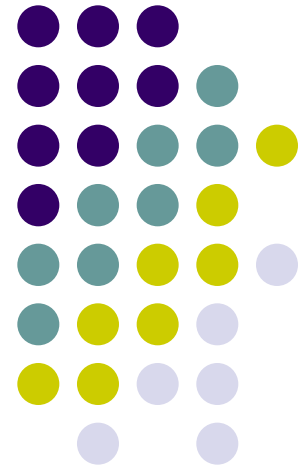


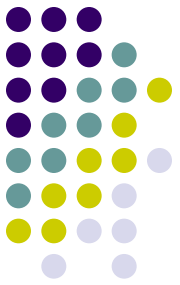
COM220

Aula 8: Arrays

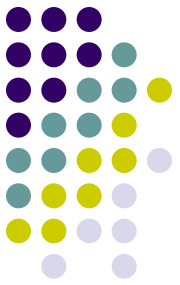
Prof. Laércio Baldochi



Conteúdo



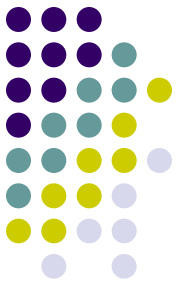
- Arrays
 - Declaração
 - Manipulação
 - Algoritmos
 - Classificação
 - Busca



Antes...

- Veremos como trabalhar com a interface gráfica do Java
 - Apenas uma classe: `JOptionPane`
 - Forma simples de ler e imprimir conteúdo em Janelas
 - Mais tarde...
 - Aprenderemos como usar todas as classes da API(Conjunto de Métodos) Java Swing

JOptionPane



- Classe *JOptionPane*

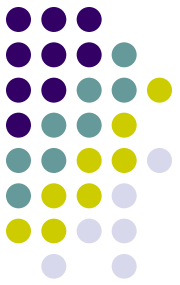
- Oferece caixas de diálogos para o usuário disponibilizar suas informações e também pedir informações ao usuário através de um campo de texto.

- Instrução *import*

- Usada para carregar classes utilizadas em um programa Java
- Classes importadas podem pertencer a pacotes do núcleo da linguagem, extensões oficiais ou extensões fornecidas por terceiros
- As instruções *import* devem aparecer sempre antes da definição das classes



JOptionPane

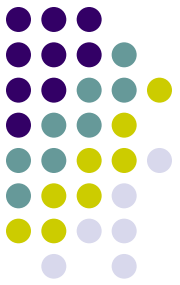
Como usar (parte 1)



```
JOptionPane.showMessageDialog(null, "\nBem-Vindo\nà  
Programação Java\n\t\t por Janelas");
```

- Método ***showMessageDialog*** tem dois argumentos
 - O primeiro é utilizado para **posicionamento** da janela. Ao ser null é ignorado e a janela é apresentada no centro da tela
 - O segundo representa a **string** que será apresentada na janela
- **System.exit(0)**
 - **Termina** o aplicativo Java. Necessário em programas com interface gráfica
 - Classe System não precisa ser importada
 - Faz parte do pacote padrão **java.lang**

- 
- Message**
-  Bem-Vindo
à Programação Java
por Janelas
- OK

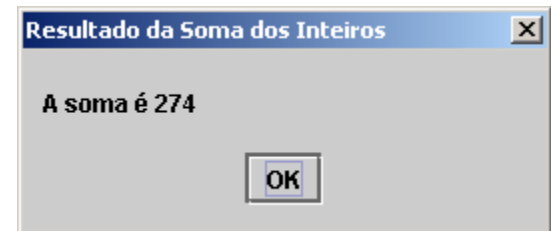
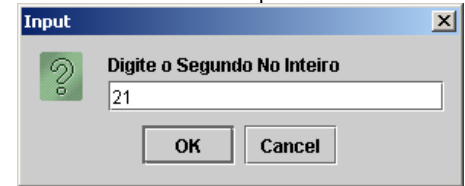
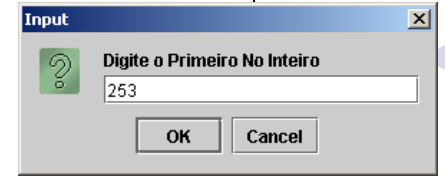


Realizando entrada de dados

- Além do método *showMessageDialog*, a classe *JOptionPane* disponibiliza também o método ***showInputDialog***
- Com esse método é possível fazer entrada de dados em janelas pré-definidas
- Veremos agora um exemplo que realiza a soma de dois números

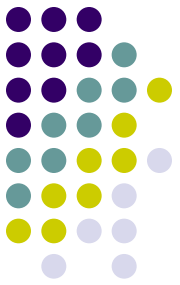


```
import javax.swing.JOptionPane; // import class JOptionPane
public class Adicao {
    public static void main( String args[] ) {
        String primeiroNumero; // 1o string informado pelo usuário
        String segundoNumero; // 2o string informado pelo usuário
        int numero1;           // primeiro operando da adição
        int numero2;           // segundo operando da adição
        int soma;              // Resultado da Adição
        // ler o primeiro número (na forma string)
        primeiroNumero = JOptionPane.showInputDialog("Digite o Primeiro No Inteiro" );
        // ler o segundo número (na forma string)
        segundoNumero = JOptionPane.showInputDialog( "Digite o Segundo No Inteiro" );
        // convertendo os strings em números inteiros
        numero1 = Integer.parseInt(primeiroNumero);
        numero2 = Integer.parseInt(segundoNumero);
        // Somando os números
        soma = numero1 + numero2;
        // Apresentando os resultados
        JOptionPane.showMessageDialog(null, "A soma é "+soma,"Resultado da Soma: ",
            JOptionPane.PLAIN_MESSAGE);
        System.exit( 0 ); // termina a aplicação
    } // fim do método main()
} // fim da classe Adicao
```



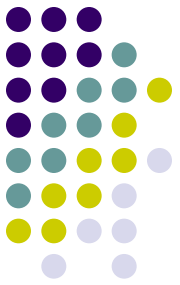
JOptionPane

Como usar (parte 2)



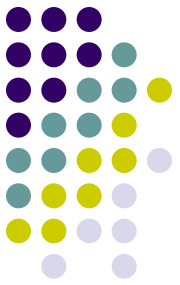
```
primeiroNumero = JOptionPane.showInputDialog("Digite o primeiro  
Número inteiro: ");
```

- O método `showInputDialog()` combina a montagem da janela de edição com o prompt de digitação da string fornecida pelo usuário
- Os valores lidos e escritos são sempre strings
 - Por essa razão o programa faz a conversão:
 - `numero1 = Integer.parseInt(primeiroNumero);`



Exercício

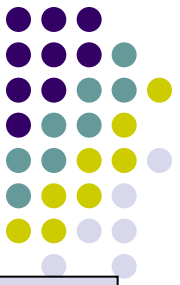
- Modifique o programa anterior de forma que seja calculada a média dos dois números lidos, ao invés de somá-los.



Arrays (Vetores)

- Estruturas de dados, na forma de um grupo de posições contíguas na memória, com valores de mesmo nome e mesmo tipo
- Funcionam de maneira "quase" idêntica àquela da linguagem C
- Vetores são estruturas estáticas
 - Uma vez criados, mantêm seu tamanho original
- Para estruturas dinâmicas, Java provê as classes Vector e Array

Arrays (Vetores)



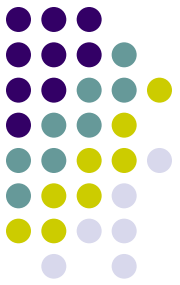
Nome do *array* (todos os elementos do vetor passam a ter o mesmo nome: 'c')

Vetores em Java podem ter seu comprimento conhecido pelo atributo **length**.
Sintaxe: ***nomeVet.length***

c [0]	- 128
c [1]	8
c [2]	0
c [3]	82
c [4]	64
c [5]	- 12
c [6]	65
c [7]	43
c [8]	76
c [9]	11

```
...  
public static void main (String args[]) {  
    int[] c = new int[10];  
    ou  
    int c[]={-128,8,0,82,64,-12,65,43,76,11};  
  
    ...  
    c[4] += c[2];    // c[4] = 64 + 0 = 64  
}
```

Número da posição do elemento dentro de um *array* (índice ou subscrito)



Declaração de vetores

- É possível declarar e, em seguida, reservar espaço para o vetor com o operador `new`

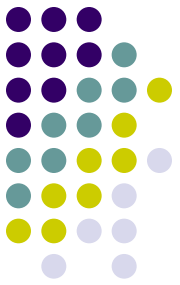
```
int c[];    // declaração do array
```

```
c = new int[12]; // declaração e reserva de espaço do array
```

- Pode-se também declarar e inicializar ao mesmo tempo

```
int c[] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

- Quando os vetores são declarados sem inicialização, o Java faz a inicialização para zeros (variáveis numéricas), `false` (variáveis lógicas do tipo `boolean`) ou `null` para referências a tipos de objetos.



Declaração múltipla

- Um programa Java pode declarar vários arrays em uma única declaração.

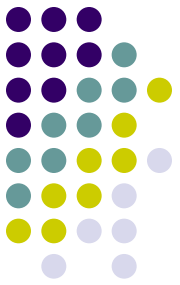
```
int[] arrayA, arrayB, arrayC;      // três arrays de  
inteiros
```

```
int[] arrayD = new int[121]; // criação de espaço com  
inicialização
```

```
String objTexto[] = new String [120],  
                    x[] = new String[21];  
// objTexto contém 120 objetos da classe String  
// x contém 21 objetos da classe String
```

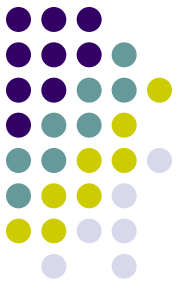
- Arrays de tipos não primitivos (ex. Objetos da classe String) guardam **referências** a objetos em seus elementos. A inicialização de referências é null.

Passando Array para um Método



```
public void somaElementos(int[] array) {  
    int soma = 0;  
    for(int i=0; i< array.length; i++)  
        soma+=array[i];  
    System.out.println("A soma do Array é :" + soma);  
}
```

Retornando um Array de um Método



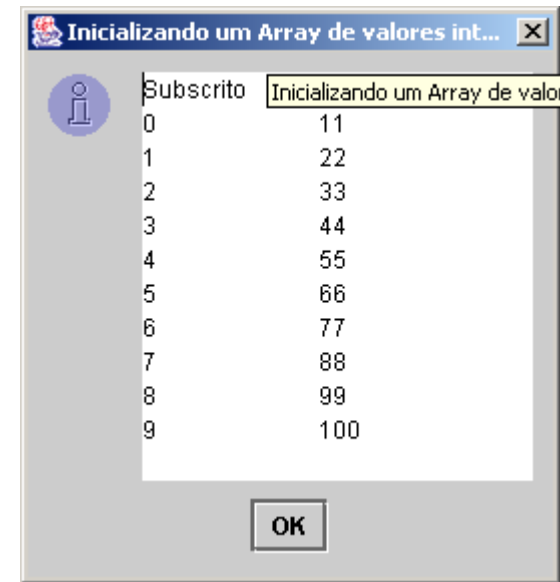
```
public int[] incrementaElementos(int[] array) {  
    for(int i=0; i< array.length; i++)  
        array[i]++;  
    return array;  
}
```

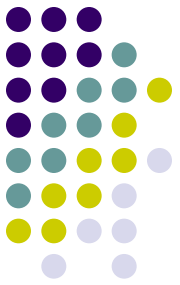

Exemplo 1

Criação e inicialização de vetores

```
import javax.swing.*;

public class InicializaVet {
    // função main
    public static void main( String args[] )    {
        // declaração com inicialização (dispensando operador new)
        int vet[] = {11,22,33,44,55,66,77,88,99,100};
        String saidaStr = "Subscrito\tValor\n"; // string alocado e inicializado
        // adiciona cada valor dos elementos do array ao String de saída
        for (int i = 0; i < vet.length; i++)
            saidaStr += i + "\t" + vet[i] + "\n";
        JTextArea saidaArea = new JTextArea();
        saidaArea.setText(saidaStr);
        JOptionPane.showMessageDialog( null, saidaArea,
            "Inicializando um Array de valores inteiros",
            JOptionPane.INFORMATION_MESSAGE );
        System.exit( 0 );
    }
}
```





Exemplo 2

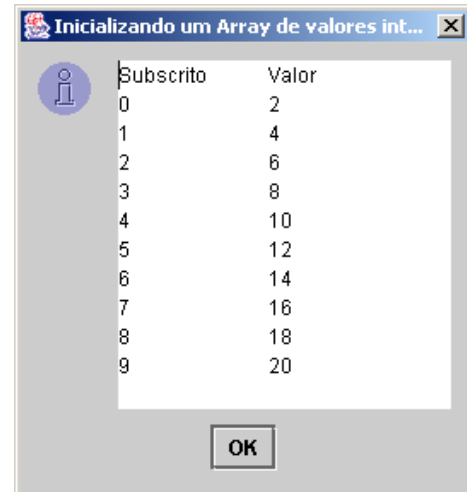
- No próximo exemplo usaremos uma constante para definir o tamanho do vetor
- O tamanho de vetores pode ser declarado com o tipo final.

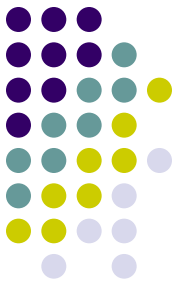
```
final int TAM_ARRAY = 10;  
int array = new int[TAM_ARRAY]; // array de 10  
itens (inicializados com zeros)
```

- O exemplo 2 guarda uma série aritmética de 2 nas posições do vetor

Exemplo 2

```
import javax.swing.*;
public class VetExemplo2 {
    // função main
    public static void main( String args[] )    {
        // declaração de vetor de inteiros com tamanho 10
        final int TAM_VET = 10;
        int vet[] = new int [TAM_VET];
        // calcula o valor para cada elemento do vetor
        for ( int i = 0; i < vet.length; i++)
            vet[i] = 2 + 2*i;
        String saidaStr = "Subscrito\tValor\n"; // string alocado e inicializado
        // adiciona cada valor dos elementos do vetor ao String de saída
        for (int i = 0; i < vet.length; i++)
            saidaStr += i + "\t" + vet[i] + "\n";
        JTextArea saidaArea = new JTextArea();
        saidaArea.setText(saidaStr);
        JOptionPane.showMessageDialog( null, saidaArea,
            "Inicializando um Array de valores inteiros",
            JOptionPane.INFORMATION_MESSAGE );
        System.exit( 0 );
    }
}
```





Exemplo 3

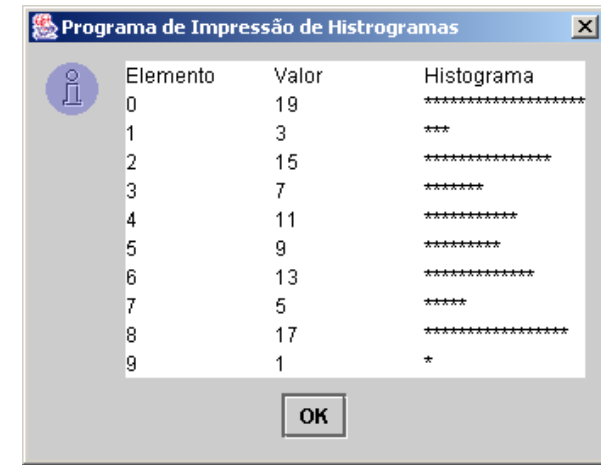
- Neste exemplo utilizaremos um histograma de barras para fazer uma melhor representação dos elementos de um vetor
- Utilizaremos uma sequência de asteriscos que representarão valores numéricos contidos nos vetores

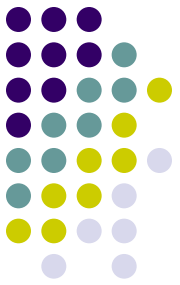
Exemplo 3

Histograma

```
import javax.swing.*;

public class Histograma {
    public static void main( String args[] )    {
        int vet[] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
        String saidaStr = "Elemento\tValor\tHistograma";
        // para cada elemento do vetor, apresentar barra em histograma
        for (int i = 0; i < vet.length; i++) {
            saidaStr += "\n" + i + "\t" + vet[i] + "\t";
            // apresenta barra de asteriscos
            for (int estrelas = 0; estrelas < vet[i]; estrelas++)
                saidaStr += "*";
        }
        JTextArea outputArea = new JTextArea();
        outputArea.setText( saidaStr );
        JOptionPane.showMessageDialog( null, outputArea,
            "Programa de Impressão de Histogramas",
            JOptionPane.INFORMATION_MESSAGE );
        System.exit( 0 );
    }
}
```





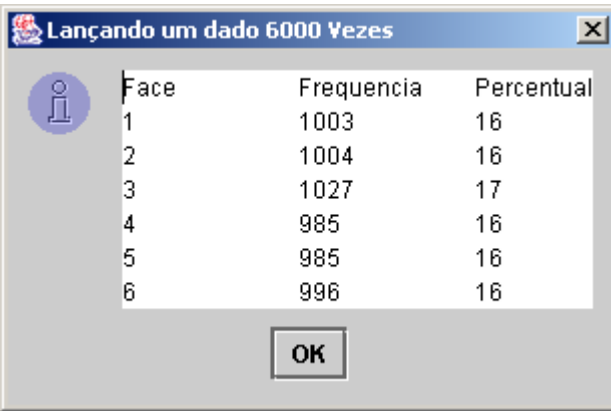
Exemplo 4

- Neste exemplo guardaremos a frequência da ocorrência de eventos em posições de um vetor
- Em seguida, faremos uma distribuição de probabilidade com base nos valores armazenados
- Programa
 - Realiza 6.000 lançamentos de um dado, utilizando o método `random()` da classe `Math`
 - Programa deve mostrar os resultados do experimento

Exemplo 4 - Lançamento de dados

```
import javax.swing.*;

public class LancamentoDados {
    public static void main( String args[] )    {
        int face, frequencia[] = new int[ 7 ];
        // lança o dado 6000 vezes
        for (int lancamento = 1; lancamento <= 6000; lancamento++) {
            face = 1 + ( int ) ( Math.random() * 6 );
            // utilizando o valor da variável face como subscrito do array
            ++frequencia[face];
        }
        String output = "Face\tFrequencia\tPercentual";
        // Adiciona frequências ao String de Saída
        for ( face = 1; face < frequencia.length; face++ )
            output += "\n" + face + "\t" + frequencia[ face ] + "\t" +
                100*frequencia[face]/6000;
        JTextArea outputArea = new JTextArea();
        outputArea.setText( output );
        JOptionPane.showMessageDialog( null, outputArea,
            "Lançando um dado 6000 Vezes",
            JOptionPane.INFORMATION_MESSAGE );
        System.exit( 0 );
    }
}
```

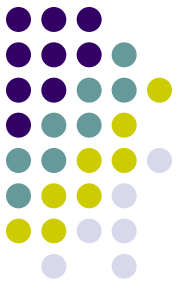


The screenshot shows a Java Swing window titled "Lançando um dado 6000 Vezes". Inside the window, there is a table with three columns: "Face", "Frequencia", and "Percentual". The table contains data for faces 1 through 6. Below the table is an "OK" button.

Face	Frequencia	Percentual
1	1003	16
2	1004	16
3	1027	17
4	985	16
5	985	16
6	996	16

Vetores

Classificação e Busca

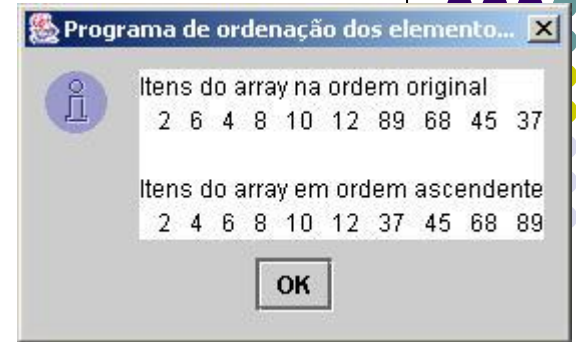


- É comum termos que localizar itens em vetores
- Para facilitar esse processo, é conveniente classificar (ordenar) o vetor e, assim, poder realizar uma busca mais eficiente, tal como a busca binária
- Veremos agora um programa que utiliza um dos métodos de ordenação mais simples, o método da bolha

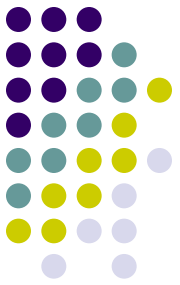
Ordenação de vetores

```
import javax.swing.*;

public class ClassificaArray extends JApplet {
    public static void main( String args[] ) {
        JTextArea saidaArea = new JTextArea();
        int array[] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
        String saida = "Itens do array na ordem original\n";
        // acrescenta valores originais do array ao String saida
        for ( int counter = 0; counter < array.length; counter++ )
            saida += "  " + array[ counter ];
        bubbleSort( array ); // ordena o array
        saida += "\n\nItens do array em ordem ascendente\n";
        // acrescenta valores ordenados do array ao String saida
        for ( int counter = 0; counter < array.length; counter++ )
            saida += "  " + array[ counter ];
        saidaArea.setText( saida );
        JOptionPane.showMessageDialog( null, saidaArea,
            "Programa de ordenação dos elementos de um array",
            JOptionPane.INFORMATION_MESSAGE );
        System.exit( 0 );
    }
}
```

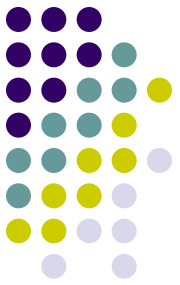


Ordenação de vetores



```
// ordena os elementos de um array considerando o algoritmo da Bolha
public static void bubbleSort(int array2[]) {
    // laço para controlar o número de passagens
    for ( int passagem = 1; passagem < array2.length; passagem++ ) {
        // laço para controlar o número de comparações
        for ( int elemento = 0; elemento < array2.length - 1;
            elemento++ ) {
            // compara elementos adjacentes e os troca de lugar se
            // o primeiro elemento for maior que o segundo elemento
            if ( array2[ elemento ] > array2[ elemento + 1 ] )
                troca( array2, elemento, elemento + 1 );
        } // fim do laço para controlar comparações
    } // fim do laço para controlar passagens
} // fim do método bubbleSort
```

```
// troca dois elementos de um array
public static void troca( int array3[], int prim, int sec ) {
    int elemento; // área de armazenamento temporário para troca
    elemento = array3[ prim ];
    array3[ prim ] = array3[ sec ];
    array3[ sec ] = elemento;
}
} // fim da classe ClassificaArray
```



Busca Binária

- Em um vetor ordenado, podemos realizar busca binária, a qual é muito mais eficiente que a busca sequencial
- Algoritmo da busca binária

Se (baixo > alto)

Retorne (-1)

$\text{meio} = (\text{alto} + \text{baixo}) / 2$

Se ($x == v[\text{meio}]$)

Retorne (meio);

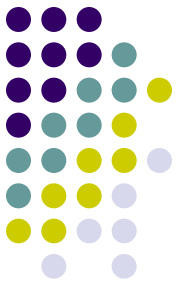
Se ($x < v[\text{meio}]$)

Busque por x em v de $v[\text{baixo}]$ até $v[\text{meio} - 1]$;
senão

Busque por x em v de $v[\text{meio} + 1]$ até $v[\text{alto}]$;

Exercício

Entrega: 02/04



Crie um programa em Java que leia um Array de 10 posições e realize as seguintes operações:

- O programa deve passar o Array como parâmetro para a função **invertePosicoes**, que vai ler o Array, inverter as posições, retornar o vetor invertido. O vetor retornado deverá então ser impresso numa JOptionPane.
- Em seguida, uma segunda função deve ordenar o Array pelo método da bolha e imprimir o vetor ordenado em uma JOptionPane
- Por fim, deve-se utilizar uma JOptionPane de entrada de dados para ler um valor numérico. Utilizando busca binária, deve-se verificar se o valor lido se encontra no Array, dando uma mensagem apropriada.