Aula 4 – POO 1
Apresentação Java - Parte 2

Profa. Elaine Faria

UFU - 2021

Arrays

- É um grupo de variáveis que contém valores que são todos do mesmo tipo
- Os *arrays* são objetos → tipos por referência
- Os elementos de um *array* podem ser tipos primitivos ou por referência
- O número de posição do elemento é chamado índice

Arrays

- Acessando um array
  - Usar o nome do *array* seguido pelo índice do elemento entre colchetes ([])
- O primeiro elemento do *array* tem índice zero
- Um *array* possui o membro length que fornece o comprimento do *array*

Arrays

- Para criar *arrays* usa-se a palavra-chave new, especificando o tipo dos elementos do *array* e o número de elementos
  - `Ex: int c[] = new int [12];`
- Um programa pode criar vários *arrays* em uma única declaração
  - Ex: `String b[] = new String[100], x[] = new String[27];`

Arrays

- Os colchetes pode aparecer
  - `double[] array1, array2;`
  - `double array1[];`
  - `double[] array1;`
- Exemplo *array*

```
int array[] = new int[10];
for (int c=0; c<array.length; c++)
   System.out.println(array[c]);
```

Arrays

- Um programa pode criar e inicializar um *array*
  - `Ex: int n[] = {10, 20, 30, 40}`
- Neste caso não é necessário utilizar a palavra-chave *new*

Estrutura *For* aprimorada

- Itera os elementos de um *array* sem utilizar um contador
  - for (parâmetro : array)
    Instrução

```
int array[] = {88,94,100,24,75,78}
int total = 0;
for (int nro: array)
   total += nro;
```

Passando *arrays* para métodos

- Para passar um *array* para um método especifique o nome do *array* sem nenhum colchete
- Quando um argumento para um método for um *array* ou elemento de um *array* de um tipo por referência, o método recebe uma cópia da referência
- Quando um argumento para um método for um elemento do *array* de um tipo primitivo, o método recebe uma cópia do valor do elemento

Passagem por valor e referência

- O Java não permite ao programador escolher entre passar por valor ou passar por referência
- Todos os argumentos são passados por valor
  - Cópias de valores primitivos
  - Cópias de referência para objetos

Arrays Multidimensionais

- Os *arrays* bidimensionais são usados para representar tabelas (linhas e colunas)
  - Para identificar um elemento deve-se informar 2 índices
- O Java permite especificar *arrays* unidimensionais cujos elementos são também *arrays* unidimensionais alcançando assim um *array* bidimensional

Arrays Multidimensionais

- Cada elemento do *array* é acessado por
  - a[linha][coluna]
- Exemplos

```
int b[][] = {{1,2}, {3,4}};

Int b[][] = {{1,2}, {3,4,5}};
```

Arrays Multidimensionais

- Exemplos

```
int b[][];
b = new int [3][4];


int b [][];
b = new int [2][];
b[0] =new int[5];
b[1] = new int[3];
```

Lista de argumentos e comprimento variável

- Programadores podem criar métodos que recebem um número não especificado de argumentos
  - Para isso usa-se reticências na lista de parâmetros (somente uma vez e no fim da lista de parâmetros)

```
public double average(double ... nros){
      double total = 0.0;
      for (double d: nros)
            total+=d;
      return total/nros.length;
```

Leitura de Dados via Teclado

- Usar a classe Scanner

```
imports java.util.Scanner;

Scanner entrada = new Scanner(System.in);
String nomecurso = entrada.nextLine();
System.out.println(nomecurso);
```

Strings

- São sequência de caracteres utilizada para representação e manipulação de texto
- São objetos ou instâncias da classe **java.lang.String**
- Devem ser declarados e instanciados.
  - `String ola = new String("Alô mundo Java");`
  - `String ola = "Alô Mundo Java !";`

Strings

- A classe String em Java possui mais de 50 métodos
  - **int length**(): Retorna o tamanho da string, ou seja, a quantidade de caracteres da string;
  - **char charAt**(**int** i): Retorna o i-ésimo caractere da string.
    - Assim como nos vetores a posição do primeiro caractere de uma string é igual a 0 (zero).
  - **concat**(**String** s): Retorna uma string com os caracteres deste objeto concatenados (no final) com os caracteres do argumento "s".

Strings

- **contains**(**String** s): Retorna verdadeiro se a sequência de caracteres do argumento "s" existe no objeto e falso caso contrário
- **equal**(**String** s): Retorna true se as strings forem "exatamente" iguais.
- **indexOf**(**int** ch): Retorna o índice dentro da sequência de caracteres da primeira ocorrência do caractere especificado (ch).
  - O valor -1 como retorno indica que não existe uma ocorrência.

Strings

- **toLowerCase**(): Retorna a string com os caracteres convertidos em "minúsculos".
- **toUpperCase**(): Retorna a string com os caracteres convertidos em "maiúsculas".
- **trim**(): Retorna a string com os espaços em branco do início e do final da cadeia removidos.
- **replace**(**char** oldChar,**char** newChar): Retorna a string resultante da troca de "todas" as ocorrências do caractere "oldChar" pelo caractere "newChar".

Strings

- **substring(int** ini,**int** fim): Retorna a "substring" da string definida a partir da posição "ini" até a posição "fim-1".
- **lastIndexOf**(**int** ch): Retorna o índice dentro da sequência de caracteres da última ocorrência do caractere especificado (ch).
  - O valor -1 como retorno indica que não existe uma ocorrência.

Formatação de dados

- Para formatar saídas do console, pode-se usar:
  - printf()
  - format

Formatação de dados

- **Método format() da classe String**

```
public class FormatExample{
public static void main(String args[]){
     String name="sonoo";
     String sf1=String.format("Meu nome
          é %s",name);
     String sf2=String.format("valor é
          %f",32.33434);
```

Formatação de dados

- Método printf do System.out

```
System.out.printf("%s\n%s","Olá
     pessoal", “Vamos aprender Java!");
System.out.printf(“\n%d\n%d",15,20);
```

Formatação de dados

Especificador	Formato
%s	String de caracteres
%d	Número inteiro decimal
%u	Número inteiro decimal sem sinal
%o	Número inteiro octal sem sinal
%x, %X	Número inteiro hexadecimal sem sinal, minúsculo ou maiúsculo
%f	Float
%2f	Double
%e, %E	Número real, em notação científica, minúsculo ou maiúsculo
%b	Boolean
%c	Caractere (char)

Formatação de dados

Caractere	Representa
\t	Tabulação
\b	Backspace
\n	Nova Linha
\r	Retorno de carro
\'	Aspa simples
\"	Aspa dupla
\\	Barra invertida
Específico para o printf:	
Caractere	**Representa**
%%	Símbolo de porcentagem

Exercícios

- Crie uma classe para testar a criação de *arrays* bidimensionais.
- Execute algumas operações neste *array* como soma e impressão do menor elemento do *array*.

Exercícios

- Crie uma classe chamada Retangulo.

  A Classe deve conter

  – Um atributo que representa o lado de um retângulo. Use um array de 2 dimensões para isso
  – Métodos para obter e alterar o valor do lado.
  – Um método que calcula a área do retângulo
  – Um método que calcula o perímetro do retângulo

Exercícios

- Crie também uma classe TestaRetangulo que faz a chamada da classe Retangulo testando todos os seus métodos