**Arrays and Constructor**

1.  Create a class illustrating all the three types of constructors

● No arguments constructor

● Default constructor

● Parameterised constructor (can create more than one with different type of parameters)

**CODE:**

public class Constructors{

public Constructors() {

System.out.println("No-argument constructor called");

}

public Constructors(int value) {

System.out.println("Default constructor called with parameter: " + value);

}

public Constructors(String name, int age) {

System.out.println("Parameterized constructor called with name: " + name + " and age: " + age);

}

public static void main(String[] args) {

Constructors obj1 = new Constructors();

Constructors obj2 = new Constructors(10);

Constructors obj3 = new Constructors("John", 20);

}

}

2.  Given a sorted integer array (in increasing order), remove duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same. Then return the number of unique elements in the array.

**Input**

[22,22,77,77,88, 89,89]

**Output**

4

**Explanation :**After removing duplicates -> [22, 77, 88, 89, \_, \_, \_ ]

No. of unique elements = 4

**CODE:**

import java.util.Arrays;

import java.util.Scanner;

public class Duplicates {

public static int removeDuplicates(int[] nums) {

if (nums == null || nums.length == 0) {

return 0;

}

int uniqueCount = 1;

int i = 1;

for (int j = 1; j < nums.length; j++) {

if (nums[j] != nums[j - 1]) {

nums[i] = nums[j];

i++;

uniqueCount++;

}

}

return uniqueCount;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] nums = new int[size];

System.out.println("Enter the elements of the sorted array:");

for (int i = 0; i < size; i++) {

nums[i] = scanner.nextInt();

}

scanner.close();

System.out.println("Original array: " + Arrays.toString(nums));

int uniqueCount = removeDuplicates(nums);

System.out.println("After removing duplicates: " + Arrays.toString(Arrays.copyOf(nums, uniqueCount)));

System.out.println("Number of unique elements: " + uniqueCount);

}

}

3 .  An array contains both positive and negative numbers in random order. Rearrange the array elements so that all negative numbers appear before all positive numbers. Don’t use .sort() method

**Input**[-12, 11, -13, -5, 6, -7, 5, -3, -6]

**Output**[-12, -13, -5, -7, -3, -6, 11, 6, 5]

CODE:

import java.util.Arrays;

import java.util.Scanner;

public class Rearrange{

public static void rearrangeArray(int[] arr) {

int left = 0;

int right = arr.length - 1;

while (left <= right) {

while (left <= right && arr[left] < 0) {

left++;

}

while (left <= right && arr[right] >= 0) {

right--;

}

if (left <= right) {

int temp = arr[left];

arr[left] = arr[right];

arr[right] = temp;

left++;

right--;

}

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] arr = new int[size];

System.out.println("Enter the elements of the array:");

for (int i = 0; i < size; i++) {

arr[i] = scanner.nextInt();

}

scanner.close();

System.out.println("Original array: " + Arrays.toString(arr));

rearrangeArray(arr);

System.out.println("After rearranging: " + Arrays.toString(arr));

}

}