Arrays

Profa. Karen Selbach Borges

Introdução

- Arrays começam na posição zero.
- Arrays podem ser uni ou multidimensionais(regulares ou não)
- Arrays apresentam limitações em função da sua estrutura estática.
- Arrays também são objetos (classe Arrays)

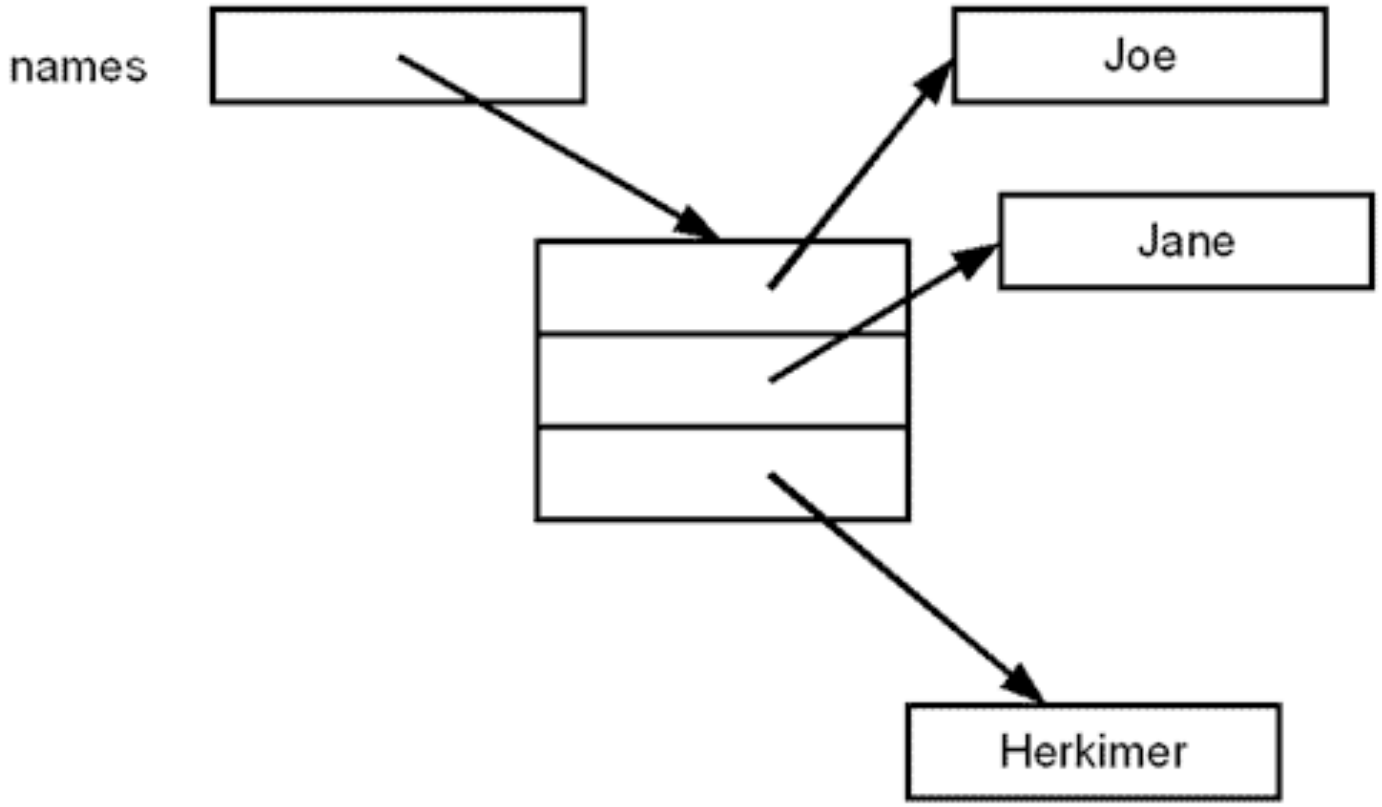
Arrays unidimensionais

- Vetores :
  - tipo nome[ ] = new tipo[tamanho]; ou
  - tipo[ ] nome = new tipo[tamanho].
    - Ex1: char vogais[ ] = new char[5];
    - Ex2: char[ ] vogais = new char[5];

Arrays unidimensionais

- Inicialização
  - As posições do array são inicializadas conforme os valores default do tipo declarado do array.
    - int[ ] vogais = new int[5] -> 5 posições contendo 0
    - boolean[ ] status = new boolean[5] -> 5 posições contendo false
  - É possível inicializar o array no momento da sua criação
    - String names[ ] = {“Joe”, “Jane”, “Herkimer" };

Arrays unidimensionais



Arrays unidimensionais

- Pesquisa
  - É feita comparando posição por posição
  - Pode-se utilizar o recurso de Foreach
    - Exemplo:

```
String[] data = { "Toronto", "Stockholm" };
  for (String s : data) {
    System.out.println(s);
  }
```

Exemplo1 - Vetores

```
class TestaArray {
    public static void main (String[] args) {
      int[ ] tabuada = new int[10];

      for (int i = 0; i<10; i++)
        tabuada[i] = i*5;

      System.out.println("Tabuada do 5");
      for (int i = 0; i<tabuada.length; i++)
        System.out.println(i + " * 5 = " + tabuada[i]);
    }
}
```

Argumentos do main

- O método main possui como parâmetro um array de String.
  - Permite passar valores para o programa no momento da execução. Exemplo:

```
class TestMain {
  public static void main (String [] args){
    System.out.println("First arg is " + args[0]);
  }
}
```

•Ao executar o código através da linha de comando

java TestMain Hello

•O Resultado será “First arg is Hello”

Arrays multidimensionais

- Matrizes :
  - tipo nome[ ] [ ] = new tipo[tamanho] [tamanho]; ou
  - tipo[ ] [ ] nome = new tipo[tamanho] [tamanho];
    - int mat[ ] [ ] = new int[2][3]; ou
    - int [ ] [ ] mat = new int[2][3].

Arrays multidimensionais

- Inicialização
  - As posições do array são inicializadas conforme os valores default do tipo declarado do array.
    - int[ ][ ] vogais = new int[2][5] -> 10 posições contendo 0
  - É possível inicializar o array no momento da sua criação
    - int mat[ ] [ ] = {{1, 2, 3}, {4, 5, 6}};
      - mat[0][0] = 1 e mat[0][1] = 2 e mat[0][2] = 3
      - mat[1][0] = 4 e mat[1][1] = 5 e mat[1][2] = 6

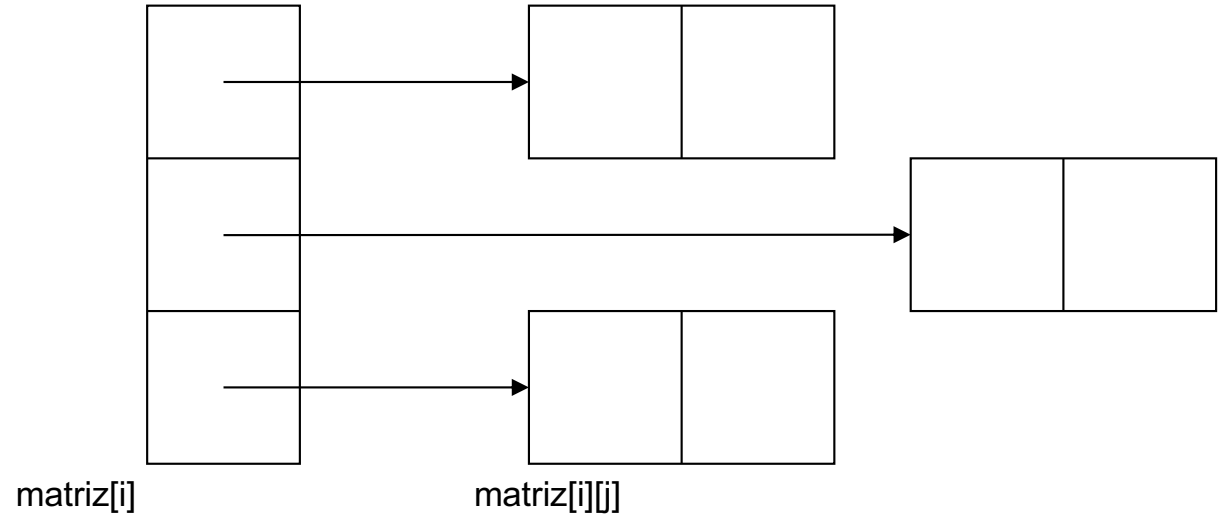
Exemplo1 - Matrizes

```
class TestaArrayMulti {
    public static void main (String[] args) {
      int[ ] [ ] matriz = new int[3][3];

      for (int i = 0; i<3; i++)
         for (int j = 0; j<3; j++)
            matriz[i][j] = i*j;

      System.out.println("Resultado da matriz");
      for (int i = 0; i<3; i++)
         for (int j=0; j<3; j++)
            System.out.println("[" +i+ "][" +j+ "]  = " +matriz[i][j]);
    } //fecha main
} //fecha class
```

Arrays multidimensionais

Em Java, as matrizes são tratadas como “arrays de arrays”

Ex : int[ ] [ ] matriz = new int[3][2];
matriz[i] é um vetor que aponta para outros vetores.

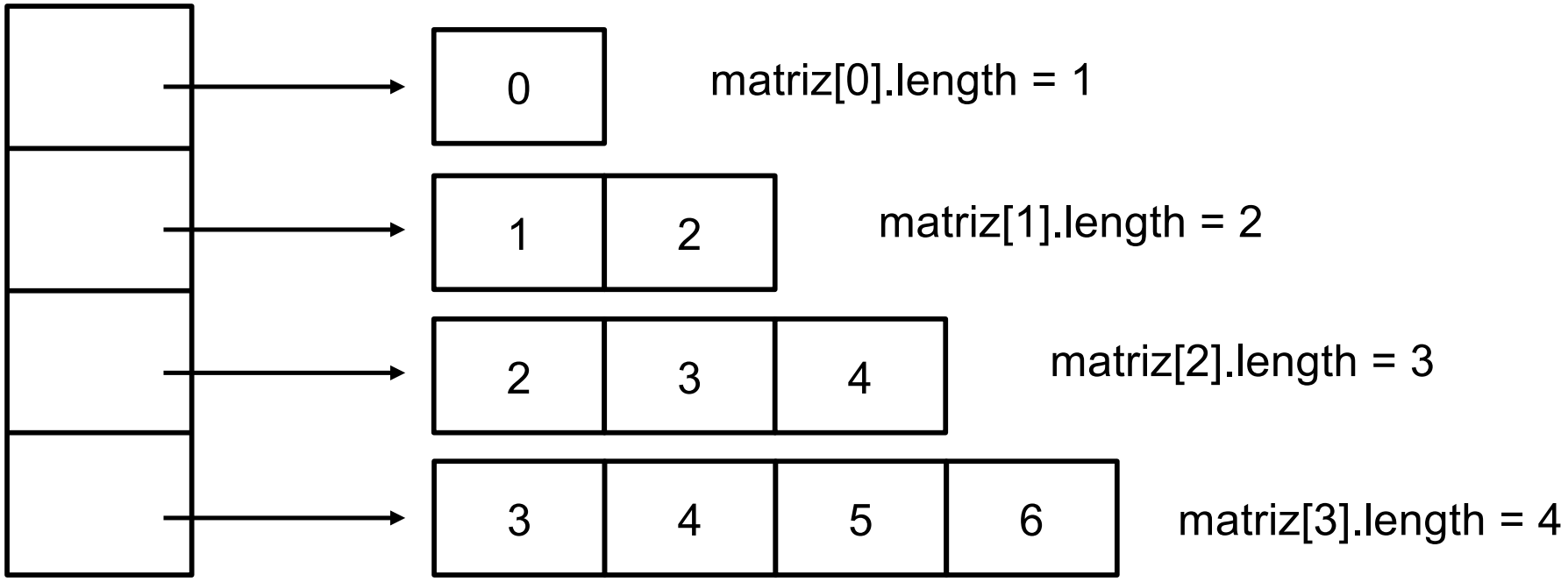


Exemplo2 - Matrizes

```
public class TestaArrayMulti2 {
        public static void main(String args[ ]) {
        int matriz[ ][ ] = new int[4][ ];
        matriz[0] = new int[1];
        matriz[1] = new int[2];
        matriz[2] = new int[3];
        matriz[3] = new int[4];
        for(int i=0;i<4;i++) {
            System.out.println();
            for (int j=0; j<=i; j++) {
                matriz[i][j]=i+j;
                System.out.print(" "+matriz[i][j]);
            }
        }
    }
}
```

Exemplo 2

Resultado da execução



Arrays como Argumentos

- Arrays podem ser utilizados como argumentos na passagem de parâmetros de métodos de uma classe

```
public class DataUtil {

    private Data[] feriados = new Data[15];

    //Método que cadastra os feriados
    public void setFeriados(Data[] d){
        feriados = d;
    }
}
```

Arrays como Argumentos

- Alterações realizadas sobre o vetor irão refletir no array original

```
public void sort(int[] v){
        int n = v.length;
        int incr = n / 2;
        while (incr >= 1){
            for (int i = incr; i < n; i++){
                int temp = v[i];
                int j = i;
                while (j >= incr && temp < v[j - incr]){
                    v[j] = v[j - incr];
                    j -= incr;
                }
                v[j] = temp;
            }
            incr /= 2;
        }
    }
```

Arrays como Valores de Retorno

- Métodos também podem retornar arrays.
- Isso é especialmente interessante quando se deseja retornar conjunto de valores.

```
public class DataUtil {

    private Data[] feriados = new Data[15];

    //Método que retorna as datas de todos os feriados cadastrados
    public Data[] getFeriados(){
        return feriados;
    }
}
```

Arrays como Coleção de Objetos

- Arrays podem ser úteis para organizar objetos.

```
public class DataUtil {

    private Data[] feriados = new Data[15];

    //Método que retorna as datas de todos os feriados cadastrados
    public Data[] getFeriados(){
        return feriados;
    }
}
```

Coleção de Datas

Arrays como Coleção de Objetos

- Observe que:
  - O que fica guardado dentro do array é a referência ao objeto
  - Logo, se existir outra referência a um objeto que está “dentro” de um array, alterações feitas utilizando essa referência irão alterar o conteúdo do array e vice-versa.

Arrays como Coleção de Objetos

- Arrays como coleção de objetos não é a melhor solução
- Soluções melhores podem ser encontradas no *Java Collection Framework*.