**QUIZ 13**

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)

public class ConstructorExample {

public ConstructorExample() {

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

}

public ConstructorExample(int dummy) {

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

}

public ConstructorExample(String param1) {

System.out.println("Parameterized constructor with one parameter called. Param1: " + param1);

}

public ConstructorExample(String param1, int param2) {

System.out.println("Parameterized constructor with two parameters called. Param1: " + param1 + ", Param2: " + param2);

}

public static void main(String[] args) {

ConstructorExample obj1 = new ConstructorExample();

ConstructorExample obj2 = new ConstructorExample(0);

ConstructorExample obj3 = new ConstructorExample("Hello");

ConstructorExample obj4 = new ConstructorExample("Hello", 42);

}

}

OUTPUT:

No arguments constructor called

Default constructor called

Parameterized constructor with one parameter called. Param1: Hello

Parameterized constructor with two parameters called. Param1: Hello, Param2: 42

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

public class RemoveDuplicates {

public static int removeDuplicates(int[] nums) {

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

return 0; // Empty array has 0 unique elements

}

int uniqueCount = 1; // At least one element is unique

int n = nums.length;

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

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

nums[uniqueCount] = nums[i];

uniqueCount++;

}

}

return uniqueCount;

}

public static void main(String[] args) {

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

int uniqueCount = removeDuplicates(nums);

System.out.println("No. of unique elements = " + uniqueCount);

System.out.print("After removing duplicates -> [");

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

System.out.print(nums[i]);

if (i < uniqueCount - 1) {

System.out.print(", ");

}

}

System.out.println("]");

}

}

OUTPUT:

No. of unique elements = 4

After removing duplicates -> [22, 77, 88, 89]

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]

public class RearrangeArray {

public static void rearrangeArray(int[] arr) {

int n = arr.length;

int negIndex = 0;

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

if (arr[i] < 0) {

if (i != negIndex) {

int temp = arr[i];

arr[i] = arr[negIndex];

arr[negIndex] = temp;

}

negIndex++;

}

}

}

public static void main(String[] args) {

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

rearrangeArray(arr);

System.out.print("Output [");

for (int i = 0; i < arr.length; i++) {

System.out.print(arr[i]);

if (i < arr.length - 1) {

System.out.print(", ");

}

}

System.out.println("]");

}

}

OUTPUT:

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