

Programação Orientada a Objetos (PO24CP)

Aula #05

- Tipos de Dados e Operadores
- Processando argumentos da linha de comando
- Vetor de objetos

Prof^a Luciene de Oliveira Marin
lucienemarin@utfpr.edu.br

Tipos de Dados e Operadores

- Um array é uma coleção de variáveis do mesmo tipo referenciadas por um nome comum.
- Cada uma das variáveis é especificada por um índice.
- Forma de declarar um array:

```
type[ ] arrayname = new type[size];
```

- Exemplo:

```
int[] categorias = new int[30];
```

No exemplo, **categorias** é o nome da coleção de 30 variáveis inteiras

Acessando variáveis array

- Você deve acessar um índice para acessar uma variável.
- Se o array tem tamanho 30, os índices são de 0 a 29.
- Exemplo:

```
int[] categorias = new int[30];
categorias[0] = 100;
categorias[29] = 0;
categorias[1] = categorias[0];
categorias[2] = categorias[3*2+1];
for(int i = 0; i < 30; i++)
    categorias[i] = i+70;
```

Inicializando arrays

- Para criar e inicializar ao mesmo tempo um array:

```
type[ ] arrayname = { val1, val2, ..., valN};
```

- Exemplo:

```
int[] fourVals = {3, 1, 4, 1 }
```

- Este exemplo cria um array de tamanho 4, armazenando quatro valores 3, 1, 4, 1.

Bubble Sort

- Ordenar um array **nums** de tamanho **size**:

```
for(int a = 1; a < size; a++)  
    for(int b = size - 1; b >= a; b--){  
        if(nums[b-1] > nums[b]) { // se esta fora de  
                                // ordem troca os elementos  
            int t = nums[b-1];  
            nums[b-1] = nums[b];  
            nums[b] = t;  
        }  
    }  
}
```

Arrays de duas dimensões

- Um array de duas dimensões é um array de arrays de uma dimensão.
- Exemplos:

```
int [][] table = new int [3][4];  
table[0][1] = 3;  
for(int i = 0; i < 3; i++)  
    for(int j = 0; j < 4; j++)  
        table[i][j] = i+j;  
  
int [][] newTable = {{1,2},{3,4},{5,6}};
```

Array irregulares bi-dimensionais

```
int [][] data = new int [3][];  
data[0] = new int [1];  
data[1] = new int [2];  
data[2] = new int [4];  
  
int [][] moreData = {{1},{2,3},{4,5,6}};
```


O membro **length** de Array

- Todos os arrays tem uma variável de instância *read-only* chamada **length**
- Exemplo:

```
int [] data = new data [5];  
    for (int i = 0; i < data.length; i++)  
        data[i] = 3*i;
```

Comando de iteração no estilo **for-each**

- Forma geral:
`for(type iterVar : collection) statement-block`
- Exemplo:

```
int [] data = {3,4,5,6};  
  
// os dois laços fazem a mesma coisa  
for(int i = 0; i < data.length; i++)  
    System.out.println(data[i]);  
  
for(int v : data)  
    System.out.println(v);
```

Construindo Strings

- Strings são objetos da classe **String**.
- Exemplo:

```
// todas as 3 declaracoes criam novos objetos  
// do tipo String  
String s1 = "hello";  
String s2 = new String("hello");  
String s3 = new String(s2);
```

Algumas Operações sobre Strings

boolean equals(str)	Retorna <i>true</i> se a string invocante contem a mesma sequência de caracteres que <i>str</i> .
int length()	Retorna o número de caracteres na string.
char charAt(index)	Retorna o caracter do índice especificado por index.
int compareTo(str)	Retorna um valor negativo se a string invocante é menor do que str, um valor positivo se a string invocante é maior do que str, e zero se as strings são iguais.
int indexOf(str)	Busca na string invocante por uma substring especificada em str. Retorna o índice da primeira posição da ocorrência ou -1 em falha.

Exemplos usando operações com **Strings**

```
String s1 = "abcde";

System.out.println(s1.length()); // prints "5"
System.out.println(s1.charAt(2)); // prints "c"

if(s1.compareTo("xyz") < 0)
    System.out.println("Yes"); // prints "Yes"

System.out.println(s1.indexOf("bc")); // prints "1"
System.out.println(s1.indexOf("f")); // prints "-1"
System.out.println(s1.indexOf("ef")); // prints "-1"
```

O operador ?

abreviação if

- Forma geral:
condition ? expression1 : expression 2
- A expressão corresponde ao valor da expressão 1 se a condição é verdadeira ou ao valor da expressão 2, caso contrário.
- Exemplos:

```
int x = (3 < 4 ? 5 : 6); // atribui 5 a x
int abs = (x < 0 ? -x : x); // atribui a abs o
                             //valor absoluto de x
```

Processando argumentos da linha de comando

Processando argumentos da linha de comando

```
public class CalculadoraDeLinhaDeComando // declaração da classe
{
    public static void main(String[] argumentos)
    {
        if (argumentos.length != 3){
            System.out.println("Este programa precisa que três argumentos sejam passados "+
                "pela linha de comando.");
            System.exit(1); // saímos do programa com o código de execução número 1
        }
        // Extraímos um valor inteiro da String correspondente ao primeiro argumento
        int primeiroValor = Integer.parseInt(argumentos[0]);
        // Extraímos o primeiro caracter da String correspondente ao segundo argumento
        char operador = argumentos[1].charAt(0);
        // Extraímos um valor inteiro da String correspondente ao terceiro argumento
        int segundoValor = Integer.parseInt(argumentos[2]);
        // Dependendo do caracter operador, efetuamos a operação
        int resultado = 0; // deve ser inicializada
        switch(operador)
        {
            case '+': resultado = primeiroValor + segundoValor; break;
            case '-': resultado = primeiroValor - segundoValor; break;
            case 'x': resultado = primeiroValor * segundoValor; break;
            case '/': resultado = primeiroValor / segundoValor; break;
        }
        // Imprimimos os argumentos passados com espaços entre eles
        for(int indice=0;indice<argumentos.length;indice++)
            System.out.print(argumentos[indice]+" ");
        // Imprimimos o resultado
        System.out.println("="+resultado);
    } // fim do método main
} // fim da classe
```


Executando na linha de comando - Exemplos

```
$java CalculadoraDeLinhaDeComando 15 x 17
```

```
$15 x 17 = 255
```

```
$java CalculadoraDeLinhaDeComando 104 / 13
```

```
$104 / 13 = 8
```

```
$java CalculadoraDeLinhaDeComando 6 +
```

```
$Este programa precisa que tres argumentos sejam  
passados pela linha de comando
```

- 1) Usando a classe `CalculadoraDeLinhaDeComando` como base, escreva uma aplicação em Java que processe a linha de comando, recebendo três ou mais argumentos, de forma que o primeiro deva ser o operador '+' ou '*', e os argumentos do segundo em diante devam ser valores numéricos. A aplicação deve efetuar a soma ou multiplicação de todos os argumentos passados e mostrar o resultado. Se, por exemplo, os argumentos + 2 4 1 5 forem passados, a aplicação deverá imprimir o resultado 12. Se os argumentos * 2 4 1 5 forem passados, a aplicação deverá imprimir o resultado 40.

Vetor de objetos

Classes Pessoa.java e UsaPessoa.java

```
public class Pessoa{
    private String nome;
    private String cpf;

    public Pessoa(String nome, String cpf) {
        this.nome = nome;
        this.cpf = cpf;
    }
    public void imprimirDados(){
        System.out.println("Nome: " + nome);
        System.out.println("CPF: " + cpf);
    }
}
```

this guarda o endereço de memória do objeto

```
public class UsaPessoa{

    public static void main(String[] args) {
        /*Criando um vetor que permite alocar ate
        3 objetos da classe Pessoa */
        Pessoa[] agenda = new Pessoa[10];

        //Criando 2 objetos da classe Pessoa
        agenda[0] = new Pessoa("joao", "123");
        agenda[1] = new Pessoa("maria", "456");
        agenda[2] = new Pessoa("jose", "789");

        //Uma forma diferente do for para
        //percorrer vetores
        for (Pessoa p : agenda) {
            if(p!=null)p.imprimirDados();
        }
    }
} //fim da classe
```

Exercícios (cont.)

- 2) Escreva uma classe `Banco` que encapsule um array de instâncias da classe `ContaBancaria2`. Escreva para esta classe, um método `total` que calcule e retorne o total dos saldos de todas as contas bancárias encapsuladas como elementos do array.

Exercícios (cont.)

- 3) Escreva uma classe `ArrayDePontos2D`, no molde da classe `ArrayDeFloats` (moodle) que encapsule um array de instâncias da classe `Ponto2D` (moodle). Esta classe deve ter os seguintes métodos:
- Construtor, que recebe como argumento um número máximo de instâncias da classe `Ponto2D` que serão encapsuladas pela classe,
 - `tamanho`, que retorna o tamanho do array encapsulado,
 - `modifica`, que recebe como argumentos um valor inteiro (posição) e uma instância da classe `Ponto2D`, e faz com que a instância naquela posição do array passe a ser a passada como argumento,
 - `valor`, que recebe como argumento um valor inteiro (posição) e retorna a instância armazenada naquela posição do array,
 - `toString`, que retorna uma única string contendo todas os valores das instâncias no array encapsulado na classe.

Para facilitar a criação da classe, crie na classe `Ponto2D` ao menos um construtor que recebe argumentos e métodos para recuperar as coordenadas `x` e `y` do ponto.