C-DAC Mumbai

Date 26/09/2024

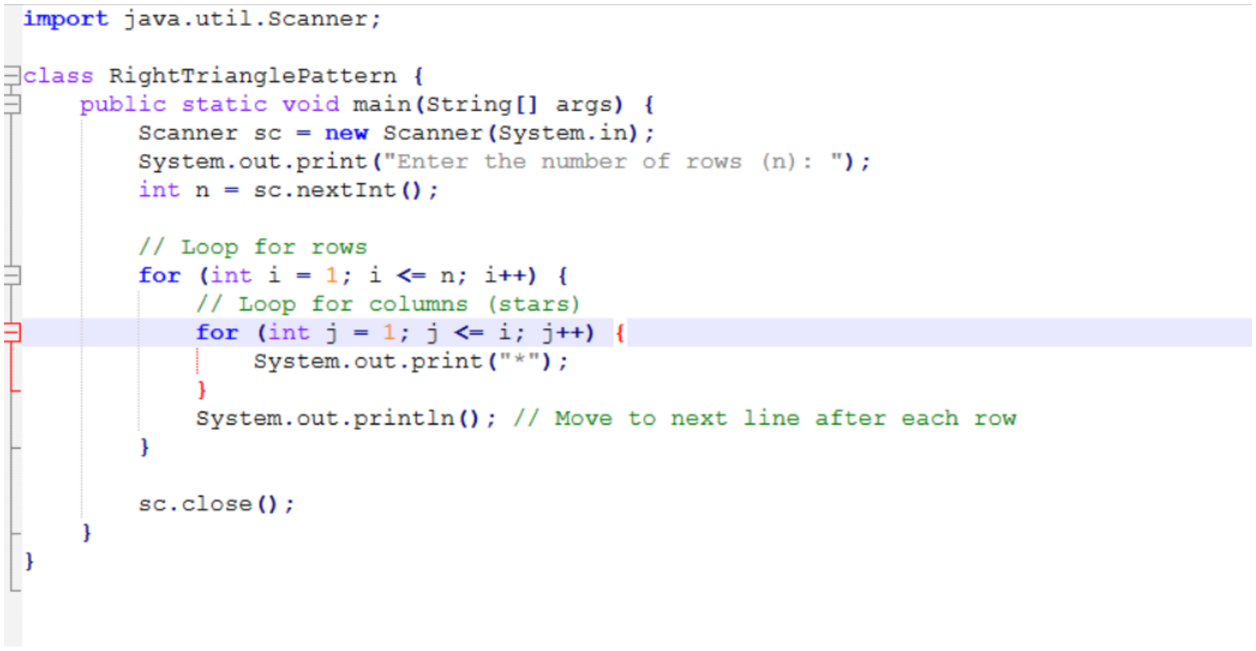
Subject: Algorithm and Data Structure
Assignment 1

Solve the assignment with following thing to be added in each question.

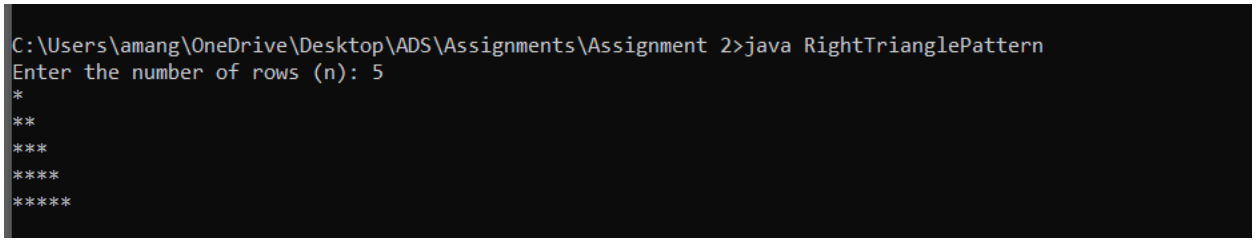
-Program
-Flow chart
-Explanation
-Output
-Time and Space complexity

1. Printing Patterns
Problem: Write a Java program to print patterns such as a right triangle of stars.

Test Cases:

Input: n = 3
Output:
```
*
**
***
```
Input: n = 5
Output:
```
*
**
***
****
*****
```


```java
import java.util.Scanner;

class RightTrianglePattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows (n): ");
        int n = sc.nextInt();

        // Loop for rows
        for (int i = 1; i <= n; i++) {
            // Loop for columns (stars)
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println(); // Move to next line after each row
        }

        sc.close();
    }
}
```



```
C:\Users\amang\OneDrive\Desktop\ADS\Assignments\Assignment 2>java RightTrianglePattern
Enter the number of rows (n): 5
*
**
***
****
*****
```

The program takes an integer input n representing the number of rows. It uses two nested loops:

- The outer loop runs i from 1 to n for each row.
- The inner loop runs j from 1 to i, printing i stars in each row.

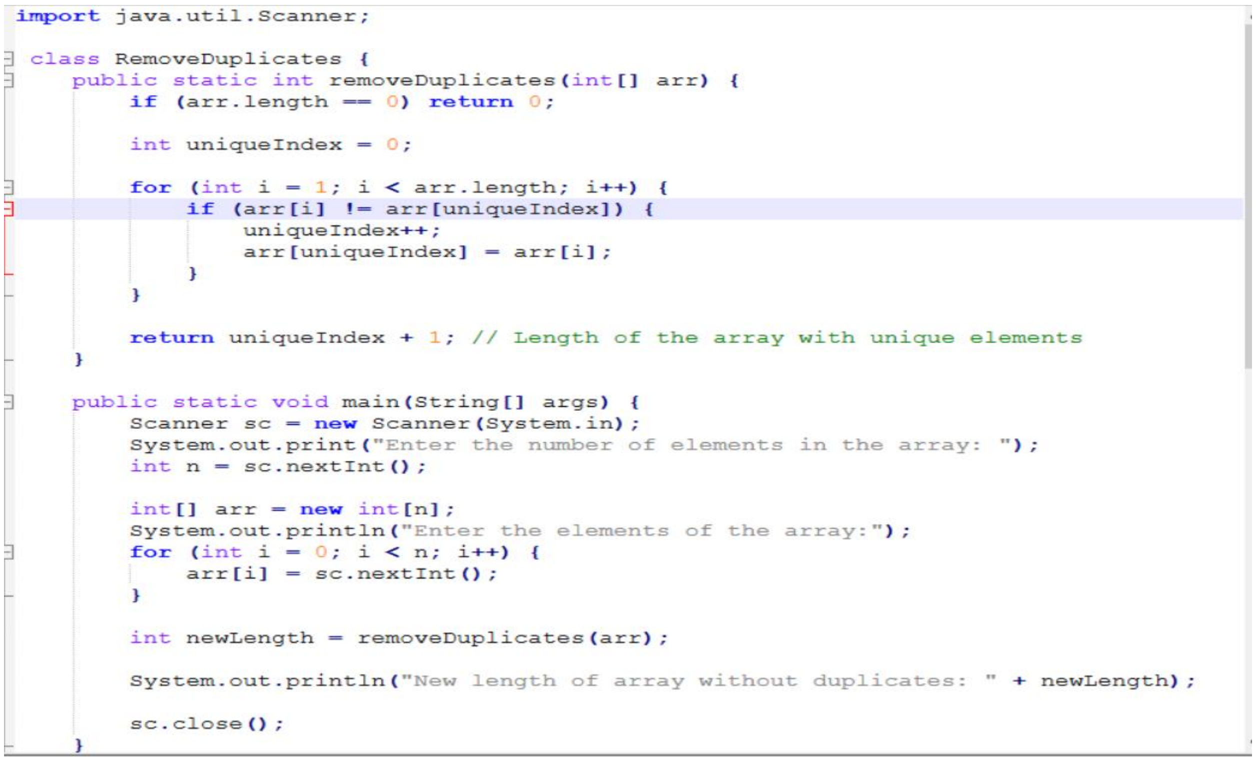
For example, if n = 3, the outer loop runs 3 times and the inner loop prints 1, 2, and 3 stars respectively for each row.

Time complexity → $O(n^2)$
Space Complexity → O(1)

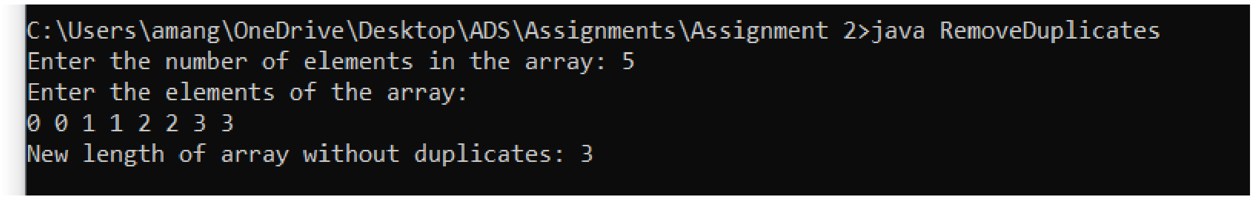
2. Remove Array Duplicates
Problem: Write a Java program to remove duplicates from a sorted array and return the new length of the array.

Test Cases:

Input: arr = [1, 1, 2]
Output: 2
Input: arr = [0, 0, 1, 1, 2, 2, 3, 3]
Output: 4



```java
import java.util.Scanner;

 class RemoveDuplicates {
    public static int removeDuplicates(int[] arr) {
        if (arr.length == 0) return 0;

        int uniqueIndex = 0;

        for (int i = 1; i < arr.length; i++) {
            if (arr[i] != arr[uniqueIndex]) {
                uniqueIndex++;
                arr[uniqueIndex] = arr[i];
            }
        }

        return uniqueIndex + 1; // Length of the array with unique elements
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = sc.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        int newLength = removeDuplicates(arr);

        System.out.println("New length of array without duplicates: " + newLength);

        sc.close();
    }
```



```
C:\Users\amang\OneDrive\Desktop\ADS\Assignments\Assignment 2>java RemoveDuplicates
Enter the number of elements in the array: 5
Enter the elements of the array:
0 0 1 1 2 2 3 3
New length of array without duplicates: 3
```

The program takes a sorted array as input and removes duplicates in-place, returning the new length of the array containing only unique elements. It uses a two-pointer approach:

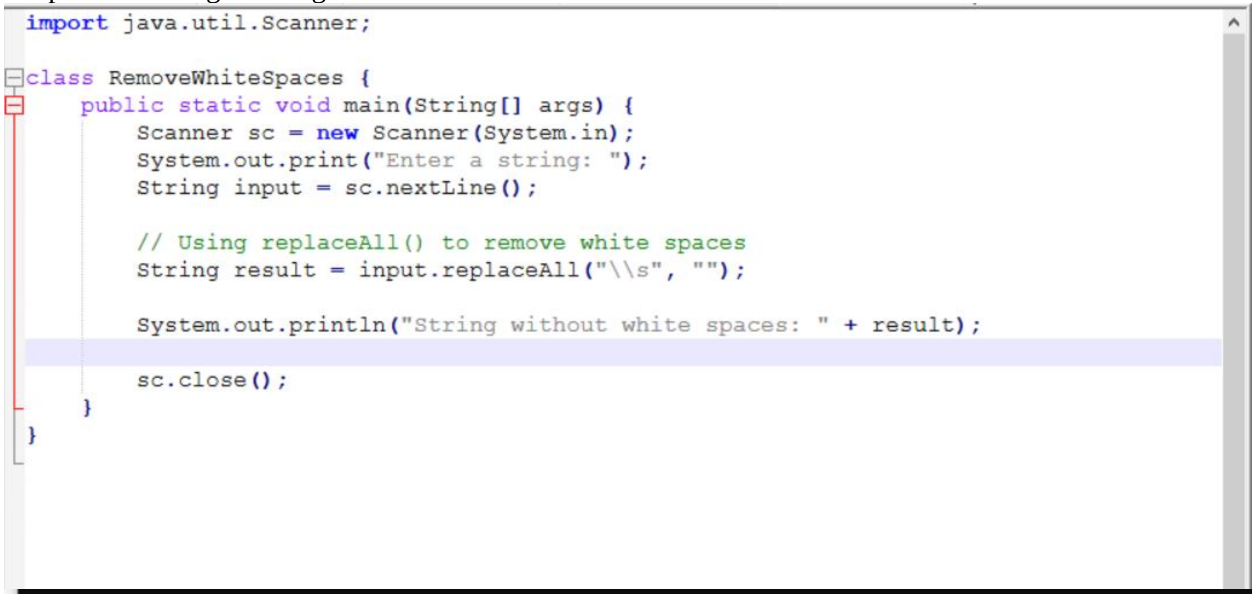
- uniqueIndex starts at 0 and keeps track of the position of the last unique element.
- The second pointer, i, iterates through the array. When a new unique element is found, uniqueIndex is incremented, and the new unique element is placed at arr[uniqueIndex].

Time complexity → O(n)
Space Complexity → O(1)

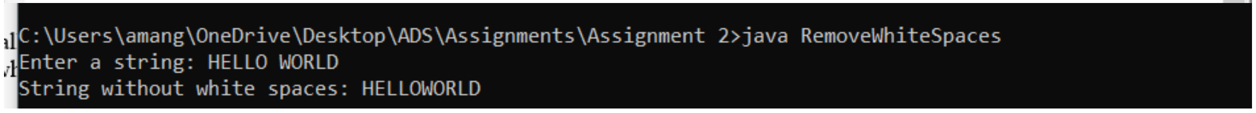
3. Remove White Spaces from String
Problem: Write a Java program to remove all white spaces from a given string.

Test Cases:

Input: "Hello World"
Output: "HelloWorld"
Input: " Java   Programming "
Output: "JavaProgramming"



```java
import java.util.Scanner;

class RemoveWhiteSpaces {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = sc.nextLine();

        // Using replaceAll() to remove white spaces
        String result = input.replaceAll("\\s", "");

        System.out.println("String without white spaces: " + result);

        sc.close();
    }
}
```



```
C:\Users\amang\OneDrive\Desktop\ADS\Assignments\Assignment 2>java RemoveWhiteSpaces
Enter a string: HELLO WORLD
String without white spaces: HELLOWORLD
```

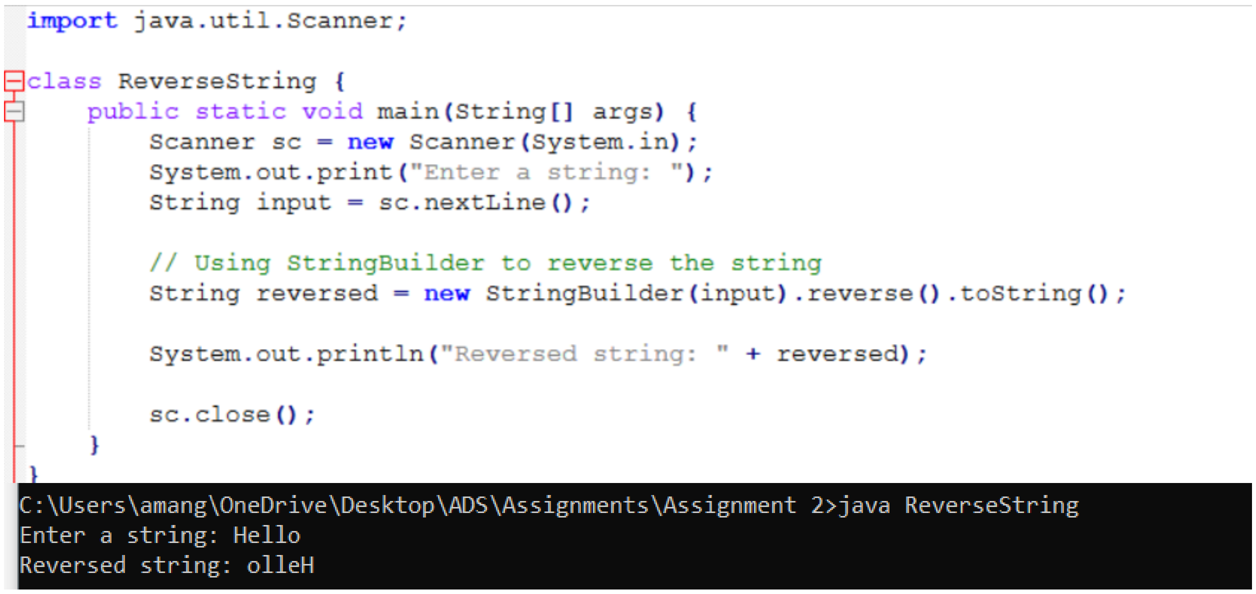
The program uses replaceAll("\\s", "") to remove all white spaces (including spaces, tabs, and newlines) from the given string. The regex \\s matches any white space, and it replaces them with an empty string.

Time complexity → O(n)
Space Complexity → O(n)

4. Reverse a String
Problem: Write a Java program to reverse a given string.

Test Cases:

Input: "hello"
Output: "olleh"
Input: "Java"
Output: "avaJ"



```java
import java.util.Scanner;

class ReverseString {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = sc.nextLine();

        // Using StringBuilder to reverse the string
        String reversed = new StringBuilder(input).reverse().toString();

        System.out.println("Reversed string: " + reversed);

        sc.close();
    }
}
```

```
C:\Users\amang\OneDrive\Desktop\ADS\Assignments\Assignment 2>java ReverseString
Enter a string: Hello
Reversed string: olleH
```

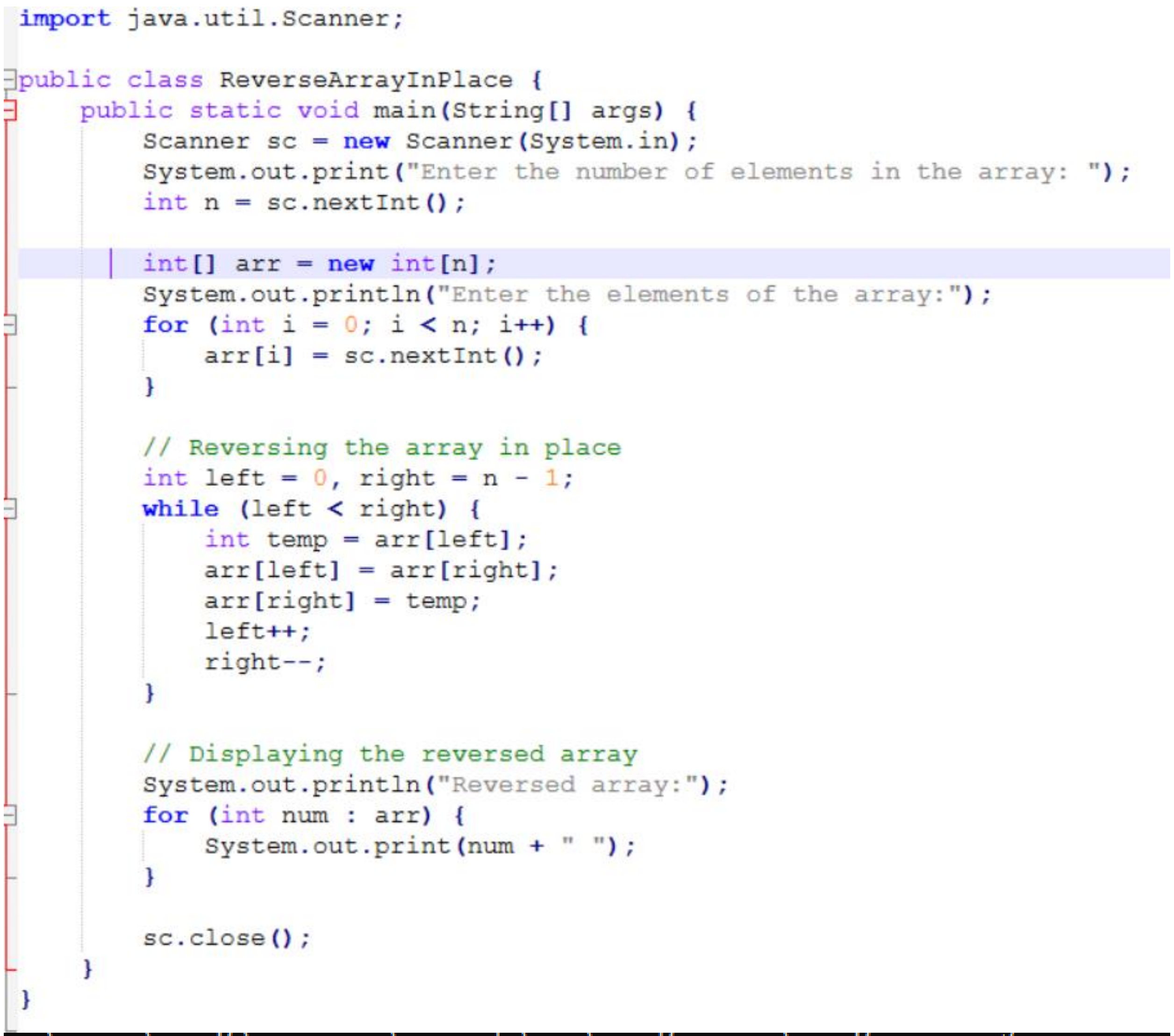
The program takes an input string and reverses it using the StringBuilder.reverse() method. This method constructs a mutable string and then reverses it in place.

Time complexity → O(n)
Space Complexity → O(n)

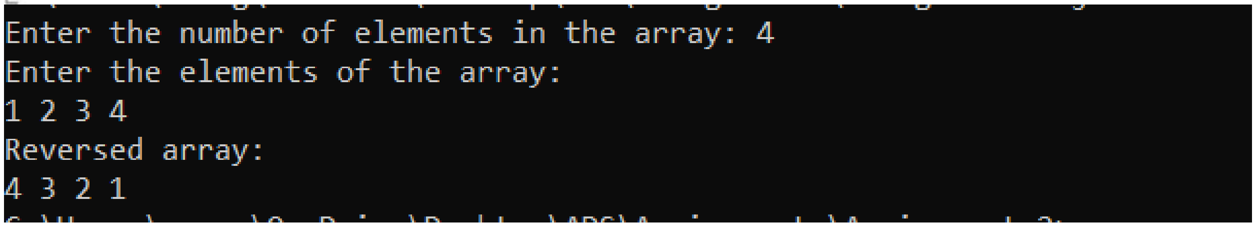
5. Reverse Array in Place
Problem: Write a Java program to reverse an array in place.

Test Cases:

Input: arr = [1, 2, 3, 4]
Output: [4, 3, 2, 1]
Input: arr = [7, 8, 9]
Output: [9, 8, 7]



```java
import java.util.Scanner;

public class ReverseArrayInPlace {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = sc.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        // Reversing the array in place
        int left = 0, right = n - 1;
        while (left < right) {
            int temp = arr[left];
            arr[left] = arr[right];
            arr[right] = temp;
            left++;
            right--;
        }

        // Displaying the reversed array
        System.out.println("Reversed array:");
        for (int num : arr) {
            System.out.print(num + " ");
        }

        sc.close();
    }
}
```



```
Enter the number of elements in the array: 4
Enter the elements of the array:
1 2 3 4
Reversed array:
4 3 2 1
```

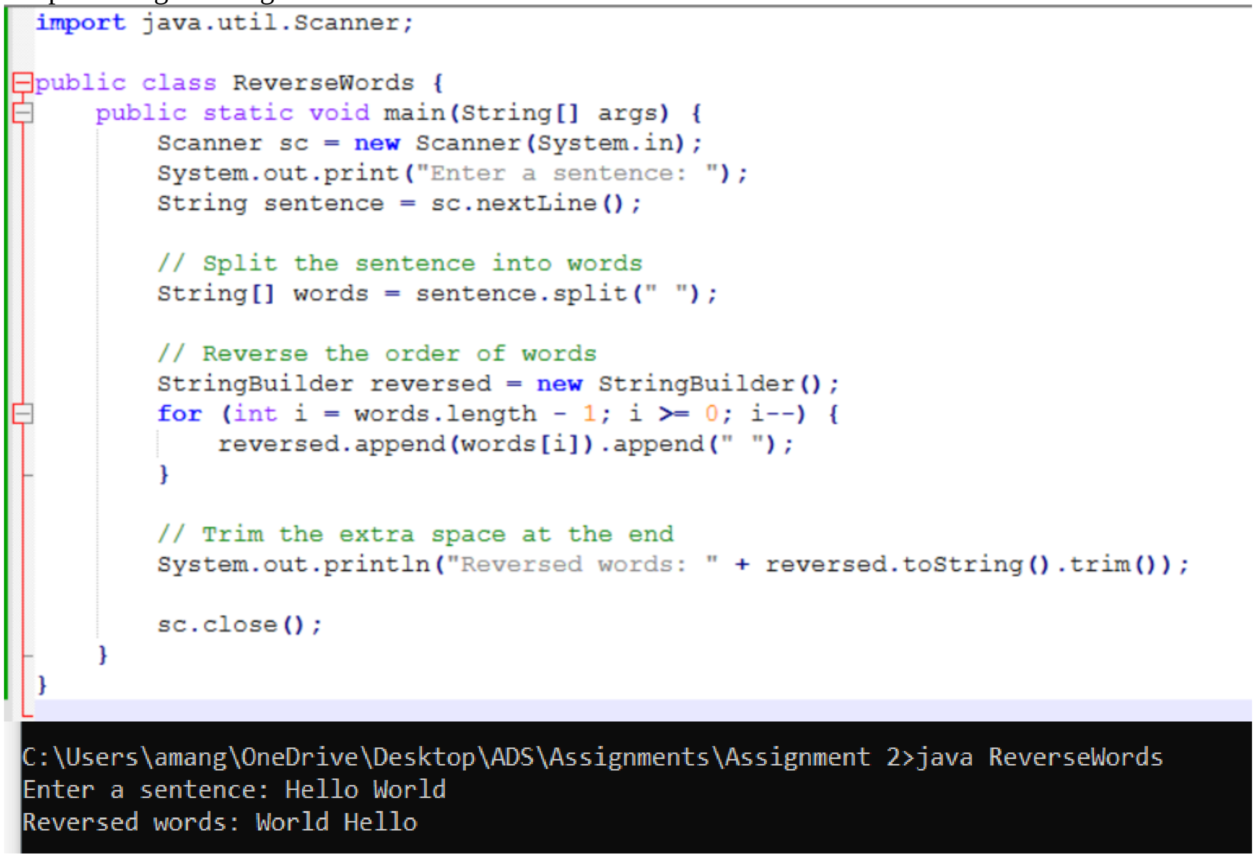
This program reverses an array in place using two pointers (left and right). It swaps the values at the left and right pointers, then increments left and decrements right until the entire array is reversed.

Time complexity → O(n)
Space Complexity → O(1)

6. Reverse Words in a String
Problem: Write a Java program to reverse the words in a given sentence.

Test Cases:

Input: "Hello World"
Output: "World Hello"
Input: "Java Programming"
Output: "Programming Java"



```java
import java.util.Scanner;

public class ReverseWords {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a sentence: ");
        String sentence = sc.nextLine();

        // Split the sentence into words
        String[] words = sentence.split(" ");

        // Reverse the order of words
        StringBuilder reversed = new StringBuilder();
        for (int i = words.length - 1; i >= 0; i--) {
            reversed.append(words[i]).append(" ");
        }

        // Trim the extra space at the end
        System.out.println("Reversed words: " + reversed.toString().trim());

        sc.close();
    }
}
```

```
C:\Users\amang\OneDrive\Desktop\ADS\Assignments\Assignment 2>java ReverseWords
Enter a sentence: Hello World
Reversed words: World Hello
```

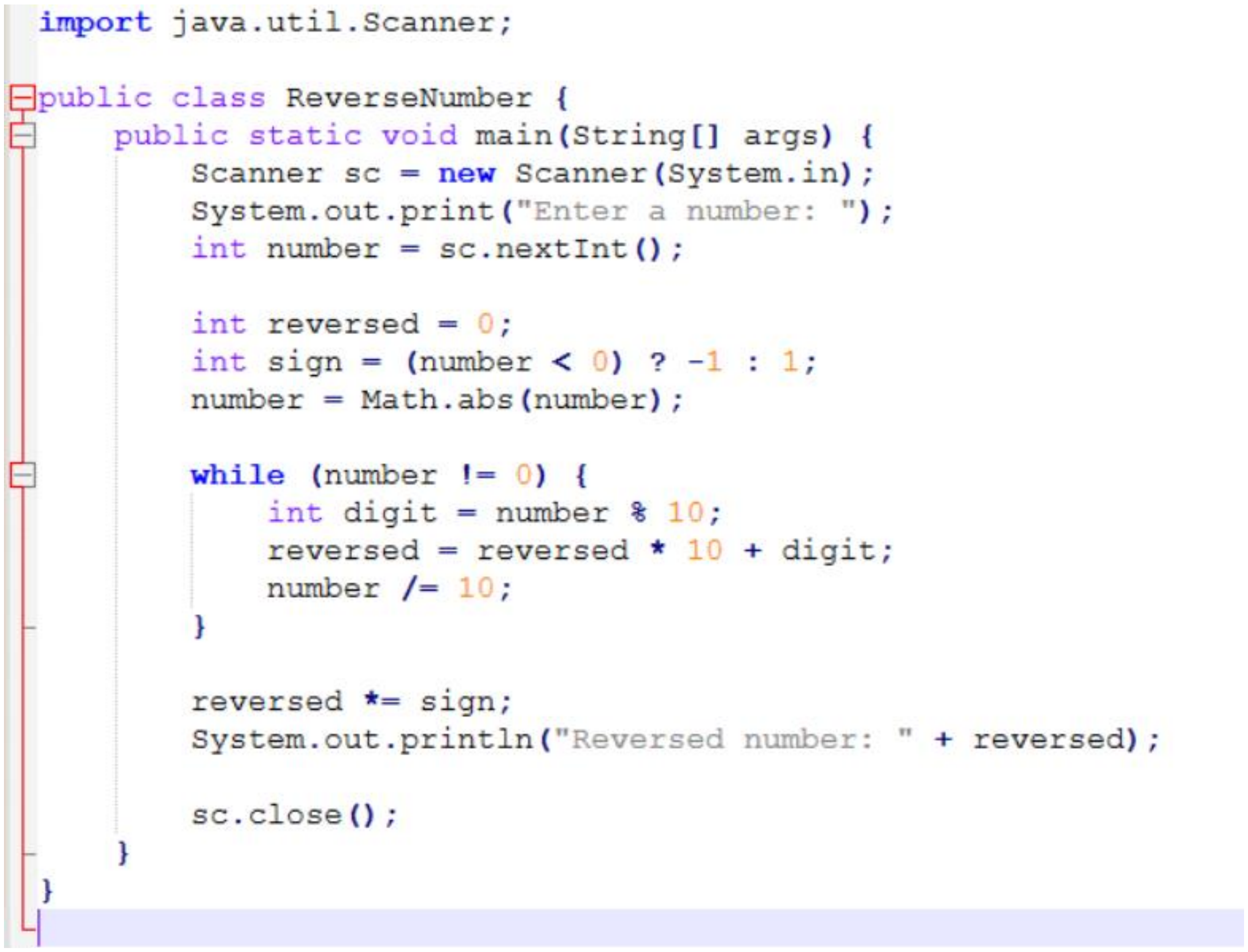
The program splits the sentence into an array of words using split(" "), then reverses the words by appending them in reverse order using a loop. Finally, it prints the reversed sentence.

Time complexity → O(n)
Space Complexity → O(n)

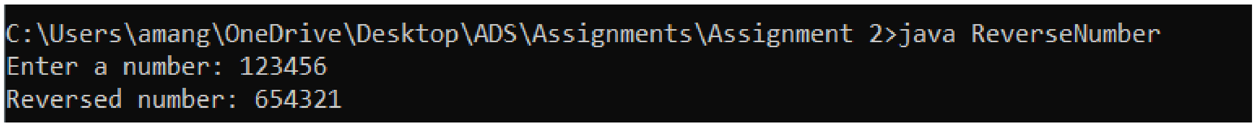
7. Reverse a Number
Problem: Write a Java program to reverse a given number.

Test Cases:

Input: 12345
Output: 54321
Input: -9876
Output: -6789



```java
import java.util.Scanner;

public class ReverseNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = sc.nextInt();

        int reversed = 0;
        int sign = (number < 0) ? -1 : 1;
        number = Math.abs(number);

        while (number != 0) {
            int digit = number % 10;
            reversed = reversed * 10 + digit;
            number /= 10;
        }

        reversed *= sign;
        System.out.println("Reversed number: " + reversed);

        sc.close();
    }
}
```



```
C:\Users\amang\OneDrive\Desktop\ADS\Assignments\Assignment 2>java ReverseNumber
Enter a number: 123456
Reversed number: 654321
```

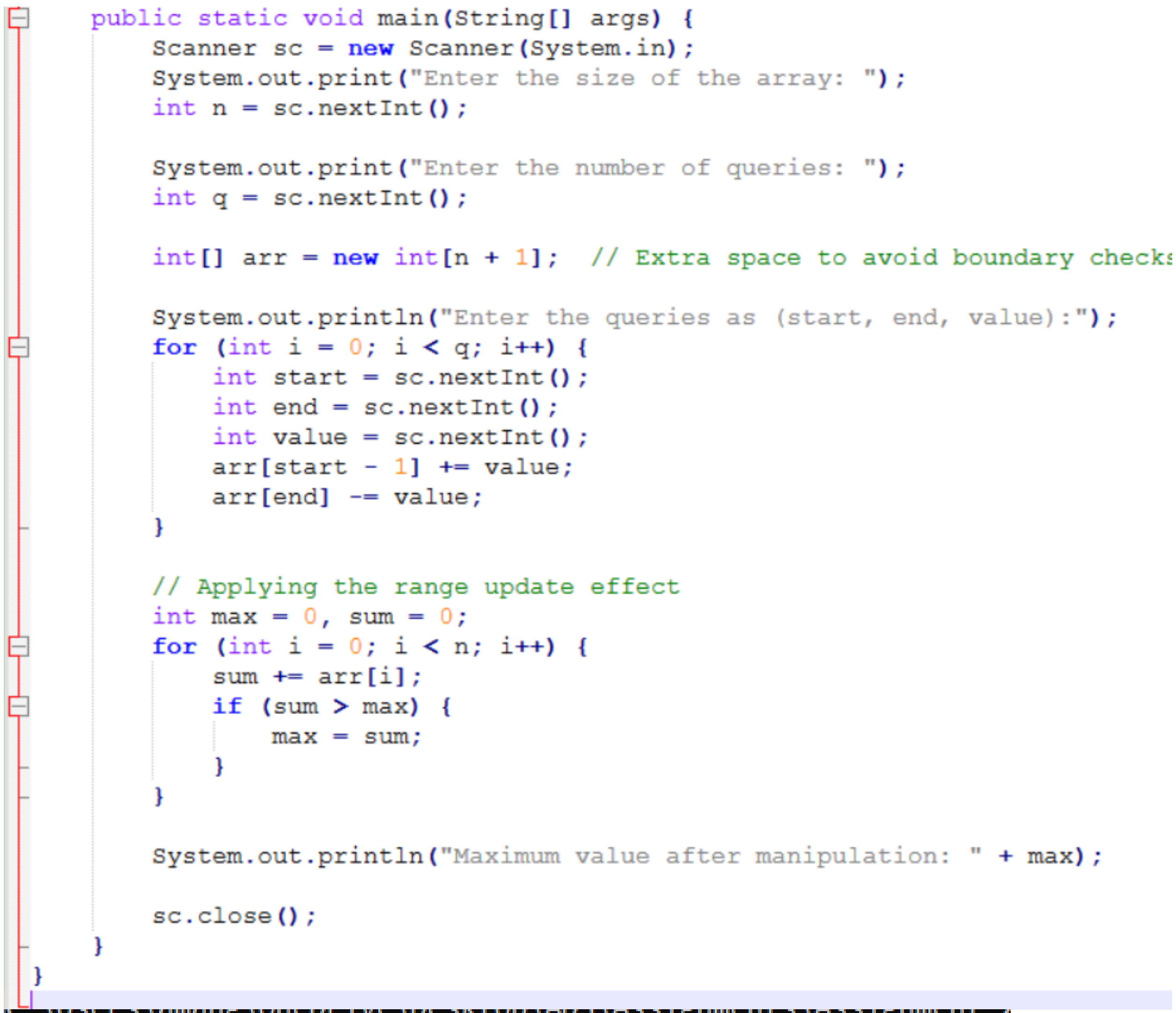
The program reverses a number by repeatedly extracting the last digit (number % 10) and appending it to the result. The sign is handled separately to accommodate negative numbers.

Time complexity → O(log(n))
Space Complexity → O(1)

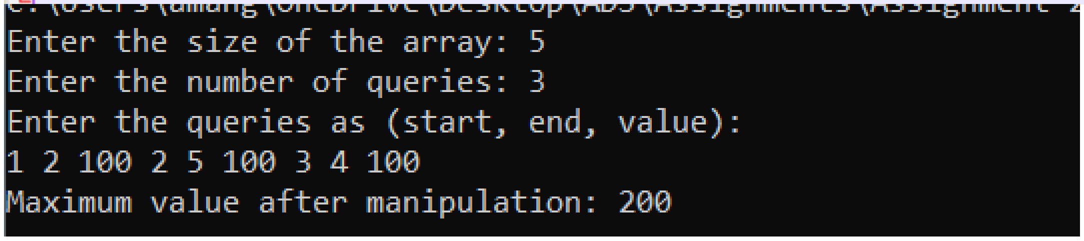
8. Array Manipulation
Problem: Perform a series of operations to manipulate an array based on range update queries. Each query adds a value to a range of indices.

Test Cases:

Input: n = 5, queries = [[1, 2, 100], [2, 5, 100], [3, 4, 100]]
Output: 200
Input: n = 4, queries = [[1, 3, 50], [2, 4, 70]]
Output: 120



```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();

        System.out.print("Enter the number of queries: ");
        int q = sc.nextInt();

        int[] arr = new int[n + 1];  // Extra space to avoid boundary checks

        System.out.println("Enter the queries as (start, end, value):");
        for (int i = 0; i < q; i++) {
            int start = sc.nextInt();
            int end = sc.nextInt();
            int value = sc.nextInt();
            arr[start - 1] += value;
            arr[end] -= value;
        }

        // Applying the range update effect
        int max = 0, sum = 0;
        for (int i = 0; i < n; i++) {
            sum += arr[i];
            if (sum > max) {
                max = sum;
            }
        }

        System.out.println("Maximum value after manipulation: " + max);

        sc.close();
    }
}
```



```
Enter the size of the array: 5
Enter the number of queries: 3
Enter the queries as (start, end, value):
1 2 100 2 5 100 3 4 100
Maximum value after manipulation: 200
```

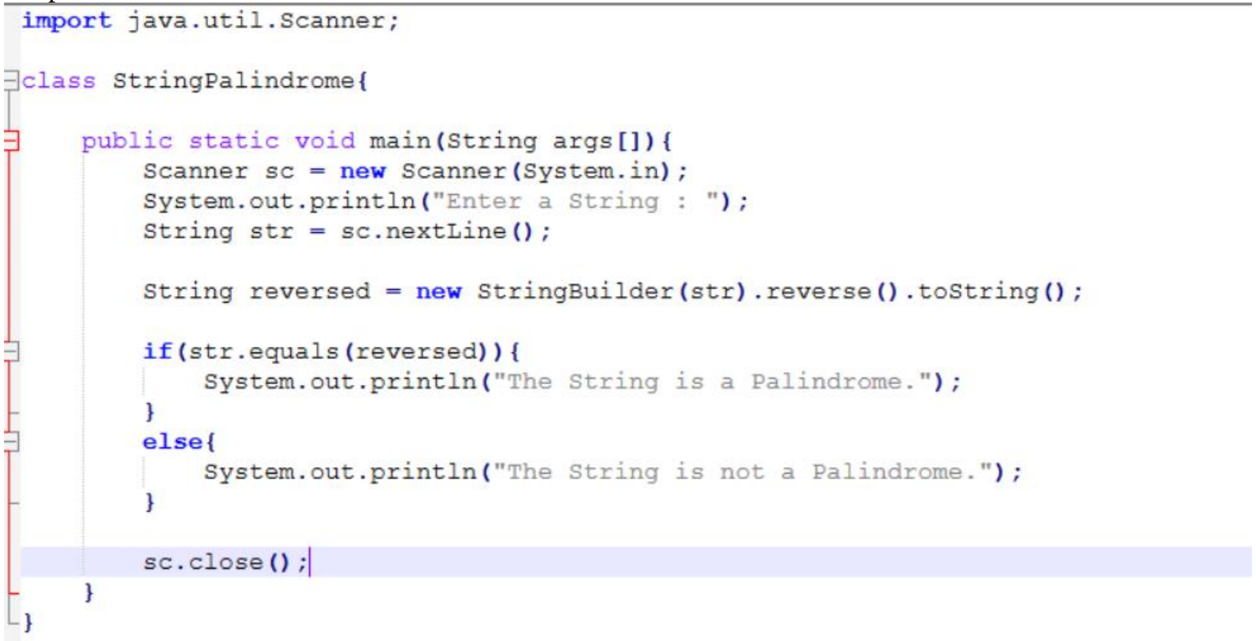
The program applies range updates using a technique where the value is added at the start index and subtracted just after the end index. Then, a cumulative sum is calculated to get the actual values in the array. The maximum value is found during this process.

Time complexity → O(n+q)
Space Complexity → O(n)

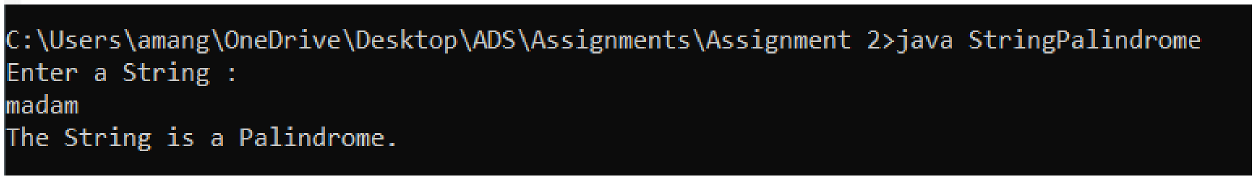
9. String Palindrome
Problem: Write a Java program to check if a given string is a palindrome.

Test Cases:

Input: "madam"
Output: true
Input: "hello"
Output: false



```java
import java.util.Scanner;

class StringPalindrome{

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a String : ");
        String str = sc.nextLine();

        String reversed = new StringBuilder(str).reverse().toString();

        if(str.equals(reversed)){
            System.out.println("The String is a Palindrome.");
        }
        else{
            System.out.println("The String is not a Palindrome.");
        }

        sc.close();
    }
}
```



```
C:\Users\amang\OneDrive\Desktop\ADS\Assignments\Assignment 2>java StringPalindrome
Enter a String :
madam
The String is a Palindrome.
```

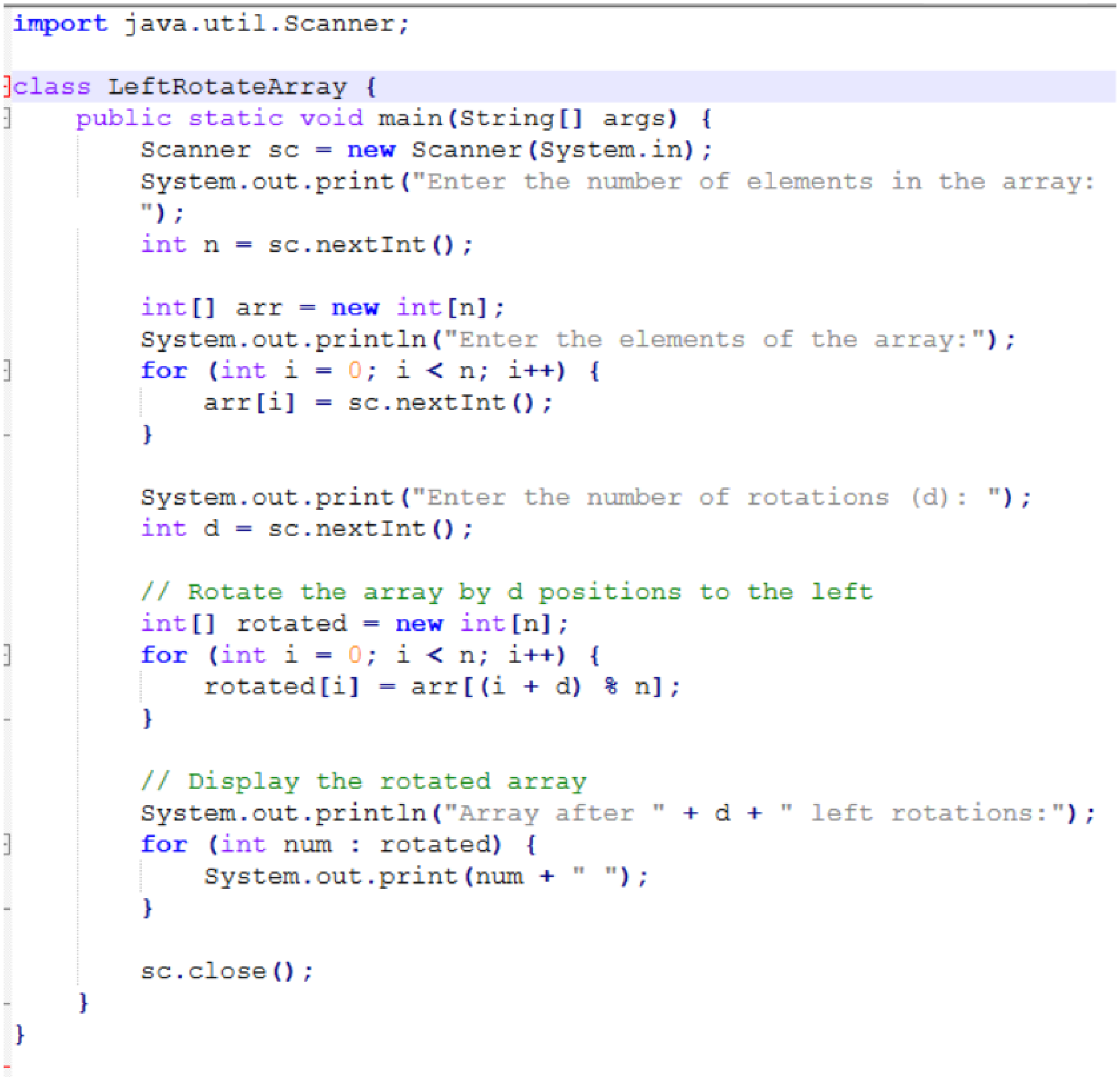
The program checks if a string is a palindrome by reversing the string and comparing it to the original. If both are equal, the string is a palindrome.

Time complexity → O(n)
Space Complexity → O(n)

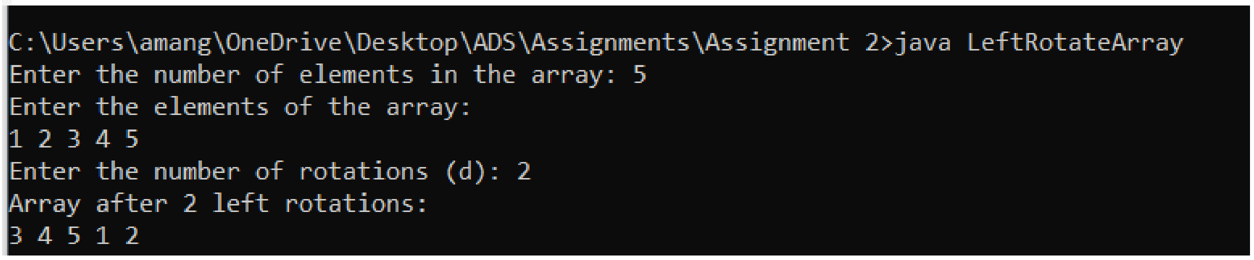
10. Array Left Rotation
Problem: Write a Java program to rotate an array to the left by d positions.

Test Cases:

Input: arr = [1, 2, 3, 4, 5], d = 2
Output: [3, 4, 5, 1, 2]
Input: arr = [10, 20, 30, 40], d = 1
Output: [20, 30, 40, 10]



```java
import java.util.Scanner;

class LeftRotateArray {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array:
");
        int n = sc.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.print("Enter the number of rotations (d): ");
        int d = sc.nextInt();

        // Rotate the array by d positions to the left
        int[] rotated = new int[n];
        for (int i = 0; i < n; i++) {
            rotated[i] = arr[(i + d) % n];
        }

        // Display the rotated array
        System.out.println("Array after " + d + " left rotations:");
        for (int num : rotated) {
            System.out.print(num + " ");
        }

        sc.close();
    }
}
```



```
C:\Users\amang\OneDrive\Desktop\ADS\Assignments\Assignment 2>java LeftRotateArray
Enter the number of elements in the array: 5
Enter the elements of the array:
1 2 3 4 5
Enter the number of rotations (d): 2
Array after 2 left rotations:
3 4 5 1 2
```

This program performs left rotation by shifting each element to the left by d positions. It calculates the new position for each element using (i + d) % n.

Time complexity → O(n)
Space Complexity → O(n)