Array in Java

What is Array?

Array is the most fundamental data structure which is used to store multiple elements together.

Features of Array

(i) Homogeneous

- It will store only similar types of elements

(ii) Fixed Size

- Array in Java is fixed in size
- Once an array is created, we cannot change/modify its size

(iii) No Built-in Methods Support

- In Java, Array will not have support of any inbuilt methods
- Methods are provided externally by **Arrays Class** or using **Stream** feature

(iv) Index-based Data Structure

- Array is an index-based data structure where indexing always starts from 0
- The last index will be `length - 1`

(v) Built-in Variable: Length

- Array has an in-built variable called `length` which is used to get the size of Array

Key Points

- **Fixed:** Arrays are fixed in size
- **No support method in Arrays:** Cannot use inbuilt methods directly
- **No direct support:** Not built-in, but we can import
- **Similar types of element:** We can add in Array
- **In Java homogeneous data stored in Array**
- **Most basic data structure is Array**
- **Only Variable Support:** `arr.length`

Declaration of Array

Syntax Options:

```
// Option 1
datatype[] varName;

// Option 2
datatype [] varName;

// Option 3
datatype varName[];
```

Examples:

```
int[] marks = {41, 23, 40};
// Index:    0   1   2

// Accessing elements
marks[0] = 41;
marks[1] = 23;
marks[2] = 40;
```

Primitive vs Non-Primitive Types

Primitive Types:

- `int[]`
- `double[]`
- `char[]`

Non-Primitive Types:

- `String[]`
- `Product[]`
- `Employee[]`

Array Creation

Array can be created in two ways:

(1) Without using `new` keyword

Array can be created without keyword only by using `{}` and by providing multiple values.

Syntax:

```
datatype[] varName = {val1, val2, val3, ...};
```

Examples:

```
// Example 1
int[] a = {10, 20, 30, 40};

// Example 2
double[] b = {24.1, 12, 23};

// Example 3
String[] c = {"RAM", "Sohan", "Mohan"};

// Example 4
Car[] cars = {new Car("Tata", 1456), new Car("BMW", 423)};
```

Memory Representation:

For array `a = {10, 20, 30, 40, 50}`:

```
Index:  [0] [1] [2] [3] [4]
Value:   10  20  30  40  50
```

System.out.println(a[0]) will give element from last index

Operations:

- `sop(a)` → Prints reference/hashcode (e.g., `I@hexno`)
- `sop(a.length)` → `5`
- `sop(a[0])` → `10`
- `sop(a[3])` → `40`
- `a[2] = 72` → Modifies value at index 2
- `a[5] = 24` → **ArrayIndexOutOfBound Exception** (index out of bounds)

---

(2) With using `new` keyword

We can create an array by using `new` keyword with appropriate size. In this manner, array is created with default value and later we have to assign the value.

Syntax:

```
int[] a = new int[5];
```

Memory Representation (with default values):

```
Index:  [0] [1] [2] [3] [4]
Value:   0   0   0   0   0
```

Assigning Values:

```
a[0] = 10;
a[1] = 20;
a[2] = 30;
a[3] = 40;
a[4] = 50;
```

Note:

- `int a = {10, 20, 30}`
- `a = {40, 20}` ❌ **Cannot modify** - We cannot reassign array

Operations:

- `sop(a)` → Prints `I@hexno`
- `sop(a.length)` → `5`
- `sop(a[0])` → `10`
- `sop(a[5])` → **ArrayIndexOutOfBound Exception**

Accessing Array Using `sop(a.length-1)`

It will give element from last index

```
sop(a[a.length-1]); // 50
```

Run a Loop Through Array

Given:

```
int a = {10, 20, 30, 40, 50};
```

(i) Using For Loop

```
for (int i = 0; i < a.length; i++) {
    sop(a[i]);
}
```

Output:

```
10
20
30
40
50
```

(ii) Using For-Each Loop

```
for (int i : a) {
    sop(i);
}
```

Output:

```
10
20
30
40
50
```

Using JavaScript

```
let nums = [10, 20, 30, 40, 50];
console.log(nums); // [10, 20, 30, 40, 50]
```

(1) For Loop

```
for (let i = 0; i < nums.length; i++) {
  console.log(a[i]);
}
```

Output:

```
10
20
30
40
50
```

(2) For-Each

```
for (int i of nums) {
    console.log(i);
}
```

Output:

```
10
20
30
40
50
```

Common Array Operations

Q1: Access all even index elements from array

```
int a[] = {10, 20, 30, 40, 50};

for (int i = 0; i < a.length; i++) {
    if (i % 2 == 0) {
        sop(a[i]);
    }
}
```

Output:

```
10
30
50
```

Q2: Access all even elements from array

```
int a[] = {10, 20, 30, 40, 50};

for (int i = 0; i < a.length; i++) {
    if (a[i] % 2 == 0) {
        sop(a[i]);
    }
}
```

Output:

```
10
20
30
40
50
```

Q3: Count all even elements from array

```
int a[] = {10, 20, 15, 30, 11};
int count = 0;

for (int i = 0; i < a.length; i++) {
    if (a[i] % 2 == 0) {
        count++;
    }
}

System.out.println(count);
```

Output:

```
3
```

Q4: Access all elements of array from end

```
int a[] = {10, 20, 30, 40, 50};

for (int i = a.length - 1; i >= 0; i--) {
    sop(a[i]);
}
```

Or:

```
for (int i = 0; i < a.length; i++) {
    sop(a[a.length - 1 - i]);
}
```

Q5: Print and Count all three digit numbers from array

```
int a[] = {121, 24, 423};
int count = 0;

for (int i = 0; i < a.length; i++) {
    if (a[i] > 99 && a[i] < 1000) {
        sop(a[i]);
        count++;
    }
}

sop("Count is: " + count);
```

Output:

```
121
423
Count is: 2
```

Q6: Print sum of all elements from array

```
int a[] = {10, 20, 30, 40};
int sum = 0;

for (int i = 0; i < a.length; i++) {
    sum = a[i] + sum;
}

sop("Sum: " + sum);
```

Output:

```
Sum: 100
```

Q7: Print Average of all elements from Array

```
int a[] = {10, 20, 30};
int sum = 0;
int count = 0;
int avg = 0;

for (int i = 0; i < a.length; i++) {
    sum = a[i] + sum;
    count++;
}

avg = sum / count;

sop(avg);
```

Output:

```
20
```

Calculation:

```
10 + 20 + 30 = 60
60 / 3 = 20
```

Q8: WAP to Print and Count all the elements of array which are bigger than average value

```
int a[] = {10, 20, 30};
int count = 0;
int avg = 0;
int sum = 0;
int num = 0;

for (int i = 0; i < a.length; i++) {
    sum = a[i] + sum;
    count++;
}

avg = sum / count;

if (a[i] > avg) {
    sop(a[i]);
    num++;
}

System.out.println(num);
// "Count: " + count
```

Alternative approach:

```
for (int i = 0; i < a.length; i++) {
    if (a[i] > avg) {
        sop(a[i]);
        num++;
    }
}

System.out.println(num);
```

Output:

```
30
1
```

Q9: Print sum of all even elements from array

```
int a[] = {10, 20, 30};
int sum = 0;

for (int i = 0; i < a.length; i++) {
    if (a[i] % 2 == 0) {
        sum = a[i] + sum;
    }
}

sop("Sum of even element: " + sum);
```

Output:

```
40
```

Q10: Print sum of all odd elements from array

```
int a[] = {10, 11, 15};
int sum = 0;

for (int i = 0; i < a.length; i++) {
    if (a[i] % 2 == -1) {
        sum = a[i] + sum;
    }
}

sop(sum);
```

Output:

```
26
```

Practice Problems Summary

1. **Access all even index** - Print elements at even indices
2. **Access all even element** - Print all even elements
3. **Count all even element** - Count how many even numbers
4. **Access all element from end** - Traverse array backwards
5. **Print and Count all 3 digit numbers** - Filter by range
6. **Print Sum of elements** - Calculate total sum
7. **Print Average** - Calculate mean value
8. **Grand done fun for product in fun** (unclear task)
9. **Access all they by only type** (unclear task)

Notes

- Always remember array indexing starts from **0**
- Be careful with **ArrayIndexOutOfBound Exception** when accessing elements
- Use `array.length` to get the size of array dynamically
- For-each loop is simpler but doesn't provide index access
- Cannot modify array size after creation in Java
- Default values: `0` for int, `0.0` for double, `null` for objects

---

End of Notes