
ORACLE
Academy

Java Fundamentals

6-1
Matrizes

ORACLE
Academy

Objetivos

- Esta aula abrange os seguintes tópicos:
  - Escrever um array unidimensional em um programa Java usando tipos de dados primitivos
  - Escrever um array unidimensional em um programa Java usando tipos de referência (Objeto)
  - Escrever um array bidimensional em um programa Java usando tipos de dados primitivos
  - Escrever um array bidimensional em um programa Java usando tipos de referência (Objeto)
  - Declarar um array, inicializar um array e atravessar o array

ORACLE Academy

Visão geral

- Esta aula abrange os seguintes tópicos:
  - Descrever a inicialização do array
  - Diferenciar o método length() da String e o valor do comprimento do array
  - Reescrever um programa Java para armazenar números inteiros em um array, realizar um cálculo matemático e exibir o resultado
  - Usar sintaxe de declaração do array alternativo

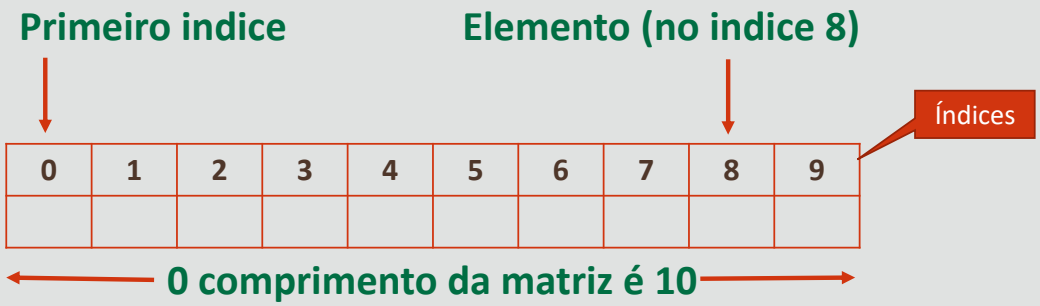
O Que é um Array?

- Um array é um conjunto de valores do mesmo tipo de dados armazenado em um objeto do contêiner
- Pode ser qualquer número de valores
- O tamanho do array é definido quando o array é declarado
- O tamanho é fixado depois que o array é declarado

ORACLE Academy

Elementos do Array

- Cada valor é chamado de um elemento
- Cada elemento é acessado por um índice
- O índice deve ser um inteiro
- O índice sempre começa em 0



Tipos de Dados do Array

- Os arrays podem conter qualquer tipo de dados, como:
  - Primitivo
  - Objetos predefinidos, como Strings da API Java
  - Objetos Definidos pelo Programador, como instâncias de uma classe que você criar

Declarando um Array

- Há três partes para declarar um array:
  - Tipo de dado
  - Nome da variável
  - Tamanho do array

Declarando Exemplos de um Array

- Uma declaração de array pode ser feita em uma ou duas linhas
- Os dois exemplos a seguir são declarações de array equivalentes

```
data_type[] variable_name;
variable_name = new data_type[size];
//declare an array using two lines of code
```

```
data_type[] variable_name = new data_type[size];
//declare an array using one line of code
```

Declarando Exemplo de um Array

- Um amigo traz um buquê de seis flores diferentes
- Você quer escrever um programa para armazenar cada tipo de flor no buquê
- Os segmentos do código abaixo são formas idênticas de declarar um array de Strings chamado myBouquet
- O tamanho é definido como seis elementos, de forma que possa armazenar cada tipo de flor em seu buquê

```
String[] myBouquet;
myBouquet = new String[6];
```

```
String[] myBouquet = new String[6];
```

Exemplo do Array myBouquet Explicado

```
String[] myBouquet;
myBouquet = new String[6];
```

```
String[] myBouquet = new String[6];
```

O tamanho do array é 6 elementos, um para cada flor.

O tipo de dados armazena os tipos de flores como Strings.

O nome do array é myBouquet.

ORACLE Academy

Identificar Componentes de uma Declaração de Array

- Identifique os três componentes de uma declaração de array para cada um desses arrays de tipos de dados primitivos

```
1.   int[] myArray;
     myArray = new int[20];

2.   char[] sentence = new char[100];

3.   double[] teamPoints = new double[5];
```

	Tipo de Dado	Nome	Tamanho
1			
2			
3			

Sintaxe de Declaração de Array Alternativa

- Uma forma alternativa de declarar arrays é colocar as chaves usadas para declarar um array no lado do identificador de array
- Sintaxe introduzida:

```
int[] primeNumbers;
int[] evenNumbers;
double[][] prices;
String[] words;
Point[] coordinates;
Rectangle[][] blocks;
```

Sintaxe alternativa:

```
int primeNumbers[];
int evenNumbers[];
double prices[][];
String words[];
Point coordinates[];
Rectangle blocks[][];
```


A sintaxe alternativa raramente é usada por programadores Java.

Sintaxe de Declaração de Array Alternativa

- A declaração funcionará
  - A Sintaxe Alternativa é semelhante àquela usada na linguagem de C
  - A sintaxe usada para introduzir arrays realmente lê de forma mais clara
  - A primeira linha deve ler "Array do inteiro primeNumbers"
- Sintaxe introduzida:

```
int[] primeNumbers;
int[] evenNumbers;
double[][] prices;
String[] words;
Point[] coordinates;
Rectangle[][] blocks;
```

Sintaxe alternativa:

```
int primeNumbers[];
int evenNumbers[];
double prices[][];
String words[];
Point coordinates[];
Rectangle blocks[][];
```

Inicializando um Array

- Depois que você declarar um array, deve inicializá-lo para definir os valores dos índices especificados
- Há três componentes para inicializar um array:
  - Nome da variável
  - Índice
  - Valor

Formas de Inicializar um Array

- Um array pode ser inicializado de duas formas diferentes:
  - Declare o array, em seguida, inicialize cada elemento
  - A declaração e a inicialização ocorre na mesma etapa
- Exemplo de declarar e inicializar em duas etapas:

```
data_type[] variable_name = new data_type[size];
variable_name[index] = value;  //repeat for each index
```

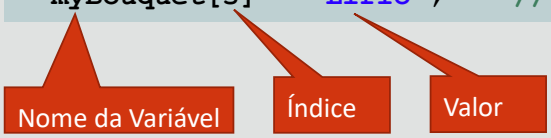
- Exemplo de declarar e inicializar em uma etapa:

```
data_type[] variable_name = {val1, val2, …};
```

Inicializando um Array - Exemplo 1

- Rechame a declaração do array myBouquet da String
- Depois que o array é declarado, o código abaixo inicializa os elementos e armazena os tipos de flores
- O índice de um array começa com 0, dessa forma, o primeiro elemento será adicionado ao índice 0

```
String[] myBouquet = new String[6];//previous declaration
 myBouquet[0] = "Rosa";      //Store "Rosa" as the first element
 myBouquet[1] = "Girassol"; //Store "Girassol" as the second
 myBouquet[2] = "Margarida";//and so on
 myBouquet[3] = "Dente-de-leão";
 myBouquet[4] = "Violeta";
 myBouquet[5] = "Lírio";     //"Lírio" is the last (sixth) element
```



ORACLE Academy

Inicializar um Array - Exemplo 1

- O array dos tipos de flores parecerá com o seguinte:

Índice:	**0**	**1**	**2**	**3**	**4**	**5**
Valor:	Rosa	Girassol	Margarida	Dente-de-leão	Violeta	Lírio

ORACLE Academy

Inicializar um Array - Exemplo 2

- O array myBouquet também pode ser declarado e inicializado usando a segunda notação, como segue:

```
String[] myBouquet = {"Rosa", "Girassol", "Margarida",
                      "Dente-de-leão", "Violeta", "Lírio"};
```

- Observe que usar este método não especifica o tamanho, mas é designado um tamanho com base no número de elementos da lista entre {}

Com frequência, cada elemento é colocado sozinho em uma linha, portanto, o código é mais óbvio.

Primeira Notação versus Segunda Notação

- Parece que a segunda notação é muito mais curta e mais fácil de codificar
- Por que você precisa da primeira notação?

Primeira Notação versus Segunda Notação

- Considere usar cinco números de um usuário e armazená-los em um array chamado myArray
- Você precisa declarar o array, em seguida, inicializar os elementos um de cada vez, conforme o usuário informa cada número

```
int[] myArray = new int[5]; //Declare the array of size 5
MyArray[0] = 7;             //The user entered 7
MyArray[1] = 24;            //The user entered 24
MyArray[2] = 352;           //The user entered 352
MyArray[3] = 2;             //The user entered 2
MyArray[4] = 37;            //The user entered 37
```

Primeira Notação versus Segunda Notação

- A segunda notação pode ser uma forma mais fácil de inicializar o array, se você já souber o conteúdo do array ao declará-lo

```
int[] myArray = {7, 24, 352, 2, 37};
```

Representação do Array

- Quando os arrays são declarados, mas não inicializados, os elementos recebem o valor padrão associado ao tipo de dado
- Por exemplo, o padrão para tipos de dados numéricos, como inteiro é 0
- O padrão para tipos de Objeto, como Strings “” (nulo)
- Quando o array é declarado, a representação na tabela é a seguinte

```
int[] myArray = new int[5];
```

Índice:	0	1	2	3	4
Valor:	0	0	0	0	0

ORACLE Academy

Representação de Array Atualizada

- Depois que você começar a inicializar os elementos, o array será atualizado
- A nova representação na tabela é a seguinte:

```
myArray[0] = 32;
myArray[3] = 27;
```

Índice:	0	1	2	3	4
Valor:	32	0	0	27	0

Tipos de Objeto dos Arrays

- Os arrays não são restritos para armazenar tipos de dados primitivos
- Eles podem armazenar qualquer tipo de objeto, incluindo tipos que você definir
- Por exemplo, se uma classe Flower existia:
  - As flores poderiam ser armazenadas no array, em vez de armazenar o tipo de flor como uma String
  - Como sabemos quais flores incluir, a segunda notação pode ser usada para inicializar myBouquet de seis Flores

```
Flower[] myBouquet = {new Flower("Rosa"), new Flower("Girassol"),
                      new Flower("Margarida"), new Flower("Dandelion"),
                      new Flower("Violeta"), new Flower("Lírio")};
```

Acessando o Tamanho do Array

- Com cada declaração de um array, você deve definir um tamanho ou comprimento do array
- O tamanho é armazenado como uma variável da instância para tal objeto e pode ser acessado usando a notação arrayName.length
- Esta técnica é útil no seguinte exemplo:
  - Estabelecer um array com um tamanho com base na entrada do usuário
  - Informar um segmento do código no qual a entrada do usuário não está mais no escopo
  - Você precisa acessar o tamanho da variável da instância para esse array
  - Em resumo, arr.length retorna o tamanho do array, arr

Com Strings, houve um método length(). Com matrizes, é apenas length sem ().

Iterar por meio de um Array

- Para iterar, ou inverter, um array, o que significa progredir por meio de cada elemento do array por número de índice.
- Iterar um array é útil quando:
  - Você quer acessar cada elemento de um array na ordem
  - Você quer inicializar os elementos de um array como todos do mesmo valor
- Use .length ao iterar, em vez do valor inteiro informado ao declarar o array
- Isso garantirá que você não recebe um índice fora do erro de limites

ORACLE Academy

Você também pode iterar em uma matriz e inicializar cada elemento para um valor diferente. Você pode solicitar ao usuário os valores ou pode lê-los em um arquivo.

Exemplo de Iterar por meio de um Array

- Para inicializar um array de inteiros chamado allTwos, de forma que cada elemento seja 2, use um loop For, conforme mostrado abaixo
- A primeira linha do código é a declaração de array
- Ela declara um array chamado allTwos com um tamanho 10
- O loop for itera pelos índices e para cada índice no array o valor nesse índice é definido como 2

```
int[] allTwos = new int[10];
for(int index = 0; index < allTwos.length; index++){
   allTwos[index] = 2;
}//fim for
```

Observe como o tamanho do array éacessado para sair dos limites do índice do array.

Quando a Iteração é Útil

- No exemplo da flor, a iteração ajuda se você quiser imprimir os tipos de flores armazenados no array myBouquet

```
String[] myBouquet = {"Rosa", "Girassol", "Margarida",
                      "Dente-de-leão", "Violeta", "Lírio"};
```

- Use um loop for para iterar por este array
- O contador inicializado no loop for pode ser usado para incrementar os índices, conforme mostrado abaixo
- O que é exibido como resultado deste código?

```
//remember that the index range is 0 to 5 for an array of size 6
  for(int index = 0; index < myBouquet.length; index++){
          System.out.println(myBouquet[index]);
```

Loop for-each

- O Java oferece um loop for-each, uma alternativa para usar o contador inicializado para iterar por um array
  - Quando usado para acessar os elementos de um array, o loop for-each trabalha do mesmo jeito que o loop For, mas é implementado de uma forma mais simples
  - Se substituirmos o código do loop For de nosso exemplo anterior pelo código a seguir, teremos o mesmo resultado

```
//remember that the index range is 0 to 5 for an array //of size 6
for (String myFlower : myBouquet)
{
    System.out.println(myFlower);
}//fim for
```


O loop for-each tem uma sintaxe mais simples do que o loop for para iterar em uma matriz, mas não tem uma variável de controle de loop. Dessa forma, se você quisesse, por exemplo, imprimir cada elemento com o número do elemento antes dele, o loop for-each não seria a melhor escolha.

Loop for-each

- O loop for - each acessa (um de cada vez) e retorna todos os elementos de um array
  - As alterações nos elementos do array não podem ser feitas usando um loop for-each
  - Se substituirmos o código do loop For de nosso exemplo anterior pelo código a seguir, teremos o mesmo resultado
  - O exemplo abaixo imprimirá o tamanho de cada string no array myBouquet

```
//remember that the index range is 0 to 5 for an array //of size 6
for (String myFlower : myBouquet)
{
    System.out.println(myFlower);
}//fim for
```

Exemplo de Loop for-each

- As duas implementações do código abaixo exibirão as informações sobre os elementos do array

```
public class Bouquet {
   public static void main(String[] args){
      String[] myBouquet = {"Rosa","Girassol","Margarida",
                            "Dente-de-leão","Violeta", "Lírio"};

      //use a for loop to iterate through the array
      for(int index = 0; index < myBouquet.length; index++){
           System.out.println(myBouquet[index]);
      }//fim for
      //use a for each to iterate through the array
      for (String myFlower : myBouquet){
        System.out.println(myFlower);
      }//fim for
   }//fim do método main
}//fim da classe Bouquet
```

O Que Sabemos sobre Arrays

- O que sabemos sobre arrays:
  - Arrays são tipos de objetos que podem armazenar qualquer tipo primitivo ou de objeto
  - No entanto, os arrays podem armazenar arrays
  - O conceito de armazenar um array de arrays é chamado array bidimensional

Arrays Bidimensionais

- Um array bidimensional, chamado de "array dos arrays" é aquele que armazena outros arrays
- O número de arrays contidos no array é definido na declaração
- O número de itens em cada array interno também é definido na declaração

Não há limite no Java para o número de dimensões que uma matriz pode ter. A limitação é a quantidade de memória que o computador precisa para armazenar os elementos da matriz (e sua integridade).

Exemplo de Array Bidimensional

- Este exemplo mostra um array de dois arrays com três elementos em cada um
- Um array bidimensional pode ser visualizada como uma tabela com linhas e colunas
- O exemplo abaixo tem duas linhas e três colunas

```
int[][] nums = { {14,51,16}, {12,73,87} };
```

Declarando um Array Bidimensional

- Componentes de arrays bidimensionais:
  - Tipo de dado
  - Nome da variável
  - Tamanho do array
- Para declarar um array bidimensional, use um dos exemplos de sintaxe abaixo

```
data_type[][] variable_name;
variable_name = new data_type[size1][size2];
//declare it using two lines of code
```

```
data_type[][] variable_name = new data_type[size1][size2];
//declare it using one line of code
```

Declarando um Array Bidimensional - Exemplo 1

- Identifique os três componentes nos seguintes exemplos de tipo de dados primitivo

```
int[][] myArray;
myArray = new int[2][3];
char[][] sentence = new char[10][10];
```

Declarando um Array Bidimensional - Exemplo 1

- Quando declarar um array bidimensional:
  - O número no primeiro conjunto de colchetes [2] é o número de arrays que o contêiner mantém (linhas)
  - O número no segundo conjunto de colchetes [3] é o número de elementos em cada um desses arrays (colunas)
- Outra forma de declarar myArray pode parece com esta:

```
int[][] myArray = { {0,0,0}, {0,0,0} };
```

Declarando um Array Bidimensional - Exemplo 2

- Seu amigo trouxe três de cada tipo de flores e cada uma das três flores tem diferentes cores
- Um array unidimensional torna entediante manter controle destes dados
- Um array bidimensional permite que você armazene seis arrays, um para cada tipo de flor e que cada array armazene as cores de cada uma das três flores
- O código a seguir declara um array que mantém seis arrays, cada um com três tamanho:

```
String[][] myBouquet = new String[6][3];
```

Inicializando um Array Bidimensional

- Os arrays bidimensionais, como os arrays unidirecionais, podem ser inicializados usando dois métodos diferentes
- Método 1:
  - i é o índice do array interno (linha) e j é o índice do elemento dentro desse array (coluna) que está sendo inicializado

```
public class TwoDTester{
  public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    int[][] nums = new int[3][2];

    for(int i = 0; i < nums.length; i++){
      for(int j = 0; j < nums[i].length; j++){
        System.out.println("Enter a value for row " + i + ", column " + j);
        nums[i][j] = in.nextInt();
      }//fim for
    }//fim for
  }//fim do método main
}//fim da classe TwoDTester
```

ORACLE Academy

Um loop dentro de outro loop é conhecido como aninhamento. Não há limite no Java para o número de níveis em que se pode aninhar um loop.

Inicializando um Array Bidimensional

- Método 2: Declara e inicializa todos os arrays e todos os elementos dentro desses arrays na mesma linha de código
- Porém, você deve conhecer os valores que deseja que o array contenha para iniciar o array no mesmo momento em que ele é declarado

```
public class TwoDTester2{
   public static void main(String[] args){
      int[][] nums = {{2,3,7},{15, 98, 2}};
   }//fim do método
}//fim da classe TwoDTester2
```

Inicializando um Array Bidimensional - Exemplo

- Método 1:

```
int[][] myArray = new int[3][2];
MyArray[0][0] = 7;
MyArray[0][1] = 24;
MyArray[1][0] = 352;
MyArray[1][1] = 2;
MyArray[2][0] = 37;
MyArray[2][1] = 65;
```

Índice do array interno

Índice do elemento dentro do array interno

- Método 2:

```
int[][] myArray = new int[][] {{7, 24}, {352, 2}, {37, 65}};
```

Usando a Segunda Notação para Inicializar o Array

- Como já sabemos o conteúdo dos arrays de nosso buquê, use a segunda notação para inicializar o array
  - Lembre que a ordem é “Rosa”, “Girassol”, “Margarida”, “Dente-de-leão”, “Violeta” e “Lírio”
  - Estas flores serão representadas pelos índices 0, 1, 2, 3, 4 e 5

```
String[][] myBouquet = {{"Red", "Peach", "Yellow"},
                        {"Yellow", "White", "Blue"},
                        {"Green", "Blue", "Purple"},
                        {"White", "White", "White"},
                        {"Purple", "Pink", "Violet"},
                        {"Pink", "Orange", "White"}};
```

Representação de Array Bidimensional

- O array bidimensional é representado como segue

Índice	0	1	2
0 (Rosa)	Vermelha	Pêssego	Amarelo
1 (Girassol)	Amarelo	Branco	Azul
2 (Margarida)	Verde	Azul	Roxo
3 (Dente-de-leão)	Branco	Branco	Branco
4 (Violeta)	Roxo	Rosa	Violeta
5 (Lírio)	Rosa	Laranja	Branco

Visualizando um Array Bidimensional

- Outra forma de visualizar um array bidirecional é alinhar os valores após a inicialização Esta técnica ajuda a controlar o rastreamento de como os dados são organizados
- Nos exemplos de código abaixo , observe como a segunda forma é mais fácil de visualizar

```
String[][] myBouquet = {{"Red", "Peach", "Yellow"}, {"Yellow",
"White", "Blue"}, {"Green", "Blue", "Purple"}, {"White", "White",
"White"}, {"Purple", "Pink", "Violet"}, {"Pink", "Orange",
"White"}};
```

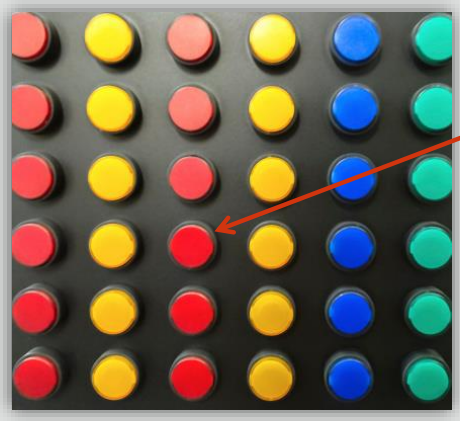
```
String[][] myBouquet = {{"Red", "Peach", "Yellow"},
                        {"Yellow", "White", "Blue"},
                        {"Green", "Blue", "Purple"},
                        {"White", "White", "White"},
                        {"Purple", "Pink", "Violet"},
                        {"Pink", "Orange", "White"}};
```

ORACLE Academy

Esse é um ótimo exemplo do por que cada linha deve estar em uma linha. Como já foi escrito, não é fácil ver imediatamente o que está acontecendo.

Acessando Dados em um Array Bidimensional

- Para acessar os dados em um array bidimensional, você deve conhecer:
  - O índice do array a ser acessado
  - O índice do conteúdo desse array para acessar



Acessando Dados em um Array Bidimensional - Exemplo

- Por exemplo, para acessar a cor da Rosa em myBouquet, use o índice do array Rose (0) e o índice da primeira cor (0)

```
//Previously initialized array
String[][] myBouquet = String[][] {
                    {"Red", "Peach", "Yellow"},
                    {"Yellow", "White", "Blue"},
                    {"Green", "Blue", "Purple"},
                    {"White", "White", "White"},
                    {"Purple", "Pink", "Violet"},
                    {"Pink", "Orange", "White"}
                    };
String rose1 = myBouquet[0][0]; //accesses first element
                                // of first array
```

Acessando Dados em um Array Bidimensional - Exemplo

- Revise a inicialização do array anterior

```
//Previously initialized array
String[][] myBouquet = String[][] {
                {"Red", "Peach", "Yellow"},
                {"Yellow", "White", "Blue"},
                {"Green", "Blue", "Purple"},
                {"White", "White", "White"},
                {"Purple", "Pink", "Violet"},
                {"Pink", "Orange", "White"}};
```

- O que será exibido na tela da console após o seguinte código ser executado?

```
System.out.println(myBouquet[0][1] + " Rose.";
System.out.println(myBouquet[5][2] + " Lilly.";
```

Acessando Dados em um Array Bidimensional - Exemplo da Solução

- O que será exibido na tela da console após o seguinte código ser executado?

```
System.out.println(myBouquet[0][1] + " Rose.";
System.out.println(myBouquet[5][2] + " Lilly.";
```

- Solução:
  - Rosa Pêssego
  - Lírio Branco

Acesando o Tamanho dos Arrays Bidimensionais

- O tamanho é uma variável da instância definida pelo tamanho de cada array declarado
- Os arrays bidimensionais têm dois tamanhos diferentes:
  - Tamanho do array externo
  - Tamanho do array interno

O tamanho da matriz externa corresponde à quantidade de linhas. O tamanho das matrizes internas corresponde à quantidade de colunas.

Tamanho dos Arrays Externos e Internos

- O tamanho do array externo, que é o tamanho que descreve o número de arrays contidos (linhas) é acessado como um array comum

```
int numArrays = arrayName.length;
```

- O tamanho dos arrays internos, que é o tamanho que descreve o número de elementos que cada array contém (colunas), é acessado usando a seguinte notação
- Os colchetes [ ] informam ao programa que ele está acessando o tamanho dos arrays internos e a linha diz qual é o array

```
int numElementsInEach = arrayName[row].length;
```

Tamanho dos Arrays Externos e Internos - Exemplo

- Revise este código

```
String[] one = new String[7];
int[][] two = new int[5][3];
double[][] three = new double[3][2];
People[] four = new People[5];
```

- Qual dos itens a seguir não é verdadeiro?
- Você pode identificar que um tem uma sintaxe imprópria para acessar o tamanho?

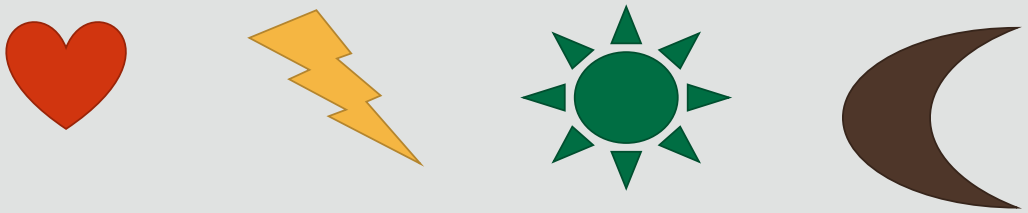
```
one.length == 7;
two.length == 3;
three.length == 3;
two[0].length == 3;
three[0].length == 2;
four[0].length == 5;
```

Tamanho dos Arrays Externos e Internos - Exemplo de Solução

- Solução:
  - `two.length == 3;` **é FALSO**
  - `four[0].length == 5;` **não é uma sintaxe válida**

Tipos de Objeto de Arrays Bidimensionais

- Como os arrays unidirecionais, os arrays bidimensionais podem armazenar qualquer tipo de objeto



Tipos de Objeto de Arrays Bidimensionais

- Como podemos organizar os alunos em uma sala em três grupos, com cinco alunos por grupo como um array?
- Resposta: Declare um array bidimensional que mantém três array, um para cada grupo.
- Cada array armazena cinco alunos.
- Uma classe chamada Student foi definida, que tem um construtor que recebe o nome do aluno
- O código a seguir declara o array que armazena o grupo de alunos

```
Student[][] groups = new Student[3][5];
```

Colocando Alunos no Array Bidirecional

- Dado um array de String que contém nomes de 15 alunos dos grupos, como você executar as seguintes tarefas?
  - Iterar pelo array existente
  - Criar um novo aluno para cada nome fornecido
  - Colocar cada aluno em um dos três grupos

Colocando Alunos no Array Bidirecional

- Você já sabe como:
  - Iterar pelo array usando um loop For
  - Criar um novo aluno para cada nome dentro do loop
  - Colocar os alunos no array bidirecional usando um aninhado para o loop

Aninhado para Loops

- Um aninhado para o loop:
  - É um loop For dentro de um loop For
  - Pode ser usado para iterar pelos arrays bidirecionais usando o contador de loop For externo como o índice dos arrays (linhas) e o contador de loop For interno como o índice dos elementos de cada array (colunas)
- Considere a seguinte declaração de array bidimensional:

```
Student[][] groups = new Student[3][5];
```

Aninhado para Loops

- Para iterar pelo array e inicializar cada aluno como "Temp", use aninhado para loops, conforme mostrado abaixo

```
//i keeps track of the rows
for(int i = 0; i < groups.length; i++){
   //j keeps track of the columns
   for(int j = 0; j < groups[i].length; j++){
      groups[i][j] = new Student("Temp");
   }//fim for
}//fim for
```

Concluir Tarefas Usando Aninhado para Loops

- Como concluímos todas as três tarefas?
  - Iterar pelo array existente
  - Criar um novo aluno para cada nome fornecido
  - Colocar cada aluno em um dos três grupos

Concluir Tarefas Usando Aninhado para Loops

```
String[] studentNames;
//Assume studentNames is initialized with 15 names

Student[][] groups = new Student[3][5];//Declare your array
   int x = 0;
   for(int i = 0; i < groups.length; i++){
      for(int j = 0; j < groups[i].length; j++){
         String name = studentNames[x];
         groups[i][j] = new Student(name);
         x++;
      }//fim inner for loop
  }//fim outer for loop
```


Para os blocos aninhados, é uma boa prática inserir um comentário depois da chave de fechamento dos loops aninhados (// Final interno do loop). Assim, será possível identificar facilmente onde cada bloco de código termina.

Argumentos da Linha de Comandos

- Um array sempre fez parte do método principal
- O args do array da String sempre é um parâmetro para o método principal

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


Os argumentos de linha de comando eram usados bem mais antes de as GUIs se tornarem comuns.

Argumentos da Linha de Comandos: Adicionando Argumentos Extra

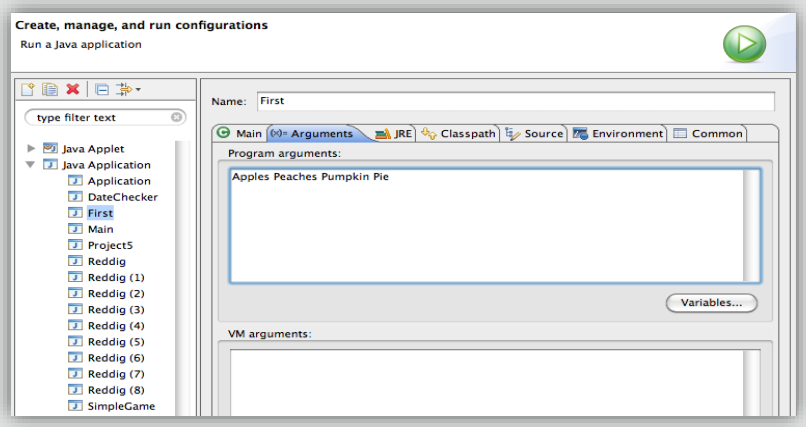
- Se você executar um programa Java de um ambiente da linha de comandos, você pode digitar argumentos extras como segue:

```
java Test apples peaches pumpkin pie
```

  - A palavra java informa ao ambiente da linha de comandos para usar a JVM para executar o Teste do programa
  - Os argumentos do array foram preenchidos com as Strings: maçãs, pêssegos, abóbora e torta
  - O uso de um IDE Java evita o ambiente de linha de comando. No entanto, é possível usar argumentos de linha de comando em um IDE Java

Usando Argumentos da Linha de Comandos no Eclipse

- Acesse a guia Arguments de seu programa indo para o menu Run e escolhendo Open Run Dialog
- Clique na guia Arguments e digite as Strings: Apples Peaches Pumpkin Pie



ORACLE Academy

JF 6-1
Matrizes

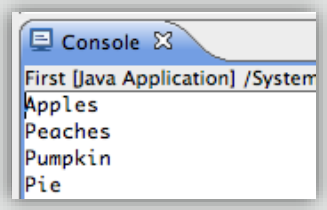
Coloque "aspas duplas" ao redor de argumentos com várias palavras. Por exemplo, verde "azul claro" vermelho. São três argumentos. Sem as aspas, teriam sido quatro argumentos.

Se você estiver usando um IDE que não seja o Eclipse, seu instrutor o orientará sobre como fazer isso.

Usando Argumentos da Linha de Comandos no Eclipse

- Clique em Run o programa
- Todas as Strings que foram digitadas na guia Arguments serão impressas na console

```
public class TestArgs {
   public static void main(String[] args){
      for(int i=0;i<args.length;i++){
         System.out.println(args[i]);
      }//fim for
   }//fim do método
}//fim da classe TestArgs
```



ORACLE Academy

Terminologia

- Os principais termos usados nesta aula foram:
  - Algoritmo
  - Array
  - Array de arrays
  - Argumento da linha de comandos
  - Índice
  - Iterar
  - Loop positivo aninhado
  - Array unidimensional
  - Atravessar
  - Array bidimensional

Resumo

- Nesta aula, você deverá ter aprendido a:
  - Escrever um array unidimensional em um programa Java usando tipos de dados primitivos
  - Escrever um array unidimensional em um programa Java usando tipos de referência (Objeto)
  - Escrever um array bidimensional em um programa Java usando tipos de dados primitivos
  - Escrever um array bidimensional em um programa Java usando tipos de referência (Objeto)
  - Declarar um array, inicializar um array e atravessar o array

Resumo

- Nesta aula, você deverá ter aprendido a:
  - Descrever a inicialização do array
  - Diferenciar o método String length() e o valor do comprimento do array
  - Reescrever um programa Java para armazenar números inteiros em um array, realizar um cálculo matemático e exibir o resultado
  - Usar sintaxe de declaração do array alternativo

ORACLE Academy


ORACLE
Academy