Aula 11 – Matrizes

Norton Trevisan Roman

20 de abril de 2013

Arranjos

- Voltemos à necessidade de se dar nomes aos materiais

Arranjos

- Voltemos à necessidade de se dar nomes aos materiais

```
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;

/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};

/* nomes dos materiais */
static char[] nAlvenaria = {'A','l','v','e','n','a',
                            'r','i','a'};
static char[] nVinil = {'V','i','n','i','l'};
static char[] nFibra = {'F','i','b','r','a'};
static char[] nPlastico = {'P','l','á','s','t','i',
                           'c','o'};
```

Arranjos

- Voltemos à necessidade de se dar nomes aos materiais
- Qual o problema?

```
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;

/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};

/* nomes dos materiais */
static char[] nAlvenaria = {'A','l','v','e','n','a',
                            'r','i','a'};
static char[] nVinil = {'V','i','n','i','l'};
static char[] nFibra = {'F','i','b','r','a'};
static char[] nPlastico = {'P','l','á','s','t','i',
                           'c','o'};
```

Arranjos

- Voltemos à necessidade de se dar nomes aos materiais
- Qual o problema?
  - Começou a crescer demais...

```
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;

/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};

/* nomes dos materiais */
static char[] nAlvenaria = {'A','l','v','e','n','a',
                            'r','i','a'};
static char[] nVinil = {'V','i','n','i','l'};
static char[] nFibra = {'F','i','b','r','a'};
static char[] nPlastico = {'P','l','á','s','t','i',
                           'c','o'};
```

Arranjos

- Voltemos à necessidade de se dar nomes aos materiais
- Qual o problema?
  - Começou a crescer demais...
- O que poderíamos fazer?

```
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;

/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};

/* nomes dos materiais */
static char[] nAlvenaria = {'A','l','v','e','n','a',
                            'r','i','a'};
static char[] nVinil = {'V','i','n','i','l'};
static char[] nFibra = {'F','i','b','r','a'};
static char[] nPlastico = {'P','l','á','s','t','i',
                           'c','o'};
```

Arranjos

- Voltemos à necessidade de se dar nomes aos materiais
- Qual o problema?
  - Começou a crescer demais...
- O que poderíamos fazer?
  - Um agrupamento semelhante ao feito com os preços.

```
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;

/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};

/* nomes dos materiais */
static char[] nAlvenaria = {'A','l','v','e','n','a',
                            'r','i','a'};
static char[] nVinil = {'V','i','n','i','l'};
static char[] nFibra = {'F','i','b','r','a'};
static char[] nPlastico = {'P','l','á','s','t','i',
                           'c','o'};
```

Arranjos

- Voltemos à necessidade de se dar nomes aos materiais
- Qual o problema?
  - Começou a crescer demais...
- O que poderíamos fazer?
  - Um agrupamento semelhante ao feito com os preços.
- E como seria esse agrupamento?

```
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;

/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};

/* nomes dos materiais */
static char[] nAlvenaria = {'A','l','v','e','n','a',
                            'r','i','a'};
static char[] nVinil = {'V','i','n','i','l'};
static char[] nFibra = {'F','i','b','r','a'};
static char[] nPlastico = {'P','l','á','s','t','i',
                           'c','o'};
```

Arranjos

- Voltemos à necessidade de se dar nomes aos materiais
- Qual o problema?
  - Começou a crescer demais...
- O que poderíamos fazer?
  - Um agrupamento semelhante ao feito com os preços.
- E como seria esse agrupamento?
  - Um arranjo de strings

```
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;

/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};

/* nomes dos materiais */
static char[] nAlvenaria = {'A','l','v','e','n','a',
                            'r','i','a'};
static char[] nVinil = {'V','i','n','i','l'};
static char[] nFibra = {'F','i','b','r','a'};
static char[] nPlastico = {'P','l','á','s','t','i',
                           'c','o'};
```

Arranjos

- Voltemos à necessidade de se dar nomes aos materiais
- Qual o problema?
  - Começou a crescer demais...
- O que poderíamos fazer?
  - Um agrupamento semelhante ao feito com os preços.
- E como seria esse agrupamento?
  - Um arranjo de strings
  - Um arranjo de arranjos

```
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;

/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};

/* nomes dos materiais */
static char[] nAlvenaria = {'A','l','v','e','n','a',
                            'r','i','a'};
static char[] nVinil = {'V','i','n','i','l'};
static char[] nFibra = {'F','i','b','r','a'};
static char[] nPlastico = {'P','l','á','s','t','i',
                           'c','o'};
```

Arranjos de Arranjos

- E como ficaria em java?

Arranjos de Arranjos

- E como ficaria em java?

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);



}
```

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:
  - A terceira letra do segundo nome:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);



}
```

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:
  - A terceira letra do segundo nome:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);
    System.out.println(nomes[1][2]);



}
```

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:
  - A terceira letra do segundo nome:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);
    System.out.println(nomes[1][2]);



}
```

Saída

```
$ java AreaCasa
Vinil
n
```

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:
  - A terceira letra do segundo nome:

Saída

```
$ java AreaCasa
Vinil
n
```

  - E o que acontece se fizermos:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);
    System.out.println(nomes[1][2]);

    System.out.println(nomes.length);

}
```

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:
  - A terceira letra do segundo nome:

Saída

```
$ java AreaCasa
Vinil
n
```

  - E o que acontece se fizermos: Dá 4

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);
    System.out.println(nomes[1][2]);

    System.out.println(nomes.length);

}
```

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:
  - A terceira letra do segundo nome:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);
    System.out.println(nomes[1][2]);

    System.out.println(nomes.length);
    System.out.println(nomes[0].length);
}
```

Saída

```
$ java AreaCasa
Vinil
n
```

  - E o que acontece se fizermos: Dá 4
  - E se fizermos:

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:
  - A terceira letra do segundo nome:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);
    System.out.println(nomes[1][2]);

    System.out.println(nomes.length);
    System.out.println(nomes[0].length);
}
```

Saída

```
$ java AreaCasa
Vinil
n
```

  - E o que acontece se fizermos: Dá 4
  - E se fizermos: Dá 9

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:
  - A terceira letra do segundo nome:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);
    System.out.println(nomes[1][2]);

    System.out.println(nomes.length);
    System.out.println(nomes[0].length);
}
```

Saída

```
$ java AreaCasa
Vinil
n
```

  - E o que acontece se fizermos: Dá 4
  - E se fizermos: Dá 9
    - ★ O comprimento de "Alvenaria"

Arranjos de Arranjos

- E como ficaria em java?
- E como acessamos isso?
  - O segundo nome:
  - A terceira letra do segundo nome:

```
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};


public static void main(String[] args) {
    System.out.println(nomes[1]);
    System.out.println(nomes[1][2]);

    System.out.println(nomes.length);
    System.out.println(nomes[0].length);
}
```

Saída

```
$ java AreaCasa
Vinil
n
```

  - E o que acontece se fizermos: Dá 4
  - E se fizermos: Dá 9
    - ★ O comprimento de "Alvenaria"
    - ★ Cada arranjo (definido por [ ]) possui um atributo *length* associado

Arranjos de Arranjos

- E qual o tamanho de cada um dos nomes?

Arranjos de Arranjos

- E qual o tamanho de cada um dos nomes?

```
public static void main(String[] args) {
    for (int i=0; i<4; i++) {
        System.out.print(
             nomes[i].length);
        System.out.print(" ");
    }
    System.out.println();



}
```

Arranjos de Arranjos

- E qual o tamanho de cada um dos nomes?

Saída

```
9 5 5 8
```

```
public static void main(String[] args) {
    for (int i=0; i<4; i++) {
        System.out.print(
             nomes[i].length);
        System.out.print(" ");
    }
    System.out.println();



}
```

Arranjos de Arranjos

- E qual o tamanho de cada um dos nomes?

Saída

```
9 5 5 8
```

- O tamanho é variável

```
public static void main(String[] args) {
    for (int i=0; i<4; i++) {
        System.out.print(
             nomes[i].length);
        System.out.print(" ");
    }
    System.out.println();



}
```

Arranjos de Arranjos

- E qual o tamanho de cada um dos nomes?

Saída

```
9 5 5 8
```

- O tamanho é variável
- Mas há outro modo de se escrever o mesmo laço:

```
public static void main(String[] args) {
    for (int i=0; i<4; i++) {
        System.out.print(
             nomes[i].length);
        System.out.print(" ");
    }
    System.out.println();



}
```

Arranjos de Arranjos

- E qual o tamanho de cada um dos nomes?

Saída

```
9 5 5 8
```

- O tamanho é variável
- Mas há outro modo de se escrever o mesmo laço:

```
public static void main(String[] args) {
    for (int i=0; i<4; i++) {
        System.out.print(
             nomes[i].length);
        System.out.print(" ");
    }
    System.out.println();

    for (char[] nome : nomes) {
        System.out.print(nome.length);
        System.out.print(" ");
    }
    System.out.println();
}
```

Arranjos de Arranjos

- E qual o tamanho de cada um dos nomes?

Saída

```
9 5 5 8
```

- O tamanho é variável
- Mas há outro modo de se escrever o mesmo laço:
- Note que *nome* é *char[]*

```
public static void main(String[] args) {
    for (int i=0; i<4; i++) {
        System.out.print(
             nomes[i].length);
        System.out.print(" ");
    }
    System.out.println();

    for (char[] nome : nomes) {
        System.out.print(nome.length);
        System.out.print(" ");
    }
    System.out.println();
}
```

Arranjos de Arranjos

- E qual o tamanho de cada um dos nomes?

Saída

```
9 5 5 8
```

- O tamanho é variável
- Mas há outro modo de se escrever o mesmo laço:
- Note que *nome* é *char[]*
  - Logo, podemos fazer *nome[2]* para acessar sua terceira letra, por exemplo

```
public static void main(String[] args) {
    for (int i=0; i<4; i++) {
        System.out.print(
             nomes[i].length);
        System.out.print(" ");
    }
    System.out.println();

    for (char[] nome : nomes) {
        System.out.print(nome.length);
        System.out.print(" ");
    }
    System.out.println();
}
```

Arranjos de Arranjos

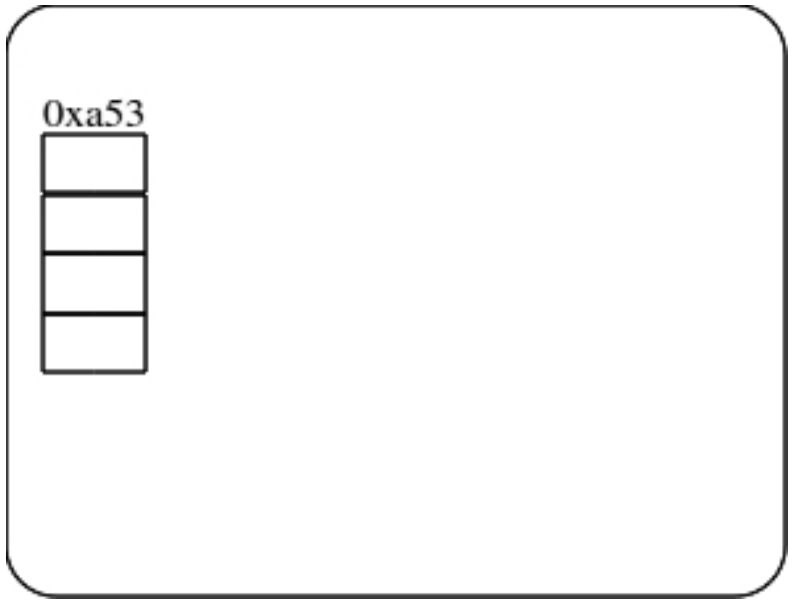
- O que acontece na memória quando declaramos o arranjo?

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

Arranjos de Arranjos

- O que acontece na memória quando declaramos o arranjo?
  - Primeiro alocamos espaço para o arranjo *nomes*

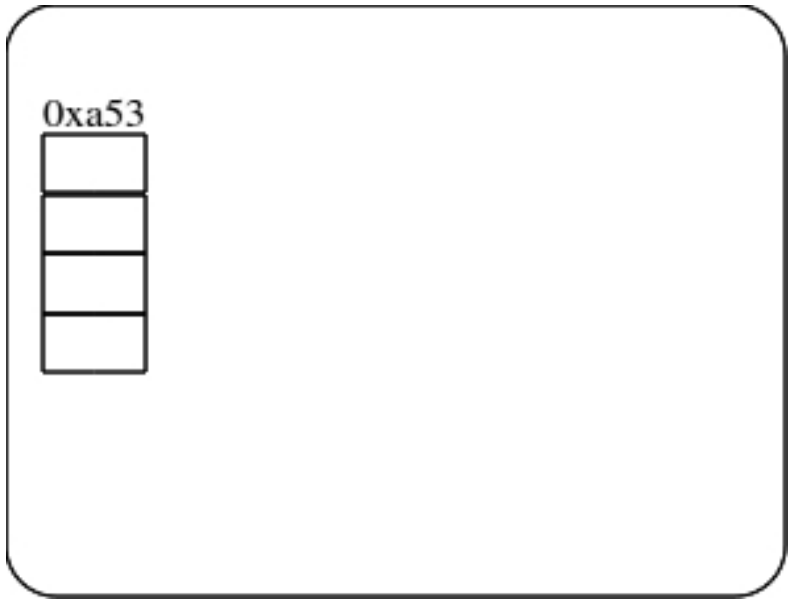
```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



Arranjos de Arranjos

- O que acontece na memória quando declaramos o arranjo?
  - Primeiro alocamos espaço para o arranjo *nomes*
    - ★ Cada posição terá espaço suficiente para um endereço

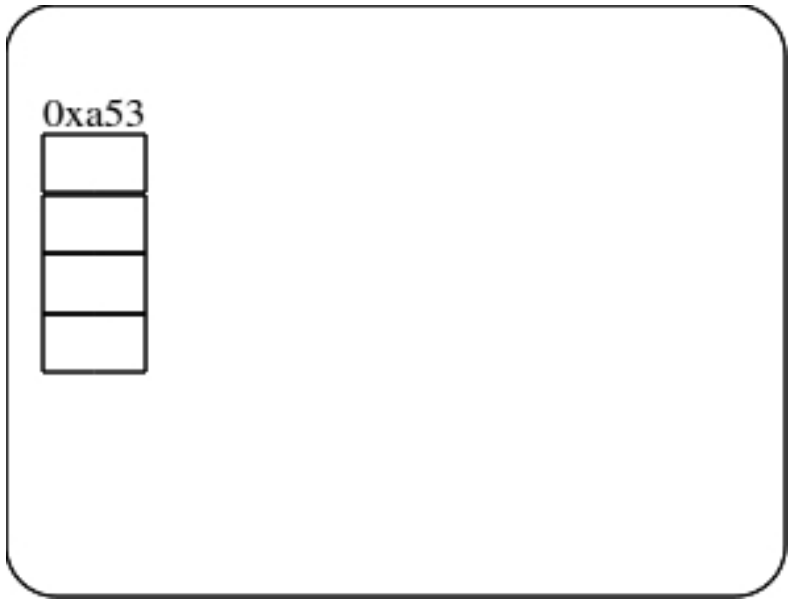
```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



Arranjos de Arranjos

- O que acontece na memória quando declaramos o arranjo?
  - Primeiro alocamos espaço para o arranjo *nomes*
    - ★ Cada posição terá espaço suficiente para um endereço
    - ★ 32 ou 64 bits

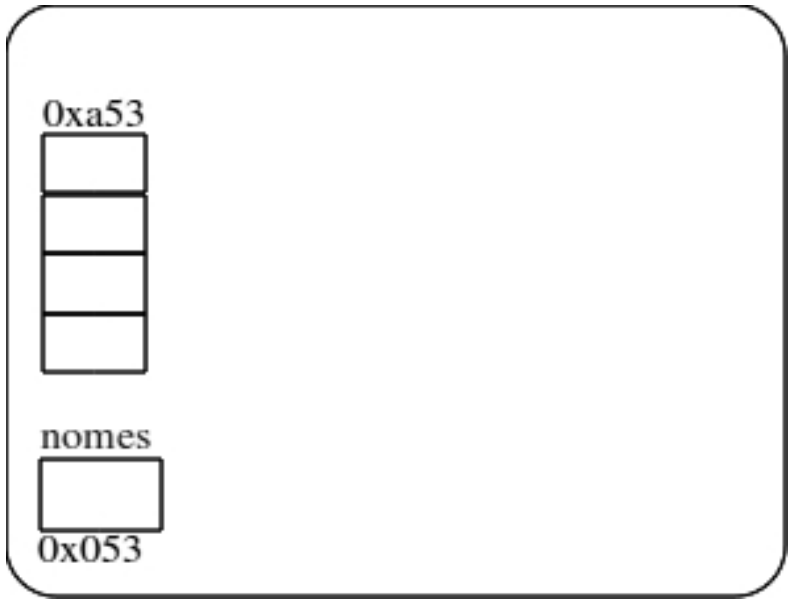
```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



Arranjos de Arranjos

- O que acontece na memória quando declaramos o arranjo?
  - Primeiro alocamos espaço para o arranjo *nomes*
    - ★ Cada posição terá espaço suficiente para um endereço
    - ★ 32 ou 64 bits
  - Em seguida, alocamos espaço para a variável *nomes*

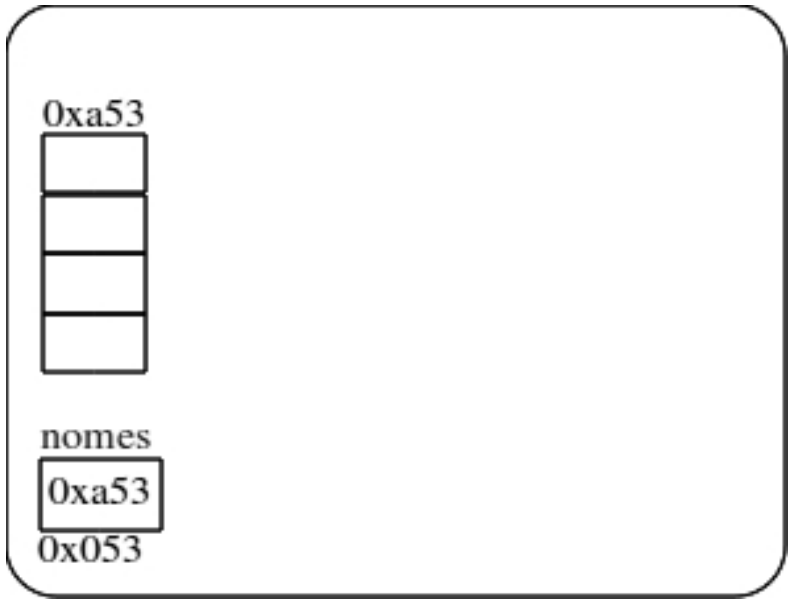
```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



Arranjos de Arranjos

- O que acontece na memória quando declaramos o arranjo?
  - Primeiro alocamos espaço para o arranjo *nomes*
    - ★ Cada posição terá espaço suficiente para um endereço
    - ★ 32 ou 64 bits
  - Em seguida, alocamos espaço para a variável *nomes*
    - ★ Conterá o endereço do arranjo correspondente a *nomes*

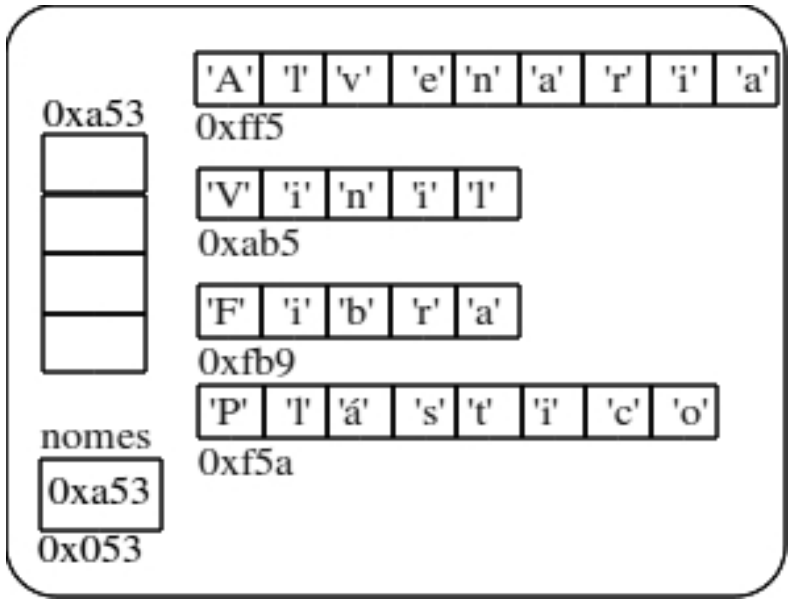
```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



Arranjos de Arranjos

- O que acontece na memória quando declaramos o arranjo?
  - Primeiro alocamos espaço para o arranjo *nomes*
    - ★ Cada posição terá espaço suficiente para um endereço
    - ★ 32 ou 64 bits
  - Em seguida, alocamos espaço para a variável *nomes*
    - ★ Conterá o endereço do arranjo correspondente a *nomes*
  - Alocamos então espaço para os arranjos que compõem *nomes*

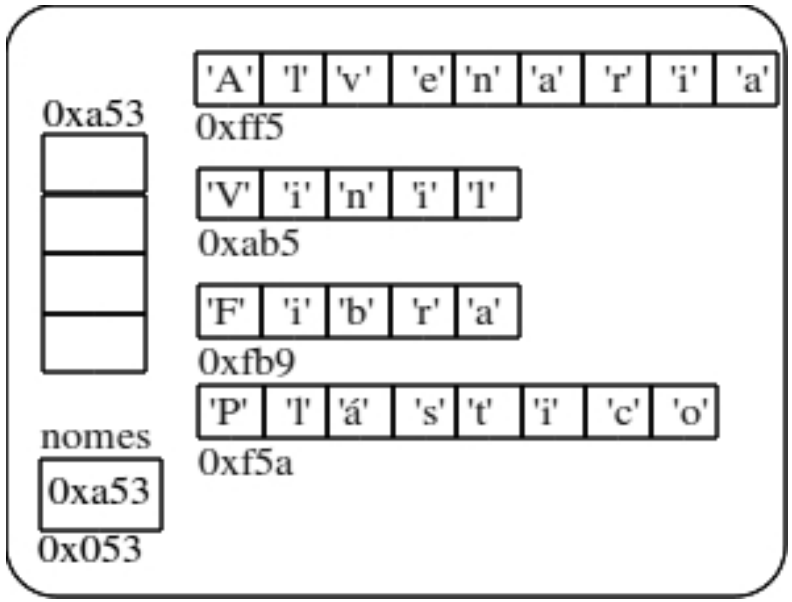
```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



Arranjos de Arranjos

- O que acontece na memória quando declaramos o arranjo?
  - Primeiro alocamos espaço para o arranjo *nomes*
    - ★ Cada posição terá espaço suficiente para um endereço
    - ★ 32 ou 64 bits
  - Em seguida, alocamos espaço para a variável *nomes*
    - ★ Conterá o endereço do arranjo correspondente a *nomes*

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

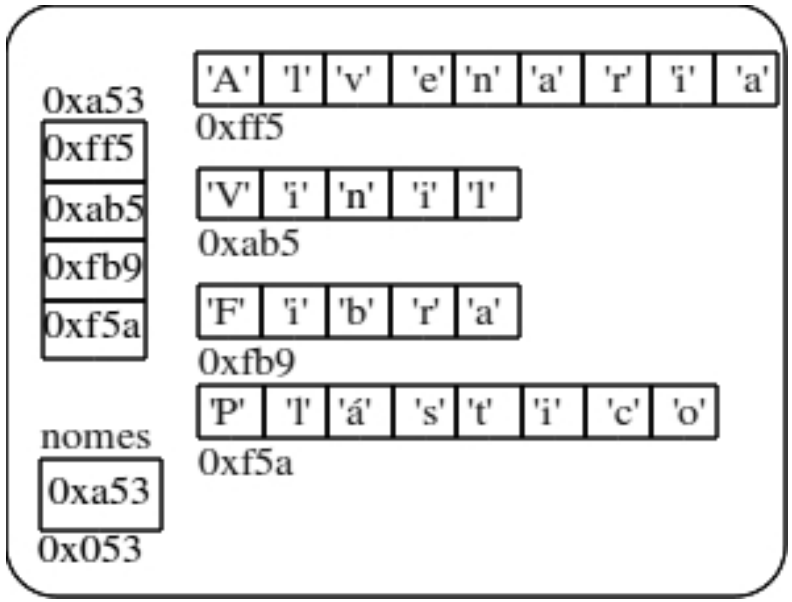


  - Alocamos então espaço para os arranjos que compõem *nomes*
    - ★ Note que não há variável que guarde seus endereços

Arranjos de Arranjos

- O que acontece na memória quando declaramos o arranjo?
  - Primeiro alocamos espaço para o arranjo *nomes*
    - ★ Cada posição terá espaço suficiente para um endereço
    - ★ 32 ou 64 bits
  - Em seguida, alocamos espaço para a variável *nomes*
    - ★ Conterá o endereço do arranjo correspondente a *nomes*

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

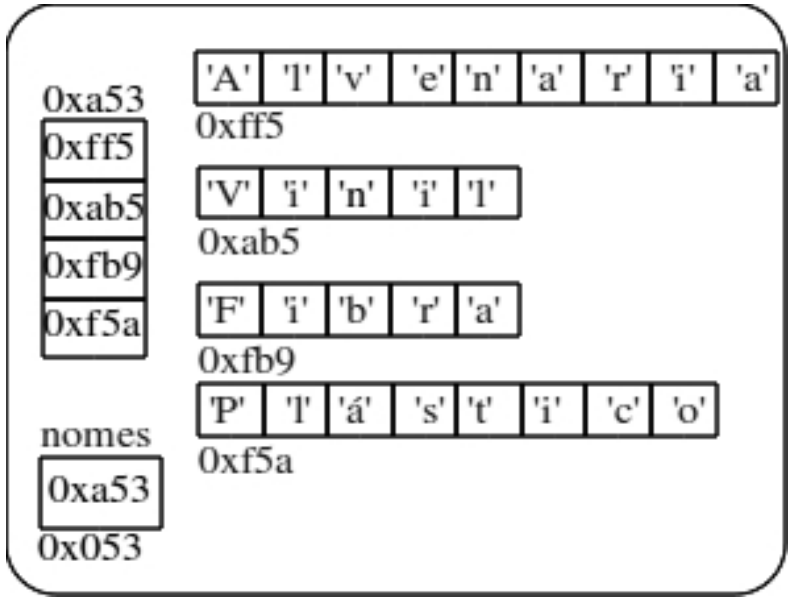


- Alocamos então espaço para os arranjos que compõem *nomes*
  - ★ Note que não há variável que guarde seus endereços

Em seguida, guardamos seus endereços nas posições de *nomes*

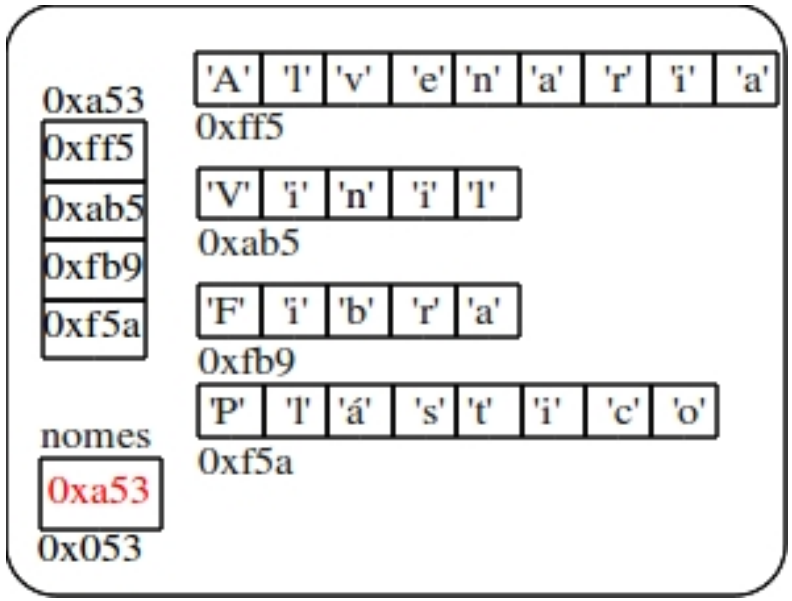
Arranjos de Arranjos

- E o que acontece na memória quando fazemos *nomes[1][2]*?



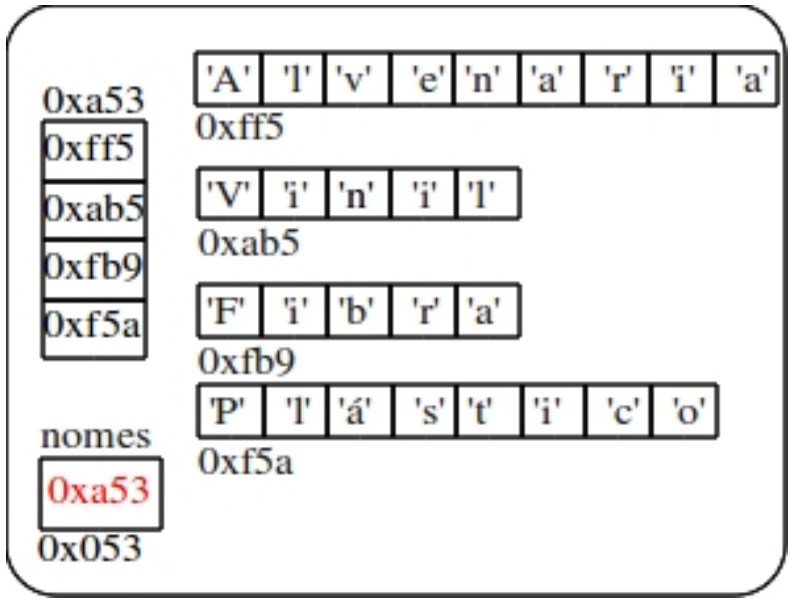
Arranjos de Arranjos

- E o que acontece na memória quando fazemos *nomes[1][2]*?
  - O conteúdo da variável *nomes* é lido



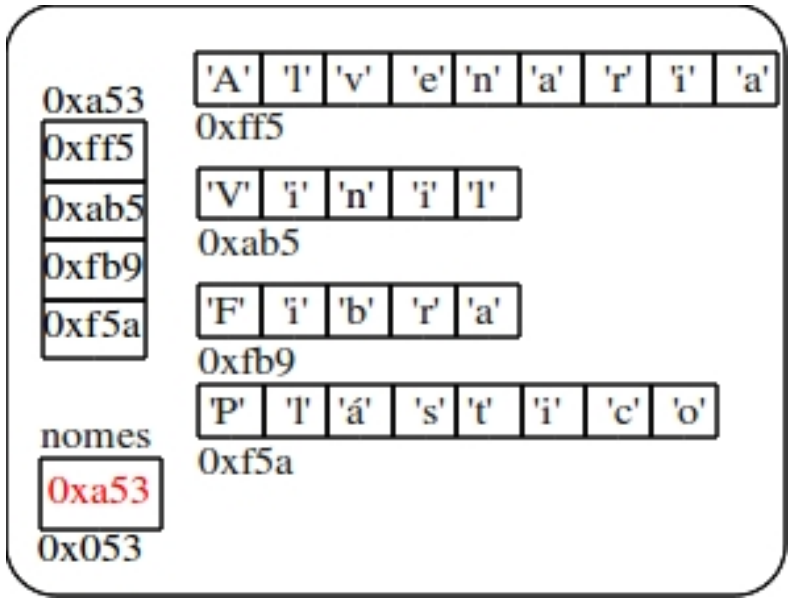
Arranjos de Arranjos

- E o que acontece na memória quando fazemos *nomes[1][2]*?
  - O conteúdo da variável *nomes* é lido
    - ★ O deslocamento é calculado $(A53 + 1 \times 8)$



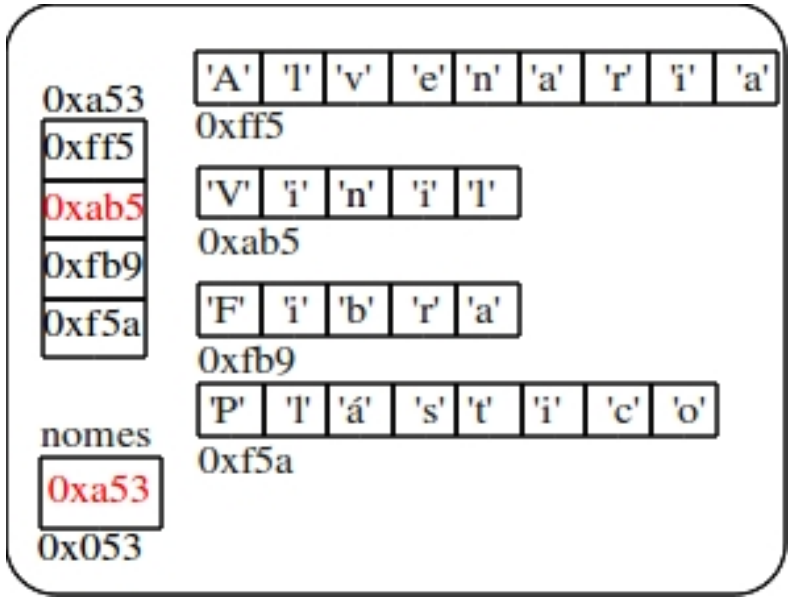
Arranjos de Arranjos

- E o que acontece na memória quando fazemos *nomes[1][2]*?
  - O conteúdo da variável *nomes* é lido
    - ★ O deslocamento é calculado ($A53 + 1 \times 8$)
    - ★ Sendo 8B o tamanho do endereço (64bits)



Arranjos de Arranjos

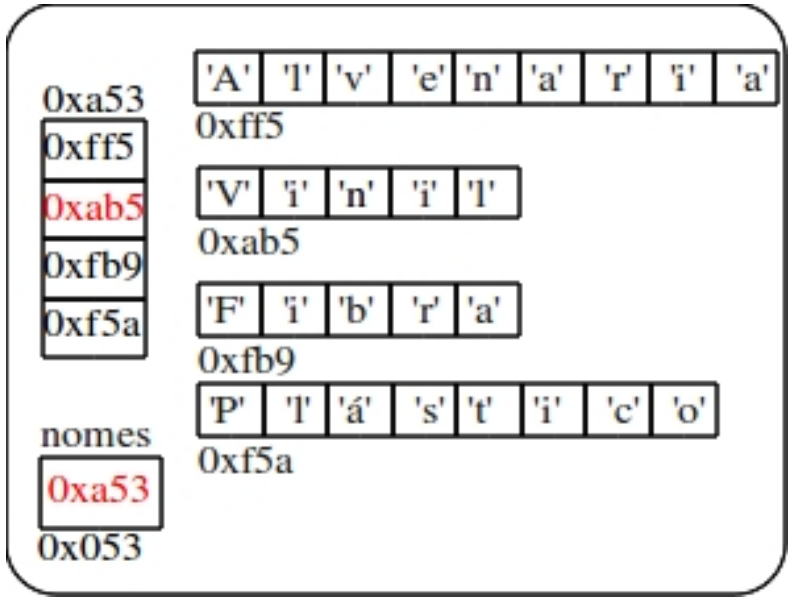
- E o que acontece na memória quando fazemos *nomes[1][2]*?
  - O conteúdo da variável *nomes* é lido
    - ★ O deslocamento é calculado ($A53 + 1 \times 8$)
    - ★ Sendo 8B o tamanho do endereço (64bits)



  - A região de memória correspondente a esse novo endereço é lida

Arranjos de Arranjos

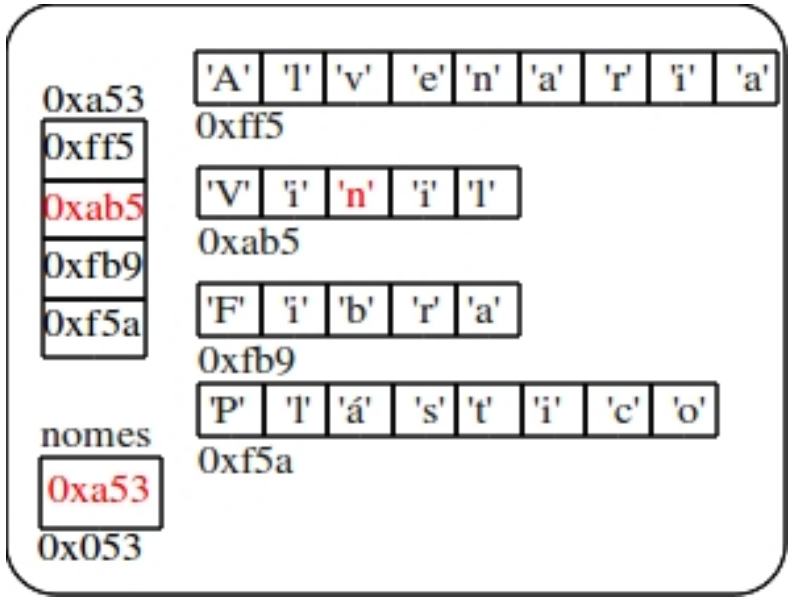
- E o que acontece na memória quando fazemos *nomes[1][2]*?
  - O conteúdo da variável *nomes* é lido
    - ★ O deslocamento é calculado ($A53 + 1 \times 8$)
    - ★ Sendo 8B o tamanho do endereço (64bits)



  - A região de memória correspondente a esse novo endereço é lida
    - ★ O deslocamento é calculado ($AB5 + 2 \times 2$ – sendo 2 o tamanho do char)

Arranjos de Arranjos

- E o que acontece na memória quando fazemos *nomes[1][2]*?
  - O conteúdo da variável *nomes* é lido
    - ★ O deslocamento é calculado ($A53 + 1 \times 8$)
    - ★ Sendo 8B o tamanho do endereço (64bits)



  - A região de memória correspondente a esse novo endereço é lida
    - ★ O deslocamento é calculado ($AB5 + 2 \times 2$ – sendo 2 o tamanho do char)
  - Finalmente, esse novo endereço é visitado, e o seu conteúdo lido

Arranjos de Arranjos

- Existe outro modo de inicializar um arranjo de arranjos

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

Arranjos de Arranjos

- Existe outro modo de inicializar um arranjo de arranjos

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};

static char[][] nomes = new char[4][9];
```

Arranjos de Arranjos

- Existe outro modo de inicializar um arranjo de arranjos
- A diferença, nesse caso, é que o segundo modo precisa ser inicializado com

```
nomes[0][0] = 'A';
nomes[0][1] = 'l';
nomes[0][2] = 'v';
...
```

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};

static char[][] nomes = new char[4][9];
```

Arranjos de Arranjos

- Existe outro modo de inicializar um arranjo de arranjos
- A diferença, nesse caso, é que o segundo modo precisa ser inicializado com

```
nomes[0][0] = 'A';
nomes[0][1] = 'l';
nomes[0][2] = 'v';
...
```

- Existe ainda um terceiro modo

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};

static char[][] nomes = new char[4][9];
```

Arranjos de Arranjos

- Existe outro modo de inicializar um arranjo de arranjos
- A diferença, nesse caso, é que o segundo modo precisa ser inicializado com

```
nomes[0][0] = 'A';
nomes[0][1] = 'l';
nomes[0][2] = 'v';
...
```

- Existe ainda um terceiro modo

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

```
static char[][] nomes = new char[4][9];
```

```
static char[][] nomes = new char[4][];
```

Arranjos de Arranjos

- Existe outro modo de inicializar um arranjo de arranjos
- A diferença, nesse caso, é que o segundo modo precisa ser inicializado com

```
nomes[0][0] = 'A';
nomes[0][1] = 'l';
nomes[0][2] = 'v';
...
```

- Existe ainda um terceiro modo
  - Mas é realmente chato de inicializar

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};

static char[][] nomes = new char[4][9];
```

```
static char[][] nomes = new char[4][];

char[] aux1 = {'A','l','v','e','n','a','r','i',
               'a'};
char[] aux2 =  {'V','i','n','i','l'};
...
nomes[0] = aux1;
nomes[1] = aux2;
...
```

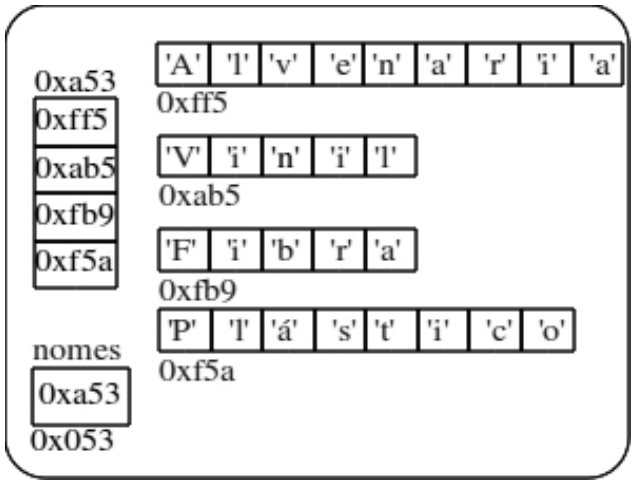
Comparando as inicializações

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

```
static char[][] nomes = new char[4][9];
```

Comparando as inicializações

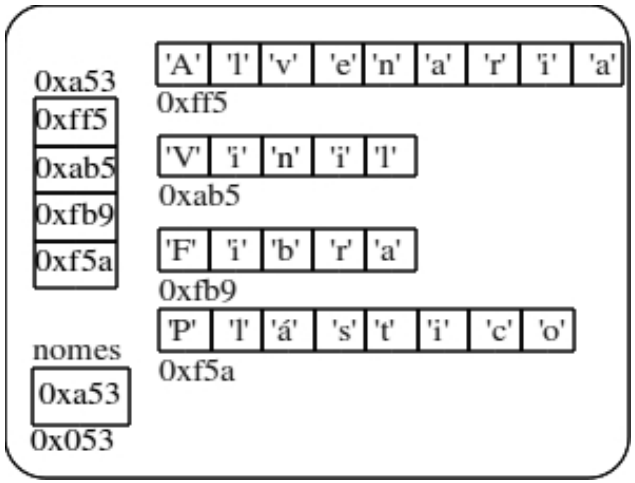
```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```




```
static char[][] nomes = new char[4][9];
```

Comparando as inicializações

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```

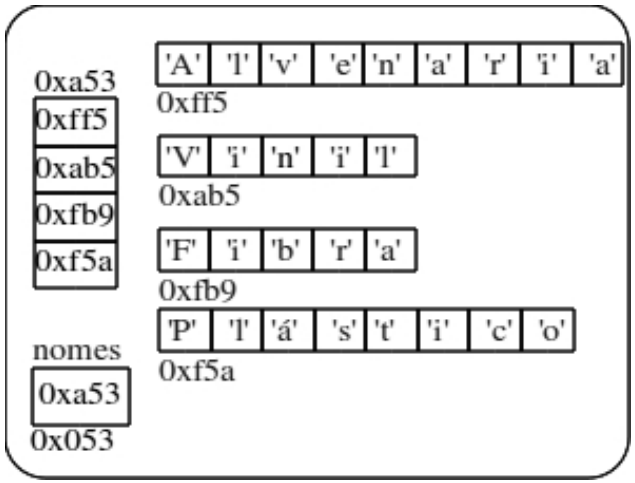


- Usado quando temos poucos dados e os conhecemos de antemão

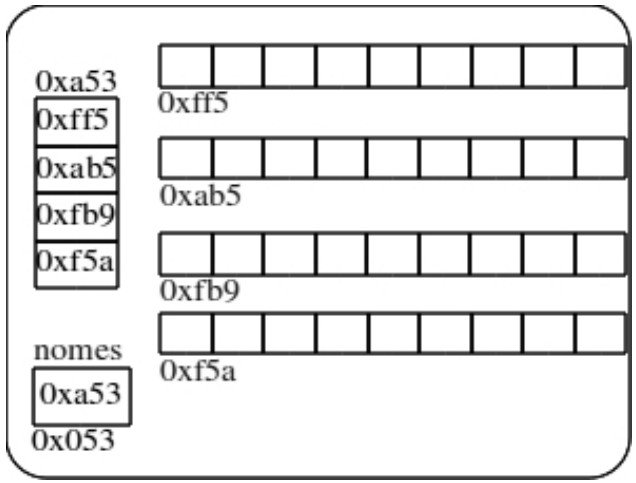
```
static char[][] nomes = new char[4][9];
```

Comparando as inicializações

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



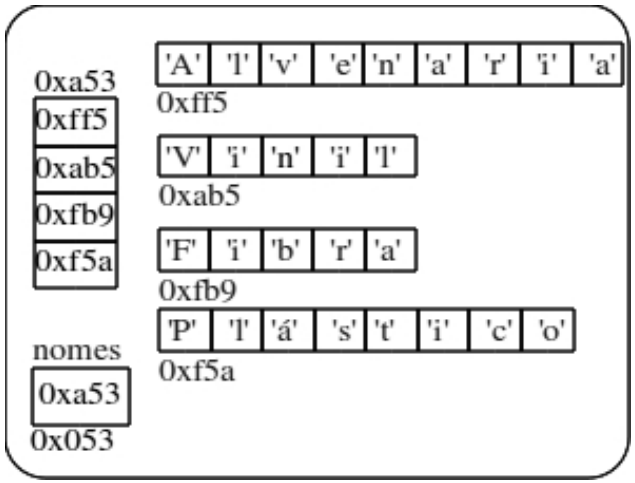

```
static char[][] nomes = new char[4][9];
```



- Usado quando temos poucos dados e os conhecemos de antemão

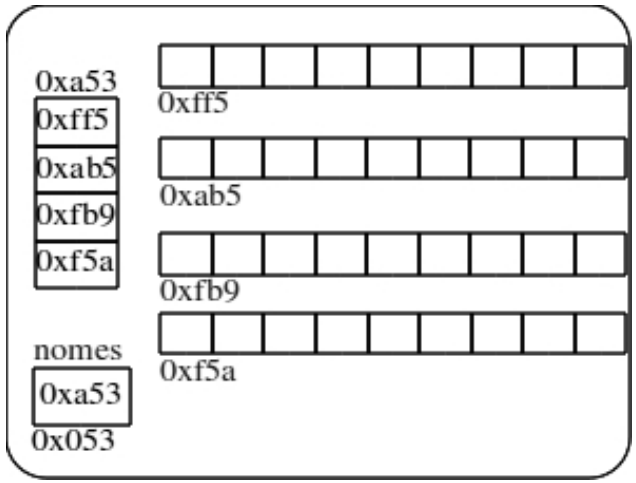
Comparando as inicializações

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



- Usado quando temos poucos dados e os conhecemos de antemão

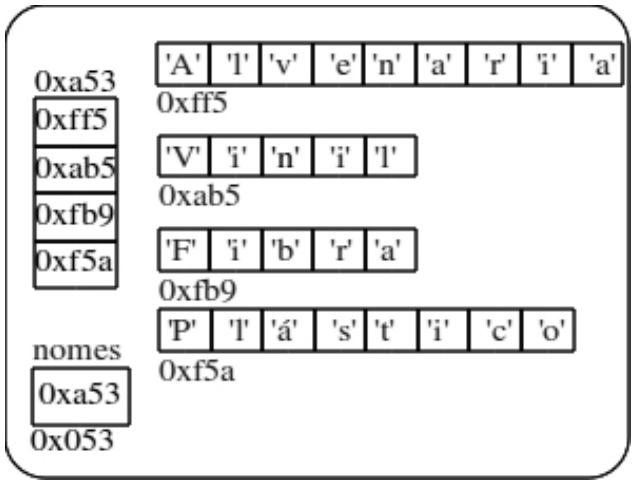
```
static char[][] nomes = new char[4][9];
```



- Usado quando temos muitos dados, ou não os conhecemos de antemão

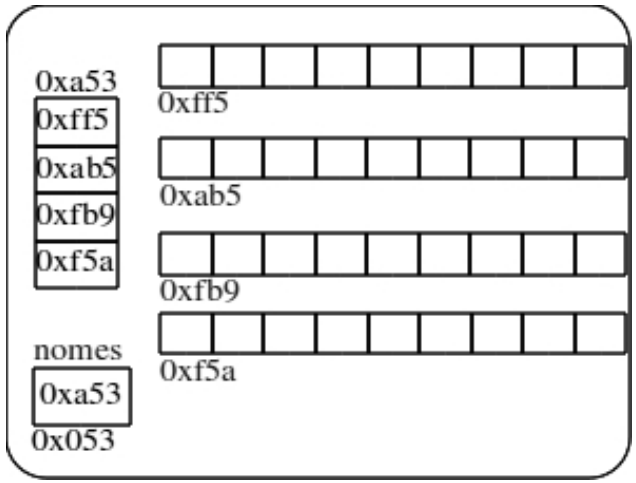
Comparando as inicializações

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



- Usado quando temos poucos dados e os conhecemos de antemão
- Poupa memória

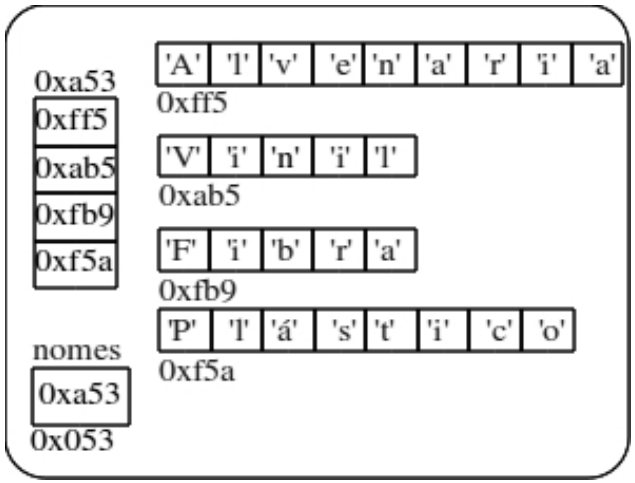
```
static char[][] nomes = new char[4][9];
```



- Usado quando temos muitos dados, ou não os conhecemos de antemão

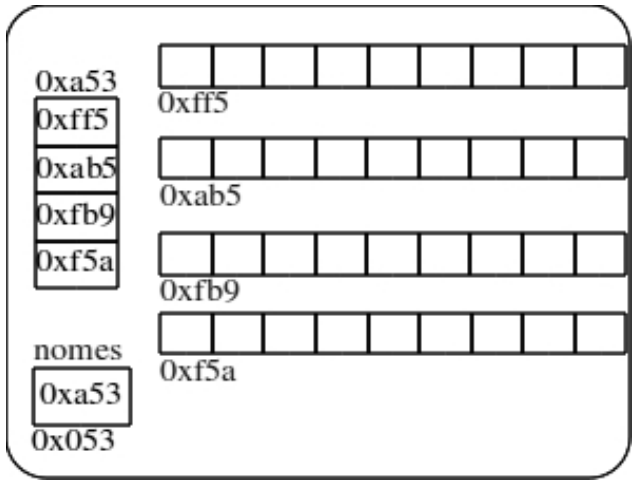
Comparando as inicializações

```
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                         {'V','i','n','i','l'},
                         {'F','i','b','r','a'},
                         {'P','l','á','s','t','i',
                          'c','o'}};
```



- Usado quando temos poucos dados e os conhecemos de antemão
- Poupa memória

```
static char[][] nomes = new char[4][9];
```



- Usado quando temos muitos dados, ou não os conhecemos de antemão
- Usa mais memória

Matrizes

- A idéia de arranjos de arranjos, em que todas as linhas têm igual tamanho, é conhecida como matriz

Matrizes

- A idéia de arranjos de arranjos, em que todas as linhas têm igual tamanho, é conhecida como matriz
  - Útil para uma gama de problemas

Matrizes

- A idéia de arranjos de arranjos, em que todas as linhas têm igual tamanho, é conhecida como matriz
  - Útil para uma gama de problemas
- Ex: Suponha que queiramos tabelar os preços das piscinas de 50, 100, 150 e 200 m^2, com os mais diversos materiais

Matrizes

- A idéia de arranjos de arranjos, em que todas as linhas têm igual tamanho, é conhecida como matriz
  - Útil para uma gama de problemas
- Ex: Suponha que queiramos tabelar os preços das piscinas de 50, 100, 150 e 200 m^2, com os mais diversos materiais
- Como fazer?

Matrizes

- A idéia de arranjos de arranjos, em que todas as linhas têm igual tamanho, é conhecida como matriz
  - Útil para uma gama de problemas
- Ex: Suponha que queiramos tabelar os preços das piscinas de 50, 100, 150 e 200 m^2, com os mais diversos materiais
- Como fazer?
  - Queremos uma tabela:

Matrizes

- A idéia de arranjos de arranjos, em que todas as linhas têm igual tamanho, é conhecida como matriz
  - Útil para uma gama de problemas
- Ex: Suponha que queiramos tabelar os preços das piscinas de 50, 100, 150 e 200 m^2, com os mais diversos materiais
- Como fazer?
  - Queremos uma tabela:

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Matrizes

- Como criamos essa tabela?

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Matrizes

- Como criamos essa tabela?
  - Como o arranjo de strings:

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];
}
```

Matrizes

- Como criamos essa tabela?
  - Como o arranjo de strings:
- E como armazenamos valores nela?

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];
}
```

Matrizes

- Como criamos essa tabela?
  - Como o arranjo de strings:
- E como armazenamos valores nela?
  - Com um método auxiliar:

Material	*Áreas (m^2)*			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

```
/*
    Carrega os valores das piscinas na matriz de área X
    material
*/
public static void carregaVal(double[][] m) {
    for (int i=0; i<m.length; i++) { // linhas (material)
        for (int j=50; j<=200; j+=50) { // colunas (áreas)
            m[i][j / 50 - 1] = precos[i] * j;
        }
    }
}
public static void main(String[] args) {
    double[][] valores = new double[4][4];
}
```

Matrizes

- Como criamos essa tabela?
  - Como o arranjo de strings:
- E como armazenamos valores nela?
  - Com um método auxiliar:
  - Note que as linhas usam o código natural dos materiais

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

```
/*
    Carrega os valores das piscinas na matriz de área X
    material
*/
public static void carregaVal(double[][] m) {
    for (int i=0; i<m.length; i++) { // linhas (material)
        for (int j=50; j<=200; j+=50) { // colunas (áreas)
            m[i][j / 50 - 1] = precos[i] * j;
        }
    }
}
public static void main(String[] args) {
    double[][] valores = new double[4][4];
}
```

Matrizes

- Como criamos essa tabela?
  - Como o arranjo de strings:
- E como armazenamos valores nela?
  - Com um método auxiliar:
  - Note que as linhas usam o código natural dos materiais
  - Já colunas precisam de um cálculo:

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

```
/*
    Carrega os valores das piscinas na matriz de área X
    material
*/
public static void carregaVal(double[][] m) {
    for (int i=0; i<m.length; i++) { // linhas (material)
        for (int j=50; j<=200; j+=50) { // colunas (áreas)
            m[i][j / 50 - 1] = precos[i] * j;
        }
    }
}
public static void main(String[] args) {
    double[][] valores = new double[4][4];
}
```

Matrizes

- Como criamos essa tabela?
  - Como o arranjo de strings:
- E como armazenamos valores nela?
  - Com um método auxiliar:
  - Note que as linhas usam o código natural dos materiais
  - Já colunas precisam de um cálculo:
- Podemos ainda usar os métodos já desenvolvidos

Material	*Áreas (m^2)*			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

```
/*
    Carrega os valores das piscinas na matriz de área X
    material
*/
public static void carregaVal(double[][] m) {
    for (int i=0; i<m.length; i++) { // linhas (material)
        for (int j=50; j<=200; j+=50) { // colunas (áreas)
            m[i][j / 50 - 1] = valorPiscina(j,i);
        }
    }
}
public static void main(String[] args) {
    double[][] valores = new double[4][4];
}
```

Matrizes

- E como mostramos os valores da matriz?

```
public static void main(String[] args) {
 double[][] valores = new double[4][4];

 carregaVal(valores);




}
```

Matrizes

- E como mostramos os valores da matriz?
  - Primeiro modo

```
public static void main(String[] args) {
 double[][] valores = new double[4][4];

 carregaVal(valores);



}
```

Matrizes

- E como mostramos os valores da matriz?
  - Primeiro modo
    - ★ Percorrendo os índices da matriz

```
public static void main(String[] args) {
 double[][] valores = new double[4][4];

 carregaVal(valores);

 for (int i=0; i<valores.length; i++) {
     for (int j=0; j<valores[i].length; j++)
         System.out.print(valores[i][j]+" ");
     System.out.println();
 }



}
```

Matrizes

- E como mostramos os valores da matriz?
  - Primeiro modo
    - ★ Percorrendo os índices da matriz
  - Segundo modo

```
public static void main(String[] args) {
 double[][] valores = new double[4][4];

 carregaVal(valores);

 for (int i=0; i<valores.length; i++) {
     for (int j=0; j<valores[i].length; j++)
         System.out.print(valores[i][j]+" ");
     System.out.println();
 }



}
```

Matrizes

- E como mostramos os valores da matriz?
  - Primeiro modo
    - ★ Percorrendo os índices da matriz
  - Segundo modo
    - ★ Percorrendo os valores da matriz

```
public static void main(String[] args) {
 double[][] valores = new double[4][4];

 carregaVal(valores);

 for (int i=0; i<valores.length; i++) {
     for (int j=0; j<valores[i].length; j++)
         System.out.print(valores[i][j]+" ");
     System.out.println();
 }

    for (double[] linha : valores) {
        for (double valor : linha)
            System.out.print(valor+" ");
        System.out.println();
    }
}
```

Matrizes

- E como mostramos os valores da matriz?
  - Primeiro modo
    - ★ Percorrendo os índices da matriz
  - Segundo modo
    - ★ Percorrendo os valores da matriz
    - ★ Por ser um arranjo de arranjos, o primeiro valor conterá um arranjo

```
public static void main(String[] args) {
 double[][] valores = new double[4][4];

 carregaVal(valores);

 for (int i=0; i<valores.length; i++) {
     for (int j=0; j<valores[i].length; j++)
         System.out.print(valores[i][j]+" ");
     System.out.println();
 }

    for (double[] linha : valores) {
        for (double valor : linha)
            System.out.print(valor+" ");
        System.out.println();
    }
}
```

Matrizes

- E como mostramos os valores da matriz?
  - Primeiro modo
    - ★ Percorrendo os índices da matriz
  - Segundo modo
    - ★ Percorrendo os valores da matriz
    - ★ Por ser um arranjo de arranjos, o primeiro valor conterá um arranjo
    - ★ Já o segundo, por correr em um dos arranjos finais, conterá o valor no arranjo

```
public static void main(String[] args) {
 double[][] valores = new double[4][4];

 carregaVal(valores);

 for (int i=0; i<valores.length; i++) {
     for (int j=0; j<valores[i].length; j++)
         System.out.print(valores[i][j]+" ");
     System.out.println();
 }

    for (double[] linha : valores) {
        for (double valor : linha)
            System.out.print(valor+" ");
        System.out.println();
    }
}
```

Matrizes

- Vamos usar essa matriz agora para:

Matrizes

- Vamos usar essa matriz agora para:
  - Dizer o preço de uma piscina de plástico com 150m^2

Matrizes

- Vamos usar essa matriz agora para:
  - Dizer o preço de uma piscina de plástico com 150m^2

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];

    carregaVal(valores);

    System.out.println("Piscina de plástico de
                    150m2: "+valores[PLASTICO][2]);



}
```

Matrizes

- Vamos usar essa matriz agora para:
  - Dizer o preço de uma piscina de plástico com 150m^2

```
Piscina de plástico de 150m2: 75000.0
```

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];

    carregaVal(valores);

    System.out.println("Piscina de plástico de
                    150m2: "+valores[PLASTICO][2]);



}
```

Matrizes

- Vamos usar essa matriz agora para:
  - Dizer o preço de uma piscina de plástico com 150m^2

    ```
    Piscina de plástico de 150m2: 75000.0
    ```

  - Dizer o preço médio das piscinas de plástico

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];

    carregaVal(valores);

    System.out.println("Piscina de plástico de
                    150m2: "+valores[PLASTICO][2]);



}
```

Matrizes

- Vamos usar essa matriz agora para:
  - Dizer o preço de uma piscina de plástico com 150m^2

    ```
    Piscina de plástico de 150m2: 75000.0
    ```

  - Dizer o preço médio das piscinas de plástico

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];

    carregaVal(valores);

    System.out.println("Piscina de plástico de
                    150m2: "+valores[PLASTICO][2]);


    double media = 0;
    for(double valor : valores[PLASTICO]) {
        media += valor;
    }
    media /= valores[PLASTICO].length;
    System.out.println("Média: "+media);
}
```

Matrizes

- Vamos usar essa matriz agora para:
  - Dizer o preço de uma piscina de plástico com 150m^2

    ```
    Piscina de plástico de 150m2: 75000.0
    ```

  - Dizer o preço médio das piscinas de plástico

    ```
    Média: 62500.0
    ```

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];

    carregaVal(valores);

    System.out.println("Piscina de plástico de
                    150m2: "+valores[PLASTICO][2]);


    double media = 0;
    for(double valor : valores[PLASTICO]) {
        media += valor;
    }
    media /= valores[PLASTICO].length;
    System.out.println("Média: "+media);
}
```

Matrizes

- Vamos usar essa matriz agora para:
  - Dizer o preço de uma piscina de plástico com 150m^2

    ```
    Piscina de plástico de 150m2: 75000.0
    ```

  - Dizer o preço médio das piscinas de plástico

    ```
    Média: 62500.0
    ```

- Alternativas para o preço médio:

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];

    carregaVal(valores);

    System.out.println("Piscina de plástico de
                    150m2: "+valores[PLASTICO][2]);


    double media = 0;
    for(double valor : valores[PLASTICO]) {
        media += valor;
    }
    media /= valores[PLASTICO].length;
    System.out.println("Média: "+media);
}
```

Matrizes

- Vamos usar essa matriz agora para:
  - Dizer o preço de uma piscina de plástico com 150m^2

    ```
    Piscina de plástico de 150m2: 75000.0
    ```

  - Dizer o preço médio das piscinas de plástico

    ```
    Média: 62500.0
    ```

- Alternativas para o preço médio:

```
double media = 0;
for(double valor : valores[PLASTICO]) {
    media += valor;
}
media /= valores[PLASTICO].length;
System.out.println("Média: "+media);
```

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];

    carregaVal(valores);

    System.out.println("Piscina de plástico de
                    150m2: "+valores[PLASTICO][2]);


    double media = 0;
    for(double valor : valores[PLASTICO]) {
        media += valor;
    }
    media /= valores[PLASTICO].length;
    System.out.println("Média: "+media);
}
```

Matrizes

- Vamos usar essa matriz agora para:
  - Dizer o preço de uma piscina de plástico com 150m^2

    ```
    Piscina de plástico de 150m2: 75000.0
    ```

  - Dizer o preço médio das piscinas de plástico

    ```
    Média: 62500.0
    ```

- Alternativas para o preço médio:

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];

    carregaVal(valores);

    System.out.println("Piscina de plástico de
                    150m2: "+valores[PLASTICO][2]);


    double media = 0;
    for(double valor : valores[PLASTICO]) {
        media += valor;
    }
    media /= valores[PLASTICO].length;
    System.out.println("Média: "+media);
}
```

```
double media = 0;
for(double valor : valores[PLASTICO]) {
    media += valor;
}
media /= valores[PLASTICO].length;
System.out.println("Média: "+media);
```

```
double media = 0;
for (int i=0; i<valores[PLASTICO].length; i++) {
    media += valores[PLASTICO][i];
}
media /= valores[PLASTICO].length;
System.out.println("Média: "+media);
```

Matrizes

- Já temos a tabela com os valores

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Matrizes

- Já temos a tabela com os valores
- Suponha agora que existam descontos nos materiais:

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Matrizes

- Já temos a tabela com os valores
- Suponha agora que existam descontos nos materiais:
  - Alvenaria está com 20% acima de 100m^2

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Matrizes

- Já temos a tabela com os valores
- Suponha agora que existam descontos nos materiais:
  - Alvenaria está com 20% acima de 100m^2
  - Vinil tem 5% para piscinas de até 100m^2, 10% para as de 150m^2 e 15% para as acima disso

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Matrizes

- Já temos a tabela com os valores
- Suponha agora que existam descontos nos materiais:
  - Alvenaria está com 20% acima de 100m^2
  - Vinil tem 5% para piscinas de até 100m^2, 10% para as de 150m^2 e 15% para as acima disso
  - Fibra tem descontos de 2%, 4%, 8% e 16%, respectivamente

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Matrizes

- Já temos a tabela com os valores
- Suponha agora que existam descontos nos materiais:
  - Alvenaria está com 20% acima de 100m^2
  - Vinil tem 5% para piscinas de até 100m^2, 10% para as de 150m^2 e 15% para as acima disso
  - Fibra tem descontos de 2%, 4%, 8% e 16%, respectivamente
  - Plástico tem desconto de 5% apenas nas piscinas com 200m^2

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Matrizes

- Já temos a tabela com os valores

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

- Suponha agora que existam descontos nos materiais:
  - Alvenaria está com 20% acima de 100m^2
  - Vinil tem 5% para piscinas de até 100m^2, 10% para as de 150m^2 e 15% para as acima disso
  - Fibra tem descontos de 2%, 4%, 8% e 16%, respectivamente
  - Plástico tem desconto de 5% apenas nas piscinas com 200m^2
- Como tabelar isso?

Matrizes

- Já temos a tabela com os valores

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

- Suponha agora que existam descontos nos materiais:
  - Alvenaria está com 20% acima de 100m^2
  - Vinil tem 5% para piscinas de até 100m^2, 10% para as de 150m^2 e 15% para as acima disso
  - Fibra tem descontos de 2%, 4%, 8% e 16%, respectivamente
  - Plástico tem desconto de 5% apenas nas piscinas com 200m^2
- Como tabelar isso?

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	0	0	0.2	0.2
Vinil	0.05	0.05	0.1	0.15
Fibra	0.02	0.04	0.08	0.16
Plástico	0	0	0	0.05

Matrizes

- Gostaríamos de recalcular a tabela de valores

Preços

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Descontos

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	0	0	0.2	0.2
Vinil	0.05	0.05	0.1	0.15
Fibra	0.02	0.04	0.08	0.16
Plástico	0	0	0	0.05

Matrizes

- Gostaríamos de recalcular a tabela de valores
  - Uma tabela de preço final

Preços

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Descontos

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	0	0	0.2	0.2
Vinil	0.05	0.05	0.1	0.15
Fibra	0.02	0.04	0.08	0.16
Plástico	0	0	0	0.05

Matrizes

- Gostaríamos de recalcular a tabela de valores
  - Uma tabela de preço final
- Como?

Preços

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Descontos

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	0	0	0.2	0.2
Vinil	0.05	0.05	0.1	0.15
Fibra	0.02	0.04	0.08	0.16
Plástico	0	0	0	0.05

Matrizes

- Gostaríamos de recalcular a tabela de valores
  - Uma tabela de preço final
- Como?
  - Crie uma tabela equivalente, vazia

Preços

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Descontos

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	0	0	0.2	0.2
Vinil	0.05	0.05	0.1	0.15
Fibra	0.02	0.04	0.08	0.16
Plástico	0	0	0	0.05

Matrizes

- Gostaríamos de recalcular a tabela de valores
  - Uma tabela de preço final
- Como?
  - Crie uma tabela equivalente, vazia

Preços

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Descontos

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	0	0	0.2	0.2
Vinil	0.05	0.05	0.1	0.15
Fibra	0.02	0.04	0.08	0.16
Plástico	0	0	0	0.05

Final

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria				
Vinil				
Fibra				
Plástico				

Matrizes

- Gostaríamos de recalcular a tabela de valores
  - Uma tabela de preço final
- Como?
  - Crie uma tabela equivalente, vazia
  - Nela, o valor final de cada célula será
    $preco - preco \times desconto$, ou
    $preco \times (1 - desconto)$

Preços

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Descontos

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria	0	0	0.2	0.2
Vinil	0.05	0.05	0.1	0.15
Fibra	0.02	0.04	0.08	0.16
Plástico	0	0	0	0.05

Final

	Áreas (m^2)			
Material	50	100	150	200
Alvenaria				
Vinil				
Fibra				
Plástico				

Matrizes

- Gostaríamos de recalcular a tabela de valores
  - Uma tabela de preço final
- Como?
  - Crie uma tabela equivalente, vazia
  - Nela, o valor final de cada célula será
    $preco - preco \times desconto$, ou
    $preco \times (1 - desconto)$
  - Ou seja:

```
pFinal[i][j] = preco[i][j] *
        (1 - desconto[i][j]);
```

Preços

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	75000	150000	225000	300000
Vinil	55000	110000	165000	220000
Fibra	37500	75000	112500	150000
Plástico	25000	50000	75000	100000

Descontos

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria	0	0	0.2	0.2
Vinil	0.05	0.05	0.1	0.15
Fibra	0.02	0.04	0.08	0.16
Plástico	0	0	0	0.05

Final

Material	*Áreas* (m^2)			
	50	100	150	200
Alvenaria Vinil Fibra Plástico				

Matrizes

- Criando as tabelas

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];
    double[][] descontos = {{0,0,0.2,0.2},
                            {0.05,0.05,0.1,0.15},
                            {0.02,0.04,0.08,0.16},
                            {0,0,0,0.05}};
    double[][] pFinal;



}
```

Matrizes

- Criando as tabelas
  - Note os modos como podemos criar

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];
    double[][] descontos = {{0,0,0.2,0.2},
                            {0.05,0.05,0.1,0.15},
                            {0.02,0.04,0.08,0.16},
                            {0,0,0,0.05}};
    double[][] pFinal;



}
```

Matrizes

- Criando as tabelas
  - Note os modos como podemos criar
  - Aloca somente espaço para *pFinal* – um endereço

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];
    double[][] descontos = {{0,0,0.2,0.2},
                            {0.05,0.05,0.1,0.15},
                            {0.02,0.04,0.08,0.16},
                            {0,0,0,0.05}};
    double[][] pFinal;



}
```

Matrizes

- Criando as tabelas
  - Note os modos como podemos criar
  - Aloca somente espaço para *pFinal* – um endereço
- Carregando valores iniciais

```
public static void main(String[] args) {
    double[][] valores = new double[4][4];
    double[][] descontos = {{0,0,0.2,0.2},
                            {0.05,0.05,0.1,0.15},
                            {0.02,0.04,0.08,0.16},
                            {0,0,0,0.05}};
    double[][] pFinal;

    carregaVal(valores);


}
```

Matrizes

- Criando as tabelas
  - Note os modos como podemos criar
  - Aloca somente espaço para *pFinal* – um endereço
- Carregando valores iniciais
- Construindo a tabela de valores finais

```
/*
    Retorna matriz com os preços finais.
    Parâmetros:
        val - matriz de valores
        desc - matriz de descontos
*/
public static double[][] calculaFinal(double[][]
                          val, double[][] desc) {
    double[][] saida = new double[val.length]
                                 [val[0].length];

    for (int i=0; i<saida.length; i++) {
        for (int j=0; j<saida[0].length; j++) {
            saida[i][j] = val[i][j] * (1-desc[i][j]);
        }
    }
    return(saida);
}

public static void main(String[] args) {
    double[][] valores = new double[4][4];
    double[][] descontos = {{0,0,0.2,0.2},
                            {0.05,0.05,0.1,0.15},
                            {0.02,0.04,0.08,0.16},
                            {0,0,0,0.05}};
    double[][] pFinal;

    carregaVal(valores);

    pFinal = calculaFinal(valores, descontos);
}
```

Matrizes

- Criando as tabelas
  - Note os modos como podemos criar
  - Aloca somente espaço para *pFinal* – um endereço
- Carregando valores iniciais
- Construindo a tabela de valores finais
  - Recebe matrizes como parâmetros

```
/*
    Retorna matriz com os preços finais.
    Parâmetros:
        val - matriz de valores
        desc - matriz de descontos
*/
public static double[][] calculaFinal(double[][]
                          val, double[][] desc) {
    double[][] saida = new double[val.length]
                                 [val[0].length];

    for (int i=0; i<saida.length; i++) {
        for (int j=0; j<saida[0].length; j++) {
            saida[i][j] = val[i][j] * (1-desc[i][j]);
        }
    }
    return(saida);
}

public static void main(String[] args) {
    double[][] valores = new double[4][4];
    double[][] descontos = {{0,0,0.2,0.2},
                            {0.05,0.05,0.1,0.15},
                            {0.02,0.04,0.08,0.16},
                            {0,0,0,0.05}};
    double[][] pFinal;

    carregaVal(valores);

    pFinal = calculaFinal(valores, descontos);
}
```

Matrizes

- Criando as tabelas
  - Note os modos como podemos criar
  - Aloca somente espaço para *pFinal* – um endereço
- Carregando valores iniciais
- Construindo a tabela de valores finais
  - Recebe matrizes como parâmetros
  - Cria a auxiliar com base nos parâmetros

```
/*
    Retorna matriz com os preços finais.
    Parâmetros:
        val - matriz de valores
        desc - matriz de descontos
*/
public static double[][] calculaFinal(double[][]
                            val, double[][] desc) {
    double[][] saida = new double[val.length]
                                    [val[0].length];

    for (int i=0; i<saida.length; i++) {
        for (int j=0; j<saida[0].length; j++) {
            saida[i][j] = val[i][j] * (1-desc[i][j]);
        }
    }
    return(saida);
}

public static void main(String[] args) {
    double[][] valores = new double[4][4];
    double[][] descontos = {{0,0,0.2,0.2},
                            {0.05,0.05,0.1,0.15},
                            {0.02,0.04,0.08,0.16},
                            {0,0,0,0.05}};
    double[][] pFinal;

    carregaVal(valores);

    pFinal = calculaFinal(valores, descontos);
}
```

Matrizes

- Criando as tabelas
  - Note os modos como podemos criar
  - Aloca somente espaço para *pFinal* – um endereço
- Carregando valores iniciais
- Construindo a tabela de valores finais
  - Recebe matrizes como parâmetros
  - Cria a auxiliar com base nos parâmetros
  - Retorna o endereço da matriz

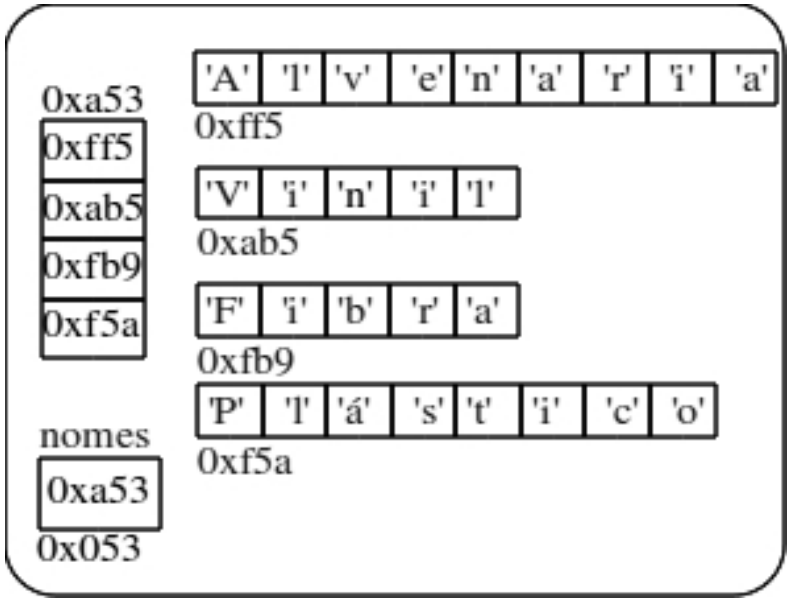
```
/*
    Retorna matriz com os preços finais.
    Parâmetros:
        val - matriz de valores
        desc - matriz de descontos
*/
public static double[][] calculaFinal(double[][]
                            val, double[][] desc) {
    double[][] saida = new double[val.length]
                                    [val[0].length];

    for (int i=0; i<saida.length; i++) {
        for (int j=0; j<saida[0].length; j++) {
            saida[i][j] = val[i][j] * (1-desc[i][j]);
        }
    }
    return(saida);
}

public static void main(String[] args) {
    double[][] valores = new double[4][4];
    double[][] descontos = {{0,0,0.2,0.2},
                            {0.05,0.05,0.1,0.15},
                            {0.02,0.04,0.08,0.16},
                            {0,0,0,0.05}};
    double[][] pFinal;

    carregaVal(valores);

    pFinal = calculaFinal(valores, descontos);
}
```

Matrizes e Memória

- Como *nomes* está na memória?

Matrizes e Memória

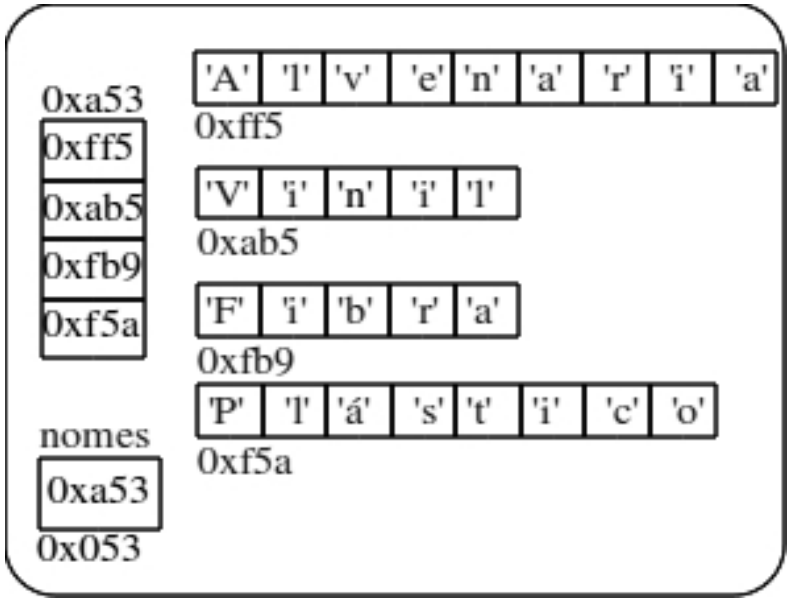
- Como *nomes* está na memória?



Matrizes e Memória

- Como *nomes* está na memória?
- Queremos fazer

```
char [][] nomes2;

nomes2 = nomes;
```

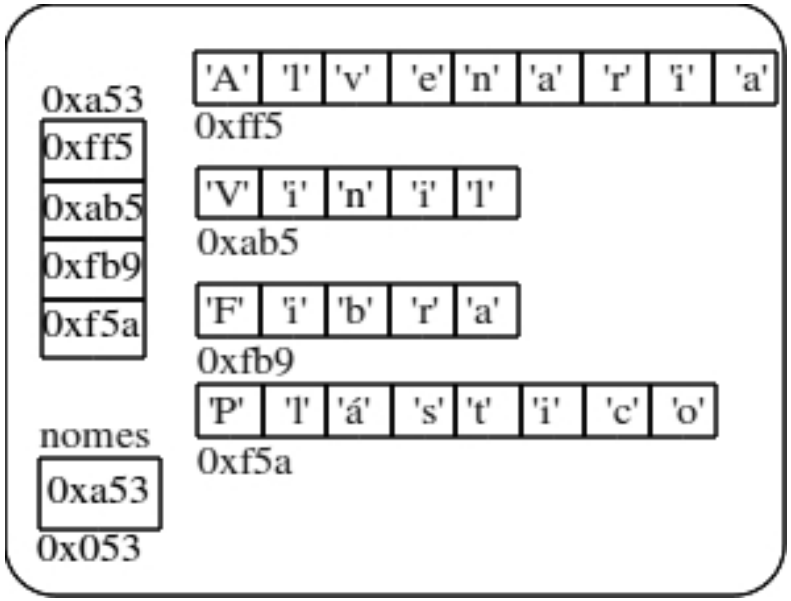


Matrizes e Memória

- Como *nomes* está na memória?
- Queremos fazer

```
char [][] nomes2;

nomes2 = nomes;
```

- O que acontece ao declararmos *nomes2*?

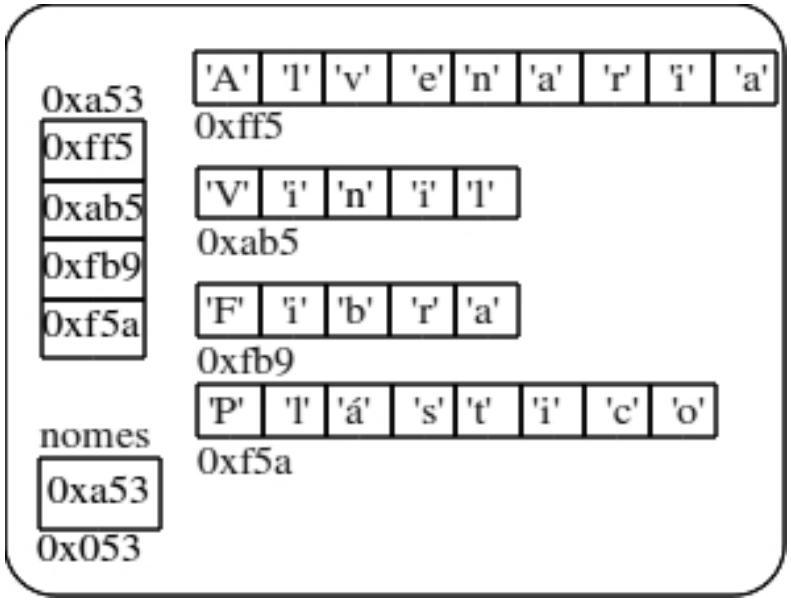


Matrizes e Memória

- Como *nomes* está na memória?
- Queremos fazer

```
char [][] nomes2;

nomes2 = nomes;
```

- O que acontece ao declararmos *nomes2*?
  - Alocamos espaço suficiente para caber um endereço

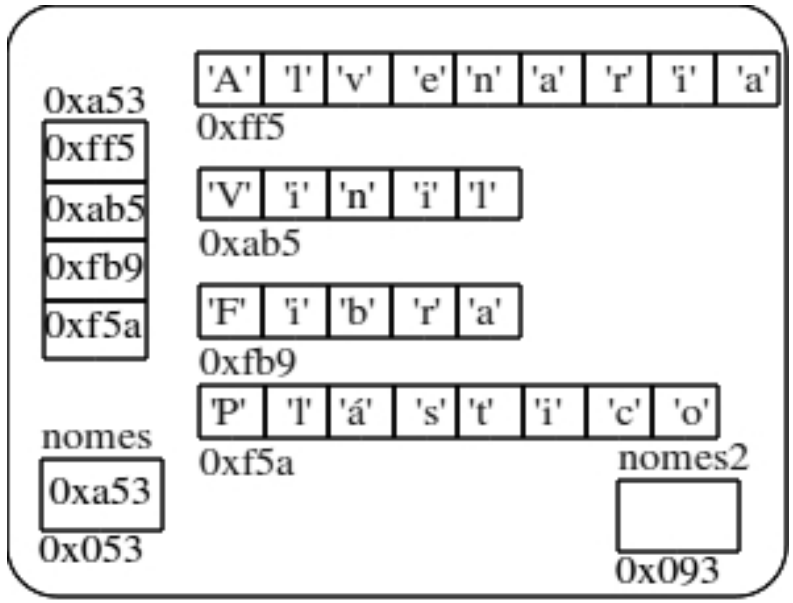


Matrizes e Memória

- Como *nomes* está na memória?
- Queremos fazer

```
char [][] nomes2;

nomes2 = nomes;
```

- O que acontece ao declararmos *nomes2*?
  - Alocamos espaço suficiente para caber um endereço

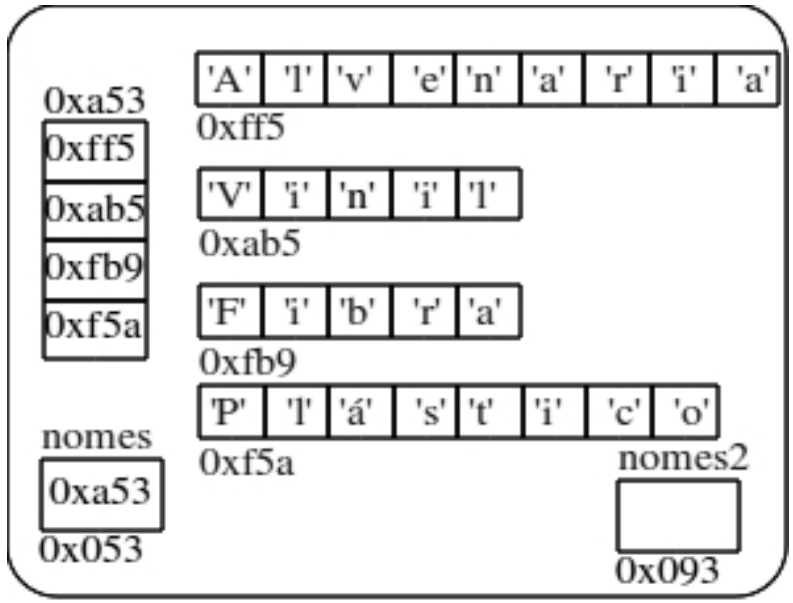


Matrizes e Memória

- Como *nomes* está na memória?
- Queremos fazer

```
char [][] nomes2;

nomes2 = nomes;
```

- O que acontece ao declararmos *nomes2*?
  - Alocamos espaço suficiente para caber um endereço



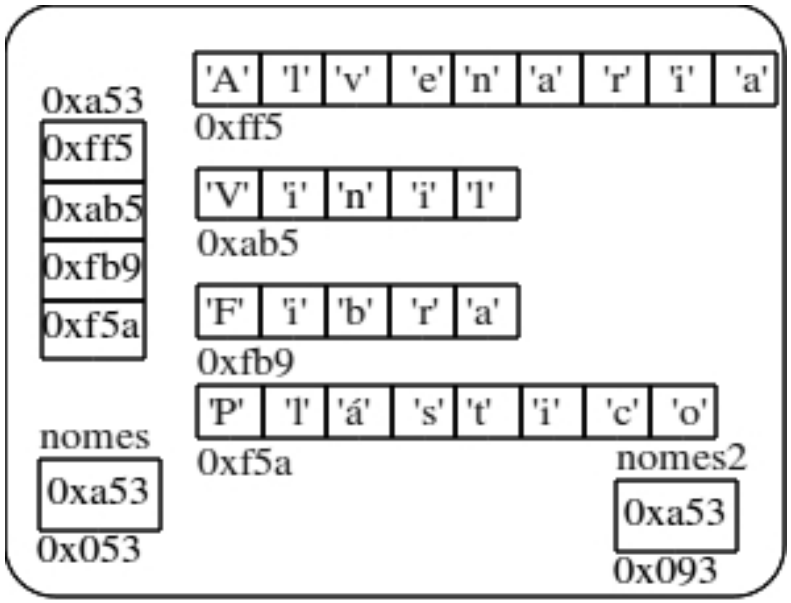
- Ao fazermos a atribuição copiamos o conteúdo de *nomes* (ou seja, o endereço que lá está) para *nomes2*

Matrizes e Memória

- Como *nomes* está na memória?
- Queremos fazer

```
char [][] nomes2;

nomes2 = nomes;
```

- O que acontece ao declararmos *nomes2*?
  - Alocamos espaço suficiente para caber um endereço



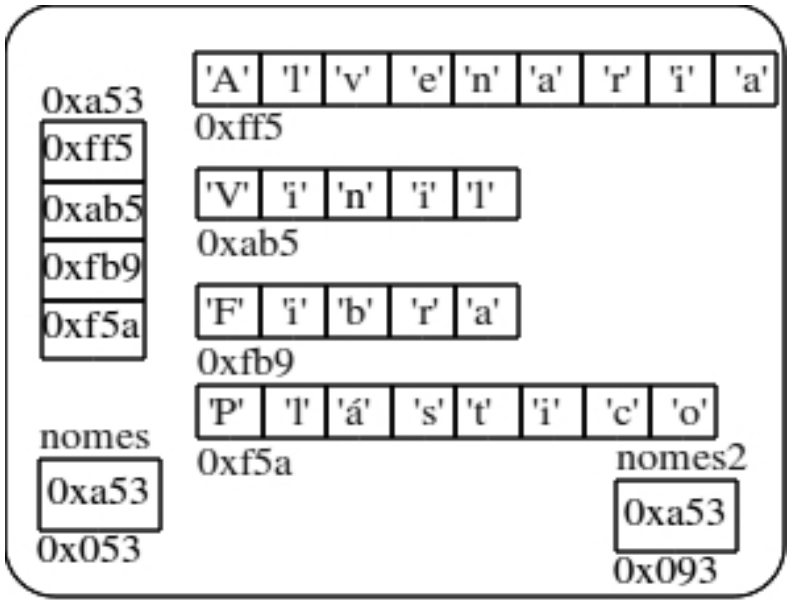
- Ao fazermos a atribuição copiamos o conteúdo de *nomes* (ou seja, o endereço que lá está) para *nomes2*

Matrizes e Memória

- Como *nomes* está na memória?
- Queremos fazer

```
char [][] nomes2;

nomes2 = nomes;
```

- O que acontece ao declararmos *nomes2*?
  - Alocamos espaço suficiente para caber um endereço



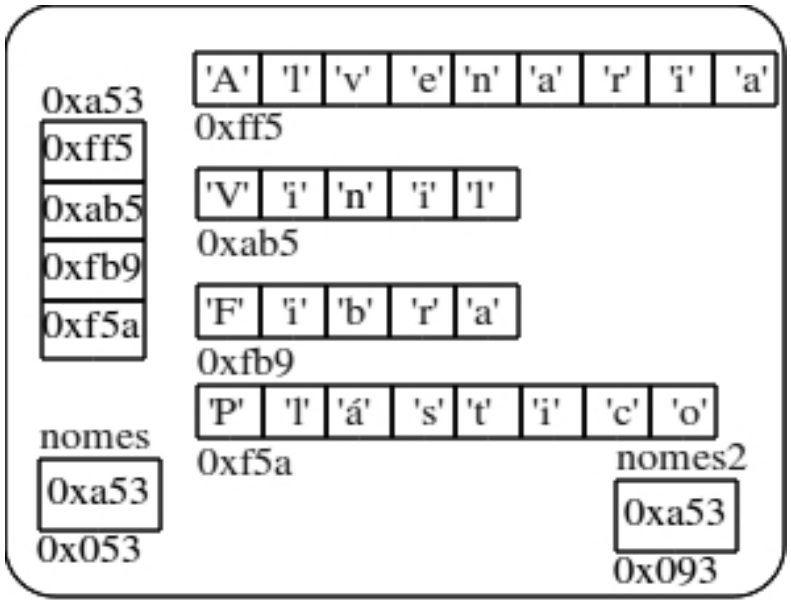
- Ao fazermos a atribuição copiamos o conteúdo de *nomes* (ou seja, o endereço que lá está) para *nomes2*
  - E tanto *nomes* quanto *nomes2* referenciam a mesma estrutura na memória

Matrizes e Memória

- Como *nomes* está na memória?
- Queremos fazer

```
char [][] nomes2;

nomes2 = nomes;
```

- O que acontece ao declararmos *nomes2*?
  - Alocamos espaço suficiente para caber um endereço



- Ao fazermos a atribuição copiamos o conteúdo de *nomes* (ou seja, o endereço que lá está) para *nomes2*
  - E tanto *nomes* quanto *nomes2* referenciam a mesma estrutura na memória
  - Então, para copiarmos uma matriz em outra, teremos que fazer elemento por elemento

Matrizes e Memória

- Considere os códigos:

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

Matrizes e Memória

- Considere os códigos:

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

- Qual será mais rápido?

Matrizes e Memória

- Considere os códigos:

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

- Qual será mais rápido?
  - Média de tempo em 50 repetições

Matrizes e Memória

- Considere os códigos:

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

- Qual será mais rápido?
  - Média de tempo em 50 repetições
- 10ms

Matrizes e Memória

- Considere os códigos:

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

- Qual será mais rápido?
  - Média de tempo em 50 repetições

- 10ms
- 38ms

Matrizes e Memória

- Por que isso acontece?

Matrizes e Memória

- Por que isso acontece?
  - Por conta da cache

Matrizes e Memória

- Por que isso acontece?
  - Por conta da cache

Cache

Circuito de memória que, juntamente com a RAM (e outros) formam a memória primária do computador

Matrizes e Memória

- Por que isso acontece?
  - Por conta da cache

Cache

Circuito de memória que, juntamente com a RAM (e outros) formam a memória primária do computador

- Características:

Matrizes e Memória

- Por que isso acontece?
  - Por conta da cache

Cache

Circuito de memória que, juntamente com a RAM (e outros) formam a memória primária do computador

- Características:
  - Rapidez (por volta de 2ns) – 5 vezes mais rápido que a RAM

Matrizes e Memória

- Por que isso acontece?
  - Por conta da cache

Cache

Circuito de memória que, juntamente com a RAM (e outros) formam a memória primária do computador

- Características:
  - Rapidez (por volta de 2ns) – 5 vezes mais rápido que a RAM
  - Caro

Matrizes e Memória

- Por que isso acontece?
  - Por conta da cache

Cache

Circuito de memória que, juntamente com a RAM (e outros) formam a memória primária do computador

- Características:
  - Rapidez (por volta de 2ns) – 5 vezes mais rápido que a RAM
  - Caro
- Funcionamento:

Matrizes e Memória

- Por que isso acontece?
  - Por conta da cache

Cache

Circuito de memória que, juntamente com a RAM (e outros) formam a memória primária do computador

- Características:
  - Rapidez (por volta de 2ns) – 5 vezes mais rápido que a RAM
  - Caro
- Funcionamento:
  - Quando um endereço de memória é buscado, o computador verifica primeiro se o conteúdo está na cache

Matrizes e Memória

- Por que isso acontece?
  - Por conta da cache

Cache

Circuito de memória que, juntamente com a RAM (e outros) formam a memória primária do computador

- Características:
  - Rapidez (por volta de 2ns) – 5 vezes mais rápido que a RAM
  - Caro
- Funcionamento:
  - Quando um endereço de memória é buscado, o computador verifica primeiro se o conteúdo está na cache
  - Se não estiver, ele é buscado na RAM, sendo então trazido, juntamente com alguns vizinhos, para a cache

Matrizes e Memória

- Por que isso acontece?
  - Por conta da cache

Cache

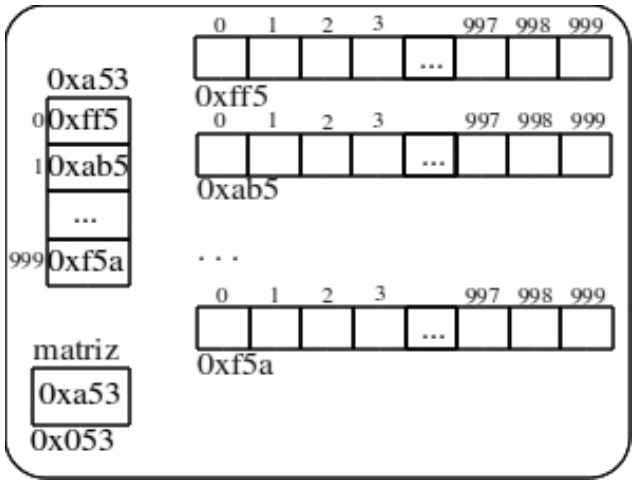
Circuito de memória que, juntamente com a RAM (e outros) formam a memória primária do computador

- Características:
  - Rapidez (por volta de 2ns) – 5 vezes mais rápido que a RAM
  - Caro
- Funcionamento:
  - Quando um endereço de memória é buscado, o computador verifica primeiro se o conteúdo está na cache
  - Se não estiver, ele é buscado na RAM, sendo então trazido, juntamente com alguns vizinhos, para a cache
  - Assim, o próximo acesso ao mesmo endereço (ou algum vizinho) será mais rápido, pois ele estará na cache

Matrizes e Memória

- Agora vejamos como cada arranjo é corrido na memória (supondo que o arranjo em 0xa53 está no cache):

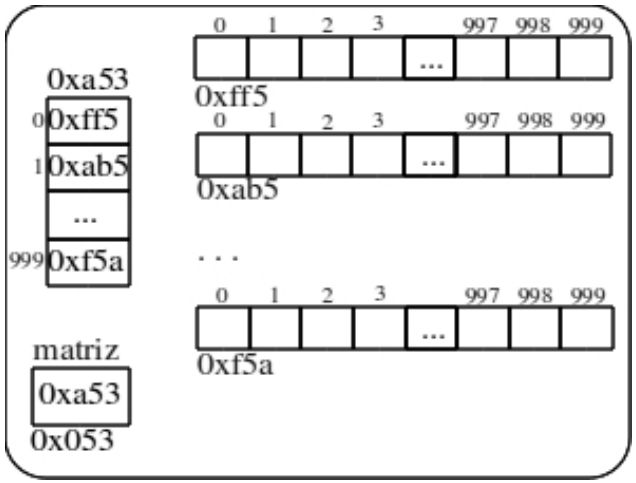
```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```



Matrizes e Memória

- Agora vejamos como cada arranjo é corrido na memória (supondo que o arranjo em 0xa53 está no cache):

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

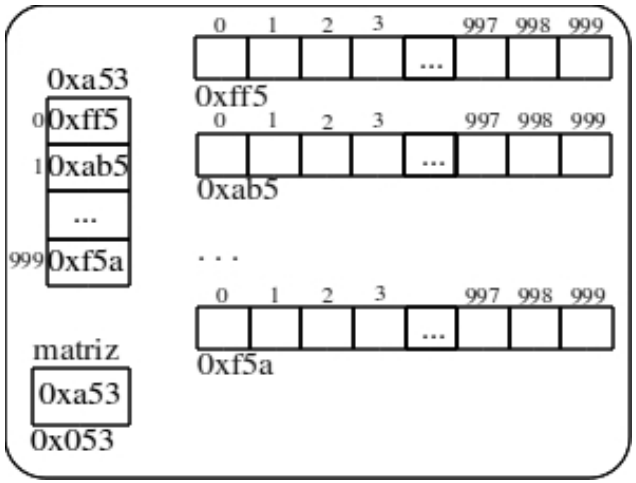


- O primeiro elemento desse vetor é buscado no cache

Matrizes e Memória

- Agora vejamos como cada arranjo é corrido na memória (supondo que o arranjo em 0xa53 está no cache):

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

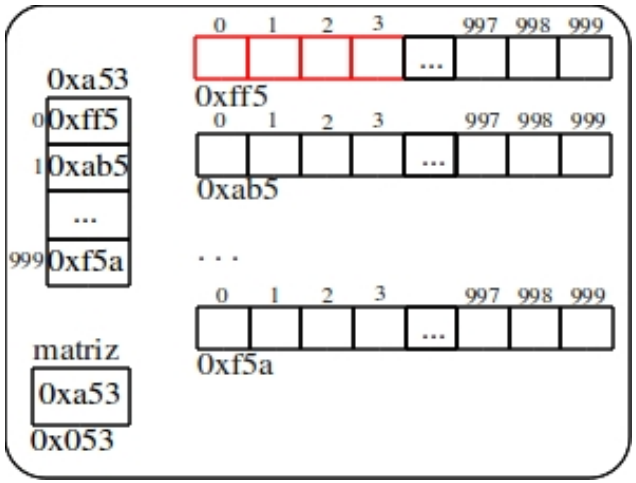


- O primeiro elemento desse vetor é buscado no cache – não é encontrado

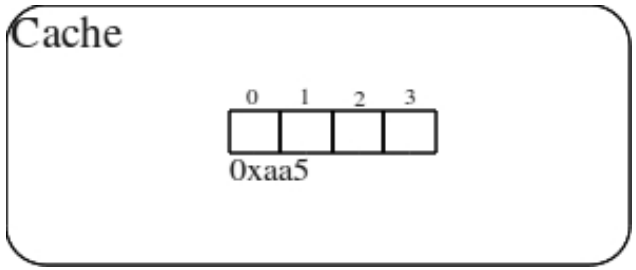
Matrizes e Memória

- Agora vejamos como cada arranjo é corrido na memória (supondo que o arranjo em 0xa53 está no cache):

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```



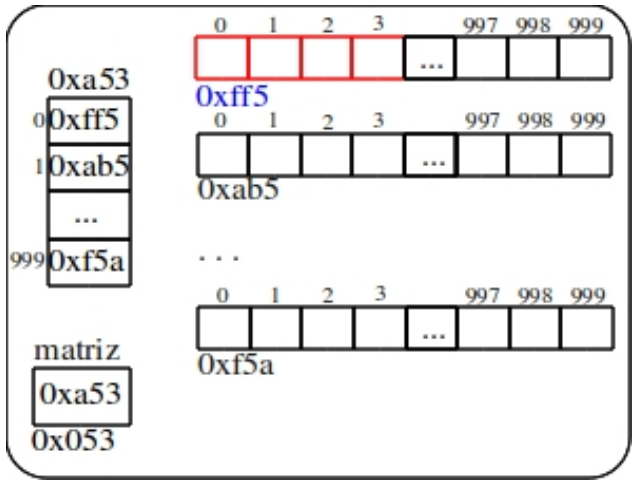
- O primeiro elemento desse vetor é buscado no cache – não é encontrado
- Então é buscado na memória, sendo alguns vizinhos trazidos para o cache

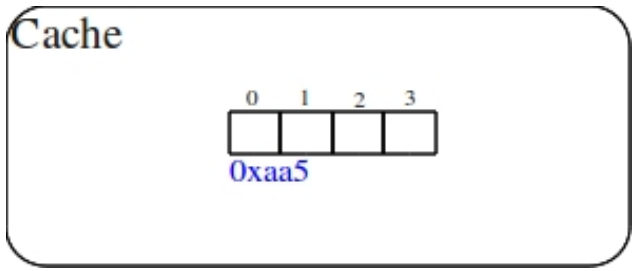


Matrizes e Memória

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

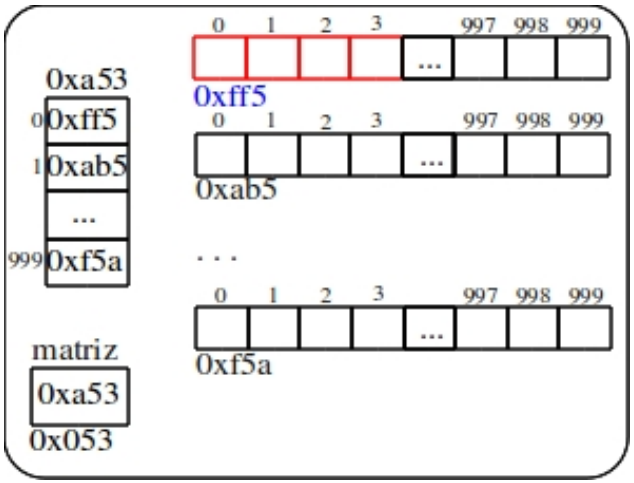
- Mudou o endereço ao ir para o cache!



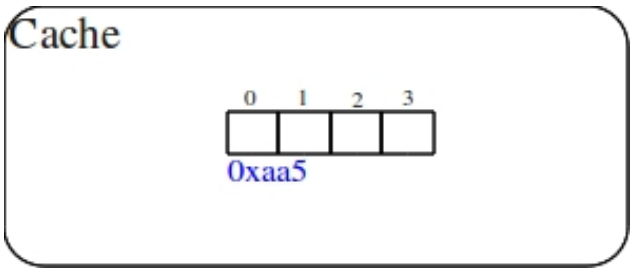


Matrizes e Memória

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```



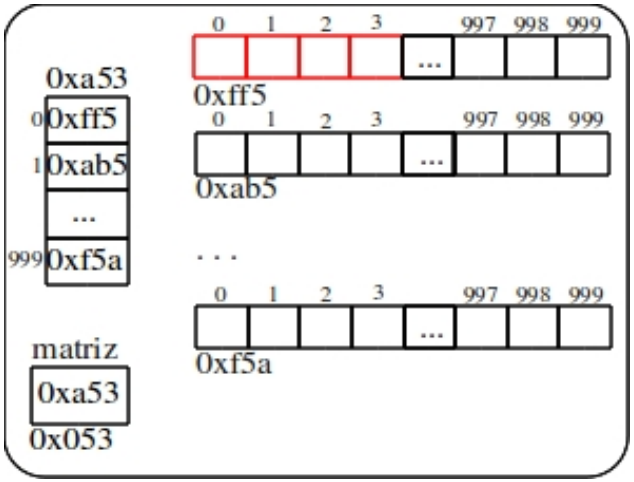
- Mudou o endereço ao ir para o cache!
  - O S.O. cuida para que isso seja trasparente ao programa

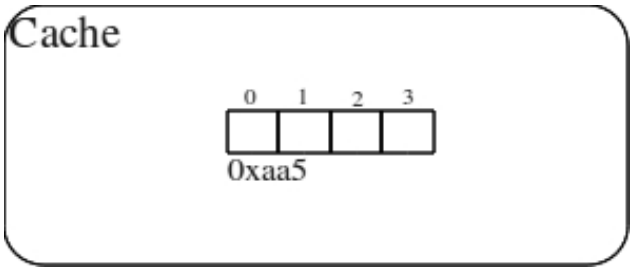


Matrizes e Memória

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

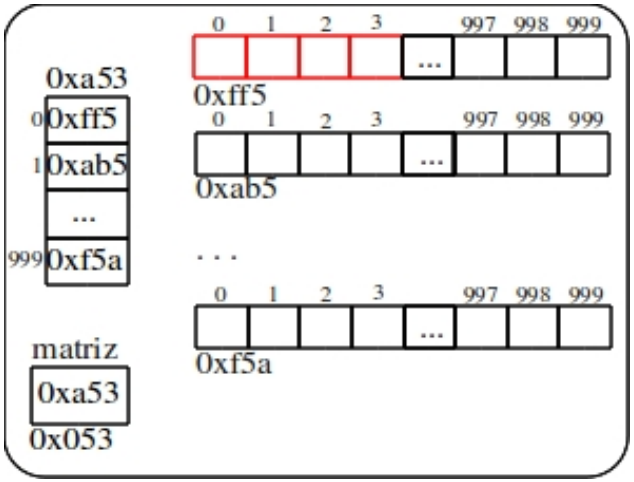
- O próximo elemento do for buscará o segundo elemento desse primeiro arranjo



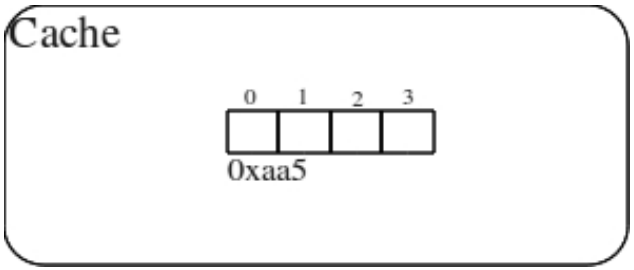


Matrizes e Memória

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

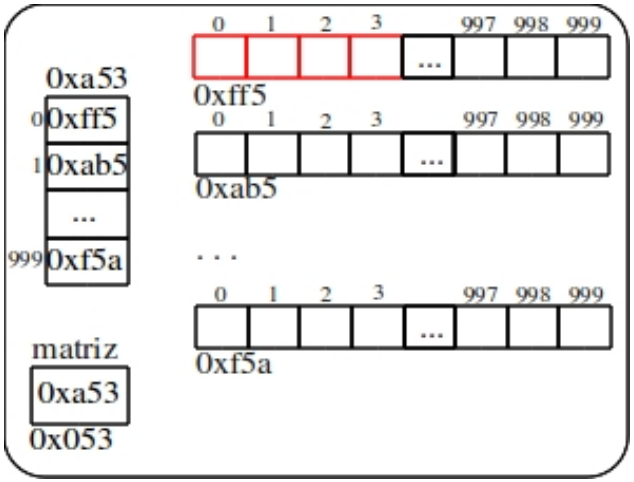


- O próximo elemento do for buscará o segundo elemento desse primeiro arranjo
- Ao buscar esse elemento, ele já está na cache

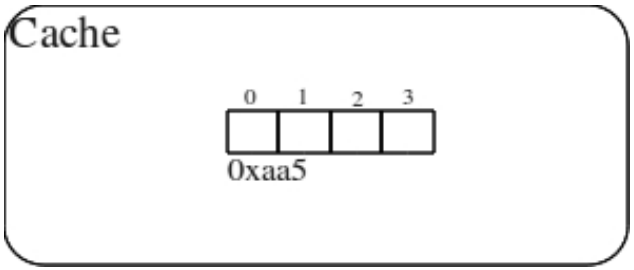


Matrizes e Memória

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[i][j];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```



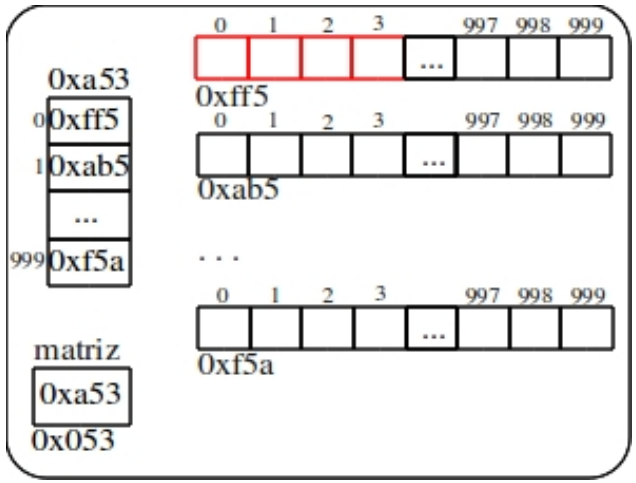
- O próximo elemento do for buscará o segundo elemento desse primeiro arranjo
- Ao buscar esse elemento, ele já está na cache
  - É recuperado mais rapidamente

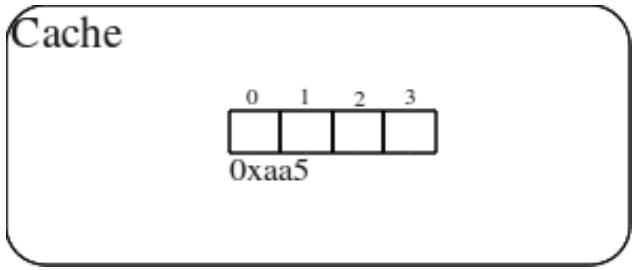


Matrizes e Memória

- E no segundo modelo?

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```

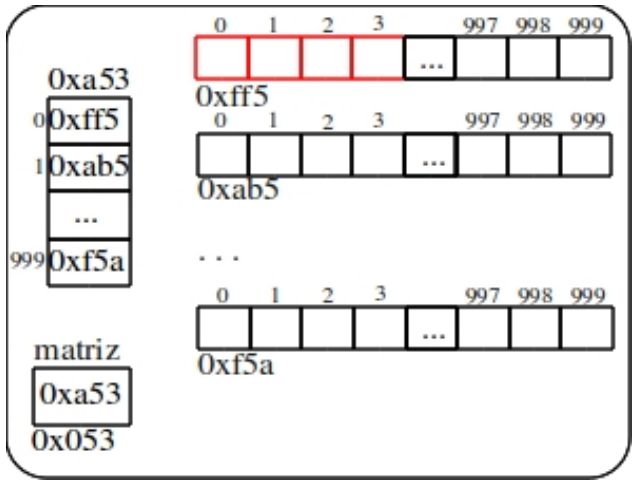




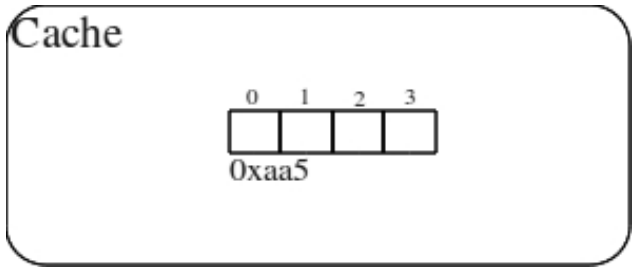
Matrizes e Memória

- E no segundo modelo?

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```



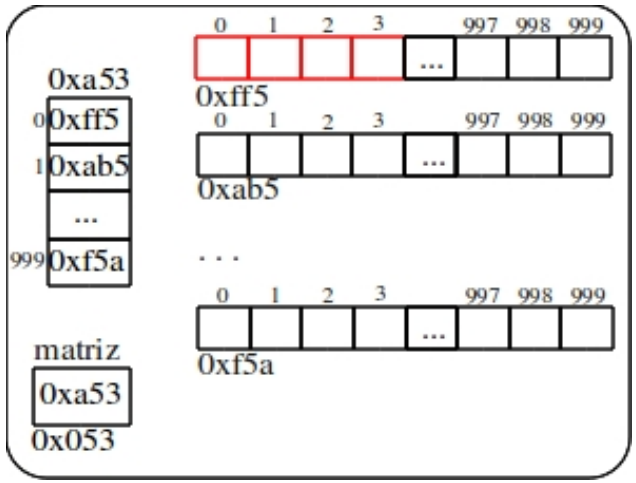
- Primeiro elemento do primeiro arranjo: idêntico ao esquema anterior



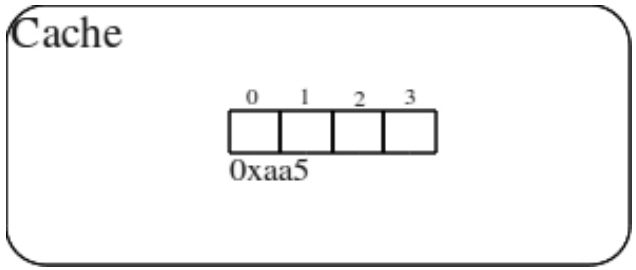
Matrizes e Memória

- E no segundo modelo?

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```



- Primeiro elemento do primeiro arranjo: idêntico ao esquema anterior

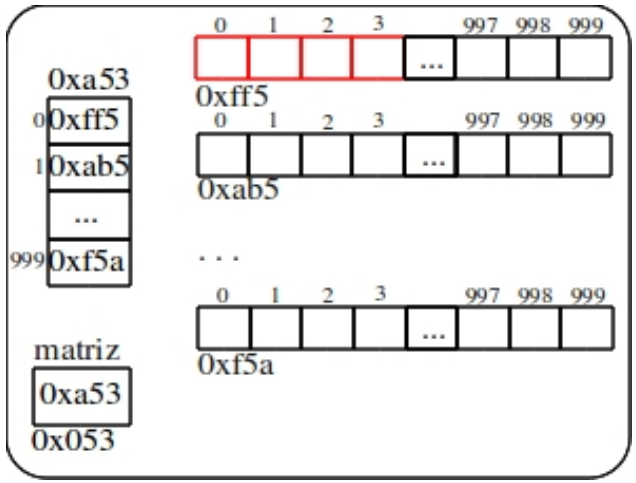


- Próximo elemento do for: primeiro elemento do segundo arranjo

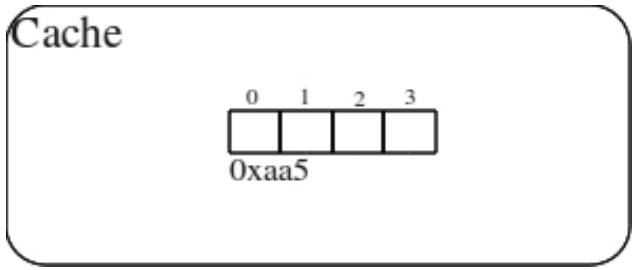
Matrizes e Memória

- E no segundo modelo?

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```



- Primeiro elemento do primeiro arranjo: idêntico ao esquema anterior

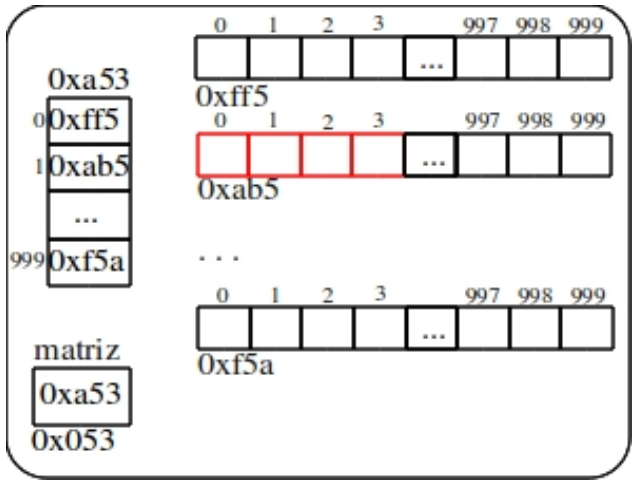


- Próximo elemento do for: primeiro elemento do segundo arranjo
- Ao buscar esse elemento, ele não estará na cache

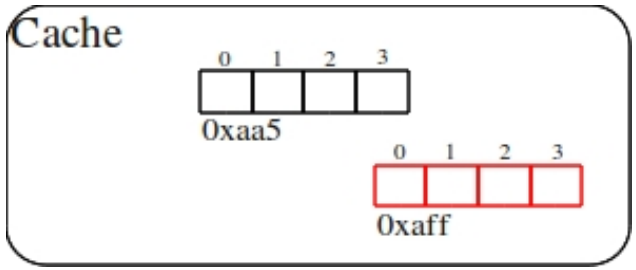
Matrizes e Memória

- E no segundo modelo?

```
public static void main(String args[]) {
    int[][] matriz = new int[1000][1000];
    int l;

    long t = System.currentTimeMillis();
    for (int i=0; i< matriz.length; i++)
        for (int j = 0; j<matriz[0].length; j++)
            l = matriz[j][i];

    long t2 = System.currentTimeMillis();

    System.out.println(t2 - t);
}
```



- Primeiro elemento do primeiro arranjo: idêntico ao esquema anterior



- Próximo elemento do for: primeiro elemento do segundo arranjo
- Ao buscar esse elemento, ele não estará na cache
  - Deve ser trazido da RAM – mais lento

Matrizes Multidimensionais

- Finalmente, matrizes não existem apenas em duas dimensões

Matrizes Multidimensionais

- Finalmente, matrizes não existem apenas em duas dimensões
- Podem ter mais:

Matrizes Multidimensionais

- Finalmente, matrizes não existem apenas em duas dimensões
- Podem ter mais:
  - Ex: `double[][][][] matriz; //quatro dimensões`

Matrizes Multidimensionais

- Finalmente, matrizes não existem apenas em duas dimensões
- Podem ter mais:
  - Ex: `double[][][][] matriz; //quatro dimensões`
  - Acessada com `matriz[1][2][0][1]`

Matrizes Multidimensionais

- Finalmente, matrizes não existem apenas em duas dimensões
- Podem ter mais:
  - Ex: `double[][][][] matriz; //quatro dimensões`
  - Acessada com `matriz[1][2][0][1]`
  - Cada dimensão pode ter tamanho diferente:

Matrizes Multidimensionais

- Finalmente, matrizes não existem apenas em duas dimensões
- Podem ter mais:
  - Ex: `double[][][][] matriz; //quatro dimensões`
  - Acessada com `matriz[1][2][0][1]`
  - Cada dimensão pode ter tamanho diferente:
    - ★ `double[][][][] matriz = new double [10][45][max][c+2];`