Tópicos

Vetores

- Declarando um vetor de inteiros:

```
int[] idades;
```

- O que a variável *idades* é?

Vetores

- Declarando um vetor de inteiros:

```
int[] idades;
```

- O que a variável *idades* é?
- Referência! Precisamos criar posições de memória para podermos acessar esse vetor.

```
idades = new int[10];
```

- O que acontece se fizermos:

```
idades[11] = 4;
```

Vetores de Referências

- Suponha a seguinte declaração:

```
Conta[] minhasContas;
minhasContas = new Conta[10];
```

- Quantas contas foram criadas aqui?

Vetores de Referências

- Suponha a seguinte declaração:

```
Conta[] minhasContas;
minhasContas = new Conta[10];
```

- Quantas contas foram criadas aqui?
- Nenhuma! Foram criados 10 espaços capazes de armazenar uma referência para uma conta.

Vetores de Referências

- O que o seguinte acesso geraria:

```
minhasContas[0].saldo = 23.45;
```

Vetores de Referências

- O que o seguinte acesso geraria:

```
minhasContas[0].saldo = 23.45;
```

- Você deve popular seu vetor antes:

```
Conta contaNova = new Conta();
contaNova.saldo = 1000.0;
minhasContas[0] = contaNova;
```

Vetores de Referências

- O que o seguinte acesso geraria:

```
minhasContas[0].saldo = 23.45;
```

- Você deve popular seu vetor antes:

```
Conta contaNova = new Conta();
contaNova.saldo = 1000.0;
minhasContas[0] = contaNova;
```

- Vetores de tipos primitivos guardam valores, enquanto vetores de objetos guardam referências.

Percorrendo um Vetor

- Quando nós o criamos:

```
public static void main(String args[]) {
    int[] idades = new int[10];
    for (int i = 0; i < 10; i++) {
        idades[i] = i * 10;
    }
    for (int i = 0; i < 10; i++) {
        System.out.println(idades[i]);
    }
}
```

Percorrendo um Vetor

- E quando recebemos um vetor como argumento?

```
void imprimeVetor(int[] vetor){
    // não compila!!
    for (int i = 0; i < ????; i++) {
        System.out.println(array[i]);
    }
}
```

Percorrendo um Vetor

- Todo vetor Java tem um atributo *length*, que você pode acessá-lo.

```
void imprimeVetor(int[] vetor){
    // não compila!!
    for (int i = 0; i < array.length; i++) {
        System.out.println(array[i]);
    }
}
```

Percorrendo um Vetor

- Todo vetor Java tem um atributo *length*, que você pode acessá-lo.

```
void imprimeVetor(int[] vetor){
    // não compila!!
    for (int i = 0; i < array.length; i++) {
        System.out.println(array[i]);
    }
}
```

- Vetores não podem mudar de tamanho!

Percorrendo um Vetor no Java 5.0

- Java 5.0 traz nova sintaxe para percorrermos vetores:

```
class AlgumaClasse{
    public static void main(String args[]) {
        int[] idades = new int[10];
        for (int i = 0; i < 10; i++) {
            idades[i] = i * 10;
        }
        // imprimindo toda a array
        for (int x : idades) {
           System.out.println(x);
        }
    }
}
```

- O mesmo é válido para vetores de referências.

Exemplos

- Crie uma classe *Empresa* que contém o conjunto de elementos de um tipo *Funcionario*. Um funcionário deve ter *nome*, *cpf*, *salario* e outros atributos que julgar necessário. Além de um conjunto de funcionários, uma *Empresa* tem um *nome*, *cnpj* e outros atributos que julgar necessário.

Exemplos

- Crie uma classe *Empresa* que contém o conjunto de elementos de um tipo *Funcionario*. Um funcionário deve ter *nome*, *cpf*, *salario* e outros atributos que julgar necessário. Além de um conjunto de funcionários, uma *Empresa* tem um *nome*, *cnpj* e outros atributos que julgar necessário.
- A empresa deve ter um método *adiciona*, que recebe uma referência a um *Funcionário* como argumento e guarda esse funcionário.

Exemplos

- Crie uma classe *Empresa* que contém o conjunto de elementos de um tipo *Funcionario*. Um funcionário deve ter *nome*, *cpf*, *salario* e outros atributos que julgar necessário. Além de um conjunto de funcionários, uma *Empresa* tem um *nome*, *cnpj* e outros atributos que julgar necessário.
- A empresa deve ter um método *adiciona*, que recebe uma referência a um *Funcionário* como argumento e guarda esse funcionário.
- Crie uma classe *TestaEmpresa* que possuirá um método *main*. Dentro dele crie algumas instâncias de *Funcionario* e passe para a empresa pelo método *adiciona*.

Exemplos

- Percorra o atributo *empregados* da sua instância da *Empresa* e imprima os salários de todos os seus funcionários. Para isso, crie um método chamado *mostraEmpregados*.

Exemplos

- Percorra o atributo *empregados* da sua instância da *Empresa* e imprima os salários de todos os seus funcionários. Para isso, crie um método chamado *mostraEmpregados*.
- Crie um método para verificar se um determinado Funcionario se encontra ou não como funcionário desta empresa.

Exemplos

- Percorra o atributo *empregados* da sua instância da *Empresa* e imprima os salários de todos os seus funcionários. Para isso, crie um método chamado *mostraEmpregados*.
- Crie um método para verificar se um determinado Funcionario se encontra ou não como funcionário desta empresa.
- Altere todas as classes criadas anteriormente para que elas passem a utilizar *ArrayList* ao invés de vetores.