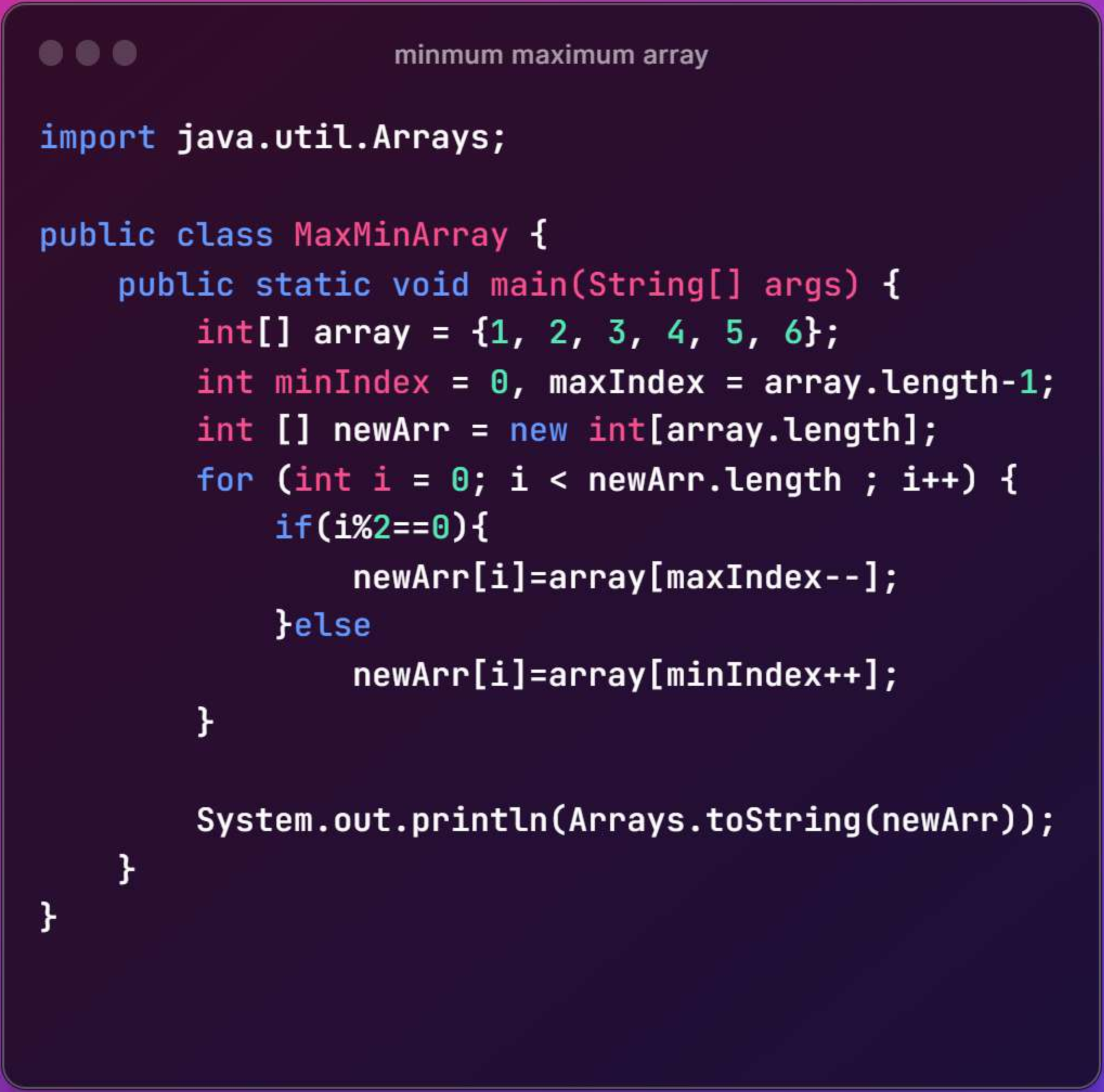
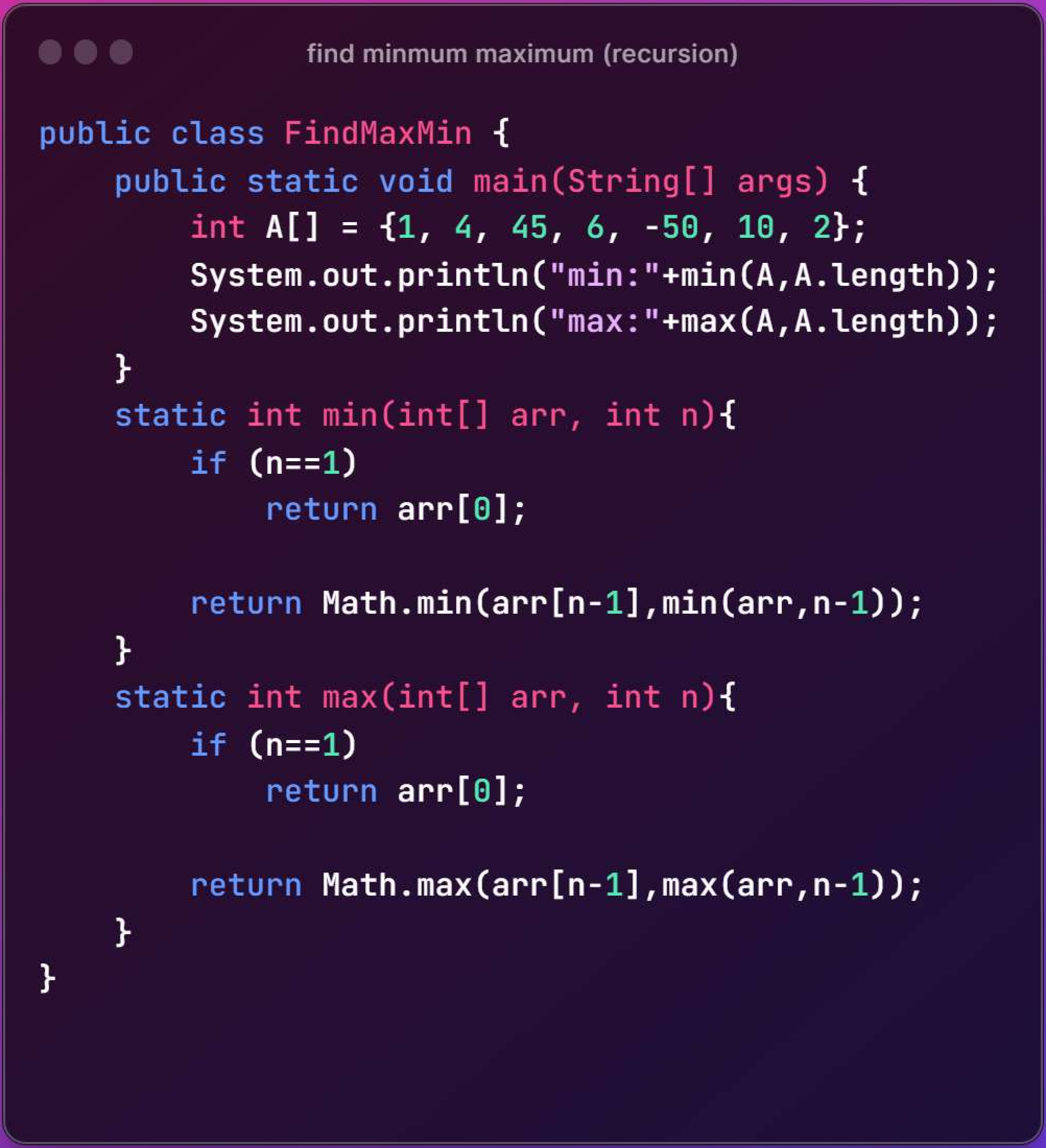
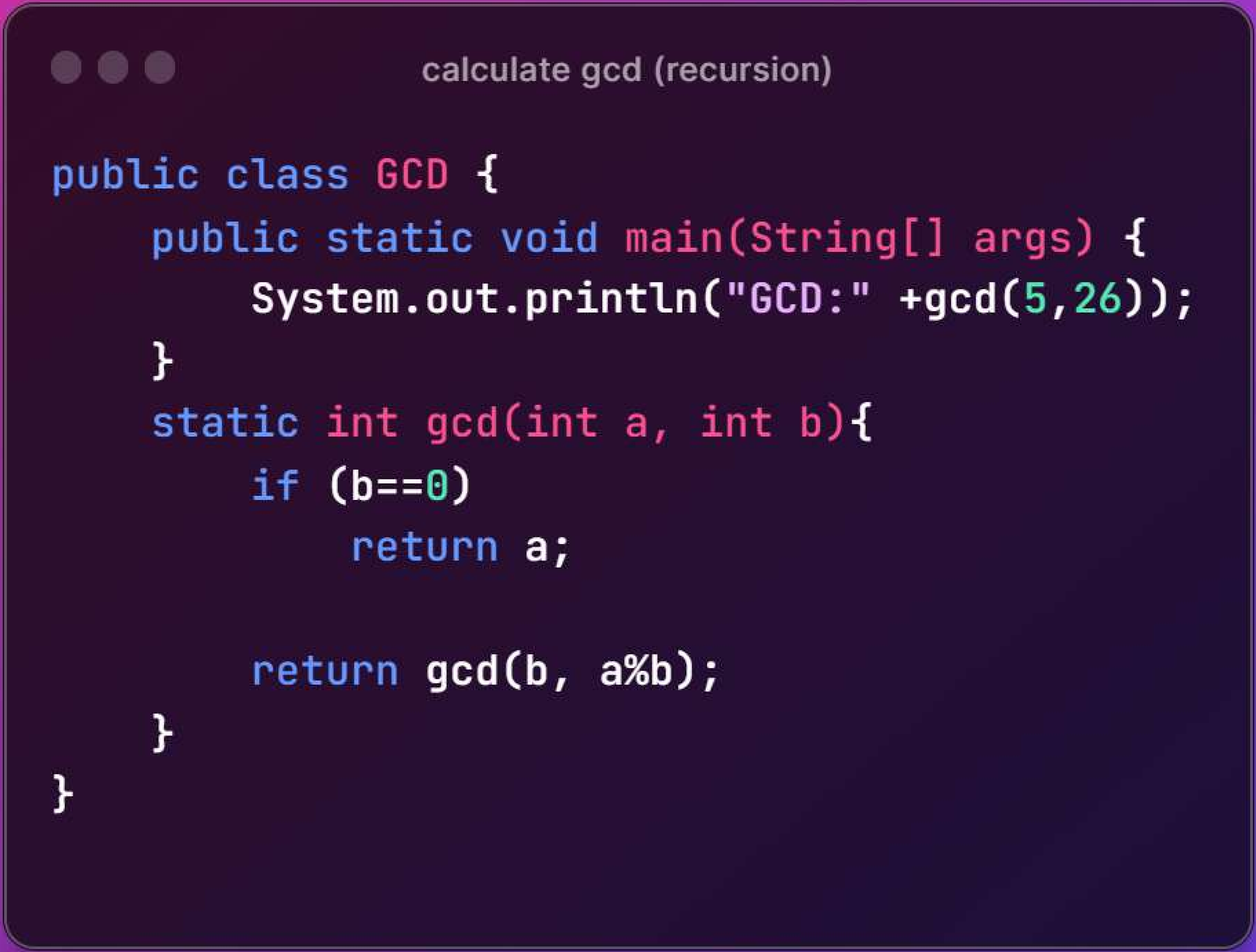


```java
import java.util.Arrays;

public class MaxMinArray {
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5, 6};
        int minIndex = 0, maxIndex = array.length-1;
        int [] newArr = new int[array.length];
        for (int i = 0; i < newArr.length ; i++) {
            if(i%2==0){
                newArr[i]=array[maxIndex--];
            }else
                newArr[i]=array[minIndex++];
        }

        System.out.println(Arrays.toString(newArr));
    }
}
```



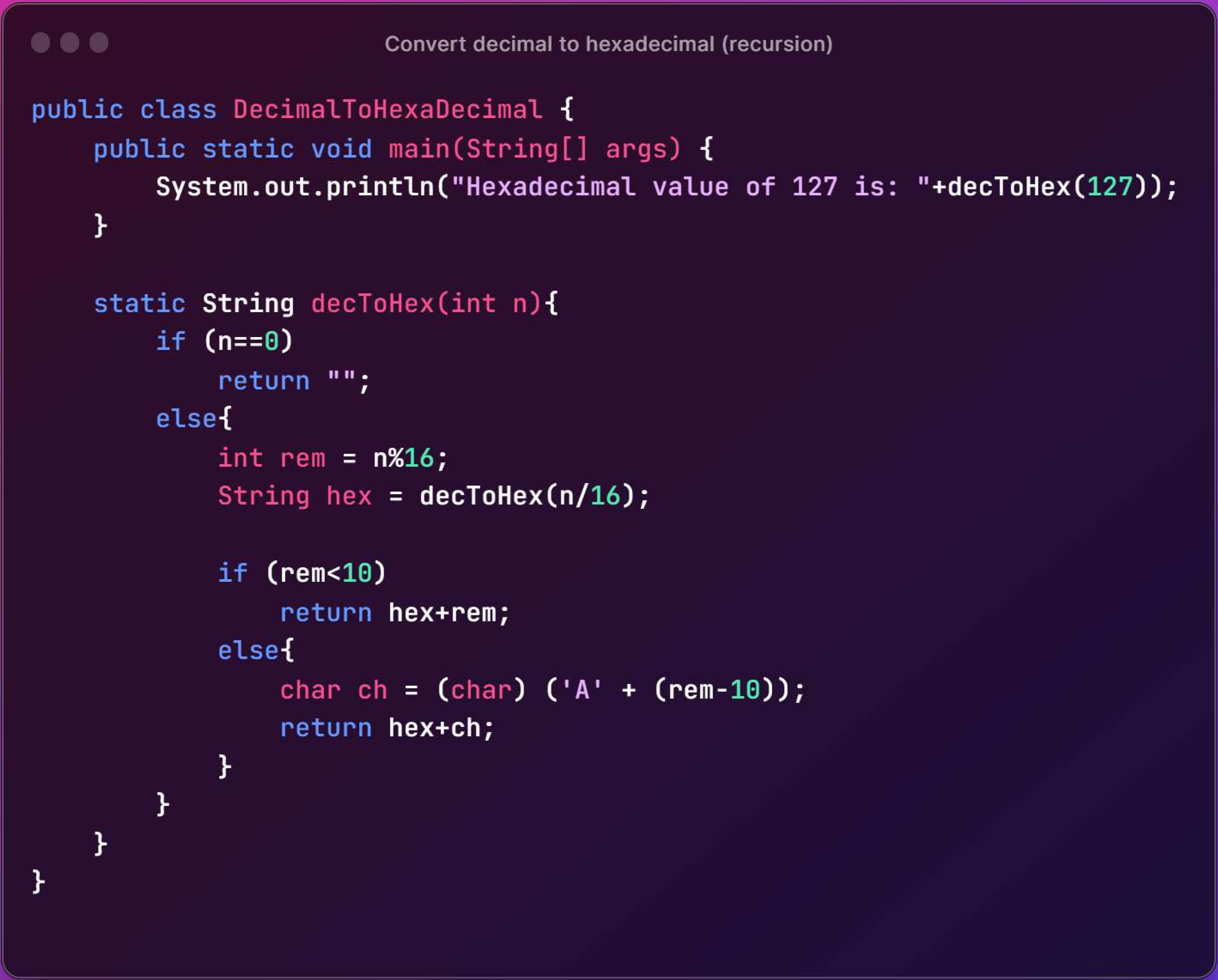
find minmum maximum (recursion)

```java
public class FindMaxMin {
    public static void main(String[] args) {
        int A[] = {1, 4, 45, 6, -50, 10, 2};
        System.out.println("min:"+min(A,A.length));
        System.out.println("max:"+max(A,A.length));
    }
    static int min(int[] arr, int n){
        if (n==1)
            return arr[0];

        return Math.min(arr[n-1],min(arr,n-1));
    }
    static int max(int[] arr, int n){
        if (n==1)
            return arr[0];

        return Math.max(arr[n-1],max(arr,n-1));
    }
}
```



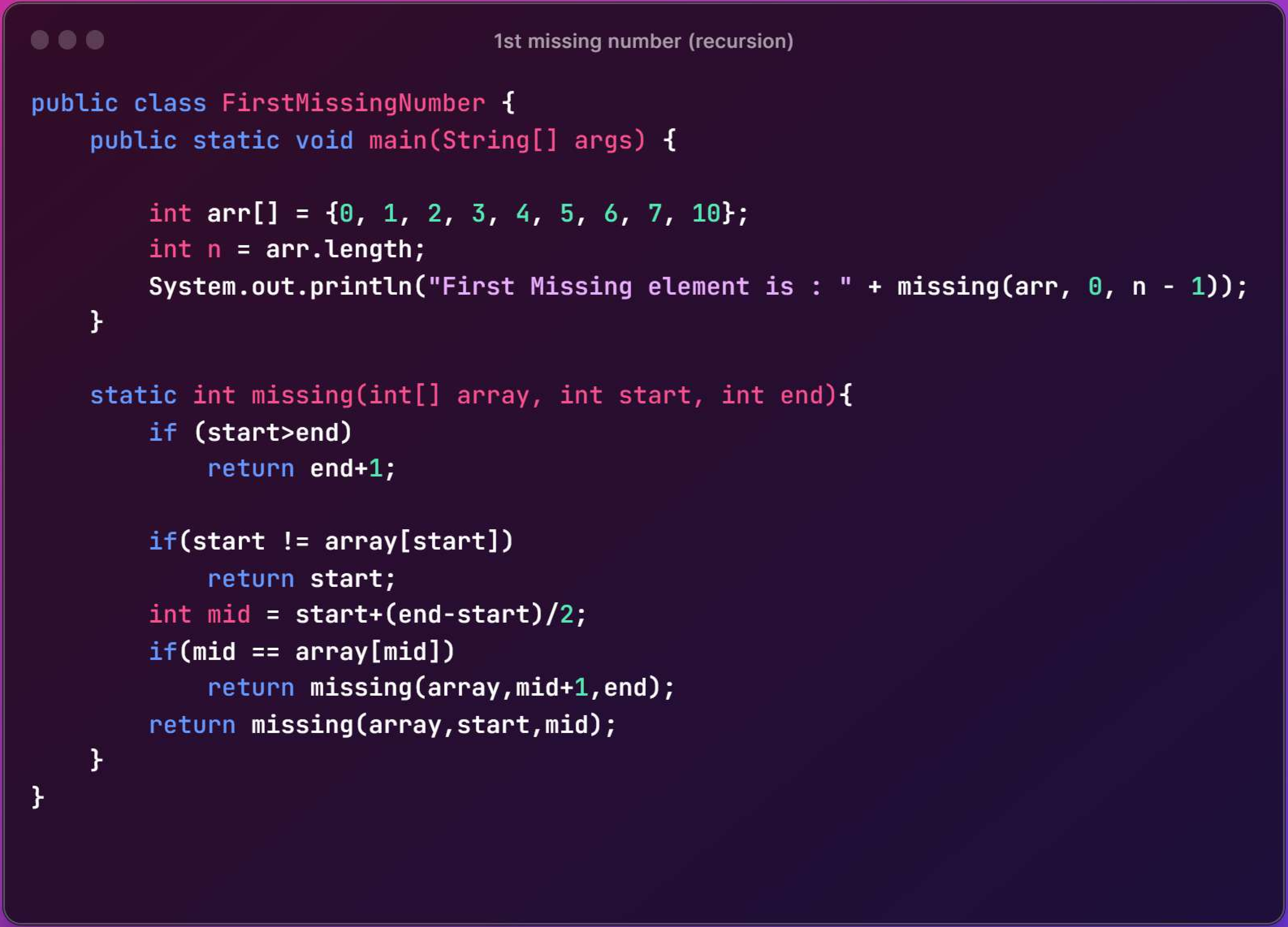
calculate gcd (recursion)

```java
public class GCD {
    public static void main(String[] args) {
        System.out.println("GCD:" +gcd(5,26));
    }
    static int gcd(int a, int b){
        if (b==0)
            return a;

        return gcd(b, a%b);
    }
}
```



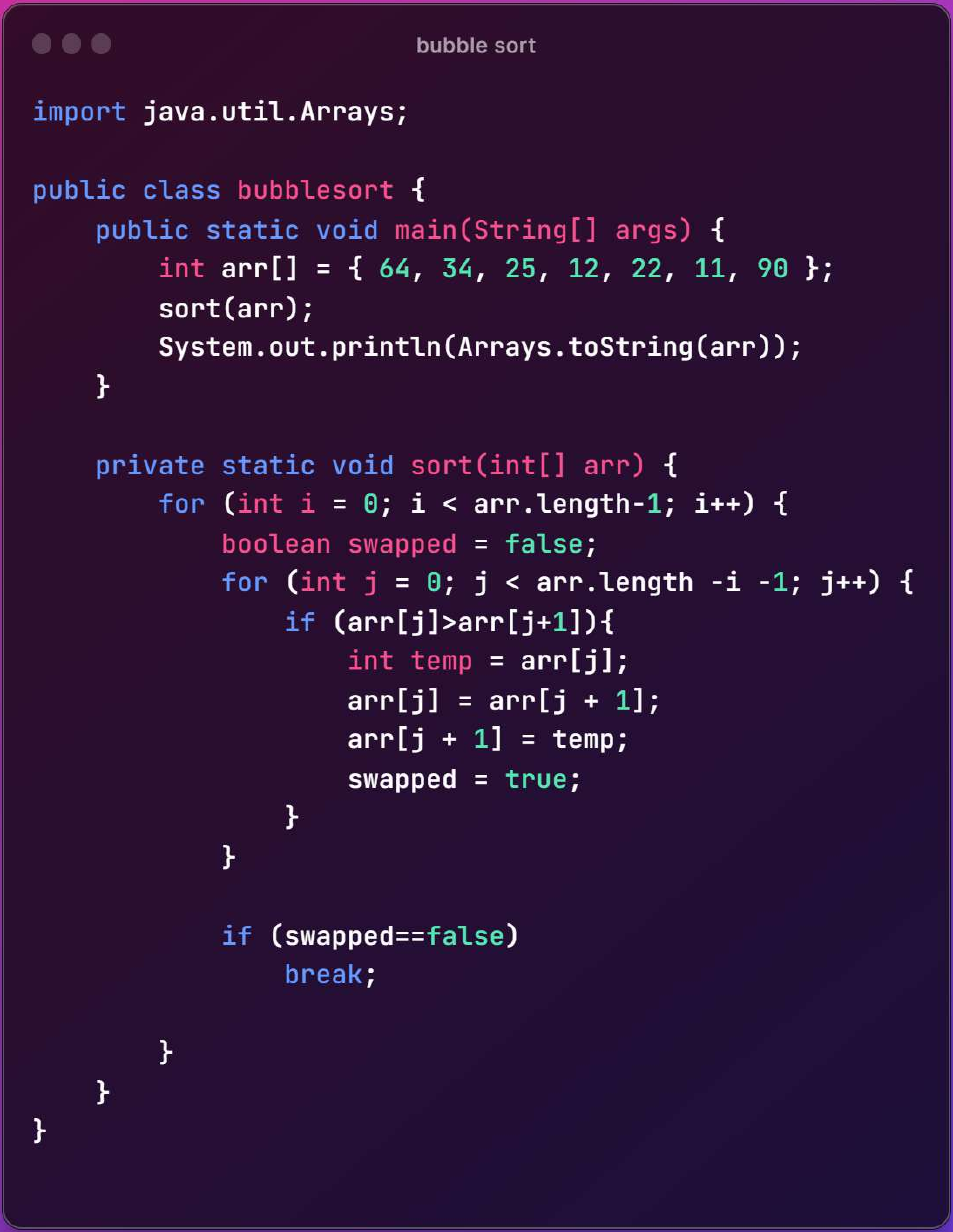
Convert decimal to hexadecimal (recursion)

```java
public class DecimalToHexaDecimal {
    public static void main(String[] args) {
        System.out.println("Hexadecimal value of 127 is: "+decToHex(127));
    }

    static String decToHex(int n){
        if (n==0)
            return "";
        else{
            int rem = n%16;
            String hex = decToHex(n/16);

            if (rem<10)
                return hex+rem;
            else{
                char ch = (char) ('A' + (rem-10));
                return hex+ch;
            }
        }
    }
}
```

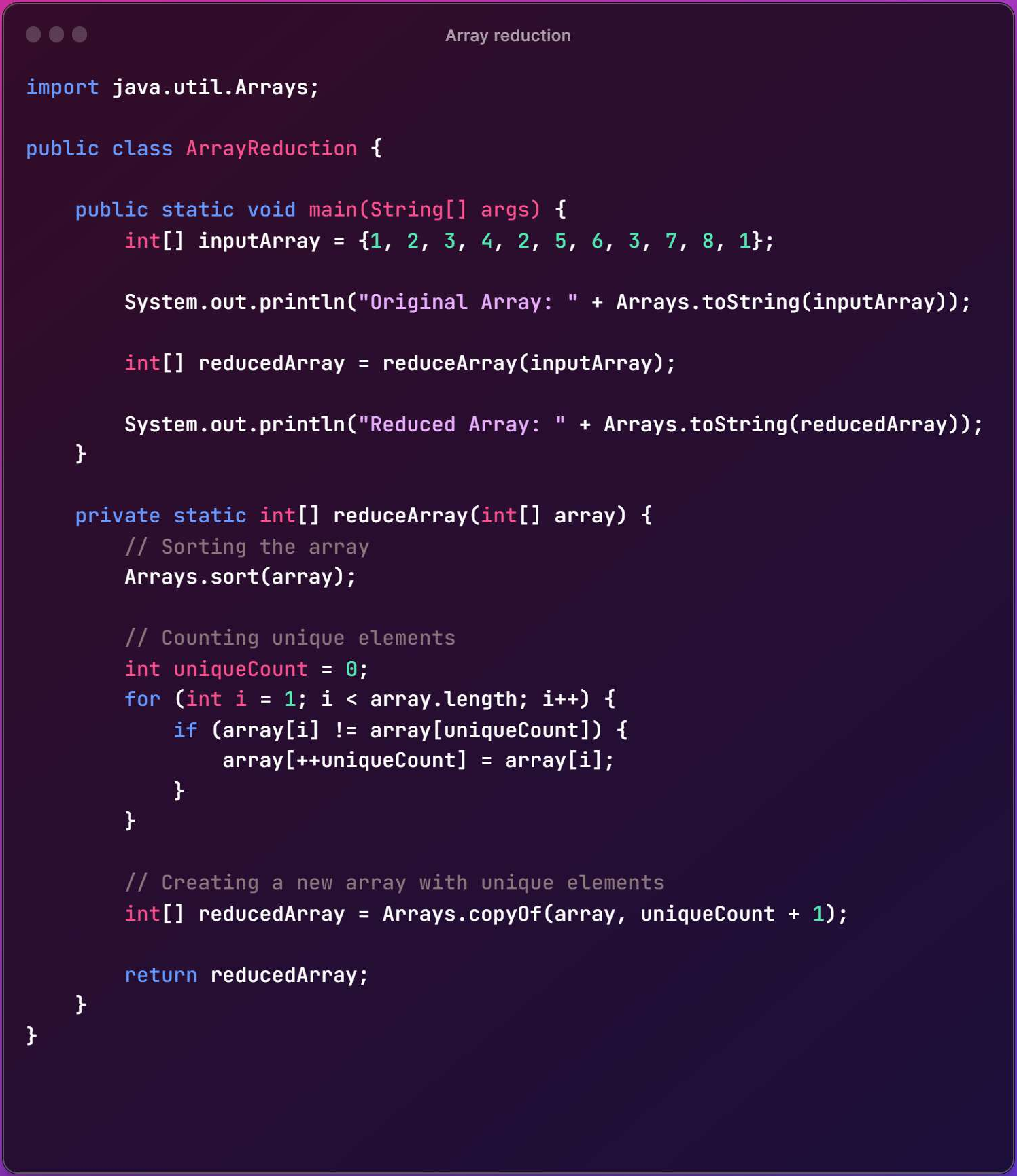


1st missing number (recursion)

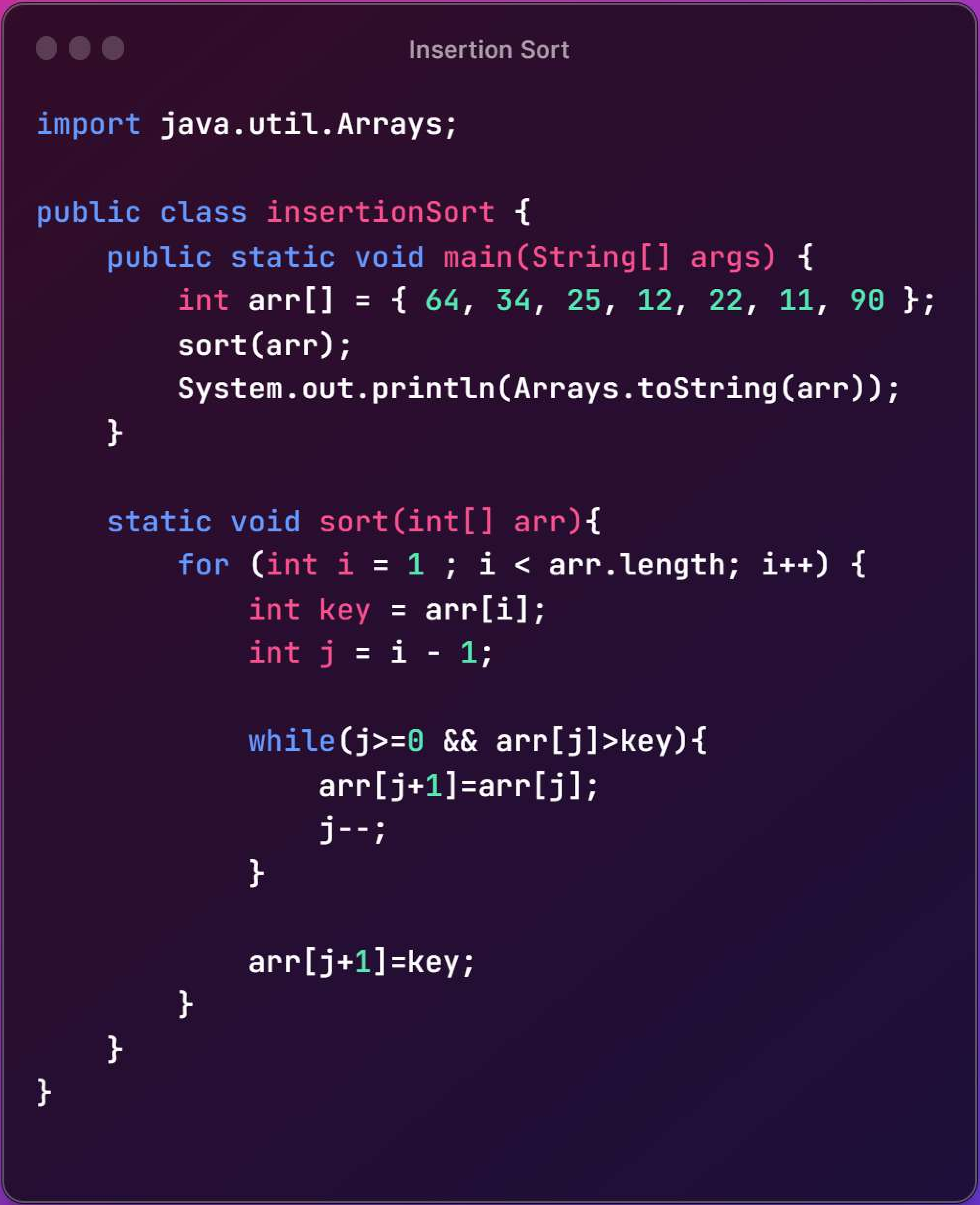
```java
public class FirstMissingNumber {
    public static void main(String[] args) {

        int arr[] = {0, 1, 2, 3, 4, 5, 6, 7, 10};
        int n = arr.length;
        System.out.println("First Missing element is : " + missing(arr, 0, n - 1));
    }

    static int missing(int[] array, int start, int end){
        if (start>end)
            return end+1;

        if(start != array[start])
            return start;
        int mid = start+(end-start)/2;
        if(mid == array[mid])
            return missing(array,mid+1,end);
        return missing(array,start,mid);
    }
}
```



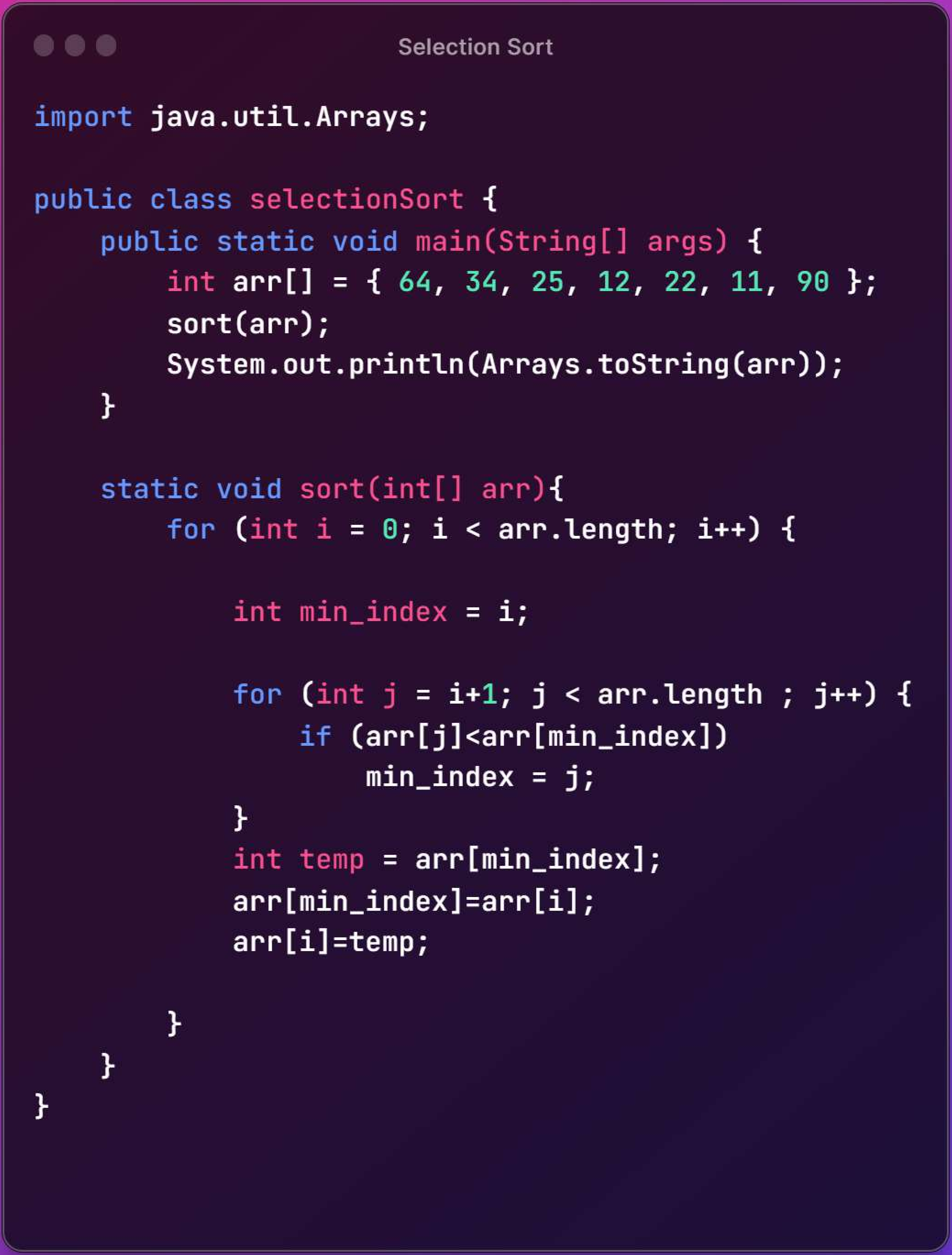
```java
import java.util.Arrays;

public class bubblesort {
    public static void main(String[] args) {
        int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
        sort(arr);
        System.out.println(Arrays.toString(arr));
    }

    private static void sort(int[] arr) {
        for (int i = 0; i < arr.length-1; i++) {
            boolean swapped = false;
            for (int j = 0; j < arr.length -i -1; j++) {
                if (arr[j]>arr[j+1]){
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }

            if (swapped==false)
                break;

        }
    }
}
```



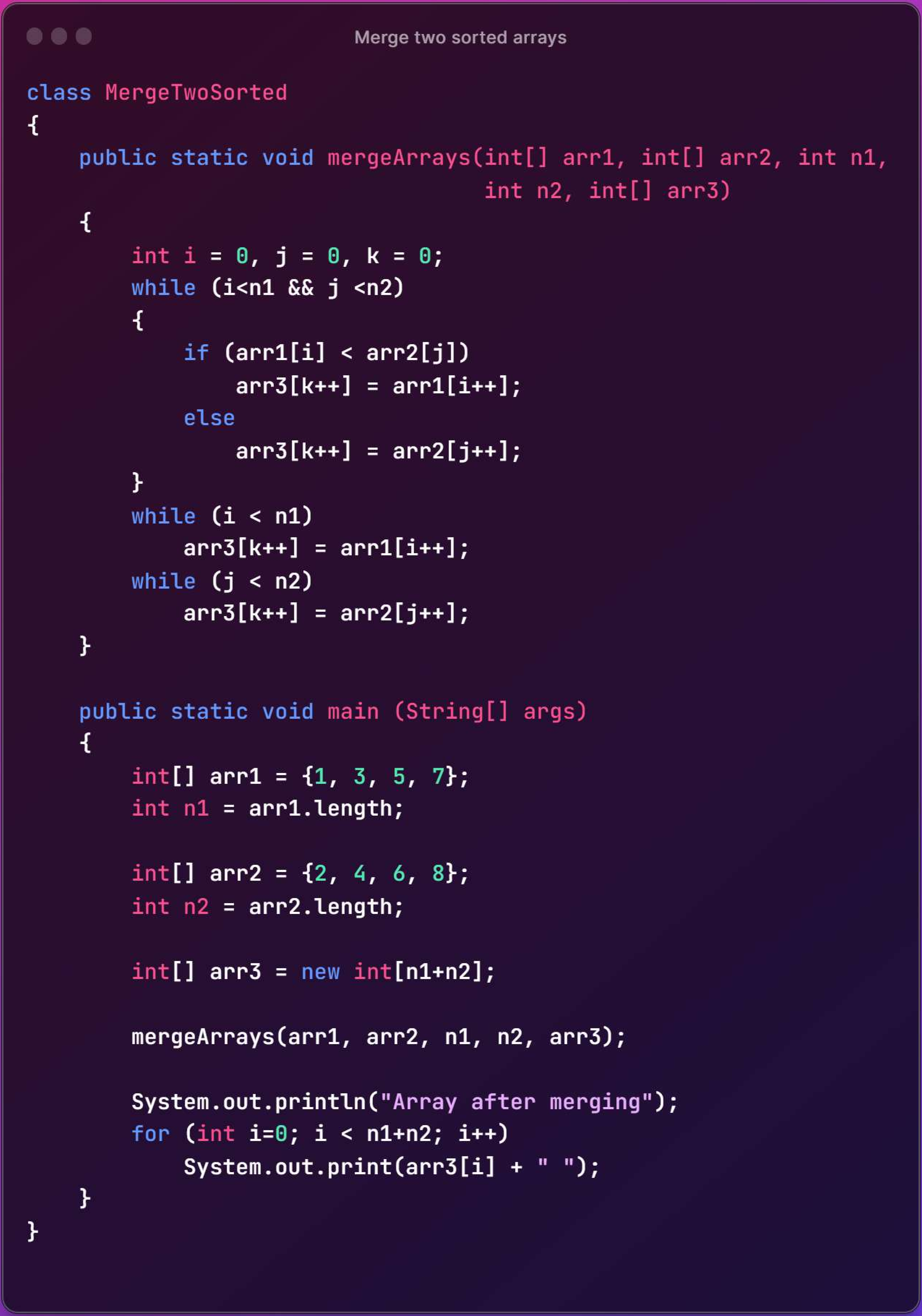
```java
import java.util.Arrays;

public class ArrayReduction {

    public static void main(String[] args) {
        int[] inputArray = {1, 2, 3, 4, 2, 5, 6, 3, 7, 8, 1};

        System.out.println("Original Array: " + Arrays.toString(inputArray));

        int[] reducedArray = reduceArray(inputArray);

        System.out.println("Reduced Array: " + Arrays.toString(reducedArray));
    }

    private static int[] reduceArray(int[] array) {
        // Sorting the array
        Arrays.sort(array);

        // Counting unique elements
        int uniqueCount = 0;
        for (int i = 1; i < array.length; i++) {
            if (array[i] != array[uniqueCount]) {
                array[++uniqueCount] = array[i];
            }
        }

        // Creating a new array with unique elements
        int[] reducedArray = Arrays.copyOf(array, uniqueCount + 1);

        return reducedArray;
    }
}
```



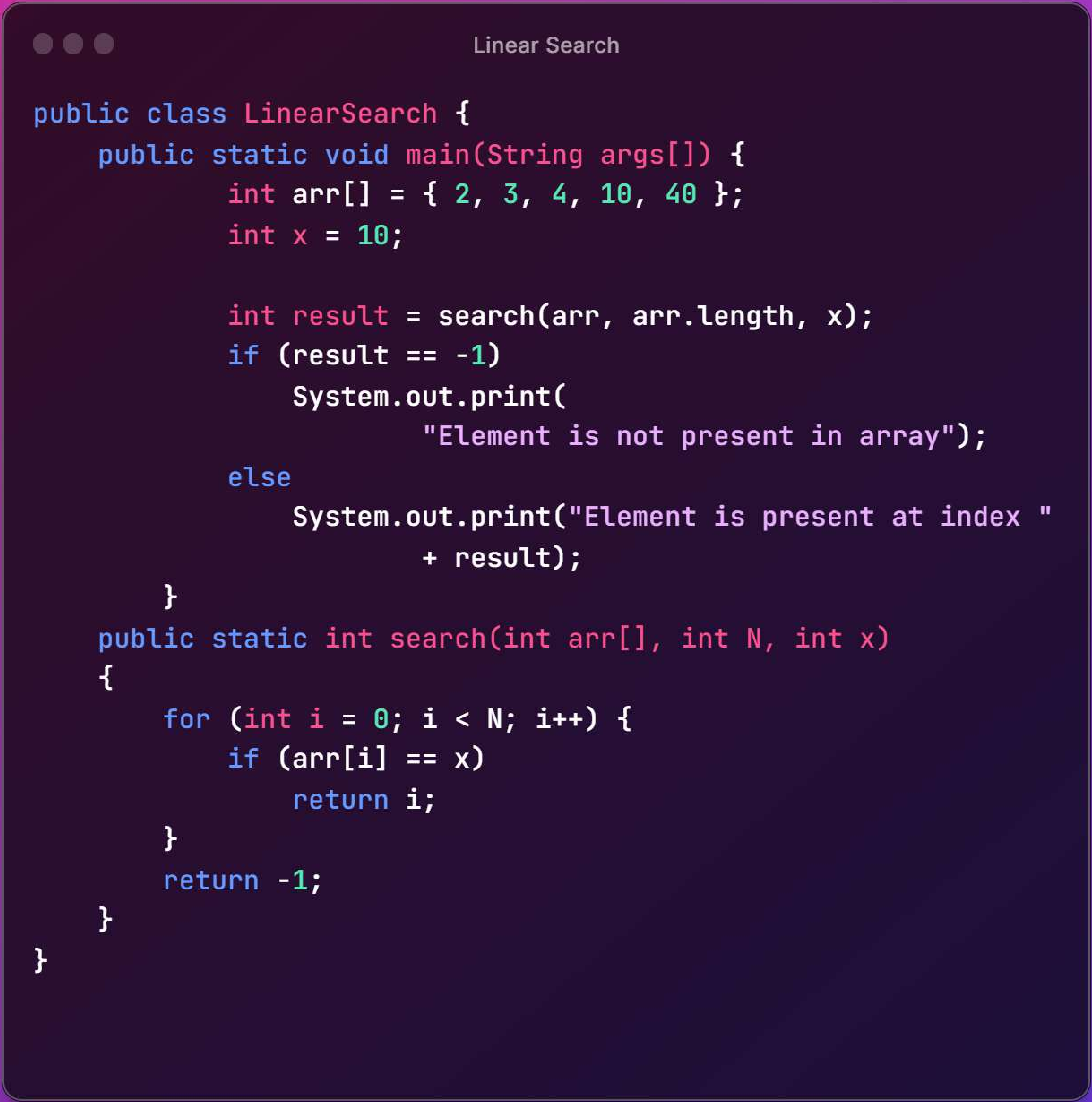
```java
import java.util.Arrays;

public class insertionSort {
    public static void main(String[] args) {
        int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
        sort(arr);
        System.out.println(Arrays.toString(arr));
    }

    static void sort(int[] arr){
        for (int i = 1 ; i < arr.length; i++) {
            int key = arr[i];
            int j = i - 1;

            while(j>=0 && arr[j]>key){
                arr[j+1]=arr[j];
                j--;
            }

            arr[j+1]=key;
        }
    }
}
```



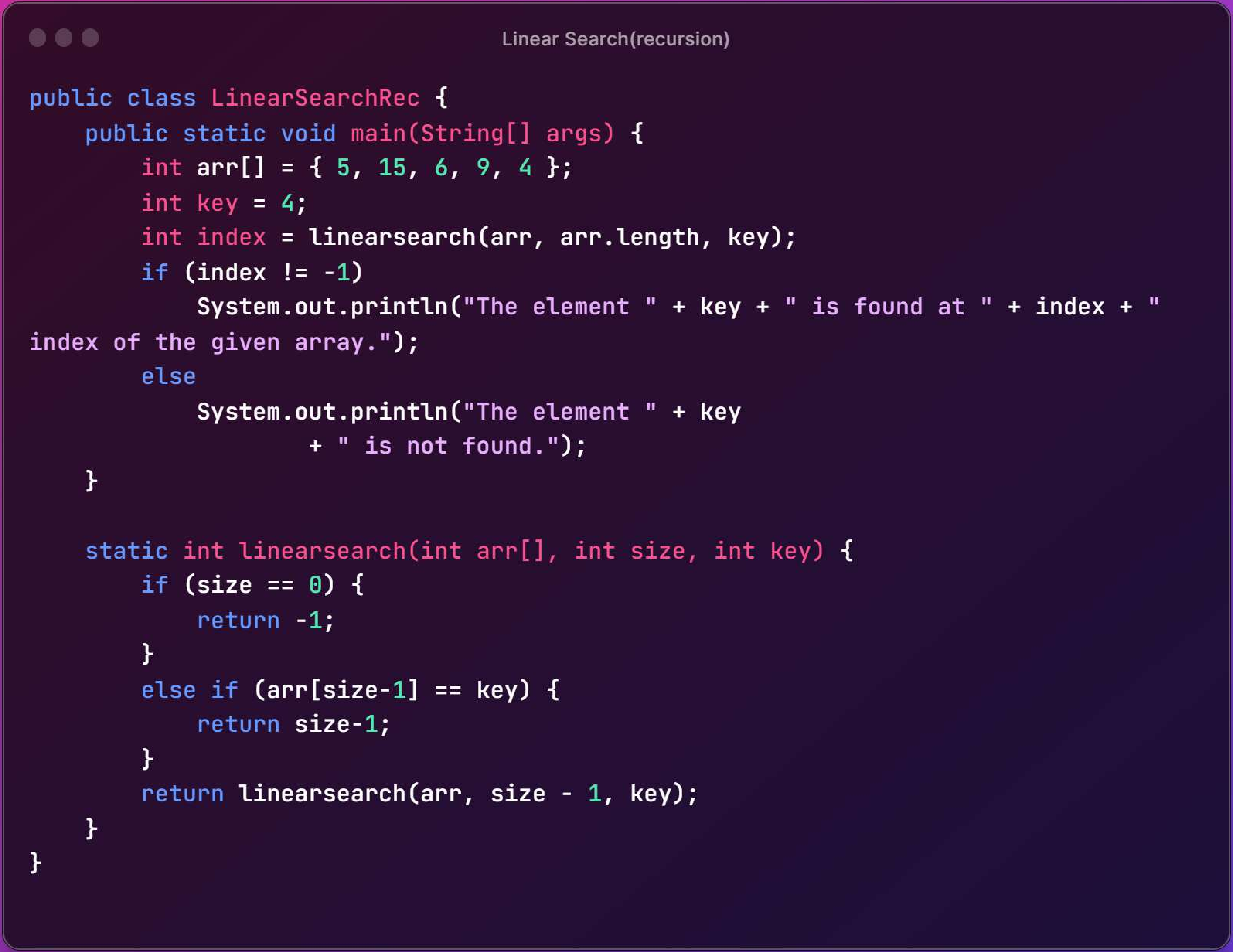
```
import java.util.Arrays;

public class selectionSort {
    public static void main(String[] args) {
        int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
        sort(arr);
        System.out.println(Arrays.toString(arr));
    }

    static void sort(int[] arr){
        for (int i = 0; i < arr.length; i++) {

            int min_index = i;

            for (int j = i+1; j < arr.length ; j++) {
                if (arr[j]<arr[min_index])
                    min_index = j;
            }
            int temp = arr[min_index];
            arr[min_index]=arr[i];
            arr[i]=temp;

        }
    }
}
```



```java
class MergeTwoSorted
{
    public static void mergeArrays(int[] arr1, int[] arr2, int n1,
                                   int n2, int[] arr3)
    {
        int i = 0, j = 0, k = 0;
        while (i<n1 && j <n2)
        {
            if (arr1[i] < arr2[j])
                arr3[k++] = arr1[i++];
            else
                arr3[k++] = arr2[j++];
        }
        while (i < n1)
            arr3[k++] = arr1[i++];
        while (j < n2)
            arr3[k++] = arr2[j++];
    }

    public static void main (String[] args)
    {
        int[] arr1 = {1, 3, 5, 7};
        int n1 = arr1.length;

        int[] arr2 = {2, 4, 6, 8};
        int n2 = arr2.length;

        int[] arr3 = new int[n1+n2];

        mergeArrays(arr1, arr2, n1, n2, arr3);

        System.out.println("Array after merging");
        for (int i=0; i < n1+n2; i++)
            System.out.print(arr3[i] + " ");
    }
}
```

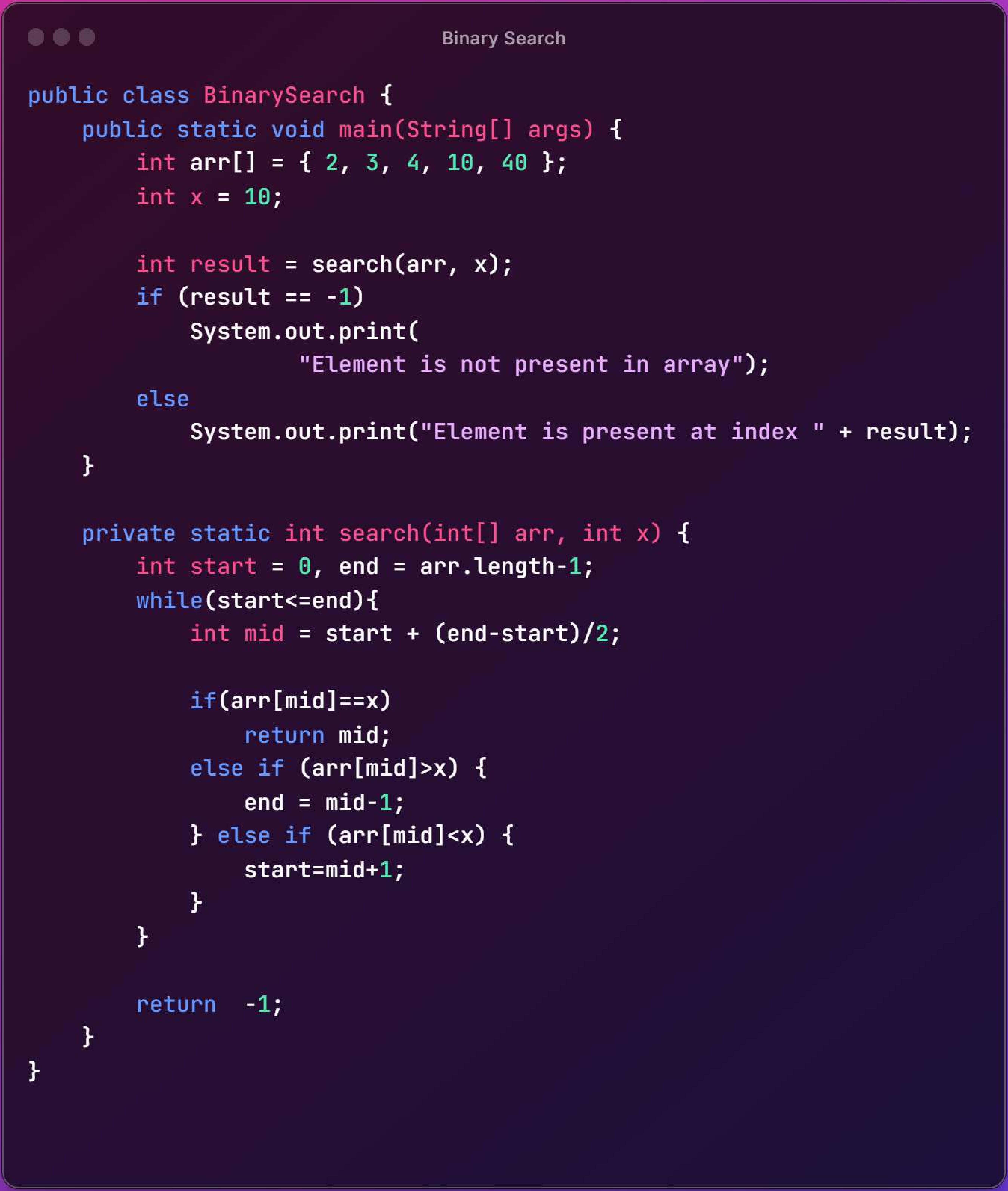


```java
public class LinearSearch {
    public static void main(String args[]) {
            int arr[] = { 2, 3, 4, 10, 40 };
            int x = 10;

            int result = search(arr, arr.length, x);
            if (result == -1)
                System.out.print(
                        "Element is not present in array");
            else
                System.out.print("Element is present at index "
                        + result);
        }
    public static int search(int arr[], int N, int x)
    {
        for (int i = 0; i < N; i++) {
            if (arr[i] == x)
                return i;
        }
        return -1;
    }
}
```

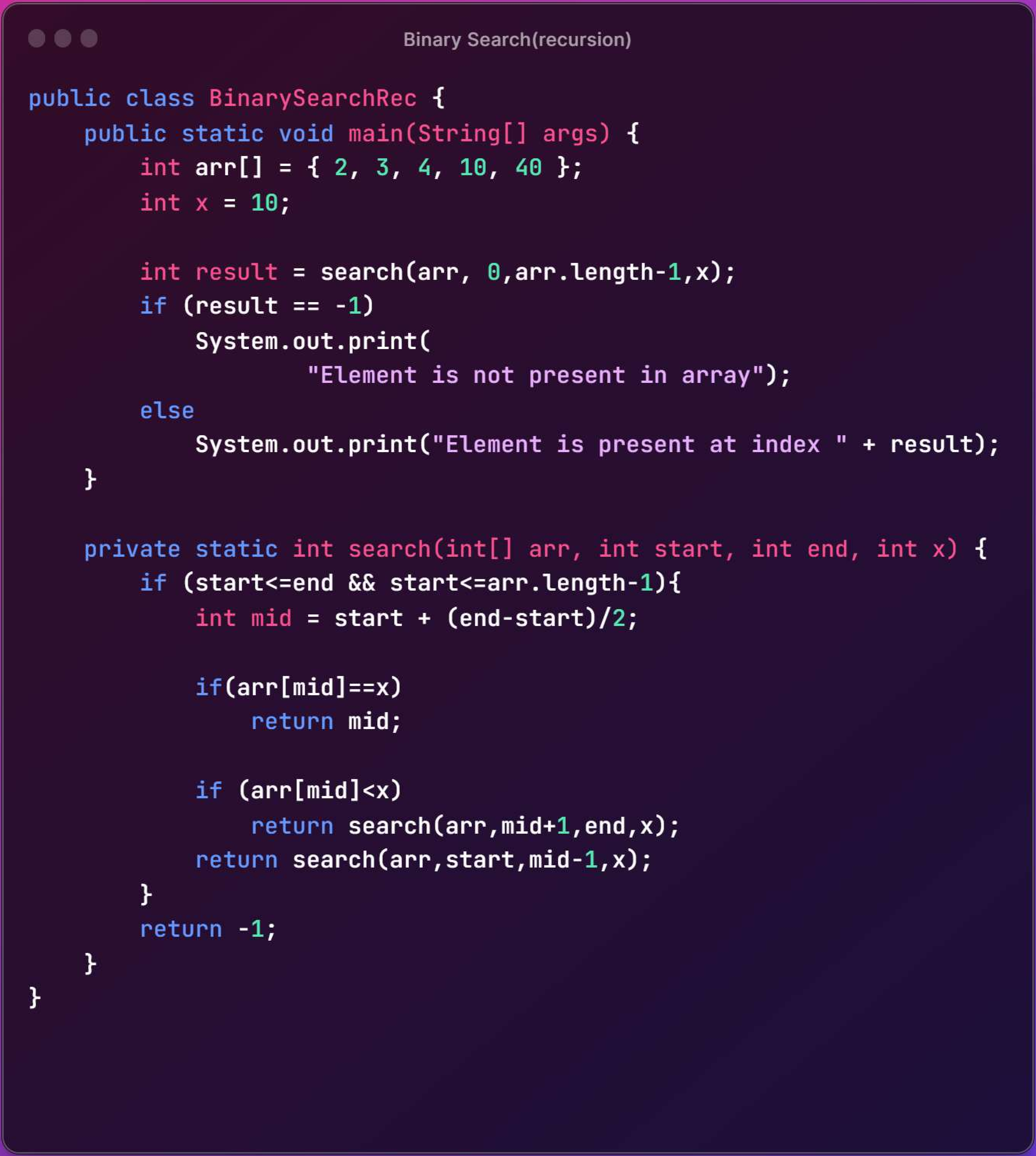


Linear Search(recursion)

```java
public class LinearSearchRec {
    public static void main(String[] args) {
        int arr[] = { 5, 15, 6, 9, 4 };
        int key = 4;
        int index = linearsearch(arr, arr.length, key);
        if (index != -1)
            System.out.println("The element " + key + " is found at " + index + "
index of the given array.");
        else
            System.out.println("The element " + key
                    + " is not found.");
    }

    static int linearsearch(int arr[], int size, int key) {
        if (size == 0) {
            return -1;
        }
        else if (arr[size-1] == key) {
            return size-1;
        }
        return linearsearch(arr, size - 1, key);
    }
}
```

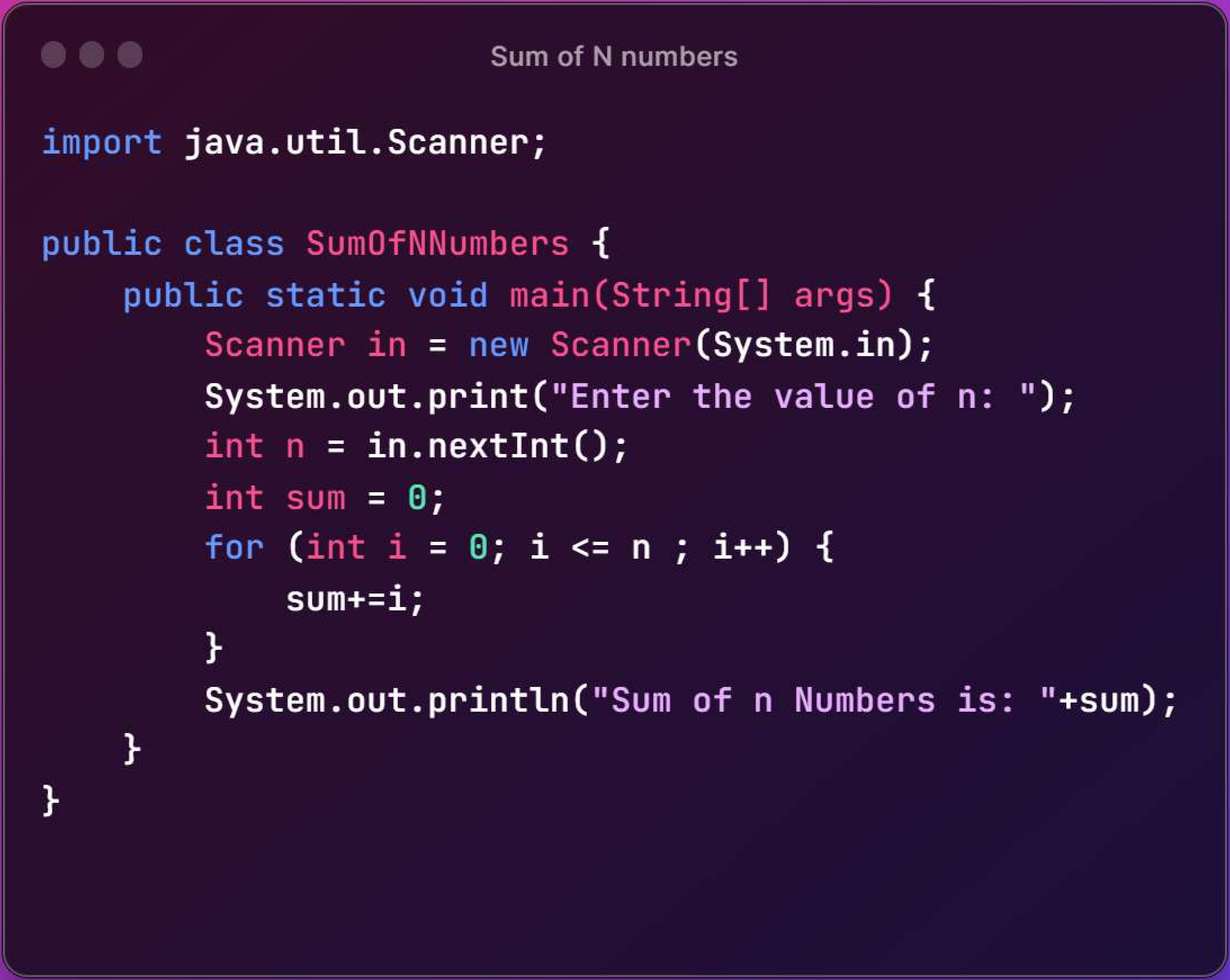


```java
public class BinarySearch {
    public static void main(String[] args) {
        int arr[] = { 2, 3, 4, 10, 40 };
        int x = 10;

        int result = search(arr, x);
        if (result == -1)
            System.out.print(
                    "Element is not present in array");
        else
            System.out.print("Element is present at index " + result);
    }

    private static int search(int[] arr, int x) {
        int start = 0, end = arr.length-1;
        while(start<=end){
            int mid = start + (end-start)/2;

            if(arr[mid]==x)
                return mid;
            else if (arr[mid]>x) {
                end = mid-1;
            } else if (arr[mid]<x) {
                start=mid+1;
            }
        }

        return  -1;
    }
}
```



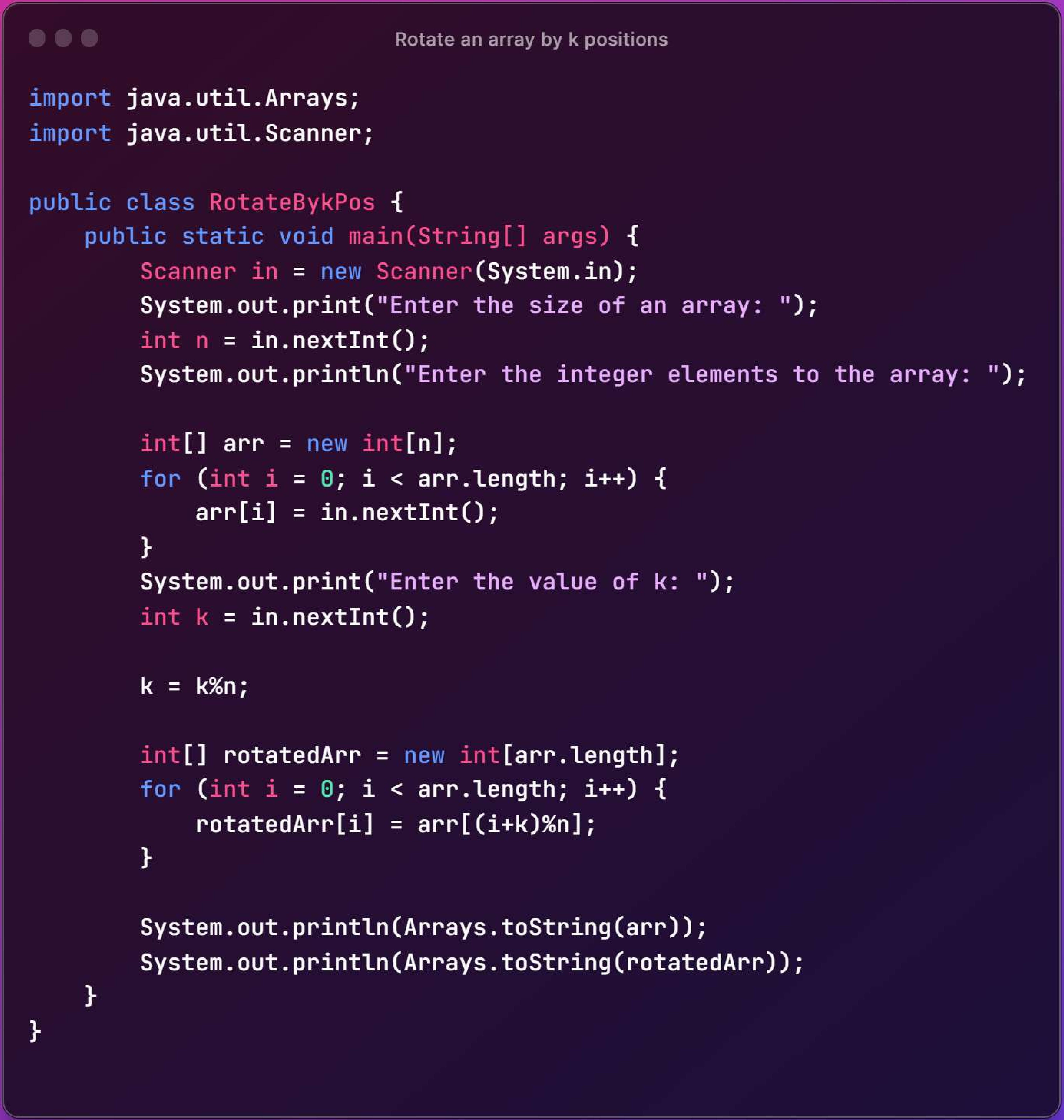
Binary Search(recursion)

```
public class BinarySearchRec {
    public static void main(String[] args) {
        int arr[] = { 2, 3, 4, 10, 40 };
        int x = 10;

        int result = search(arr, 0,arr.length-1,x);
        if (result == -1)
            System.out.print(
                    "Element is not present in array");
        else
            System.out.print("Element is present at index " + result);
    }

    private static int search(int[] arr, int start, int end, int x) {
        if (start<=end && start<=arr.length-1){
            int mid = start + (end-start)/2;

            if(arr[mid]==x)
                return mid;

            if (arr[mid]<x)
                return search(arr,mid+1,end,x);
            return search(arr,start,mid-1,x);
        }
        return -1;
    }
}
```



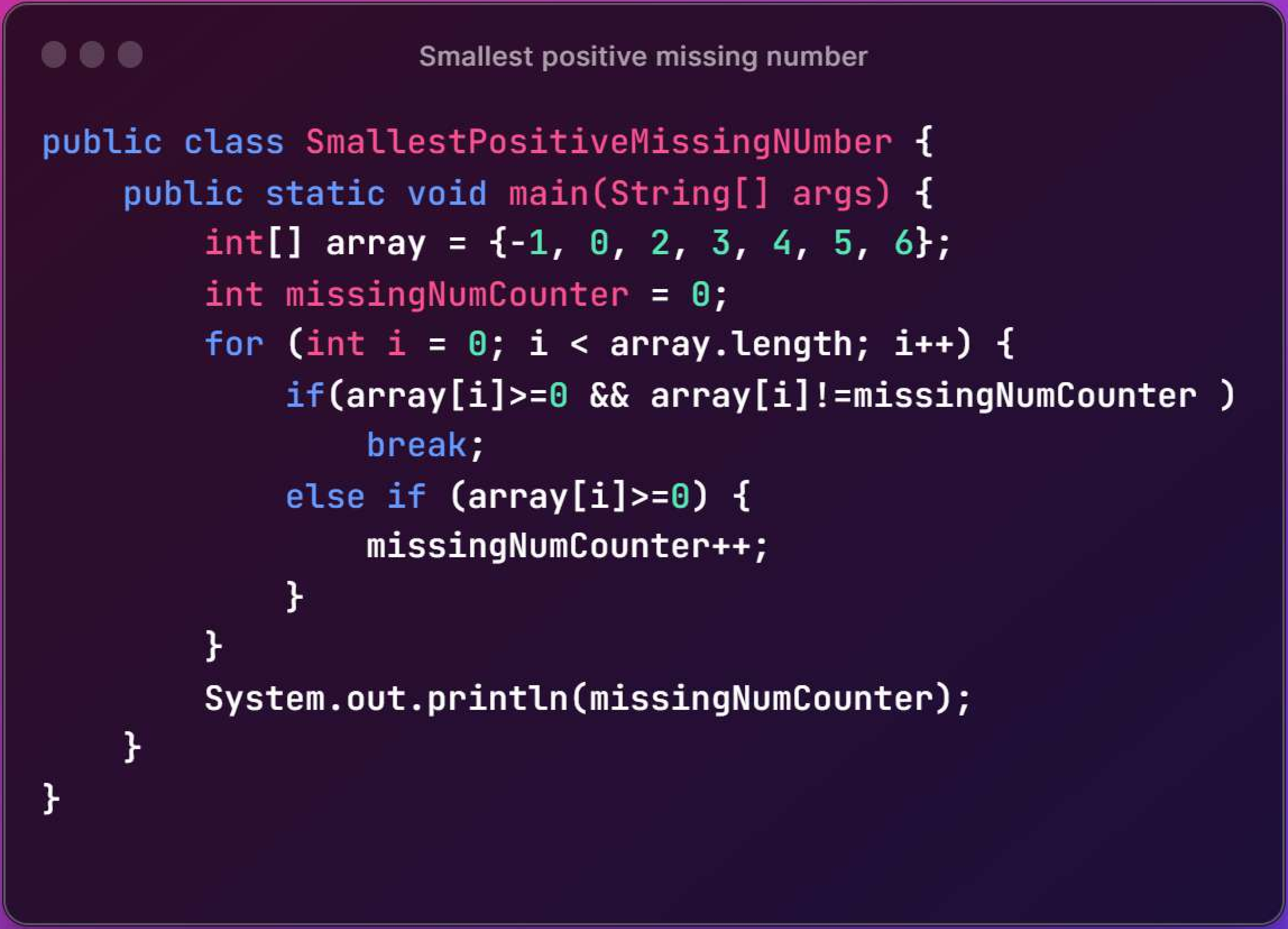
Sum of N numbers

```java
import java.util.Scanner;

public class SumOfNNumbers {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the value of n: ");
        int n = in.nextInt();
        int sum = 0;
        for (int i = 0; i <= n ; i++) {
            sum+=i;
        }
        System.out.println("Sum of n Numbers is: "+sum);
    }
}
```

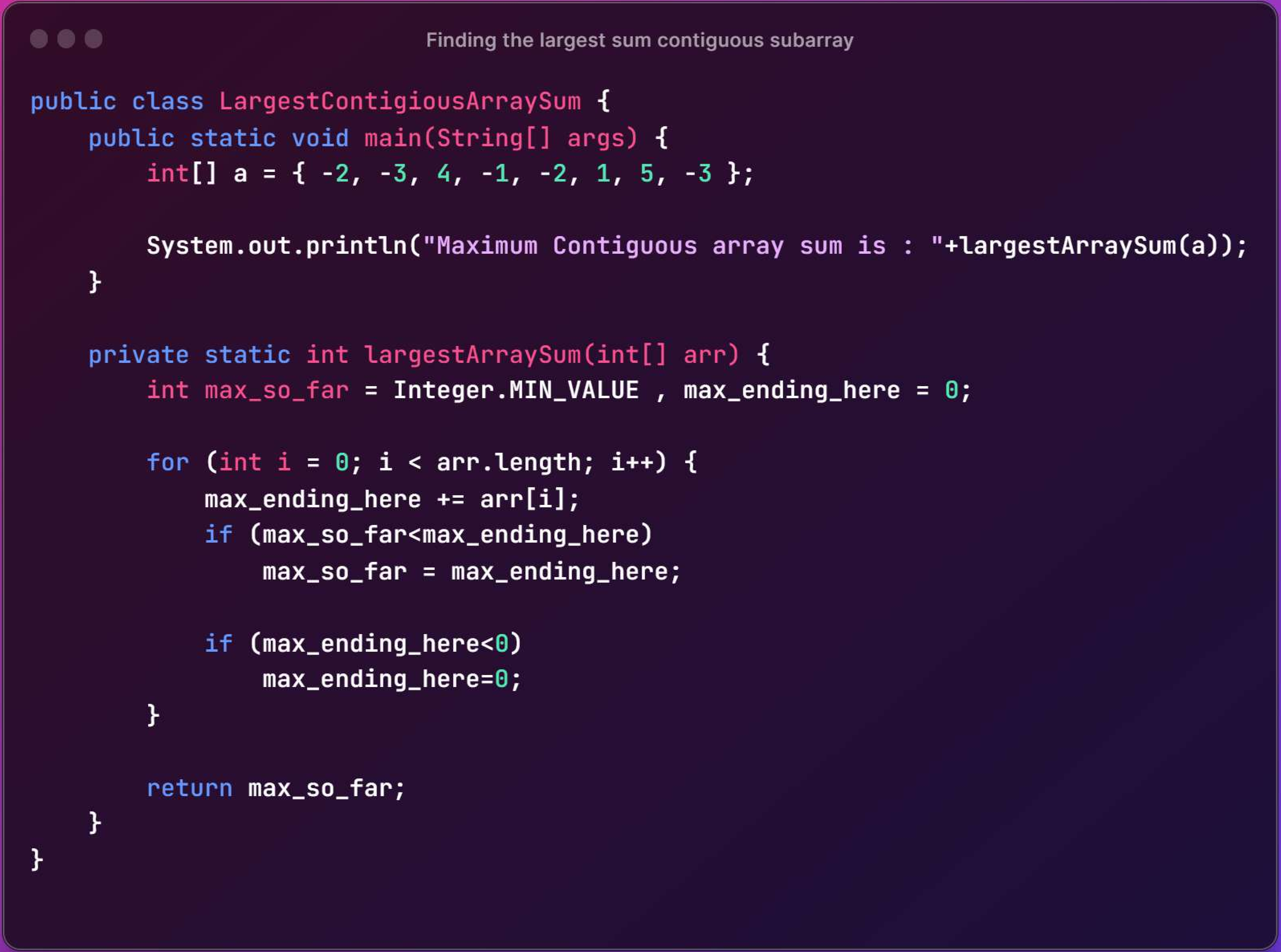


Rotate an array by k positions

```java
import java.util.Arrays;
import java.util.Scanner;

public class RotateBykPos {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size of an array: ");
        int n = in.nextInt();
        System.out.println("Enter the integer elements to the array: ");

        int[] arr = new int[n];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = in.nextInt();
        }
        System.out.print("Enter the value of k: ");
        int k = in.nextInt();

        k = k%n;

        int[] rotatedArr = new int[arr.length];
        for (int i = 0; i < arr.length; i++) {
            rotatedArr[i] = arr[(i+k)%n];
        }

        System.out.println(Arrays.toString(arr));
        System.out.println(Arrays.toString(rotatedArr));
    }
}
```



```java
public class SmallestPositiveMissingNUmber {
    public static void main(String[] args) {
        int[] array = {-1, 0, 2, 3, 4, 5, 6};
        int missingNumCounter = 0;
        for (int i = 0; i < array.length; i++) {
            if(array[i]>=0 && array[i]!=missingNumCounter )
                break;
            else if (array[i]>=0) {
                missingNumCounter++;
            }
        }
        System.out.println(missingNumCounter);
    }
}
```



```java
public class LargestContigiousArraySum {
    public static void main(String[] args) {
        int[] a = { -2, -3, 4, -1, -2, 1, 5, -3 };

        System.out.println("Maximum Contiguous array sum is : "+largestArraySum(a));
    }

    private static int largestArraySum(int[] arr) {
        int max_so_far = Integer.MIN_VALUE , max_ending_here = 0;

        for (int i = 0; i < arr.length; i++) {
            max_ending_here += arr[i];
            if (max_so_far<max_ending_here)
                max_so_far = max_ending_here;

            if (max_ending_here<0)
                max_ending_here=0;
        }

        return max_so_far;
    }
}
```