==========Arrays==========

What is an Array:

An array is a collection of elements of the same data type stored in contiguous memory locations. Arrays are fixed in size, meaning that once they are created, their size cannot be changed.

Example:

int[] numbers = new int[5]; // Declares an array of integers with 5 elements

Declaring an Array:

To declare an array, you specify the type of elements it will hold, followed by square brackets, and the variable name.

Syntax:

type[] arrayName;

Example:

int[] numbers;
String[] names;

Initializing an Array:

You can initialize an array at the time of declaration or later in the code. Initialization sets the elements of the array to specific values.

Example:

```
// Initialize at declaration
int[] numbers = {1, 2, 3, 4, 5};

// Initialize after declaration
numbers = new int[5]; // Creates an array with 5 elements, all initialized to 0
```

Accessing Array Elements:

Array elements are accessed using their index, which starts at 0 for the first element.

Example:

```
int[] numbers = {1, 2, 3, 4, 5};
int firstNumber = numbers[0]; // Accesses the first element
numbers[2] = 10; // Changes the third element to 10
```

Looping Through an Array:

You can use loops to iterate through the elements of an array.

Example:

```
int[] numbers = {1, 2, 3, 4, 5};
```

```
// Using a for loop
for (int i = 0; i < numbers.length; i++) {
    System.out.println(numbers[i]);
}

// Using a for-each loop
for (int number : numbers) {
    System.out.println(number);
}
```

Length of an Array:
The length of an array can be obtained using the length property.

Example:
```
int[] numbers = {1, 2, 3, 4, 5};
int length = numbers.length; // Length is 5
```

Multidimensional Arrays:
Java supports multidimensional arrays, which are arrays of arrays.
The most common type is the two-dimensional array.

Example:
```
// Declaring and initializing a two-dimensional array
int[][] matrix = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};

// Accessing elements of a two-dimensional array
int element = matrix[1][2]; // Accesses the element in the second row, third column (value is 6)
```

Common Array Operations:
Copying Arrays
You can copy arrays using loops or utility methods from the
java.util.Arrays class.

Example:

```
import java.util.Arrays;

public class CopyArrayExample {
    public static void main(String[] args) {
        int[] original = {1, 2, 3, 4, 5};
        int[] copy = Arrays.copyOf(original, original.length);
```

```
        System.out.println("Original array: " + Arrays.toString(original));
        System.out.println("Copied array: " + Arrays.toString(copy));
    }
}
```

Output:

```
Original array: [1, 2, 3, 4, 5]
Copied array: [1, 2, 3, 4, 5]
```

Sorting Arrays

You can sort arrays using the Arrays.sort method.

Example:

```
import java.util.Arrays;

public class SortArrayExample {
    public static void main(String[] args) {
        int[] numbers = {5, 3, 1, 4, 2};
        Arrays.sort(numbers);

        System.out.println("Sorted array: " + Arrays.toString(numbers));
    }
}
```

Output:

```
Sorted array: [1, 2, 3, 4, 5]
```

Searching Arrays

You can search for elements in an array using loops or utility methods from the java.util.Arrays class.

Example:

```
import java.util.Arrays;

public class SearchArrayExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        int index = Arrays.binarySearch(numbers, 3);

        System.out.println("Index of element 3: " + index);
    }
}
```

Output:

```
Index of element 3: 2
```

Filling Arrays

You can fill arrays with a specified value using the Arrays.fill method.

Example:

```
import java.util.Arrays;
```

```
public class FillArrayExample {
    public static void main(String[] args) {
        int[] numbers = new int[5];
        Arrays.fill(numbers, 10);

        System.out.println("Filled array: " + Arrays.toString(numbers));
    }
}
```
Output:
Filled array: [10, 10, 10, 10, 10]
Comparing Arrays
You can compare arrays using the Arrays.equals method.

Example:
```
import java.util.Arrays;

public class CompareArrayExample {
    public static void main(String[] args) {
        int[] array1 = {1, 2, 3};
        int[] array2 = {1, 2, 3};
        boolean isEqual = Arrays.equals(array1, array2);

        System.out.println("Arrays are equal: " + isEqual);
    }
}
```
Output:
Arrays are equal: true

Example:
```
public class SumArrayExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        int sum = 0;

        for (int number : numbers) {
            sum += number;
        }

        System.out.println("Sum of elements: " + sum);
    }
}
```
Output:
Sum of elements: 15
Finding the Maximum and Minimum Elements
You can find the maximum and minimum elements in an array using loops.

Example:

```
public class MaxMinArrayExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        int max = numbers[0];
        int min = numbers[0];

        for (int number : numbers) {
            if (number > max) {
                max = number;
            }
            if (number < min) {
                min = number;
            }
        }

        System.out.println("Maximum element: " + max);
        System.out.println("Minimum element: " + min);
    }
}
```

Output:

```
Maximum element: 5
Minimum element: 1
```

Reversing an Array

You can reverse an array in place using a loop.

Example:

```
public class ReverseArrayExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        // Reverse the array
        for (int i = 0; i < numbers.length / 2; i++) {
            int temp = numbers[i];
            numbers[i] = numbers[numbers.length - 1 - i];
            numbers[numbers.length - 1 - i] = temp;
        }

        System.out.println("Reversed array: " + Arrays.toString(numbers));
    }
}
```

Output:

```
Reversed array: [5, 4, 3, 2, 1]
```

Removing Duplicates

You can remove duplicates from an array by converting it to a Set.

Example:

```
import java.util.Arrays;
import java.util.LinkedHashSet;
import java.util.Set;

public class RemoveDuplicatesArrayExample {
   public static void main(String[] args) {
      Integer[] numbers = {1, 2, 2, 3, 4, 4, 5};

      // Remove duplicates
      Set<Integer> set = new LinkedHashSet<>(Arrays.asList(numbers));
      Integer[] uniqueNumbers = set.toArray(new Integer[0]);

      System.out.println("Array after removing duplicates: " + Arrays.toString(uniqueNumbers));
   }


}
```

Output:

```
Array after removing duplicates: [1, 2, 3, 4, 5]
```