Introdução a Programação – IF669
http://www.cin.ufpe.br/~if669

Arrays

AULA 09

Ricardo Massa F. Lima
rmfl@cin.ufpe.br

Sérgio C. B. Soares
scbs@cin.ufpe.br

Centro de Informática — Universidade Federal de Pernambuco — CIn.ufpe.br

Até aqui . . .

- Conceitos de programação
  - tipos, entrada e saída de dados, operadores, comandos)
- Introdução a orientação a objetos
  - recursão e passagem de parâmetros
- HOJE: Array

Array

- Um array é uma coleção ordenada de valores
- Os valores podem ser primitivos, objetos ou outros arrays
- Todos os valores de um array devem ser do mesmo tipo
- Um array é um objeto!

Array

- São objetos especiais de Java
- Uma variável do tipo array é definida usando a notação:

```
Tipo[] arrayTipo;
```

- `Tipo[]` é uma classe, mas não pode-se herdar dela

ainda não estudamos!

Array - Exemplos

```
byte b;
byte[] arrayOfBytes;
byte[][] arrayOfArrayOfBytes;
Conta[] contas;
```

`byte`
array de `byte`
array de array de `byte`
array de `Conta`

SIMILAR

```
byte b;
byte arrayOfBytes[];
byte arrayOfArrayOfBytes[][];
Conta contas[];
```

Mas vamos padronizar usar os colchetes próximo do tipo

Criando um array

- Arrays não precisam ser inicializados no ato de sua criação
  - precisa especificar o tamanho do array
- Arrays têm tamanho fixo:
  - uma vez criado, não pode crescer ou diminuir

```
byte[] buffer = new byte[1024];
String[] lines = new String[50];
```

Cada posição do array é inicializada com valor **default** do tipo do array

Tamanho do array

- Acessando o tamanho de um array
  - Através do atributo `length`

```
Conta[] contas = new Conta[1];
if (contas.length != 2) {
  contas.length = 2;
}
```

`length` é um atributo constante (`final`) e `public` de todo objeto array

Acessando elementos de um array

- Elementos de um array são acessados usando o índice de sua posição (começa com zero)

```
String[] responses = new String[2];
responses[0] = "Yes";
responses[1] = "No";
System.out.println(responses[0]);
```

- Em linguagens como C/C++ é comum cometer um erro em que o código tenta acessar um valor com um índice superior ao limite do array
- Java checa o limite do array!
  - `ArrayIndexOutOfBoundsException`

Acessando elementos de um array

- Os índices de um array são do tipo inteiro
  - Não pode indexar arrays com float-point, `boolean`, objeto ou outro array
- Caractere pode ser convertido para inteiro e ser usado como índice
- `long` não pode ser usado como índice
  - um `int` é capaz de indexar dois bilhões de elementos (8Gb de memória)

Inicializando Array

índice do primeiro elemento sempre zero

```
int[] values = new int[100];
for(int i = 0; i < values.length; i++) {
  values[i]=i*2;
}
```

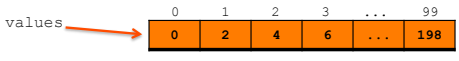
último elemento sempre `arrayName.length-1`

```
int[] powersOfTwo = {1, 2, 4, 8, 16, 32, 64};
```

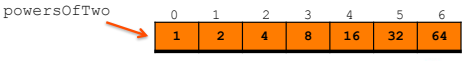
não usa `new`! permitido apenas na declaração

Visualizando um array

```
int[] values = new int[100];
for(int i = 0; i < values.length; i++) {
  values[i]=i*2;
}
```




```
int[] powersOfTwo = {1, 2, 4, 8, 16, 32, 64};
```

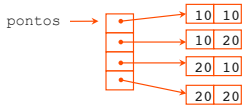


Exercício 1 (20 minutos)

- Crie a classe `MeuArray` que
  - Possui um array de inteiros como atributo
  - Possui um construtor que inicializa o atributo com um array para 5 inteiros e já define os 5 valores inteiros
  - Possui os seguintes métodos:
    - `getSum` - retorna a soma dos inteiros do array;
    - `getGreater` - retorna o maior inteiro do array;
    - `countNumber` – recebe um número inteiro (como parâmetro) e retorna o número de ocorrências desse inteiro no array;
    - `changePosition` – troca a posição de todos os elementos do array (o primeiro será o último, o segundo o penúltimo e assim por diante)
- Crie a classe `ProgramaMeuArray` para testar a classe `MeuArray`

Arrays multidimensionais

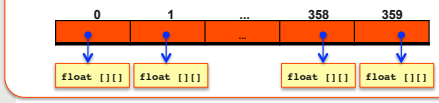
```
int[][] pontos = {{10,10},
                  {10,20},
                  {20,10},
                  {20,20}};
```



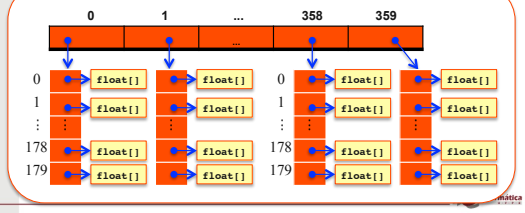

```
int[][] pontos = new int[4][2];
for(int i=0; i<4; i++)
  for(int j=0; j<2; j++)
    pontos[i][j] = in.nextInt();
```

Arrays multidimensionais

```
float[][][] globalTemperatureData = new float[360][][];
```

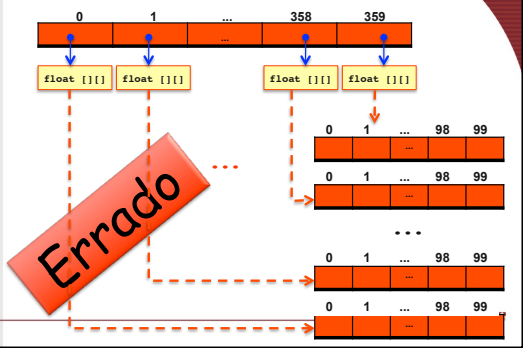



```
float[][][] globalTemperatureData = new float[360][180][];
```



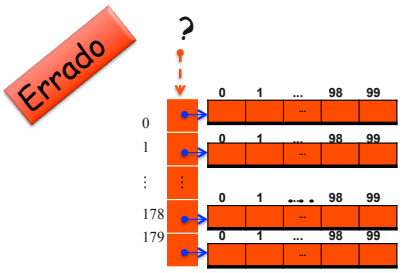
Arrays multidimensionais

```
float[][][] globalTemperatureData = new float[360][][100]
```



Arrays multidimensionais

```
float[][][] globalTemperatureData = new float[][180][100]
```



Arrays multidimensionais

inicialização array 5x5

```
int[][] products = { {0, 0, 0, 0, 0},
                     {0, 1, 2, 3, 4},
                     {0, 2, 4, 6, 8},
                     {0, 3, 6, 9, 12},
                     {0, 4, 8, 12, 16} };
```

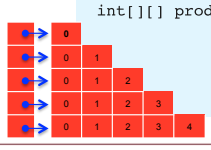
array anônimo

```
Teste t = new Teste();
t.m( new int[][] {{1,2}, {3,4}} );
```

Arrays multidimensionais não retangulares

- Em Java, arrays multidimensionais são considerados arrays de arrays
- Diferente de outras linguagens, que consideram arrays como arrays de blocos idênticos

```
int[][] products = { {0},
                     {0, 1},
                     {0, 1, 2},
                     {0, 1, 2, 3},
                     {0, 1, 2, 3, 4} };
```



Exercício 2

- Defina a classe `AulaMatriz` com:
  - Os seguintes atributos (sempre `private`):
    - **`matriz`** - um array de inteiros de duas dimensões
  - Um construtor que:
    - recebe um valor inteiro como parâmetro
    - usa o parâmetro para inicializar o array bidimensional
    - incializa cada posição do array com a soma dos índices da posição
  - Os seguintes métodos:
    - **`getMatriz`** - retorna o atributo
    - **`getDiagonal`** - retorna os elementos da diagonal principal do array (em um array de inteiros)
    - **`getLinha`** - recebe um inteiro e retorna todos os elementos da linha referente (em um array de inteiros); 0 é a 1ª linha
    - **`getColuna`** - recebe um inteiro e retorna todos os elementos da coluna referente (em um array de inteiros); 0 é a 1ª coluna
  - Crie uma classe **`ProgramaAulaMatriz`** com um método **`main`** para testar a classe **`AulaMatriz`**

Solução (1/3)

```
public class AulaMatriz {
    private int[][] matriz;

    public AulaMatriz(int dimensao) {
        this.matriz = new int[dimensao][dimensao];
        for (int i=0; i<dimensao; i++)
            for (int j=0; j<dimensao; j++)
                this.matriz[i][j] = i + j;
    }
    public int[][] getMatriz() {
        return this.matriz;
    }
    public int[] getDiagonal() {
        int[] resposta = new int[this.matriz.length];
        for(int i=0; i<this.matriz.length; i++)
            resposta[i] = this.matriz[i][i];
        return resposta;
    }
```

Solução (2/3)

```
    //considerando que a linha eh sempre a primeira dimensao
    public int[] getLinha(int linha) {
        int[] resposta = new int[this.matriz.length];
        for(int i=0; i<this.matriz.length; i++)
            resposta[i] = this.matriz[linha][i];
        return resposta;
    }

    public int[] getColuna(int coluna) {
        int[] resposta = new int[this.matriz.length];
        for(int i=0; i<this.matriz.length; i++)
            resposta[i] = this.matriz[i][coluna];
        return resposta;
    }
}
```

Solução (3/3)

```
public class Programa {

    public static void main(String[] args) {
        AulaMatriz m = new AulaMatriz(5);
        Programa.imprimeMatrizQuadrada(m.getMatriz());
        int[] diagonal = m.getDiagonal();
        Programa.imprimeArray(diagonal);
        int[] coluna1 = m.getColuna(1);
        Programa.imprimeArray(coluna1);
        int[] linha3 = m.getLinha(3);
        Programa.imprimeArray(linha3);
    }
    private static void imprimeArray(int[] array) {
        for (int i=0; i<array.length; i++)
            System.out.print(array[i] + " ");
    }
    private static void imprimeMatrizQuadrada(int[][] matriz) {
        for (int i=0; i<matriz.length; i++) {
            Programa.imprimeArray(matriz[i]);
            System.out.print("\n"); //quebrando a linha
        }
    }
}
```

Exercício 3 - Defina as classes

1. `DVD` com:
   - Os seguintes atributos (sempre `private`):
     - `codigo` - um inteiro que representa um identificador único do `DVD`
     - `descricao` - um string com a descrição do `DVD`
   - Um construtor que recebe o código e a descrição do `DVD` a ser criado
   - Métodos get e set para os dois atributos
2. `RepositorioDVDsArray` com:
   - Os seguintes atributos (sempre `private`):
     - `dvds` - um array de `DVD`
     - `indice` - um inteiro que guarda o índice do array onde será inserido o próximo `DVD` (inicia pelo índice zero)
   - Um construtor que recebe a capacidade do repositorio (inteira) e inicializa o array com esse tamanho (inicialize o `indice`)
   - Os métodos (lembre-se de usar o atributo `indice`):
     - `inserir` - recebe um `DVD` e insere no array
     - `procurar` - recebe o `codigo` de um `DVD` e retorna o `DVD`
     - `remover` - recebe o `codigo` de um `DVD` e remove o `DVD`
3. `Programa` com um método `main` que cria um objeto da classe `RepositorioDVDsArray` e testa todos os métodos da mesma pelo menos duas vezes.

Exercício 3 - Continuação

- Agora crie um método `toString` na classe `DVD` que retorna um `String` com os dados do `DVD`, formatados da seguinte forma:

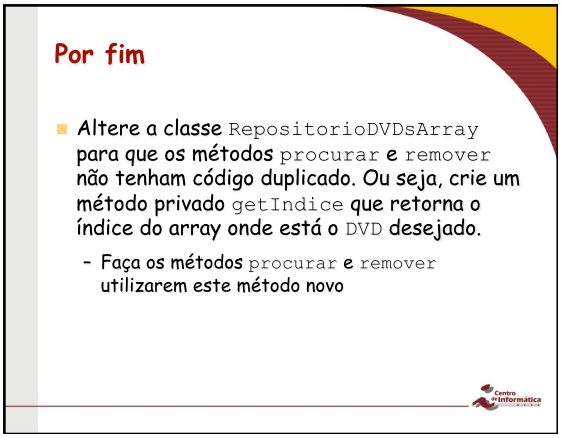
```
Descrição: Matrix
Código: 21
```

- Modifique a classe `RepositorioDVDsArray` para implementar o método `toString` que retorna um `String` com os dados de todos os DVDs inseridos no array, formatados da seguinte forma:

```
############
Descrição: Matrix
Código: 21
############
Descrição: Hulk
Código: 10
############
```

DICA: use o método `toString` definido na classe `DVD`

- Modifique a classe `Programa` para imprimir a variável do tipo `RepositorioDVDsArray`, ou seja, a referência. Faça o mesmo sempre que precisar imprimir os dados de um `DVD`, imprima a variável, ou seja, a referência.


Por fim
Altere a classe RepositorioDVDsArray para que os métodos procurar e remover não tenham código duplicado. Ou seja, crie um método privado getIndice que retorna o índice do array onde está o DVD desejado.
Faça os métodos procurar e remover utilizarem este método novo