INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA Campus Irecê	**Instituto Federal de Educação, Ciência e Tecnologia da Bahia – *Campus* Irecê** **Disciplina: Linguagem Técnica I** **Prof° Jonatas Bastos**

Nome: _____

LISTA DE EXERCÍCIO 3 – ARRAYS

1) Volte ao nosso sistema de Funcionario e crie uma classe Empresa dentro do mesmo arquivo .java. A Empresa tem um nome, cnpj e uma referência a uma array de Funcionario, além de outros atributos que você julgar necessário.

```
class Empresa {

    // outros atributos
    Funcionario[] empregados;
    String cnpj;
}
```

2) A Empresa deve ter um método adiciona, que recebe uma referência a Funcionario como argumento, e guarda esse funcionário. Algo como:

```
...

    void adiciona(Funcionario f) {
        // algo tipo:
        //   this.empregados[ ??? ] = f;
        // mas que posição colocar?
    }
...
```

Você deve inserir o Funcionario em uma posição da array que esteja livre. Existem várias maneiras para você fazer isso: guardar um contador para indicar qual a próxima posição vazia ou procurar por uma posição vazia toda vez. O que seria mais interessante?

É importante reparar que o método adiciona não recebe nome, rg, salário, etc. Essa seria uma maneira nem um pouco estruturada, muito menos orientada a objetos de se trabalhar. Você antes cria um Funcionario e já passa a referência dele, que dentro do objeto possui rg, salário, etc.

3) Crie uma classe TestaEmpresa que possuirá um método main. Dentro dele crie algumas instâncias de Funcionario e passe para a empresa pelo método adiciona. Repare que antes você vai precisar criar a array, pois inicialmente o atributo empregados da

classe Empresa não referencia lugar nenhum (null):

```
 Empresa empresa = new Empresa();

empresa.empregados = new Funcionario[10];
//      ....
```

Ou você pode construir a array dentro da própria declaração da classe Empresa, fazendo com que toda vez que uma Empresa é instanciada, a array de Funcionario que ela necessita também é criada.

Crie alguns funcionários e passe como argumento para o adiciona da empresa:

```
Funcionario f1 = new Funcionario();

f1.salario = 1000;
empresa.adiciona(f1);


Funcionario f2 = new Funcionario();
f2.salario = 1700;
empresa.adiciona(f2);
```

Você pode criar esses funcionários dentro de um loop, e dar valores diferentes de salários:

```
for (int i = 0; i < 5; i++) {

    Funcionario f = new Funcionario();
    f.salario = 1000 + i * 100;
    empresa.adiciona(f);
}
```

Repare que temos de instanciar Funcionario dentro do laço. Se a instanciação de Funcionario ficasse acima do laço, estaríamos adicionado cinco vezes a **mesma** instância de Funcionario nesta Empresa, e mudando seu salário a cada iteração, que nesse caso não é o efeito desejado.

Opcional: o método adiciona pode gerar uma mensagem de erro indicando quando o array já está cheio.

4) Percorra o atributo empregados da sua instância da Empresa e imprima os salários de todos seus funcionários.

Para fazer isso, você pode criar um método chamado mostraEmpregados dentro da classe Empresa:

```
...
    void mostraEmpregados() {
        for (int i = 0; i < this.empregados.length; i++) {
            System.out.println("Funcionário na posição: " + i);

            // preencher para mostrar o salário!!
        }
    }
...
```

Cuidado ao preencher esse método: alguns índices do seu array podem não conter referência para um Funcionario construído, isto é, ainda se referirem para null. Se preferir, use o for novo do java 5.0.

Aí, através do seu main, depois de adicionar alguns funcionários, basta fazer:

```
empresa.mostraEmpregados();
```

Em vez de mostrar apenas o salário de cada funcionário, você pode chamar o método mostra() de cada Funcionario da sua array.

5) Crie um método para verificar se um determinado Funcionario se encontra ou não como funcionário desta empresa:

```
 boolean contem(Funcionario f) {

    // ...
}
```

Você vai precisar fazer um for na sua array e verificar se a referência passada como argumento se encontra dentro da array. Evite ao máximo usar números hard-coded, isto é, use o .length.

6) Caso a array já esteja cheia no momento de adicionar um outro funcionário, criar uma nova maior e copiar os valores. Isto é, fazer a realocação já que java não tem isso: uma array nasce e morre com o mesmo length.