DSA

Module 5

Arrays

Author
Srinivas Dande

Java Learning Center

5. Arrays

- Array is a collection of elements which are of same data type.
- Array is also called as Homogeneous data structure.
- Elements of an array will be stored in contiguous memory locations.
- Arrays are objects in Java.
- Three tasks to remember while you are working with arrays:
  - Array Declaration
  - Array Construction
  - Array Initialization
- Arrays can be constructed with multiple dimensions i.e 1-D Array, 2-D Array etc.

5.1. Single Dimensional Arrays

- Single Dimensional array can be called as 1-D Array.

5.1.1. Array Declaration and Construction

- You can declare and construct an array in one statement also.

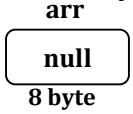
Syntax:

```
<dataType> <refVarName>[]= new <dataType>[<size>];
<dataType> []<refVarName>= new <dataType>[<size>];
```

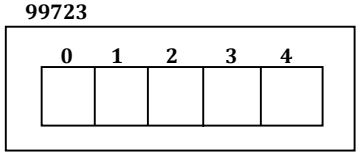
Ex:

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

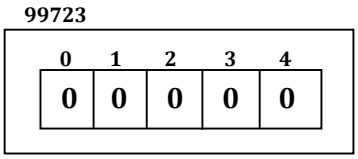
1) Allocates 8 bytes of memory for the reference variable and initializes with null value.



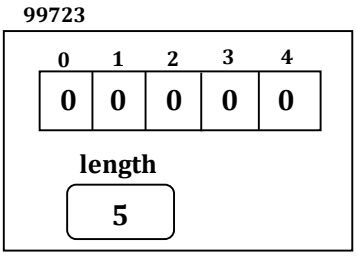
2) Memory blocks will be allocated as per array size.



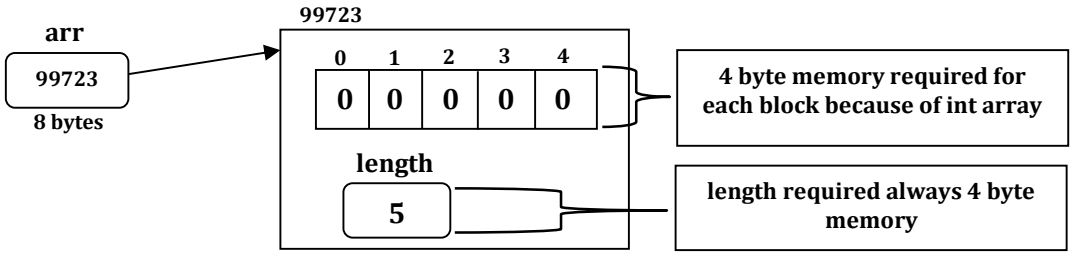
3) Memory blocks will be initialized with default value as per array data type.



4) Memory will be allocated for length variable and will be initialized with size of an array.



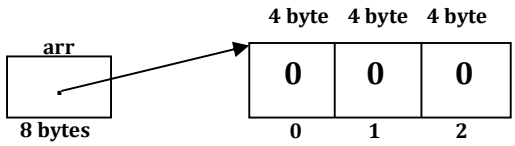
5) Array object address will be assigned to reference variable.



5.1.2. Array Initialization

- Once array is constructed then array elements will be initialized with default values as per array data type.
- You can initialize array elements with your own value using index representation.

int arr[] = new int[3];



- Here we are creating an array with size 3, so only three elements will be stored in this array.

  arr[0] -> Represents the first element which contains value 0.

  arr[1] -> Represents the second element which contains value 0.

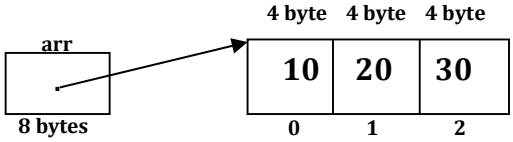
  arr[2] -> Represents the third element which contains value 0.

- You can initialize the array elements as follows:

  arr[0] =10;

  arr[1] =20;

  arr[2] =30;



- Accessing the elements after initialization

  arr[0] -> Represents the first element which contains value 10.

  arr[1] -> Represents the second element which contains value 20.

  arr[2] -> Represents the third element which contains value 30.

Lab1.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab1 {
	public static void main(String[] args) {

		// 1. Create Array
		int arr[] = new int[5];

		// 2.find the Length of Array
		int n = arr.length;
		System.out.println(n);

		// 3.Access Elements of Array
		for (int i = 0; i < n; i++)
			System.out.println(arr[i]);

		// 4. Initialize the Array
		arr[0] = 11;
		arr[2] = 33;
		arr[4] = 55;
		System.out.println("-----------------");

		// Access Elements of Array
		for (int i = 0; i < n; i++)
			System.out.println(arr[i]);
	}

}
```

5.1.3 Array Declaration, Construction and Initialization

- You can declare, construct and initialize an array in one statement also.

Syntax:

<dataType> <refVarName>[]= {<val1>,<val2>,<val3>,... ,<valN>};

Ex:

```
int arr[]={10,20,30};
String names[]={"Hello","Srinivas","Dande"};
```

Lab2.java

```
package com.jlcindia.arrays;

/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab2{
	public static void main(String[] args) {

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

		int n= arr.length;

		// Access Elements of Array
		for (int i = 0; i < n; i++)
			System.out.println(arr[i]);
	}

}
```

5.1.4. Why and Why Not Arrays?

- **Limitations with array:**
  - ✓ Array is a homogeneous data structure i.e only one type of elements can be stored in the Array.
  - ✓ Array is static in nature i.e size of an array cannot be modified.
  - ✓ Insert and delete operations require more shifting of elements.
  - ✓ You need to write the Code for doing any Operations on Arrays. No Built-In Funcatioality provided in Arrays.

- **Advantages with array:**
  - ✓ Arrays are faster in Accessing the Elements.
  - ✓ Arrays are Cache-Friendly

5.1.5 Dynamic Size Array - ArrayList

- ArrayList is implemented as a resizable array.
- ArrayList elements will be stored internally using indexing notation.
- This is one of the most widely used concrete class.
- It is fast to access the elements, but slow to insert and delete the elements.
- Size of ArrayList will be growable dynamically.
- Default Capacity is 10. i.e When You create ArrayList, It allocates the Space for 10 Elements inadvance.
- When you add add elements then size will be increased.
- Following the Formula for Capacity Calculation.

In Java 6

```
newCapacity = (oldCapacity * 3)/2 + 1;
```

In Java 7

```
newCapacity = oldCapacity + (oldCapacity >> 1)
```

- ArraList comes with many built-methods as follows.
  - int size()
  - boolean isEmpty()
  - Object get(int idx)
  - boolean add(Object obj)
  - void add(int idx, Object obj)
  - boolean remove(Object obj)
  - Object remove(int idx)
  - Object set(int idx, Object obj)
  - boolean contains(Object obj)

Lab3.java

```
package com.jlcindia.arrays;

/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

import java.util.ArrayList;
import java.util.List;

public class Lab3{
	public static void main(String[] args) {

		List<Integer> mylist = new ArrayList<>();

		mylist.add(10);
		mylist.add(20);
		mylist.add(30);
		mylist.add(40);

		System.out.println(mylist);
		mylist.add(50);
		System.out.println(mylist);
		mylist.remove(Integer.valueOf(40));
		System.out.println(mylist);
		//ArrayList has many Built-In methods to Use
	}

}
```

5.2. Operations on Arrays

Lab4.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//Search for given Element in the Unsorted  Array

public class Lab4 {
	static int linearSearch(int arr[], int element) {
		int n = arr.length;

		for (int i = 0; i < n; i++) {
			if (arr[i] == element) {
				return i;
			}
		}
		return -1;
	}
	public static void main(String[] args) {
		int arr[] = { 25, 10, 5, 15, 30, 20, 50 };
		int element = 25;

		int index = linearSearch(arr, element);

		System.out.println(index);
		if (index == -1)
			System.out.println("Element Not Found");
		else
			System.out.println("Element Found at Index : " + index);
	}
}

// Time Complexity - - L.S  O(n)
// Aux Space - O(1)
```

Lab5A.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Insert the Element at given position in an Array
// If Capacity is not there Dont Insert

public class Lab5A {

	static int insert(int arr[], int element,int position) {

		int n = arr.length;
		int index = position - 1;

		for (int i = n-1; i > index; i--) {
			arr[i] = arr[i-1];
		}

		arr[index] = element;

		return n+1;
	}

	public static void main(String[] args) {

		int arr[] = new int[5];
		arr[0] = 10;
		arr[1] = 20;
		arr[2] = 30;
		arr[3] = 40;
		arr[4] = 50;

		int element = 99;
		int position = 3;
```

```
                for(int i=0;i<arr.length;i++) {
                        System.out.print(arr[i]+" ");
                }

                 insert(arr, element,position);

                 System.out.println("----");
                for(int i=0;i<arr.length;i++) {
                        System.out.print(arr[i]+" ");
                }

        }
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Lab5B.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Insert the Element at given position in an Array
// Create New Array and Insert

public class Lab5B {

    static int[] insert(int arr[], int element, int position) {

                int index = position - 1;
                int tempArr[] = new int[arr.length + 1];
                int n = tempArr.length;
```

```
            for (int i = n - 1; i > index; i--) {
                tempArr[i] = arr[i - 1];
            }

            // Insert
            tempArr[index] = element;

            for (int i = 0; i < index; i++) {
                tempArr[i] = arr[i];
            }
            return tempArr;
        }

        public static void main(String[] args) {

            int arr[] = {10,20,30,40,50};
            int element = 99;
            int position = 3;

            for (int i = 0; i < arr.length; i++) {
                System.out.print(arr[i] + " ");
            }

            System.out.println();

            int newArr[] = insert(arr, element, position);

            System.out.println("----");
            for (int i = 0; i < newArr.length; i++) {
                System.out.print(newArr[i] + " ");
            }
        }
}
// Time Complexity - -  O(n)
// Aux Space - O(n)
```

Lab6.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Delete the Element at given position in an Array

public class Lab6 {

	static void deleteAt(int arr[],  int position) {

		int index = position - 1;
		int n = arr.length;

		if(position>n) {
			return ;
		}

		for (int i = index; i < n-1; i++) {
			arr[i] = arr[i + 1];
		}

		arr[n-1] = 0;
	}

	public static void main(String[] args) {

		int arr[] = {10,20,30,40,50};
		int position = 5;

		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + " ");
		}
		System.out.println();
```

```
                deleteAt(arr, position);

                System.out.println("----");
                for (int i = 0; i < arr.length; i++) {
                        System.out.print(arr[i] + " ");
                }

        }
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Lab7.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Delete the given Element from an Array

public class Lab7 {

	static void deleteElement(int arr[],  int element) {

		int n = arr.length;
		int index=-1;

		for(int i=0;i<n;i++) {
			if(arr[i]==element) {
				index=i;
				break;
			}
		}

		if(index==-1) {
			return ;
		}

		for (int i = index; i < n-1; i++) {
			arr[i] = arr[i + 1];
		}

		arr[n-1] = 0;

	}
```

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

                int arr[] = {10,20,30,40,50};
                int element = 55;

                for (int i = 0; i < arr.length; i++) {
                        System.out.print(arr[i] + " ");
                }

                System.out.println();

                deleteElement(arr, element);

                System.out.println("----");
                for (int i = 0; i < arr.length; i++) {
                        System.out.print(arr[i] + " ");
                }

        }
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Lab8.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Access the Elements of an Array

public class Lab8 {

	public static void main(String[] args) {

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

		System.out.println(arr[0]);
		System.out.println(arr[1]);
		System.out.println(arr[2]);
		System.out.println(arr[3]);

		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + " ");
			//Do something with Element
		}

	}
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

5.3. Time Complexity of Array Operations

Operation	Time Complexity
Access One Element	O(1)
Access All Elements	O(n)
Update One Element	O(1)
Linear Search	O(n)
Binary Search	O(log n)
Insert Element at nth postion	O(n)
Insert Element at First postion	O(n)
Insert Element at Last postion	O(1)
Delete Element at nth postion	O(n)
Delete Element at First postion	O(n)
Delete Element at Last postion	O(1)

5.4. Problems on Arrays

Problem1A.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//Reverse the given Array

public class Problem1A {
	static void reverse(int arr[]) {

		int n = arr.length;
		int start = 0;
		int end = n - 1;

		while (start < end) {
			int temp = arr[start];
			arr[start] = arr[end];
			arr[end] = temp;

			start++;
			end--;
		}
	}
	public static void main(String[] args) {

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

		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + "  ");
		}
		System.out.println( "\n  ");

		reverse(arr);
```

```
                for (int i = 0; i < arr.length; i++) {
                        System.out.print(arr[i] + "  ");
                }
        }
}

// Time Complexity - -  O(n/2) / O(n)
// Aux Space - O(1)
```

Problem1B.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//Reverse the given Array

public class Problem1B {

        static void reverse(int arr[]) {
                int n = arr.length;

                for (int i=0,j=n-1;i < j;i++,j--) {
                        int temp = arr[i];
                        arr[i] = arr[j];
                        arr[j] = temp;
                }
        }
        public static void main(String[] args) {
            //Same as Problem1 main() method
        }
}
// Time Complexity - -  O(n/2)
// Aux Space - O(1)
```

Problem2.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//Remove the Duplicates from Sorted Array

public class Problem2 {

	static int removeDuplicates(int arr[]) {
		int n = arr.length;
		int temp = 1;

		for (int i = 1; i < n; i++) {

			if (arr[i] != arr[temp - 1]) {
				arr[temp] = arr[i];
				temp++;
			}
			arr[i]=0;
		}
		return temp;
	}

	public static void main(String[] args) {

		int arr[] = { 0,0,10, 10, 20, 20, 20, 30, 40, 40, 40 };

		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + "  ");
		}

		int tempCount = removeDuplicates(arr);
```

```
                System.out.println(" \n  ");

                for (int i = 0; i < arr.length; i++) {
                        System.out.print(arr[i] + "  ");
                }

                System.out.println(" \n  ");

                for (int i = 0; i < tempCount; i++) {
                        System.out.print(arr[i] + "  ");
                }
        }
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Problem3.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Move All Zeros to End

public class Problem3 {

	static int  moveZeros(int arr[]) {

		int n = arr.length;
		int temp=0;

		for(int i=0;i<n;i++) {
			if(arr[i]!=0) { //Non-Zero values

				int x = arr[i];
				arr[i]=arr[temp];
				arr[temp]=x;

				temp++;
			}
		}
		return temp;
	}

	public static void main(String[] args) {

		int arr[] = { 10,15,0,0,25,0,20 };
		//int arr[] = { 10,15,25,20,0,0,0 };

		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + "  ");
		}
```

```
            int nonZeroCount = moveZeros(arr);

            System.out.println("\n  ");

            //Print All
            for (int i = 0; i < arr.length; i++) {
                  System.out.print(arr[i] + "  ");
            }

            System.out.println("\n  ");

            //Print Non-Zeors
            for (int i = 0; i < nonZeroCount; i++) {
                  System.out.print(arr[i] + "  ");
            }

      }
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Problem4.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//Find whether the Array is Sorted in ASC or Not

public class Problem4 {
	static boolean  isSorted(int arr[]) {

		int n = arr.length;

		for(int i=1;i<n;i++) {
			if(arr[i-1]>arr[i]) {
				return false;
			}
		}
		return true;
	}
	public static void main(String[] args) {

		int arr[] = { 10,20,30,40,50 };
		// int arr[] = { 10,20,30,50,40 };

		boolean flag = isSorted(arr);

		if(flag)
			System.out.println("Yes Sorted");
		else
			System.out.println("Not Sorted");
	}
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Problem5.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
// Rotate the Elements Left side 1 time.
public class Problem5 {
	static void leftRotateOne(int arr[]) {

		int n = arr.length;
		int temp = arr[0];

		for (int i = 1; i < n; i++) {
			arr[i - 1] = arr[i];
		}
		arr[n - 1] = temp;
	}
	public static void main(String[] args) {
		int arr[] = { 1, 2, 3, 4, 5 };

		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + "  ");
		}

		leftRotateOne(arr);

		System.out.println("\n  ");
		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + "  ");
		}
	}
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Problem6A.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
// Rotate the Elements Left side K times.

public class Problem6A {
	static void leftRotateOne(int arr[]) {

		int n = arr.length;
		int temp = arr[0];

		for (int i = 1; i < n; i++) {
			arr[i - 1] = arr[i];
		}

		arr[n - 1] = temp;
	}

	static void leftRotate(int arr[], int k) {

		for (int i = 1; i <= k; i++) {
			leftRotateOne(arr);
		}
	}

	public static void main(String[] args) {

		int arr[] = { 1, 2, 3, 4, 5 };
		int k = 3; //

		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + "  ");
		}
```

```
                leftRotate(arr,k);

                System.out.println("\n  ");

                for (int i = 0; i < arr.length; i++) {
                        System.out.print(arr[i] + "  ");
                }

        }
}

// Time Complexity - -  O(kn) => O(n)
// Aux Space - O(1)
```

Problem6B.java

```
package com.jlcindia.arrays;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
// Rotate the Elements Left side K times.

public class Problem6B {
        static void reverse(int arr[],int start,int end) {

                while (start < end) {
                        int temp = arr[start];
                        arr[start] = arr[end];
                        arr[end] = temp;

                        start++;
                        end--;
                }
        }
```

```
        static void leftRotate(int arr[], int k) {

                int n= arr.length;

                reverse(arr,0,k-1);
                reverse(arr,k,n-1);
                reverse(arr,0,n-1);

        }

        public static void main(String[] args) {

                int arr[] = { 1, 2, 3, 4, 5};
                int k = 5; // k<=n

                for (int i = 0; i < arr.length; i++) {
                        System.out.print(arr[i] + "  ");
                }

                leftRotate(arr,k);

                System.out.println("\n  ");

                for (int i = 0; i < arr.length; i++) {
                        System.out.print(arr[i] + "  ");
                }

        }
}

// Time Complexity - -  O(kn) => O(n)
// Aux Space - O(1)
```

Note: Remaining Array Problems will be discussed in LeetCode Sessions