Ordenando e buscando um elemento em um Array no Java

Nesse exemplo de código deste tutorial de java, iremos usar o método 'sort', que significa ordenar em inglês e o método 'binarySearch', que faz uma busca por um determinado elemento e retorna sua localização no Array.

Inicialmente definimos um Array de inteiros quaisquer.
Para exibir os elementos de um Array, em Java, como uma string, usamos o método:
Arrays.toString(array);
que recebe um array como argumento.

Depois vamos usar o método 'sort', que ordena o Array de ordem crescente. Ou seja, vai trocar a posição dos elementos de modo que estejam organizados do menor para o maior.
Para usar o método sort em Java fazemos:
Arrays.sort(array);

Depois, mostramos o array novamente, mas na forma de string para você ver como ficou a organização do array depois do método sort ter entrado em ação.

Depois, vamos armazenar em um inteiro, 'posicao', a posição do elemento '2112' no array: *Arrays.binarySearch(array, numero_que_estamos_procurando) ;*

IMPORTANTE: esse método só funciona se o array estiver na ordem crescente! Ou seja, só use o 'binarySearch' após usar o 'sort', pois a Binary Search é um tipo de busca inteligente (ele não sai simplesmente procurando os elementos, um por um, seu algoritmo se baseia na ordenação).

```
import java.util.Arrays;

public class arraysClass{

    public static void main(String[] args){
            int[] numeros={1, 4, 0, -13, 2112, 14, 17};
            int posicao;

            System.out.println("Os elementos do array são: "+
Arrays.toString(numeros));
            System.out.println("Ordenando...");

            Arrays.sort(numeros);

            System.out.println("Array ordenado: "+Arrays.toString(numeros));

            posicao=Arrays.binarySearch(numeros, 2112);
            System.out.println("Posição do elemento '2112': "+ posicao);
        }

}
```

Métodos da classe Arrays (Array class methods):

Dependendo dos métodos, podemos usar inteiros, float, double, char, short, List, Object etc.
Para saber exatamente se existe um método para aplicar no que você deseja usar, olhe a documentação da classe Arrays em:
http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html

Arrays.binarySearch:
Serve para encontrar um elemento específico dentro do array. Retorna a posição no array (inteiro).

Caso passe como argumento um array e um valor, a busca é feita em todo o array.
Podemos também passar um intervalo de busca. Por exemplo, para procurar o elemento 'x', no Array 'vetor', a partir do elemento 'daqui' até o elemento 'ate_aqui', faça:
Arrays.binarySearch(vetor, daqui, ate_aqui, x);

Arrays.copyOf:
Esse método copia um array e retorna outro. Esse que ele retorna é uma cópia do primeiro.
Se receber dois argumentos - um array e um valor, esse array que você passa é aquele que você deseja copiar e o valor é o número de elementos que você deseja copiar (ou o número de elementos que você quer que seu novo array tenha. Caso deseje ter um array maior, esse método preenche com 0 ou nulls):
novoArray[] = Arrays.copyOf(arrayOriginal, numero_de_elementos_a_serem_copiados);

Você também pode especificar uma faixa de valores:
novoArray[] = Arrays.copyOf(arrayOriginal, daqui, ate_aqui);

Arrays.equals:
Recebe dois arrays. Retorna true caso sejam iguais e false caso contrário.

Arrays.fill:
Vai preencher os valores de um array com determinado valor.
Caso deseje que todos os elementos de 'array' tenham o valor 'valor':
Arrays.fill(array, valor);

Para preencher só determinada faixa de valores:
Arrays.fill(array, daqui, ate_aqui, valor);

Arrays.sort:
Ordena os elementos em ordem crescente:
Arrays.sort(array);

Para ordenar uma faixa de valores:
Arrays.sort(array, daqui, ate_aqui);

Arrays.toString:
Retorna todos os elementos de um array na forma de string:
Arrays.toString(array);