**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 ConstructorsExample {

private int value;

public ConstructorsExample() {

System.out.println("No arguments constructor called");

this.value = 0;

}

public ConstructorsExample(int value) {

System.out.println("Default constructor called");

this.value = value;

}

public ConstructorsExample(String message) {

System.out.println("Parameterized constructor with String parameter called: " + message);

this.value = 0;

}

public ConstructorsExample(int value, String message) {

System.out.println("Parameterized constructor with int and String parameters called: " + value + ", " + message);

this.value = value;

}

public int getValue() {

return this.value;

}

public static void main(String[] args) {

ConstructorsExample obj1 = new ConstructorsExample();

System.out.println("Value from no arguments constructor: " + obj1.getValue());

ConstructorsExample obj2 = new ConstructorsExample(5);

System.out.println("Value from default constructor: " + obj2.getValue());

ConstructorsExample obj3 = new ConstructorsExample("Hello");

System.out.println("Value from String parameterized constructor: " + obj3.getValue());

ConstructorsExample obj4 = new ConstructorsExample(10, "World");

System.out.println("Value from int and String parameterized constructor: " + obj4.getValue());

}

}

**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 :**

public class RemoveDuplicates {

public static int removeDuplicates(int[] nums) {

if (nums.length == 0 || nums.length == 1) {

return nums.length;

}

int uniqueCount = 1;

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

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

nums[uniqueCount] = nums[i];

uniqueCount++;

}

}

return uniqueCount;

}

public static void main(String[] args) {

int[] inputArray = {22, 22, 77, 77, 88, 89, 89};

int uniqueElements = removeDuplicates(inputArray);

System.out.println("No. of unique elements: " + uniqueElements);

System.out.print("Array after removing duplicates: ");

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

System.out.print(inputArray[i] + " ");

}

}

}

**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 :**

public class RearrangeArray {

private int[] array;

public RearrangeArray(int[] array) {

this.array = array;

}

public void rearrangePositiveNegative() {

int left = 0;

int right = array.length - 1;

while (left <= right) {

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

left++;

}

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

right--;

}

if (left <= right) {

int temp = array[left];

array[left] = array[right];

array[right] = temp;

left++;

right--;

}

}

}

public void printArray() {

for (int num : array) {

System.out.print(num + " ");

}

System.out.println();

}

public static void main(String[] args) {

int[] inputArray = {-12, 11, -13, -5, 6, -7, 5, -3, -6};

RearrangeArray rearrangeArray = new RearrangeArray(inputArray);

System.out.println("Input Array:");

rearrangeArray.printArray();

rearrangeArray.rearrangePositiveNegative();

System.out.println("\nOutput Array after rearranging positive and negative numbers:");

rearrangeArray.printArray();

}

}