Exercícios obrigatórios com arrays:

1) No mesmo projeto que contém a classe funcionário, crie uma classe Empresa que terá nome, cnpj e uma referência a um array de Funcionário.

```
public class Empresa {
	private String nome, cnpj;
	private Funcionario[] empregados;

	//...
}
```

2) A Empresa deve ter um método adiciona que recebe uma referência a um Funcionário como argumento e guarda esse funcionário no array. Ficaria mais ou menos assim:

```
public class Empresa {
	private String nome, cnpj;
	private Funcionario[] empregados;

	public void adiciona(Funcionario func) {
		this.empregados[???] = func;
	}
}
```

Em que posição guardaríamos esse funcionário? Precisamos guardá-lo em uma posição do array que esteja livre. Existem algumas maneiras para se fazer isso. Uma delas seria percorrer o array em busca de uma posição vazia, que aponte para null. Outra forma mais elegante é guardar um contador que vai ter sempre a posição livre e a cada Funcionário adicionado este contador seria incrementado.

É importante salientar que o método *adiciona* não recebe nome, departamento, salário, etc. A referência de um funcionário aponta para um objeto da memória que já possui estes atributos. Então, passando essa referência, você já está passando o Funcionário completo, com todos os seus atributos. Veja como ficaria:

```
public class Empresa {
	private String nome, cnpj;
	private Funcionario[] empregados;
	private int posicaoLivre;

	public void adiciona(Funcionario f){
		this.empregados[this.posicaoLivre]=f;
		this.posicaoLivre++;
	}
	//métodos get e set...
}
```

3) Crie uma um programa TestaEmpresa, que obviamente deverá possuir um método main. Crie uma empresa e várias instâncias de Funcionário, passando para a Empresa através do método adiciona. Antes você vai precisar criar o array já que empregados por enquanto referencia null (lugar nenhum).

```
public class TestaEmpresa {
	public static void main(String[] args) {
		Empresa empresa = new Empresa();
```

```
            empresa.setEmpregados(new Funcionario[5]);
            //...
        }
}
```

Outra forma de se fazer a mesma coisa é na criação da Empresa já criar um array para empregados. Veja:

```
public class Empresa {
      private String nome, cnpj;
      private Funcionario[] empregados = new Funcionario[5];
      private int posicaoLivre;

      public void adiciona(Funcionario f){
            this.empregados[this.posicaoLivre]=f;
            this.posicaoLivre++;
      }
      //métodos get e set...
 }
```

No programa TestaEmpresa, crie alguns funcionários, preencha seus atributos e adicione à Empresa através do método adiciona.

```
public class TestaEmpresa {
    public static void main(String[] args) {
        Empresa empresa = new Empresa();

        // Funcionário 1
        Funcionario f1 = new Funcionario();
        f1.setNome("Rafael");
        f1.setDepartamento("INFORMÁTICA");
        f1.setSalario(2000);
        f1.aumentarSalario(10);
        f1.getDataDeNascimento().setDia("06");
        f1.getDataDeNascimento().setMes("11");
        f1.getDataDeNascimento().setAno("1974");

        // Funcionário 2
        Funcionario f2 = new Funcionario();
        f2.setNome("Maria");
        f2.setDepartamento("BIBLIOTECA");
        f2.setSalario(3000);
        f2.aumentarSalario(5);
        Data data = new Data();
        data.setDia("10");
        data.setMes("04");
        data.setAno("1995");
        f2.setDataDeNascimento(data);

        // adicionando os funcionários à Empresa
        empresa.adiciona(f1);
        empresa.adiciona(f2);
    }
}
```

4) Usando o foreach, percorra o atributo empregados de sua Empresa e imprima todas as informações a respeito de cada funcionário. Para fazer isso crie um método mostra empregados dentro de sua classe empresa. Cuidado, alguns

índices do seu array podem não conter uma referência para um Funcionário construído, ainda se referindo a null. Trate isso.

5) Lembra que na classe Funcionário você havia criado um método mostra()? No método mostraEmpregados() da Empresa, ao invés de imprimir atributo a atributo, use o métdo mostra() de Funcionário.
6) Na classe Empresa, crie um método para verificar se um determinado funcionário faz parte da Empresa (ou seja: verificar se ele existe no array empregados). Se você tem um array de 100 posições e o funcionário que vc procura foi encontrado na posição 8, você precisa percorrer o restante do array? Pense nisso antes de escrever este método!
7) E se na hora de adicionar um novo funcionário o array já estiver cheio? Um array não muda de tamanho, lembra? Caso isso aconteça, criar um novo array, realocando os valores do array antigo para o novo array. Que tal criar um novo método encapsulado chamado aumentarArray?
8) Para que nosso sistema continue funcionando normalmente, ao remover um funcionário o array precisa ser reorganizado. Escreva um método para remover um funcionário do array. Escreva também outro método encapsulado para reorganizar o array e saiba como chama-lo toda vez que um Funcionário for removido. Que tal chamá-lo de reorganiza?
9) Faça testes na bancada de objetos do BlueJ.
10) **Dica:** Existe uma classe utilitária em Java chamada Arrays. Pesquise sobre ela. Com o uso da classe Arrays você poderia, por exemplo, utilizar este código para redimensionar o array:

```
this.empregados = Arrays.copyOf(this.empregados, this.empregados.length+1);
```

O método copyOf é o que chamamos de método estático. Observe que pudemos chama-lo diretamente, sem precisar instanciar um objeto da classe Array. Na classe Array existem métodos estáticos para ordenar os elementos do array, entre outros.