OCJA

CENTRO DE TREINAMENTO PROFISSIONAL

TREINAMENTO@KEES.COM.BR - WWW.KEES.COM.BR

PREPARATÓRIO PARA A CERTIFICAÇÃO SCJP

Oracle Certified Java Associate

Éver Santoro

OCA, OCP, SCJA, SCJP, SCWCD, OCJD I, PDA, PDT, MCTS

ORACLE

Linguagem Java

Éver Santoro – santoro@kees.com.br

Palavras Reservadas

abstract	char	double	for
int	private	strictfp	throws
boolean	class	else	goto
interface	protected	super	transient
break	const	extends	if
long	public	switch	try
byte	continue	final	implements
native	return	synchronized	void
case	default	finally	import
new	short	this	volatile
catch	do	float	instanceof
package	static	throw	while
true	false	null	assert

Convenção de Código

- Nome da classe começa com maiúscula
  - Pessoa, Produto, Endereço
- Métodos, atributos e variáveis começam com minúsculo
  - gravar, consultaSaldo, inserir
- Constantes – todos os caracteres em maiúsculo
  - GENERO_PRODUTO, SEGUNDA

- FUGIR DA REGRA ESTÁ ERRADO!!!

Variáveis

- Tipo Primitivo
  - Numérico
  - Caracter
  - Booleanos
- Tipo Reference
  - Objetos
- Arrays

Tipos Primitivos

Data Types

Type Name	Minimum Value	Maximum Value	Default	Size	Literal
byte	-128	127	0	8-bit +/-	———
short	-32768	32767	0	16-bit +/-	———
int	-2147483648	2147483647	0	32-bit +/-	3, 077, 0xBAAC
long	-9223372036854775808	9223372036854775807	0	64-bit +/-	3L
float	-1.40239846e-45	3.40282347e+38	0.0	32-bit IEEE float	3.0F, 3.0E2F
double	-4.94065645841246533e-324	1.79769313486231570e+308	0.0	64-bit IEEE float	3.0, 3.0E2, 3.0e2D
boolean	false	true	false	N/A	true, false
char	\u0000	\uffff	\u0000	16-bit Unicode	'3'

Números Inteiros

28 = Decimal

034 = Octal

0x1c = Hexadecimal

Para forçar um valor long você deve utilizar a letra “L” ou “l” no final do número. Exemplo: 10L

Exercício

Para executar no console a classe deverá possuir o método main corretamente assinado.

Para imprimir você deve utilizar o comando

System.out.println(“Aqui vai o texto” + variavel);

Crie as seguintes variáveis e imprima seus valores:

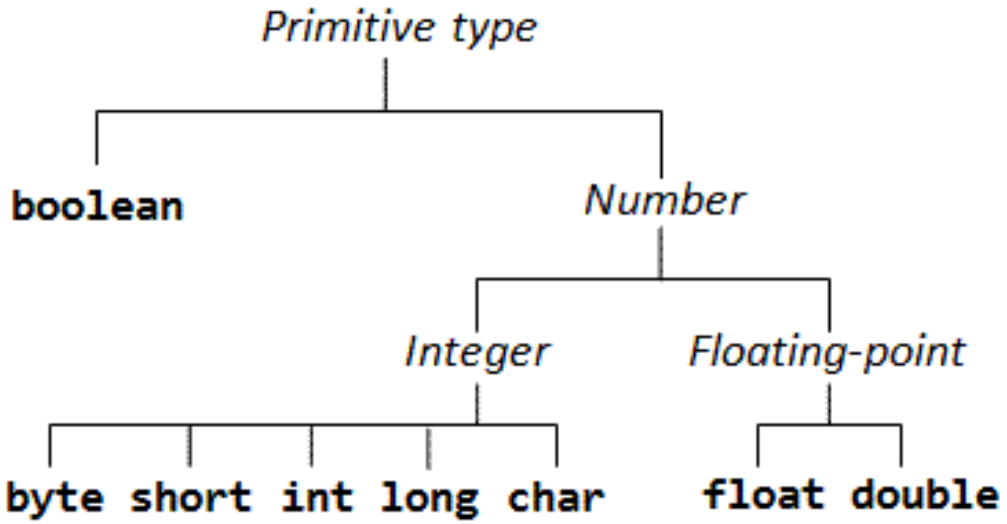
- **int i = 10;**
- **long l = 2566L;**
- **long l2 = 2236565656L;**
- **byte b = (byte) 123;**
- **byte b2 = (byte) 123568545;**
- **short s = (short) 12565;**
- **short s2 = (short) 1231321;**

Números com ponto flutuante

- float – 4 bytes
- double – 8 bytes

- Para “forçar” um determinado valor a ser um float devemos utilizar o caracter “F” ou “f” no final do número

- Exemplo: 10f ou 10F



Exercício

Para executar no console a classe deverá possuir o método main corretamente assinado.

Para imprimir você deve utilizar o comando

System.out.println(“Aqui vai o texto” + variavel);

Crie as seguintes variáveis e imprima seus valores:

- **float f = 10F;**
- **float f2 = 10.45454f;**
- **double d = 12565484546d;**
- **double d2 = 12565484546.0;**

Variável Caracter (char)

- **Os literais *char* são expressos incluindo o caracter desejado entre aspas simples**
- **Exemplo:**
  - **meuChar = ‘x’;**
- **É possível também receber valores unicode**
  - **meuCharUnicode = ‘\u4567’;**
- **Java também suporta caracteres especiais**
  - **\n = Nova linha**
  - **\t = Tabulação**
  - **\” = Aspas duplas**
  - **\\ = Barra invertida**

Exercício

Para executar no console a classe deverá possuir o método main corretamente assinado.

Para imprimir você deve utilizar o comando

System.out.println(“Aqui vai o texto” + variavel);

Crie as seguintes variáveis e imprima seus valores:

- **char c = ‘a’;**
- **char asc = 64;**
- **char espaco = ‘\u0000’;**

Variáveis Booleanas

Variáveis booleanas podem ser representadas apenas por dois tipos

- **true**
- **false**

Não pode ser utilizado 0 e 1 como em outras linguagens

Exemplo:

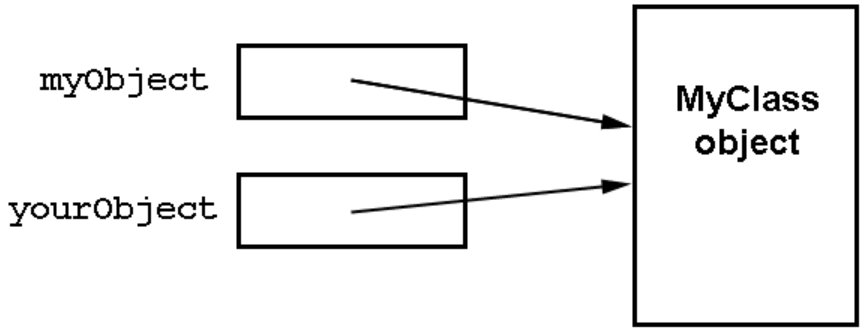
```
boolean ok = true;
```

Reference

Variáveis do tipo Reference armazenam o endereço de memória, ou seja, neste momento estamos falando de Objetos.

Podem receber os seguintes valores:

- **null**
- **Referência a um objeto de uma classe compatível com seu tipo**
- **Referência para um array**
  - **Exemplo:**
  - **String s = new String(“a”);**
  - **String s2 = “teste”;**

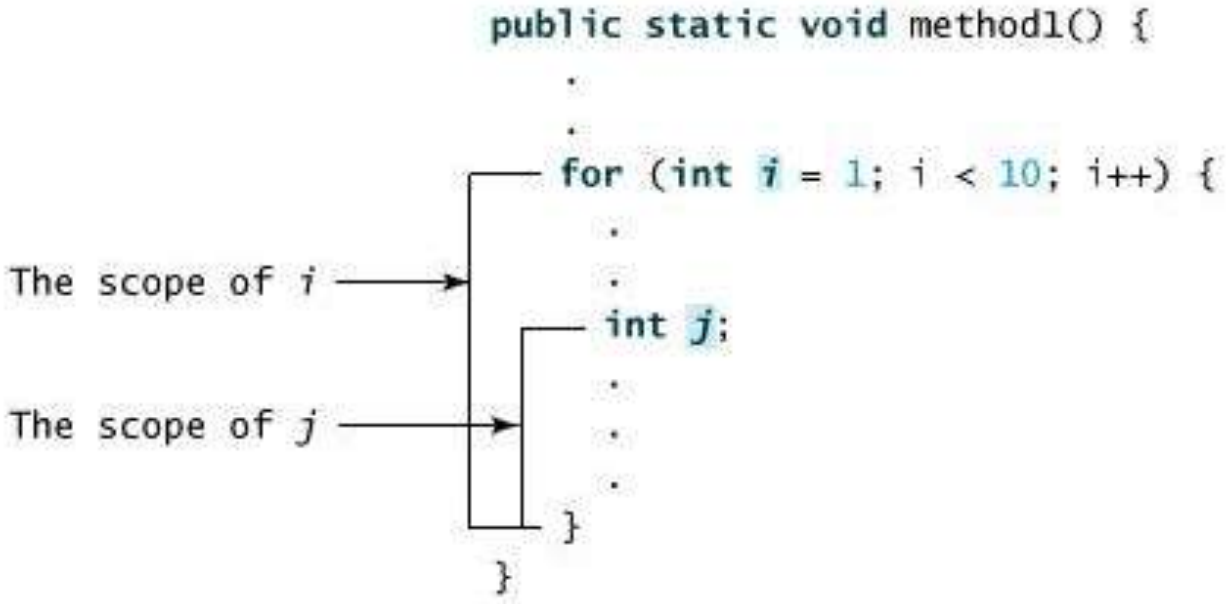


Variáveis Locais

- **Variáveis declaradas dentro de métodos ou blocos de código são definidas como variáveis locais.**
- **Ela não possui valor de inicialização padrão, logo se não indicarmos um valor para a mesma o programa dará um erro.**

Escopo

- **O Escopo define em qual parte do programa a variável estará acessível**
- **Até agora utilizamos somente variáveis dentro do escopo do método main**



Laboratório

- **Crie uma classe chamada DeclaracaoVariaveis.java**
- **Declare uma variável tipo String e armazene seu nome**
- **Declare uma variável tipo char e armazene seu sexo**
- **Declare uma variável tipo int e armazene sua idade**
- **Declare uma variável tipo double e armazene o seu salário ideal**
- **Imprima os valores no seguinte formato:**

Eu <seu nome>, com <idade> anos, do sexo <sexo>, estou registrado com o salário de R$ <salario>

System.out.println(“Eu “ + nome + “, com “ + idade);

Éver Santoro – santoro@kees.com.br

Certificação

Selecione a(s) alternativa(s) correta(s):

```
class Question{
    public static void main (String [] args)
    {
            int y = 0;  // linha 1
            int x = z = 1; // linha 2
            System.out.println(y+”,”+x+”,”+z); // linha 3
    }
}
```

A) Imprime 0,1,1
B) Erro na linha 1
C) Imprime 0,0,1
D) Erro na linha 3
E) Erro na linha 2

Operadores

Operadores Unários

- Negação: !
  - boolen ok = true; System.out.println(!ok);
- Incremento: ++
  - int i = 0; i++; System.out.println(i);
- Decremento: --
  - int i = 0; i--; System.out.println(i);
- Operadores aritméticos
  - + - * / %
- Operadores de comparação
  - < > <= >= == !=

Cast

- Cast Explícito (preciso informar)

- byte → short → int → long → float → double

  char ↗

- Cast automático (o compilador faz para você)

Exercício

Para executar no console a classe deverá possuir o método main corretamente assinado.

```
int i = 0;
System.out.println(i++);
System.out.println(i);
System.out.println(++i);
System.out.println(i);
System.out.println(i--);
System.out.println(i);
System.out.println(--i);
System.out.println(i);
```

Fundamentos

Comparação de Tipos

- instanceof

```
String nome = “”;
 if (nome instanceof String) {
    System.out.println(“Nome eh uma String”);
 }
 else
{
     System.out.println(“Nome não eh uma String”);
}
```

Condições

- & = AND → verifica todas as condições
- | = OR

- && = AND → sai no primeiro falso
- || = OR

Condições

```
int i = 0;
int j = 4;

if (i != 0 && j++ > 2) {System.out.println("Entrou no &&");}
System.out.println("&& = i: " + i + " j: " + j);

i = 0;
j = 4;

if (i != 0 & j++ > 2) {System.out.println("Entrou no &");}
System.out.println("& = i: " + i + " j: " + j);
```

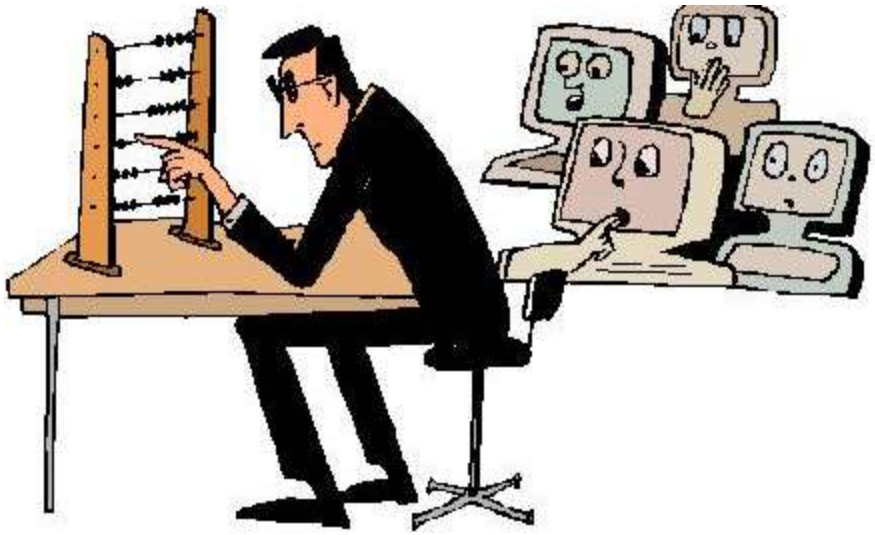
Qual a saída deste programa, por que?

Operadores de Atribuição

Operador Ternário:

```
int a = 2;
int b = 3;
int c = 4;

a = b > c ? b : c;
```



Traduzindo...

Se b for maior que c, então a = b senão a = c

Exercício: Faça uma classe que verifique se um número é par ou impar e imprima o resultado. Utilize o operador ternário.

Controle de Fluxo

Éver Santoro – santoro@kees.com.br

Controle de fluxo (if / else)

Sintaxe:

```
if ( expressão booleana ) {
                    // instruções
} else {
                    // instruções
}
```

- As chaves podem ser omitidas se houver apenas uma instrução no bloco

Éver Santoro – santoro@kees.com.br

Controle de fluxo (if / else)

A instrução else é opcional e pode ser seguida de outra estrutura if/else

```
int x = 23;
if (x == 20) {
        // instruções
}else if (x == 23) {
        //instruções
}else
        //instrução única
```


Éver Santoro – santoro@kees.com.br

Controle de fluxo (if / else)

A expressão booleana deve retornar **true** ou **false**, podendo ser:

- Variável boolea

```
boolean casado = true;
if (casado)
    System.out.println("é casado");
```

Dois ou mais operandos booleanos ou inteiros com operadores lógicos

```
int x = 5;
if (x > 0 && x < 10)
   System.out.println("x entre 0 e 10");
```

Controle de fluxo (switch)

- Maneira de expressar várias instruções if/else encadeadas

- Sintaxe:

```
switch ( variavel ){
    case valor1 : //instruções
    case valor2 : //instruções
    default : //instruções
}
```

```
int x = 2;
switch ( x ) {
        case 1 :
                System.out.println("é um");
        case 2 :
                System.out.println("é dois");
        default:
                System.out.println("é diferente de 1 e 2";
}
```

Controle de fluxo (switch)

- A variável utilizada pode ser dos tipos:
  - ✓ **char**, **byte**, **short**, **int e enum**
- A instrução case suporta:
  - ✓ Literais do tipo da variável utilizada no switch
  - ✓ Variáveis constantes do tipo da variável utilizada
- A instrução default é opcional e pode estar em qualquer parte da estrutura, não fazendo diferença para o resultado final do switch
- Não é necessário usar chaves para delimitar o bloco de código
  - ✓ A instrução **break** é utilizada para definir o limite de cada case
    - *Se o break não é utilizado e um dos cases é satisfeito, todas as instruções dos demais cases são executadas em seqüência.*

Exercício

Para executar no console a classe deverá possuir o método main corretamente assinado.

Para imprimir você deve utilizar o comando

System.out.println(“Aqui vai o texto” + variavel);

Crie uma variável x do tipo int, atribua um valor a variável

Se o valor estiver entre 1 e 7 imprima o dia da semana

Caso contrário dê uma mensagem de valor inválido

1 – Domingo 2- Segunda-feira ...

Loops e Arrays

Loops e Arrays (while)

- Sintaxe:

```
while ( condição ) {
    // instruções
}
```

- A estrutura de repetição é interrompida quando a condição retorna false
  - ✓ A condição de parada fica no início da estrutura

- As chaves podem ser omitidas se houver apenas uma instrução no bloco

```
int x = 0;
while (x < 7){
        System.out.print(x + " ");
        x += 2;
}
System.out.println("fim");
```

```
Resultado: 0 2 4 6 fim
```


Éver Santoro – santoro@kees.com.br

Loops e Arrays (do/while)

- Sintaxe:

```
do{
    // instruções
} while ( condição );
```

- A estrutura é interrompida quando a condição retorna false
  - ✓ A condição de parada fica no final da estrutura, as instrução são executadas ao menos 1 vez

- As chaves podem ser omitidas se houver apenas uma instrução no bloco

```
int x = 0;
do {
        System.out.print(x + " ");
        x += 2;
} while (x < 7);
System.out.println("fim");
```

```
Resultado: 0 2 4 6 fim
```

Exercício

Para executar no console a classe deverá possuir o método main corretamente assinado.

Para imprimir você deve utilizar o comando

System.out.println(“Aqui vai o texto” + variavel);

Crie uma variável x do tipo int, atribua um valor a variável

Crie uma classe para exibir a tabuada do valor x

Sendo a saída:

X * 1 = valor

X * 2 = valor ... (até 10)

Éver Santoro – santoro@kees.com.br

Loops e Arrays (for)

- Repetição com controle de sobre o número de repetições
- Sintaxe:

```
for([inicilização]; [condição]; [incremento]){
  //instruções
}
```

- As chaves podem ser omitidas se houver apenas uma instrução no bloco
- A repetição será interrompida quando a condição retornar false
- Inicialização, condição e incremento são opcionais
  - ✓ A seguinte instrução define um loop infinito:

```
for ( ; ; ){
   // instruções
}
```

Loops e Arrays (for)

- Inicialização:
  - ✓ É possível declarar e inicializar variáveis nesta área, mas somente de um tipo
  - ✓ A separação acontece por meio de vírgulas
- Condição:
  - ✓ Pode ser utilizada qualquer expressão que retorne true ou false
- Incremento:
  - ✓ Incrementa/decrementa o valor das variáveis declaradas na área de inicialização do for
  - ✓ Pode ser utilizado operador de incremento/decremento ou operadores aritméticos

```
for (int x = 0; x < 7 ; x+=2){
        System.out.print(x + " ");
}
System.out.println("fim");
```

```
Resultado: 0 2 4 6 fim
```


Éver Santoro – santoro@kees.com.br

Funções break e continue

- **break**
  - ✓ Interrompe a estrutura de repetição na qual essa instrução se encontra

```
for (int x = 0; x < 7 ; x += 2){
    if (x == 4)
        break;
    System.out.print(x + " ");
}
System.out.println("fim");
```

Resultado: 0 2 fim

- **continue**
  - ✓ Ignora as instruções seguintes e volta ao incremento da estrutura de repetição

```
for (int x = 0; x < 7 ; x += 2){
    if (x == 4)
        continue;
    System.out.print(x + " ");
}
System.out.println("fim");
```

Resultado: 0 2 6 fim

Certificação

Selecione a opção correta:

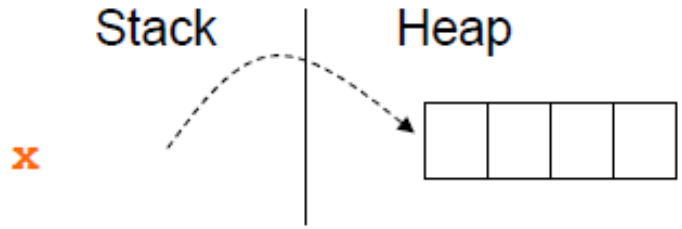
```
class Question{
    public static void main(String [] args){
      while (false); //linha 1
      if (false); // linha 2
      do() while (false); // linha 3
      for(;false;); // linha 4
    }
}
```

1. **Erro na linha 1, 3 e 4**
2. **Erro na linha 1, 2 e 4**
3. **Erro na linha 1, 2**
4. **Erro na linha 1 e 3**
5. **Código compila e roda normalmente**

Arrays

Arrays

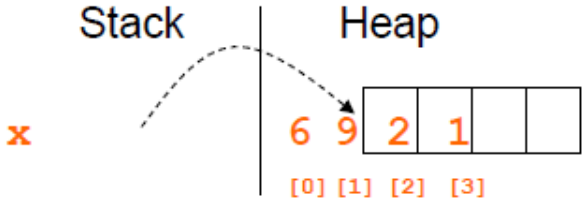
- Representa coleções de dados
- Indicado pelo uso do [ ] na declaração da variável
  - ✓ Ex.: `int x[];` ou `int[] x;`
- Tem tamanho fixo e não é possível redimensionar
- É inicializado usando a palavra reservada new
  - ✓ Ex.: `x = new int[4];`



Arrays

- O acesso a cada posição é por meio de índices que começam em 0 (zero)
  - ✓ Ex.:

```
x[0] = 6;
x[1] = 9;
x[2] = 2;
x[3] = 1;
```



- É possível declarar arrays de qualquer tipo, incluindo String ou qualquer outro tipo reference:
  - ✓ Ex.:

```
String palavras[] = new String[3];
palavras[0] = "palavra1";
palavras[1] = "palavra2";
```

Arrays

- O tipo do array define o tipo de elemento que pode ser armazenado em cada posição
  - ✓ Ex.: array de inteiros só aceita inteiros, array de Strings só aceita Strings, etc

- A propriedade `length` define o tamanho do array e pode ser usada para fazer *loops*.

```
String palavras[] = new String[3];

palavras[0] = "palavra1";
palavras[1] = "palavra2";

for(int pos = 0; pos < palavras.length; pos++) {
        System.out.print( palavras[pos] + " " );
}
```

```
Resultado: palavra1 palavra2 null
```


Éver Santoro – santoro@kees.com.br

Arrays

```
int x[] = new int[] {6, 9, 2, 1};
```

- *Declara e inicializa o array x com 4 posições já preenchidas com valores*
- *Pode ser usado depois da declaração do array, por ex. para definir um novo array para a mesma variável.*

- Arrays podem ter 2 dimensões: bidimensionais

  ✓ Declaração: dois pares de []

  ```
  int x[][]; ou
  int[][] x; ou
  int[] x[];
  ```

Arrays

- As posições do array são inicializadas com valores default:
  - ✓ Primitivos numéricos: 0
  - ✓ Booleanos: false
  - ✓ Caracteres: \u0000
  - ✓ Tipos reference = null

- Há duas formas de inicializar o array já atribuindo valores a cada uma das suas posições:

```
int x[] = { 6, 9, 2, 1};
```

  - *Declara e inicializa o array x com 4 posições já preenchidas com valores*
  - *Só pode ser usada na linha da declaração do array, caso contrário ocorre erro de compilação*

Exercício

```
/*
 * Crie um codigo que inicialize (utilizando a inicializaçao com chaves {val1, val2,..})
 * um array de String de 5 posicoes com os seguintes valores:
 *
 * 1: Cliente 1 email: cliente1@yahoo.com.br
 * 2: Cliente 2 email: cliente2@yahoo.com.br
 * 3: Cliente 3 email: cliente3@yahoo.com.br
 * 4: Cliente 4 email: cliente4@yahoo.com.br
 * 5: Cliente 5 email: cliente5@yahoo.com.br
 *
 */
class Arrays08a {

   public static void main(String args[]) {
   }
}
```

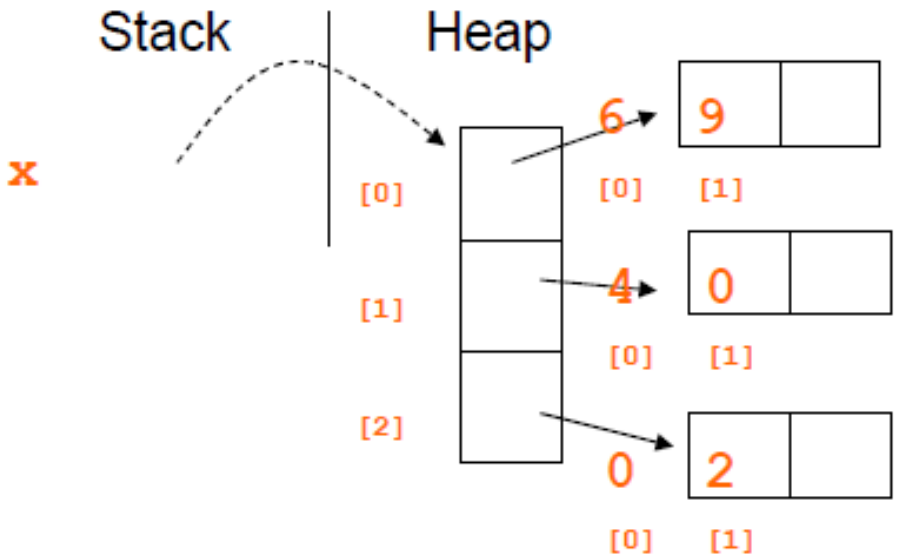
Arrays

- ✓ Inicialização:

```
x = new int[3][2];
```

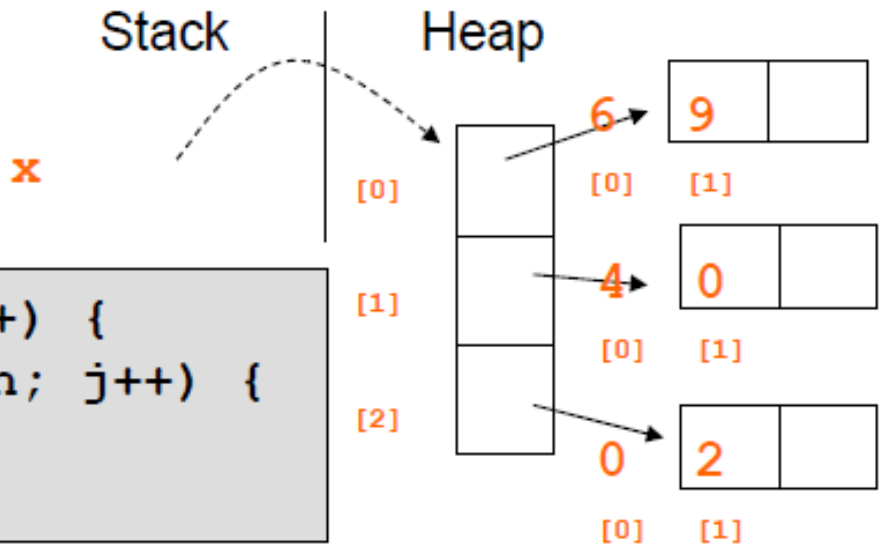
- ✓ Atribuição de valores:

```
x[0][0] = 6;
x[0][1] = 9;
x[1][0] = 4;
x[2][1] = 2;
```



Arrays

- ✓ Iteração sobre o array:




```
for(int i = 0; i < x.length; i++) {
  for(int j = 0; j < x[i].length; j++) {
    System.out.print( x[i][j]);
}
```

```
Resultado: 694002
```

Inicialização direta:

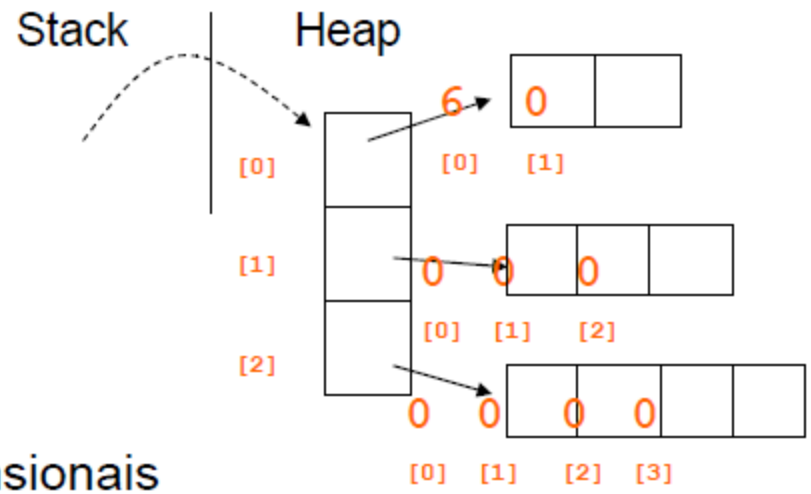
int x[][] = {{6,9},{4,0},{0,2}};

int x[][] = new int[][] {{6,9},{4,0},{0,2}};

Arrays

- ✓ É possível inicializar apenas a primeira dimensão e, posteriormente, inicializar a segunda dimensão (com tamanhos diferentes):

```
int x[][] = new int[3][];
x[0] = new int[2];
x[1] = new int[3];
X[2] = new int[4];
X[0][0] = 6;
```



- ✓ Arrays podem ser multidimensionais
  - *As regras para arrays bidimensionais são aplicáveis usando-se a quantidade de `[]` necessária para a quantidade de dimensões*

```
int x[][][] = new int[3][2][2];
```

Arrays

```
public static void main (String args[]) ...
```

Array de Strings

- Recebe os parâmetros passados para a classe no momento da sua execução

  > java Teste “São Paulo” valor1 valor2

- Na classe:

```
class Teste {
    public static void main (String[] params) {
       for(int i = 0; i < params.length; i++ )
           System.out.print("*" + params[i] + "*");
    }
}
```

```
Resultado: *São Paulo*valor1*valor2
```


Éver Santoro – santoro@kees.com.br

Exercício

```
/*
 * Utilizando dois lacos for, imprima todos os elementos do seguinte array:
 *
 * int array1[][] = {{1,6,7},{2,5,1,0},{2,4,1,2,1}};
 */
class Arrays08b {

    public static void main(String args[]) {
    }
}
```


Éver Santoro – santoro@kees.com.br