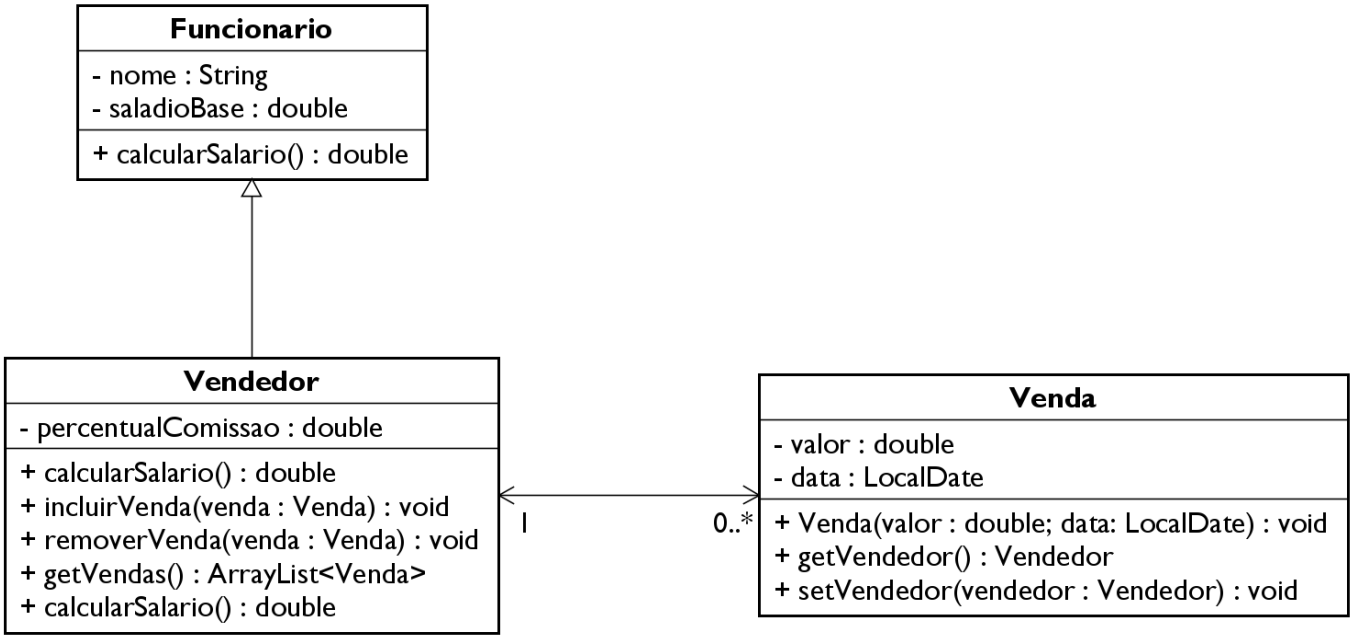
Uma empresa quer saber quanto deve pagar de salário para todos os seus funcionários



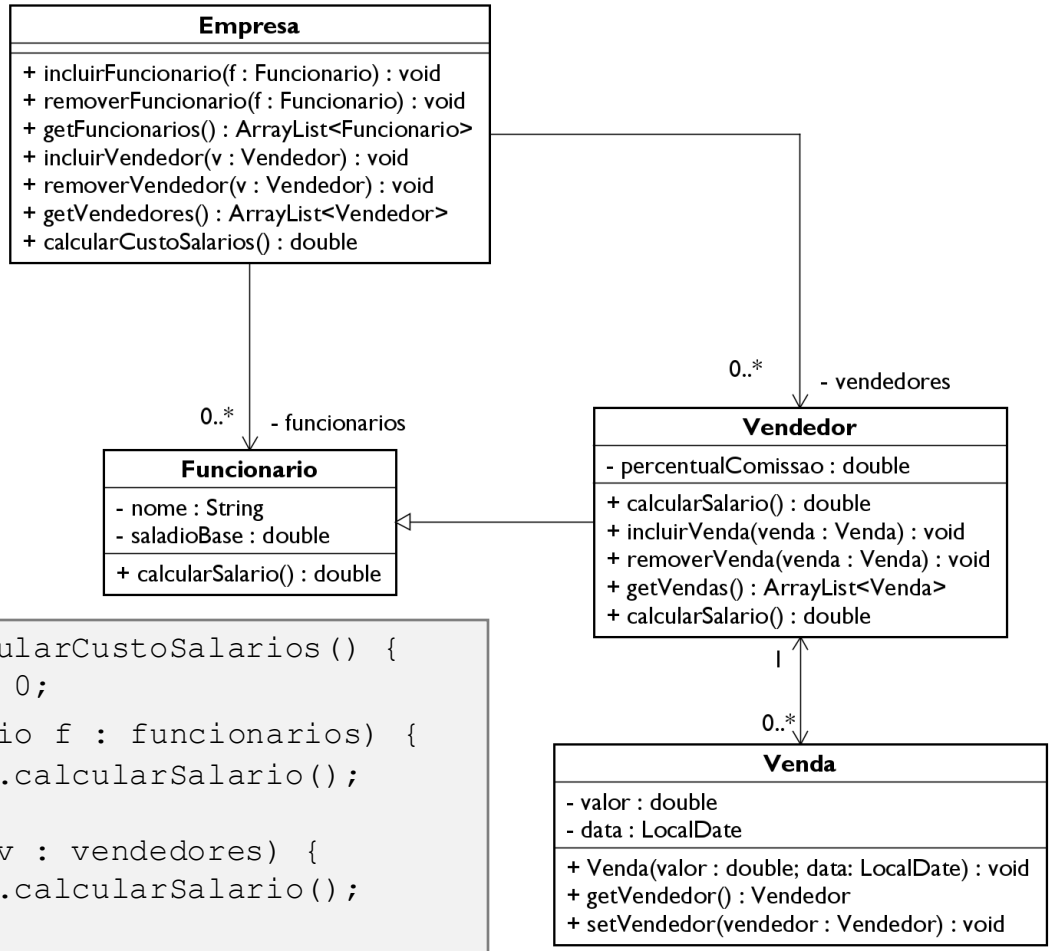

```
public double calcularCustoSalarios() {
    double total = 0;
    for (Funcionario f : funcionarios) {
        total += f.calcularSalario();
    }
    return total;
}
```

Motivação

Considerar que, existem alguns funcionários que trabalham com vendas. Os vendedores são comissionados



Motivação



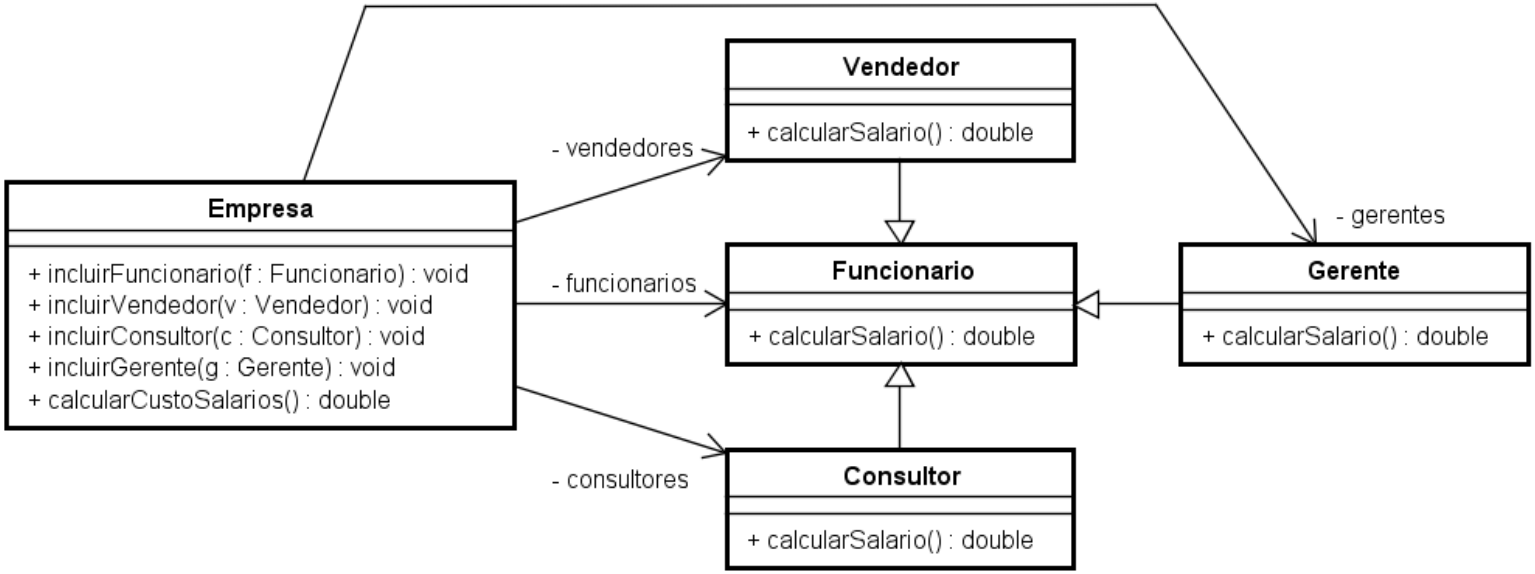

```
public double calcularCustoSalarios() {
    double total = 0;
    for (Funcionario f : funcionarios) {
        total += f.calcularSalario();
    }
    for (Vendedor v : vendedores) {
        total += v.calcularSalario();
    }
    return total;
}
```

Motivação

Considerar que a empresa possua também outros tipos de funcionários com cálculos de salários distintos.

Consultor: além de um valor fixo, o consultor recebe um valor extra a cada viagem a negócios que realizar

Gerente: além de um valor fixo, o gerente recebe um valor extra quando suas metas são atingidas



Motivação

```
public double calcularCustoSalarios() {
    double total = 0;

    for (Funcionario f : funcionarios) {
        total += f.calcularSalario();
    }

    for (Vendedor v : vendedores) {
        total += v.calcularSalario();
    }

    for (Consultor c : consultores) {
        total += c.calcularSalario();
    }

    for (Gerente g: gerentes) {
        total += g.calcularSalario();
    }

    return total;
}
```

Polimorfismo

Polimorfismo é um conceito utilizado em Programação Orientada a Objetos para definir a habilidade que diferentes objetos tem de responder, cada um da sua maneira, à chamadas idênticas de mensagens (métodos).

Polimorfismo

- Para atingir o polimorfismo, utilizamos uma **variável polimórfica**
  - Uma variável que pode referenciar tipos de objetos distintos
  - (por enquanto) é uma variável cujo tipo de dado é uma superclasse

- Dada uma variável polimórfica, ela pode referenciar:
  - Objetos de sua própria classe
  - Objetos cuja classe são subclasse (direta ou indireta) da classe da variável

Exemplo

- Declaração de variável polimórfica:

```
Funcionario f;
```

- Criação de objeto:

```
f = new Consultor();
```

- Observar a existência de dois tipos:
  - Tipo declarado na variável – tipo ***estático***
  - Tipo utilizado para instanciar o objeto – tipo ***dinâmico***

Exemplo de polimorfismo

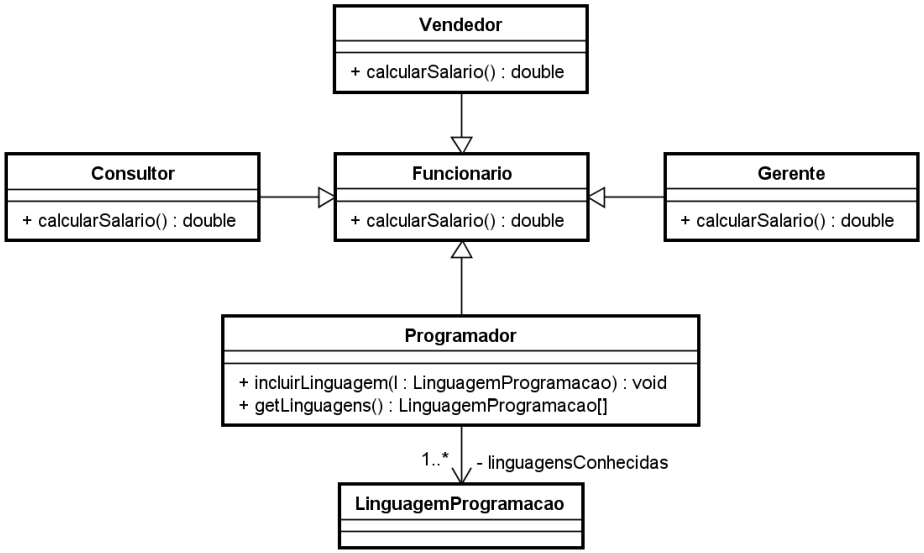
```
Funcionario[] funcionarios = new Funcionario[10];

funcionarios[0] = new Funcionario("José Silva");
funcionarios[1] = new Vendedor("Pedro Alcantara");
funcionarios[2] = new Funcionario("Cristina Lima");
funcionarios[3] = new Vendedor("Jorge Luna");
funcionarios[4] = new Consultor("Marcia Cristina de Souza");
...
funcionarios[9] = new Gerente("Lucas Gentil");
...

for (Funcionario f : funcionarios) {
    System.out.println(f.calcularSalario());
}
```

Variáveis polimórficas

O código Java que é acionado para atender **calcularSalario()** depende do tipo dinâmico

Quando **f** referenciar o tipo dinâmico **Vendedor**, será acionado o método **calcularSalario()** da classe **Vendedor** e não o da classe **Funcionario**.

Exemplo de polimorfismo




```
Funcionario f = new Programador("Márcio Santiago");

f.calcularSalario();
```

Quando o tipo dinâmico não sobrescreve o método, é acionado o código da superclasse

Polimorfismo

- Um objeto da classe **`Vendedor`** pode ser acessado através de uma variável cujo tipo é igual a sua superclasse (direta ou indireta), isto é, através de uma variável polimórfica

- Por padrão, ao utilizar uma variável polimórfica, somente os membros da classe do tipo desta variável podem ser utilizados

- Isto é:

```
Funcionario f = new Programador();
...
f.calcularSalario();
```
➡ compilável

```
f.incluirLinguagem(l);
```
➡ Não compilável

Downcasting

- Para acessar um método específico da subclasse, a partir de uma variável polimórfica, é necessário efetuar uma operação de conversão (cast). Esta operação é conhecida como *downcasting*.

```
Funcionario f = new Programador("Pedro");

LinguagemProgramacao l = new LinguagemProgramacao("C");

((Programador) f).incluirLinguagem(l);
```

Polimorfismo

- Esta é uma operação ilegal e não compila.

```
Vendedor v = new Funcionario("André Simas");
```

- Dada uma variável do tipo referência, é possível conferir se uma determinada classe pertence a sua hierarquia de classes.
- Exemplo:

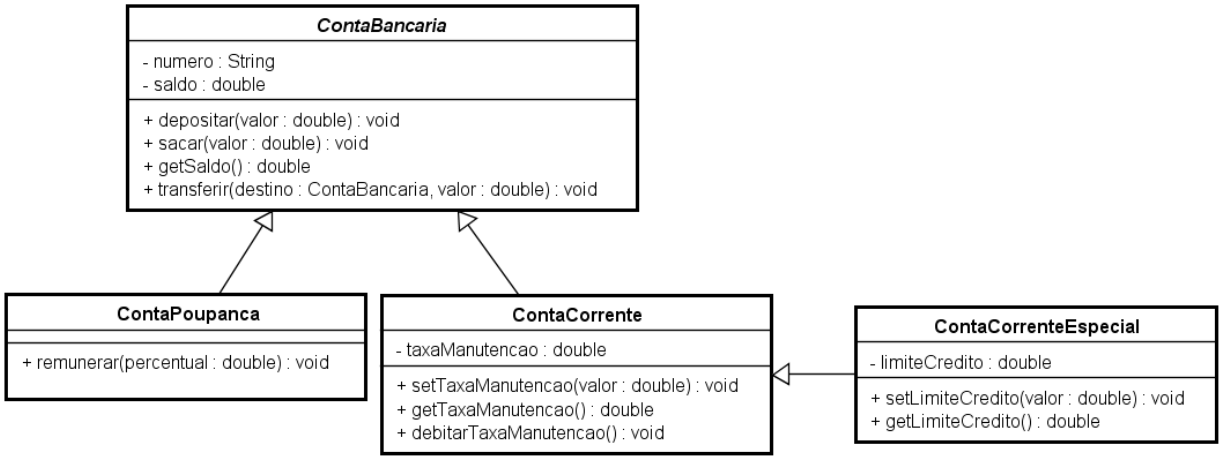
```
for (Funcionario f : funcionarios) {
    System.out.print("\nNome: " + f.getNome() +
                     ". Salário Base: " + f.getSalarioBase());
    if (f instanceof Vendedor) {
        System.out.print(". Total vendas: " +
                         ((Vendedor) f).calcularTotalVendas());
    }
}
```

Exemplo de polimorfismo

```
public class PrintStream {

    public void println(Object x) {
        String s = String.valueOf(x);
        print(s);
        newLine();
    }

}
```

```
public final class String {

    public static String valueOf(Object obj) {
        return (obj == null) ? "null" : obj.toString();
    }

}
```

Exemplo 2



Método **`transferir()`** da classe **`ContaBancaria`**:

```
public void transferir(ContaBancaria contaDestino, double valor) {
    this.sacar(valor);
    contaDestino.depositar(valor);
}
```

Polimorfismo

- O polimorfismo permite que os desenvolvedores programem de forma mais abstrata

- Permite evitar comandos condicionais para tratamento de casos especiais