java.util.Arrays

Java™

Objectives

- With help of Arrays utility methods do the following:

  - Sorting primitive and String arrays by using the sort() method.
  - Use binarySearch() to find correct indexes in sorted arrays.
  - Be able to use fill() to set all elements to the same value.
  - Be able to copy arrays with help of copyOf() method.
  - Be able to copy parts of arrays to a new array with help of copyOfRange().
  - Convert an array to a String with toString() method.

Arrays.sort() method

- Useful for sorting primitives and Strings in arrays.

```
public static void main(String[] args) {
    String[] names = {
        "Simon", "erik", "Ulf", "Fredrik", "Jonas", "Kent", "Marcus", "Martina"
    };

    Arrays.sort(names);

    for(String name: names) {
        System.out.println(name);
    }
}
```

OUTPUT:

```
Jonas
Kent
Marcus
Martina
Simon
Ulf
erik
```

”erik” is put last because sort cares about upper and lower case.

Arrays.sort() with ignore case comparator

```
public static void main(String[] args) {
    String[] names = {
        "Simon", "erik", "Ulf", "Fredrik", "Jonas", "Kent", "Marcus", "Martina"
    };

    Arrays.sort(names, String.CASE_INSENSITIVE_ORDER);

    for(String name: names) {
        System.out.println(name);
    }
}
```

Now it sort the array in a case insensitive order because of a **Comparator** in the String class.

Output:

```
erik
Fredrik
Jonas
Kent
Marcus
Martina
Simon
Ulf
```

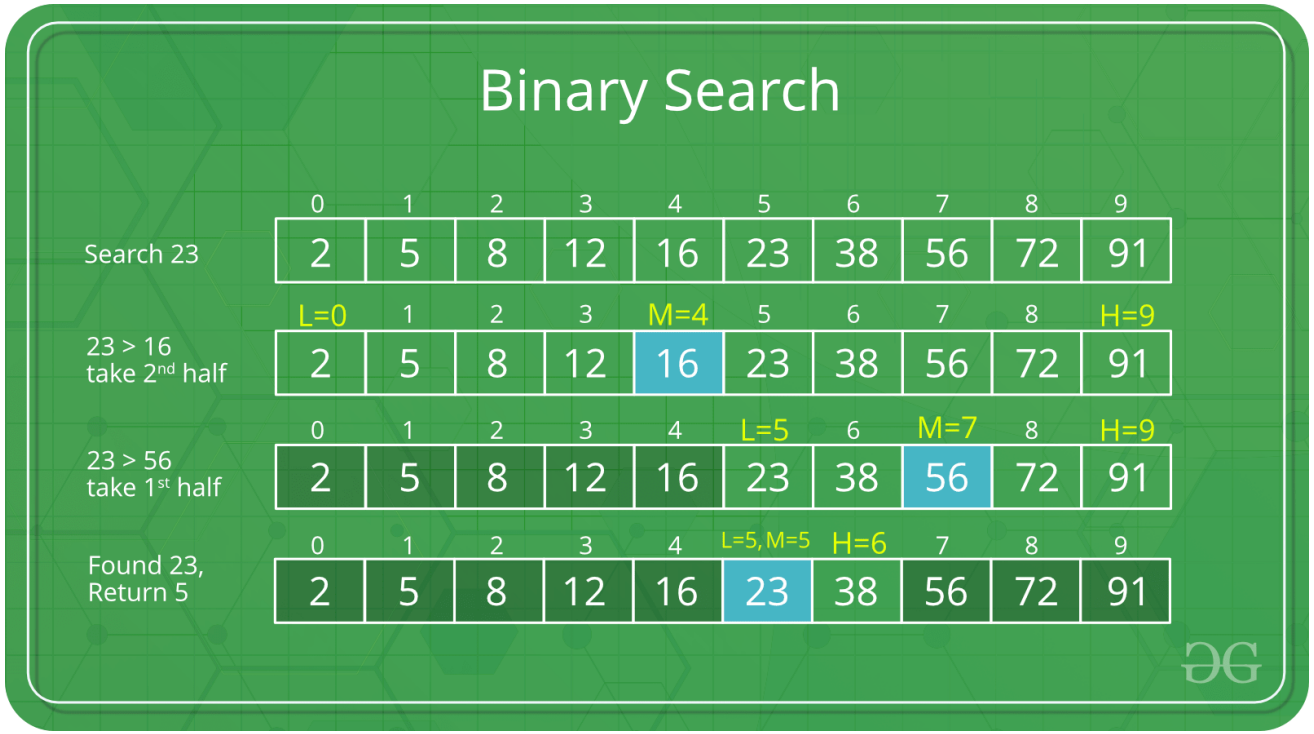
Arrays.sort() summary

- Can most common datatypes in Java.
- User defined types need to either implement Comparable<T> interface or supply a Comparator<T>.

We will look at Comparable and Comparators later on in the course….

Arrays.binarySearch method

- Used when we want to find index of an element in any array.
- Only works reliably when array is sorted by that element.



Picture taken from GeeksforGeeks.
See the original with explaination.

Arrays.binarySearch example

```
int[] numbers = {200,700,900,4555,34500,445000,500000};

int indexFound = Arrays.binarySearch(numbers, 900);
int indexNotFound = Arrays.binarySearch(numbers, 4450);

System.out.println(indexFound);          //2
System.out.println(indexNotFound);       //A negative number
```

Arrays.copyOf() method

- Copy existing array content to a new array.
- Can create a copy with a different length.
- Length of the new array can be longer or shorter than original.
- Useful when we want arrays to be more dynamic.

Arrays.copyOf() examples

Shrinking example

```
int[] array1 = {1,2,3,4};
int[] array2 = Arrays.copyOf(array1, 3);

for(int number : array2) {
    System.out.println(number);
}
```

OUTPUT:
1
2
3

Expanding example

```
int[] array1 = {1,2,3,4};
int[] array2 = Arrays.copyOf(array1, array1.length + 2);
array2[4] = 5;
array2[5] = 6;

for(int number : array2) {
    System.out.println(number);
}
```

OUTPUT:
1
2
3
4
5
6

Copy vs Reference Demo

```
public static void main(String[] args) {
    //Defining an array
    char[] letters = {'J','a','v','a'};

    //Making a reference that points to letters
    char[] notACopy = letters;

    //Making a real new array that is a copy of letters
    char[] realCopy = Arrays.copyOf(letters, letters.length);

    letters[0] = 'L’;

    printArray(notACopy); //Lava
    printArray(realCopy); //Java
}

public static void printArray(char[] toPrint) {
    for(char letter : toPrint) {
        System.out.print(letter);
    }
    System.out.println();
}
```

”Expandable” arrays with Arrays.copyOf()

You can use Arrays.copyOf() to create dynamic arrays.

```
public class App {

    public static void main(String[] args) {
        String[] names = new String[0]; //Empty Array
        names = addNameToArray(names, "Fredrik");//[Fredrik]
    }

    public static String[] addNameToArray(final String[] source, String name) {
        String[] newArray = Arrays.copyOf(source, source.length + 1);
        newArray[newArray.length-1] = name; //Adding the name to last index of newArray
        return newArray;
    }

}
```

Combining arrays with Arrays.copyOf()

```
public class App {

    public static void main(String[] args) {
        String[] names = {"Fredrik"};
        String[] moreNames = {"Erik", "Ulf", "Simon", "Kent"};
        names = arrayConcat(names, moreNames); //[Fredrik, Erik, Ulf, Simon, Kent]
    }

    public static String[] arrayConcat(String[] source, String[] elementsToAdd) {
        String[] combined = Arrays.copyOf(source, source.length + elementsToAdd.length);
        for(int i=source.length, j=0; i<combined.length; i++, j++) {
            combined[i] = elementsToAdd[j];
        }
        return combined;
    }
}
```

Arrays.copyOfRange()

```
public static void main(String[] args) {

    int[] numbers = {1,2,3,4,5,6,7,8,9};

    int startIndex = 2;              //Position we want to start from, INCLUSIVE
    int endIndex = numbers.length;   //Position we want to end at, EXCLUSIVE

    //range = {3,4,5,6,7,8,9}
    int[] range = Arrays.copyOfRange(numbers, startIndex, endIndex);

}
```

Arrays.fill()

```
public static void main(String[] args) {

    char[] letters = new char[10];
    Arrays.fill(letters, 'X'); //['X','X','X','X','X','X','X','X','X','X']

}
```

Arrays.toString()

```
public class App {

    public static void main(String[] args) {

        String[] javaPrinciples = {
            "1. DRY - Don't Repeat Yourself",
            "2. KISS - Keep It Simple Stupid"
        };

        //[Ljava.lang.String;@15db9742
        System.out.println(javaPrinciples);

        //[1. DRY - Don't Repeat Yourself, 2. KISS - Keep It Simple Stupid]
        System.out.println(Arrays.toString(javaPrinciples));

    }

}
```

If you want to turn an array into a String you can use Arrays.toString() method.

If you dont use it you will only get the memory address of the array back.

Questions?