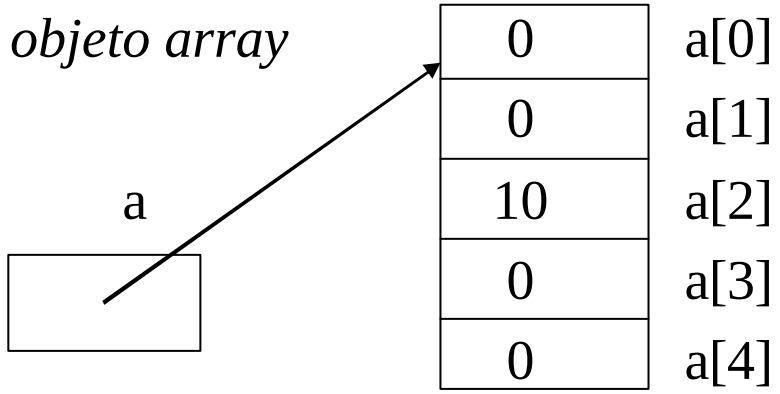
6. Manipulação de arrays

- Um array pode ser considerado como um conjunto de posições de memória onde cada posição armazena um determinado valor, sendo todos os valores de um mesmo tipo.

- Em Java, arrays são tratados como **objetos**.

Exemplo:

```
int [ ]  a;    // declara  a  como sendo um objeto array de inteiros
a = new int[5];  // aloca (constrói) o objeto array
a[2] = 10;
a[7] = 12;  // erro de execução
```



Outro exemplo:

```
.....................
double[ ] x = new double[4];
int i = 0;
i++;
x[0] = 1.0;
x[i] = 4.5;
x[i+1] = x[i]*2;
int tam = x.length;   // tam  armazenará o valor 4
double [ ] z = new double[10];
x = z;
// a partir deste ponto x e z referem o mesmo array de 10 posições
```

Lembre-se:

- Em Java arrays são objetos. Logo ao ser alocado espaço, Java inicializa cada uma das posições com determinado valor.

- O primeira posição de um array é zero.

- Tentar acessar uma posição que não existe é uma situação de erro.

- Um array pode ser declarado e já inicializado:

```
int [ ] s = { 10, 20 , 30 , 40 };
```

- Ao passar o nome de um array como argumento de determinado método, está-se passando todo o array. Lembre-se: arrays são objetos.

Exercício

Considere um objeto que represente uma coleção de valores inteiros. Escreva a respectiva classe.

```
public class ColecaoInteiros
{    protected int[] x; // array que conterá os elementos da coleção
     protected int quant;  // conterá a quantidade de elementos da coleção
     public ColecaoInteiros( int n)
     {
          x = new int[n];
          quant = 0;
      }
      public ColecaoInteiros( int[] valores)
      {     x = valores;
            quant = valores.length; /* atributo length foi definido na classe
             como final e portanto não pode ser alterado.*/
       }
```

```
public int informeQuant()
{   return quant ; }
public int informeElemento ( int i)
{   return x[i]   ;  }
 public void recebaNovoValor( int valor)
 {
       if (quant == x.length)
       { int[] y = new int[quant + 10];
          for (int i=0; i<quant; i++)
               y[i] = x[i];
           x = y;
        }
        x[quant] =  valor;
         quant++;
}
```

```
public int fornecaSoma()
{
    // implementar
 }
public double fornecaMedia()
{
    // implementar
 }
 public boolean estaVazia()
{
    // implementar
 }
public int fornecaMaiorValor()
{
    // implementar
 }
```

```
public boolean temValorRepetido()
{
    // implementar
 }
public boolean contemValor(int valor)
{
    // implementar
 }
 public boolean eliminouPrimeiraOcorrencia( int valor )
{
    // implementar
 }
public void ordene_se()
{
    // implementar
 }
```

```
public void ElimineTodasOcorrencias( int valor)
{
    // implementar
 }
 public int[] fornecaNegativos()
{    int cont = 0;
     for (int i=0; i<quant; i++)
         if (x[i] < 0 ) cont++;
     int[] y = new  int[cont];
     int k = 0;
     for (int i=0; i<quant; i++)
         if (x[i] < 0 )
         {  y[k] = x[i];
            k++;
          }
      return y;
 }
```

public void ElimineRepeticoes()

{ /* *o método deve eliminar as repetições de valores. Assim, se um valor aparece mais que uma vez, a repetições devem ser eliminadas e o valor deve passar a contar apenas uma vez.*/*

}

public ColecaoInteiros fornecaMultiplos(int n)

{ */* O método deve retornar com a coleção formada por cópias dos valores integrantes da coleção representada pelo objeto executor que são múltiplos de n, onde n é um inteiro positivo**.** */*

}

public ColecaoInteiros fornecaIntersecao(ColecaoInteiros outra)

{ */* O método deve retornar a coleção formada pela interseção da coleção representada pelo objeto executor com a coleção representada pelo parâmetro outra. Assumir que na coleção executora não existem valores repetidos */*

}

Exercício

Escreva uma classe contendo métodos estáticos que gerem valores pseudo-aleatórios.

```
public class Sortear
{       public static int sortearNumero(int v)
        {  return (int)(Math.random()*v+1); }
        public static double sortearNumero(double v)
        {
                return Math.random()*v;
        }
        public static int[] sortearNumerosDiferentes(int v, int quant)
        {
        // sorteia quant valores inteiros todos diferentes e na faixa 0 – v
        //  implementar
        }
}
```

Obs.: A classe ***Arrays*** declarada em *java.util* apresenta os seguintes métodos estáticos:

fill(*ar* , *val*) --> armazena o valor *val* em todas as posições do array *ar*

equals(*ar1*, *ar2*) --> retorna true se o array *ar1* for igual ao array *ar2*
(Dois arrays são iguais se tiverem a mesma quantidade de elementos e se, para todas as posições, o valor armazenado em uma posição do primeiro array for igual ao valor armazenado na mesma posição do segundo array.)

sort(*ar*) --> ordena em ordem nâo descendente os elementos do array *ar*

binarySearch(*ar*, *val*) --> realiza uma pesquisa no array *ar* retornando true caso o valor *val* estiver no array *ar*. O array *ar* deve estar ordenado.

Exercícios:

1) Escreva uma classe que implemente uma fila genérica. Além de construtor e possíveis métodos de acesso, coloque métodos para colocar um novo elemento na fila e retirar um elemento da fila.

2) Escreva uma classe que implemente uma fila de pessoas. Além das operações tradicionais de uma fila, coloque métodos para:

   a) Fornecer o nome da pessoa mais idosa da fila;

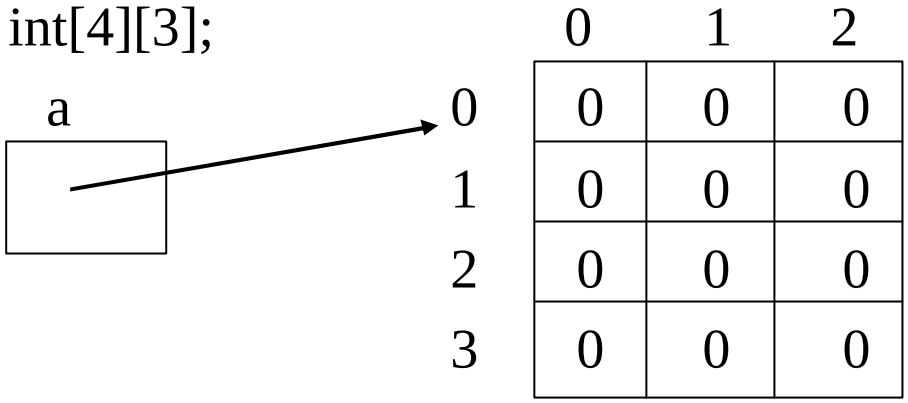
   b) Fornecer a idade média das pessoas da fila;

   c) Retirar da fila uma determinada pessoa;

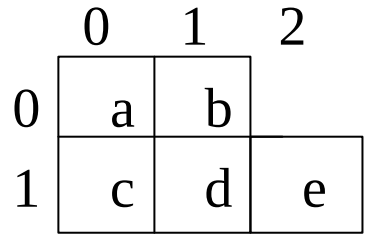
   d) Retirar da fila a pessoa mais jovem;

   e) Retirar da fila todas as pessoas do sexo feminino e colocá-las em uma nova fila, respeitando a ordem que as mesmas tinham na fila original.

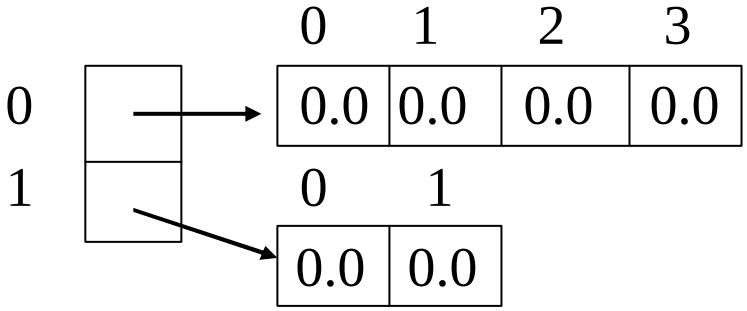
```
int [][] a;
a = new int[4][3];
```




```
char [][] letra = {{'a' ,'b'},{'c','d','e'}}
```




```
double [][] d;
d = new double[2][];
d[0] = new double[4];
d[1] = new double[2];
```



```
Pessoa [][] x;
x = new Pessoa[2][2];
for (int i=0; i<2; i++)
   for (int j=0; j<2; j++)
   { Pessoa p = new Pessoa (“Maria”+i+j, i+j, ‘M’);
      x[i][j] = p;
    }

// Cálculo da soma das idades
int soma =0;
for (int i=0; i<2; i++)
   for (int j=0; j<2; j++)
       soma += x[i][j].informeIdade();
```

Exercício:

Escreva uma classe de forma que uma matriz numérica seja tratada como um objeto, implementando as principais operações sobre matrizes.

```
public class Matriz
{   protected double [][] x;
    public Matriz( int nl, int nc)
    {   x = new double[nl][nc];       }
     public Matriz ( double [][] valor )
     {   int nl = valor.length;
         int nc = valor[0].length;
         x = new double [nl][nc];
         for (int i=0; i< nl; i++)
              for ( int j=0; j<nc; j++)
                   x[i][j] = valor[i][j];
      }
```

```
public int informeNumeroLinhas()
{    return x.length;     }
public int informeNumeroColunas()
{   return x[0].length; }
 public boolean recebeuValorElemento( int linha, int coluna, double val)
 {   if (linha>=0 & linha<x.length)
        if (coluna>=0 & coluna < x[0].length)
            {    x[linha][coluna] = val;
                 return true;    }
        else
             return false;
     else
         return false;
 }
 public double informeElemento( int linha, int coluna)
 { return x[linha][coluna]; }
```

```
 public double fornecaSomaLinha( int l)
{       // implementar
}

public double fornecaSomaColuna ( int c)
{
   // implementar
}
 public double fornecaSomaElementos()
{
   double soma = 0;
   for ( int i=0; i < this.informeNumeroLinhas();  i++ )
       for ( int j=0; j< this.informeNumeroColunas();  j++ )
           soma += x[i][j];
   return soma;
}
```

```
public double fornecaMaiorValor()
{
    // implementar
}
 public int linhaMaiorSoma()
{
   // implementar
 }
 public boolean e_Quadrada()
{
  // implementar
}
public boolean e_Simetrica()
{  // implementar
 }
```

```
public boolean temLinhasIguais()
{        // implementar
 }
 public Matriz fornecaSuaSomaCom(Matriz outra)
 {          // implementar
 }
 public double[] fornecaPontosMaximos()
 {
    // deve devolver a relação dos valores que sendo o máximo de sua linha
    //  seja  também o máximo de sua coluna
  }
  public boolean e_QuadradoMagico()
  {  /* A matriz será um quadrado mágico se for quadrada e se a soma dos
   valores em cada linha, cada coluna ou diagonal for sempre o mesmo
    valor. */       }
```

```
public Matriz fornecaTransposta()
{
        // implementar
 }
public Matriz fornecaMultiplicacao (Matriz outra)
{
       // implementar
}
```