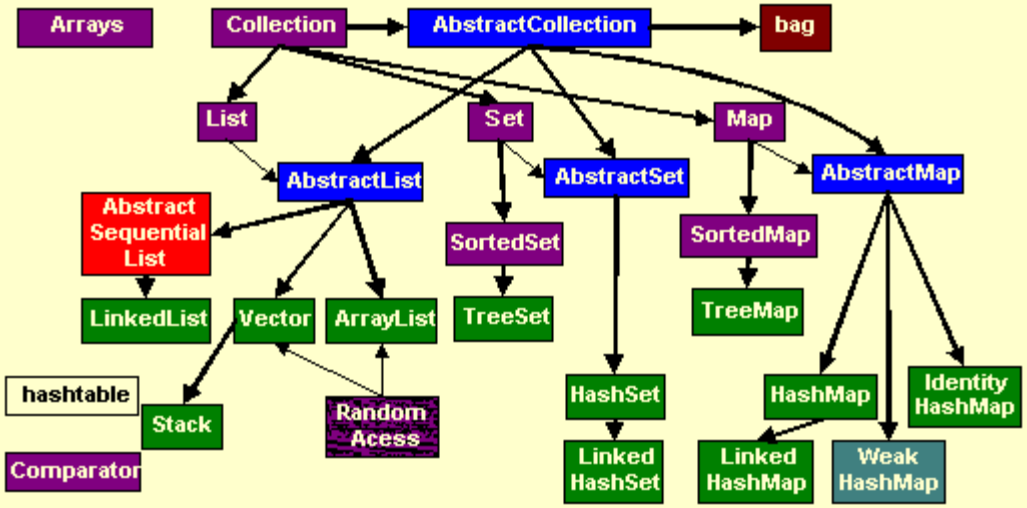
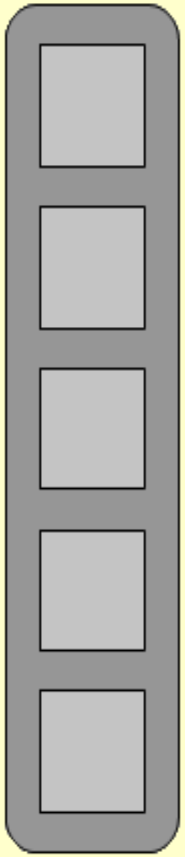
Arrays e Coleções em Java



Arrays



- Também chamados de vetores
  - Estrutura de dados homogênea (português estruturado)
  - É uma coleção ordenada, ou lista numerada, de valores do mesmo tipo
- Declaração através de colchetes ([])

```
int numeros[];                  int[] valores; // recomendado
String parametros[];            String[] nomes;
```

- Iniciando arrays:
  - Declarando e/ou especificando tamanho do vetor

```
byte[] num = new byte[10]; // tamanho 10
numeros = new int[256]; // declaração feita antes
String[] diasFolga = {"sab","dom"}; // tamanho 2
int[] idades = new int[] {42,30,1}; // tamanho 3
```

Arrays

- Utilização:
  - Somente após declaração e criação
  - Índice inicial: 0 (zero)
- Exemplos:

```
long[] numeros;
numeros = new long[10];
numeros[3] = 22;
numeros[7] = -988;
/* primeiro elemento */
numeros[0] = 26428;
/* último elemento → 9  */
numeros[numeros.length - 1] = 3;
System.out.println(numeros[9]);
```

```
String[] nome = new String[2];
nome[0] = "João";
nome[1] = "Maria";
nome[2] = "Ana"; // erro!!!
```

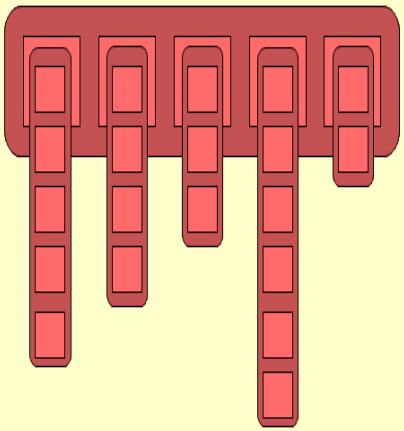
```
byte[] numeros = new byte[10];
for(int i=0;i<10;i++) {
  numeros[i] = (byte)i;
}
```

Arrays Multidimensionais

- São arrays de elementos arrays
  - Funcionam como matrizes;
  - Podem ter duas ou mais dimensões.
  - Exemplo:

```
int[][] valor = new int[5][10];
```

  - Funcionamento:
    - Declara *valor* para conter um array de array de **int;**
    - Cria 1 array de 5 elementos**;**
    - Cria 5 arrays, cada um é array de 10 elementos **int;**
    - Cada array de 10 elementos **int** criado é vinculado a cada elemento do array inicial de 5 elementos.



Arrays Multidimensionais

- Pode-se também declarar:

```
String[][] agenda = {{"JOÃO","2265555"},{"ANA","2345678"}};
float[][] salarios = new float[12][];
```

  - Neste caso, depois tem que criar `float[]` para cada elemento:

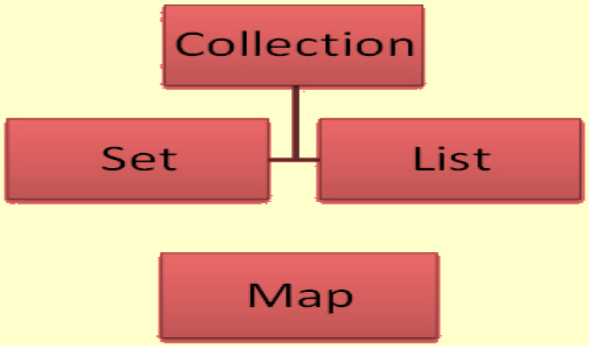
```
for (int i = 0; i < 12; i++)  {
  salarios[i] = new float[10]; // 10 funcionários
}
```

  - Mesma coisa para o seguinte:

```
float[][][] salarioAnual = new float[10][12][];
```

Coleções

- São classes oferecidos pelo Java que simulam e aperfeiçoam as funções de um vetor
- Funcionam como uma lista de objetos
  - Obs.: Tipos primitivos são convertidos automaticamente como objetos das suas respectivas classes
- Objetos de vários tipos são permitidos
  - Vetor só permite elementos de um único tipo
- Tamanho variável
  - Vetor possui número de elementos fixos e imutáveis



Coleções

- Vamos focar em duas classes de coleções:
  - **ArrayList** e **Vector**
    - Coleções de objetos quaisquer
    - ➔ *ArrayList* é mais utilizado pela comunidade devido a sua estrutura mais flexível e leve
- Principais métodos:

- Construtores:
  - ***ArrayList()*** ou ***Vector()***
  - ***ArrayList(****Collection c****)*** ou ***Vector(****Collection c****)***
  - Exemplos:

```
ArrayList lista = new ArrayList ();
Vector lista2 = new Vector();
```

  - ➔ Nos construtores é possível especificar o tipo de objeto da lista, se conveniente, em caso de necessidade de lista de único tipo de objeto:
    - Neste caso, utilize **<>** para especificar o tipo. Exemplo:

```
ArrayList<String> lista = new ArrayList<String>();
```

- Para seus objetos
  - ***add(****Object obj****)***
  - ***remove(****Object obj****)*** ou ***remove(****int index****)***
  - ***set(****int index, Object obj****)***
  - ***get(****int index****)***
  - ***size()***
  - ***clear()***
  - ***isEmpty()***

Coleções

- Exemplo de utilização
  - Objetos da mesma classe

```
ArrayList<String> listaStr = new ArrayList<String>(); // instancia/inicializa coleção
listaStr.add("primeiro"); // adiciona objetos (no caso, objetos String)
listaStr.add("segundo");
listaStr.add("terceiro");
listaStr.remove(0); // remove o elemento "primeiro"
listaStr.add(5); // o que acontece?
```

  - Objetos de classes diferentes

```
ArrayList lista = new ArrayList(); // instancia/inicializa coleção
lista.add("primeiro"); // adiciona objetos
lista.add("segundo");
lista.add("terceiro");
lista.remove(0); // remove o elemento "primeiro"
lista.add(5); // o que acontece?
```

Coleções

- Percorrendo elementos de uma lista:
  - Usando **get**

```
for(int i=0; i<lista.size(); i++) {
  String elem = lista.get(i);
  System.out.println("elemento " + i + " = " + elem);
}
```

  - Usando **iterator**

```
for(Iterator it = lista.iterator(); it.hasNext();) {
   String elem = (String) it.next();
   System.out.println(elem);
}
```

  - Usando **for** avançado (para arrays e coleções)

```
for(String elem : listaStr) {
    System.out.println(elem);
}
```

Exercícios para Coleções

1) Crie uma classe **Agenda** que deve possuir um ArrayList com nome e telefone dos contatos. Crie métodos que:

a) Adicione um determinado contato;

b) Remova um contato pelo seu nome;

c) Remova um contato pelo seu telefone;

d) Altere o nome do contato pelo telefone;

e) Liste todos os contatos;

f) Retorne a quantidade total de contatos cadastrada

Obs.: **Contato** é uma classe apenas com atributos nome e telefone, além de construtor que receba esses dois parâmetros.

Exercícios para Coleções

2) Crie uma classe **Venda** que possui como atributos o nome do cliente e uma coleção de itens de venda. Cada item de venda contém um objeto da classe Produto e a quantidade. A classe **Produto** contém o nome do produto, seu código e o preço unitário dele. A classe **Venda** deve possuir métodos que adicione um item de venda, remova um item de venda (pelo produto ou ordem do item na venda) e que liste todos os itens comprados com os respectivos preços, valores unitários e valor total por item. Deve também possuir um método que totalize o valor da venda.

A classe **Venda** possui apenas um construtor, que recebe o nome do cliente como parâmetro.