**Assignment 2**

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:

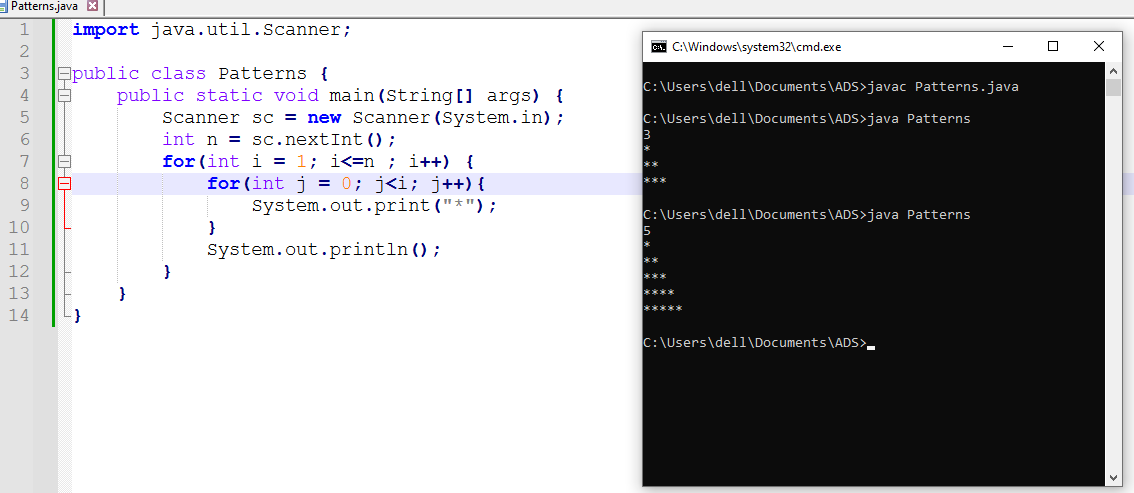
\*

\*\*

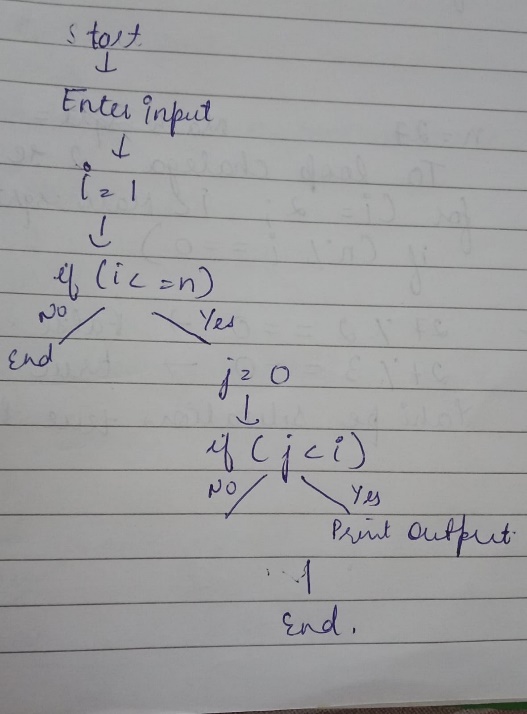
\*\*\*

\*\*\*\*

\*\*\*\*\*

****

**Flowchart-**

****

**Explanation –** Firstly, I’ve created a public class named Patterns. Inside the main method I’ve taken the input from the user using Scanner class. The outer for loop starts with i = 1 and runs until i is less than or equal to n. It controls the number of rows that will be printed. The value of i represents the current row number. Inside the outer loop, there is another for loop that initializes j to 0 and runs until j is less than i. This loop controls how many asterisks (\*) will be printed in the current 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.

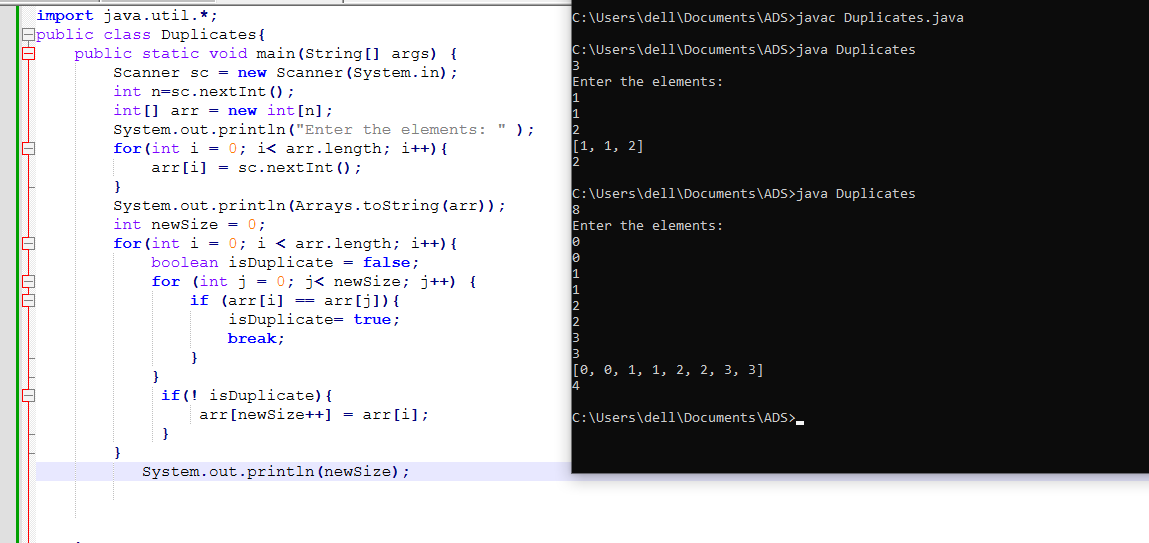
Test Cases:

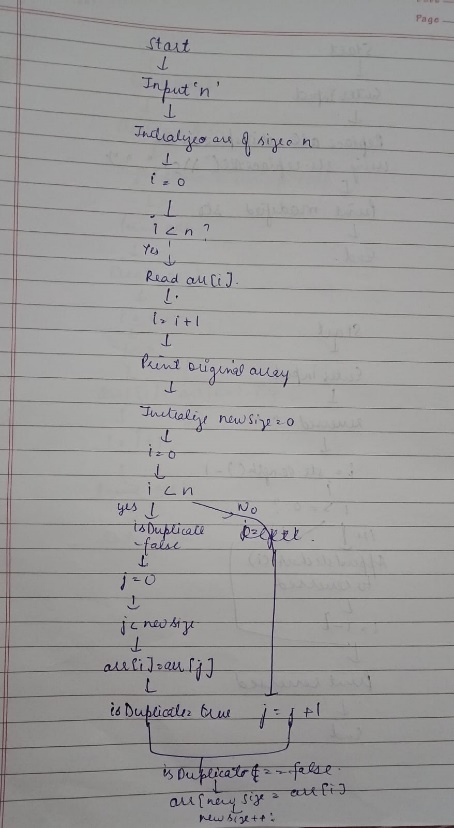
Input: arr = [1, 1, 2]

Output: 2

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

Output: 4

**  
Flowchart-**

****

**Explanation -** Firstly, I’ve created a public class named Duplicates. Inside the main method I’ve declared an integer variable n and assign it the value read from the user input using sc.nextInt() and initialized an integer array named arr with a size of n. The for loop iterates from 0 to arr.length – 1. For each iteration, it reads an integer from the user using sc.nextInt() and assigns it to the i-th index of the array arr and the original array is printed. After this I’ve declared an integer variable newSize and initialize it to 0. This variable will track the number of unique elements found in the array. There is boolean variable isDuplicate and initialize it to false. This flag will help determine if the current element arr[i] is a duplicate. if the current element arr[i] is equal to any of the previously found unique elements arr[j]. If a match is found, isDuplicate is set to true.

**Time Complexity** – O(n^2)

**Space Complexity** – O(n)

3. Remove White Spaces from String

Problem: Write a Java program to remove all white spaces from a given string.

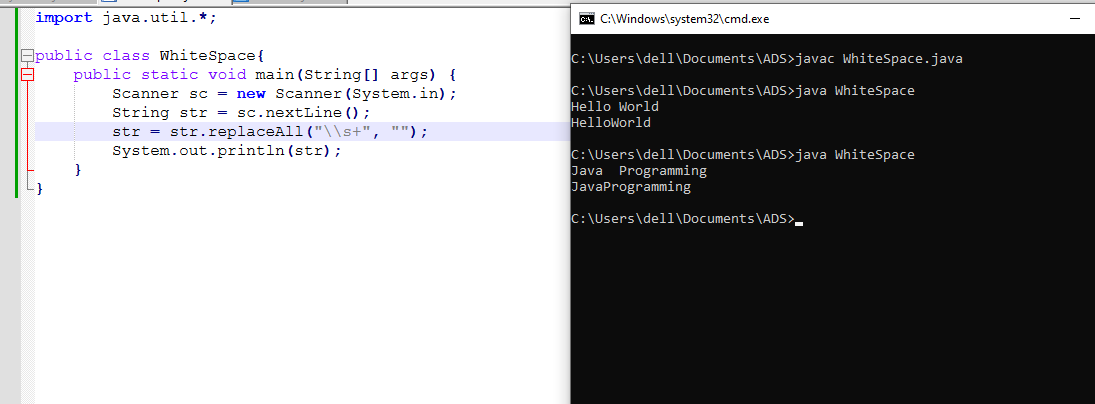
Test Cases:

Input: "Hello World"

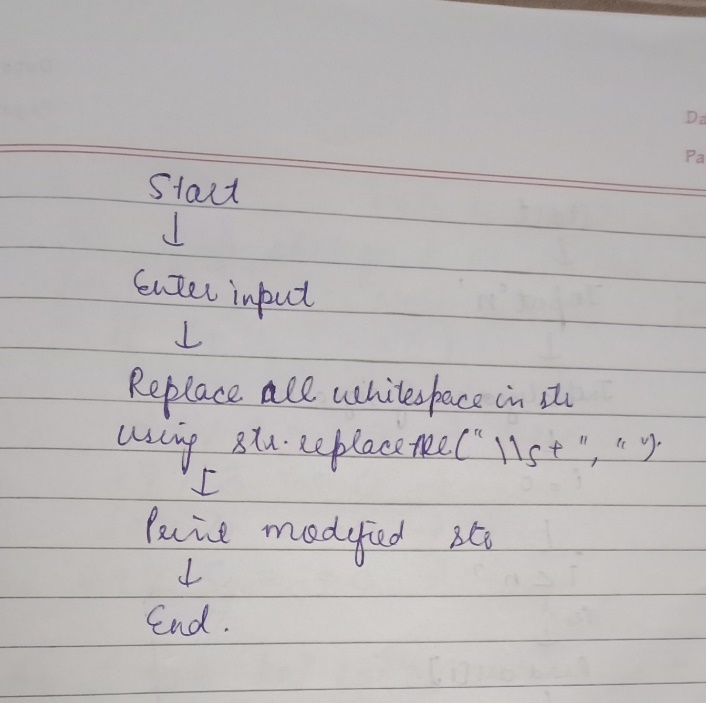
Output: "HelloWorld"

Input: " Java Programming "

Output: "JavaProgramming"

****

**Flowchart –**

****

**Explanation -** Firstly, I’ve created a public class named WhiteSpace. Inside the main method I’ve taken a string as input with the help of Scanner class. Then I used the replaceAll method on the String class. The argument \\s+ is a regular expression that matches one or more whitespace characters. The second argument is an empty string "", which means all matched whitespace characters will be replaced with nothing.

**Time Complexity** – O(n)

**Space Complexity** – O(n)

4. Reverse a String

Problem: Write a Java program to reverse a given string.

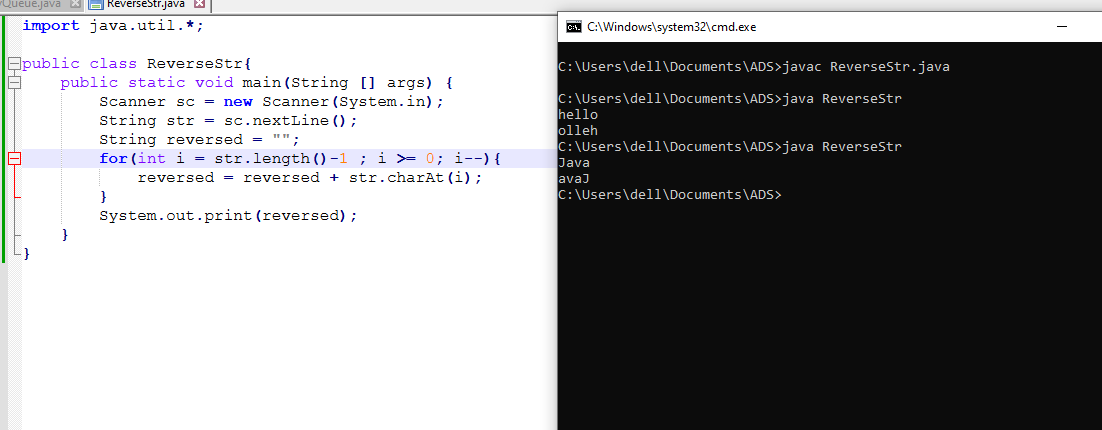
Test Cases:

Input: "hello"

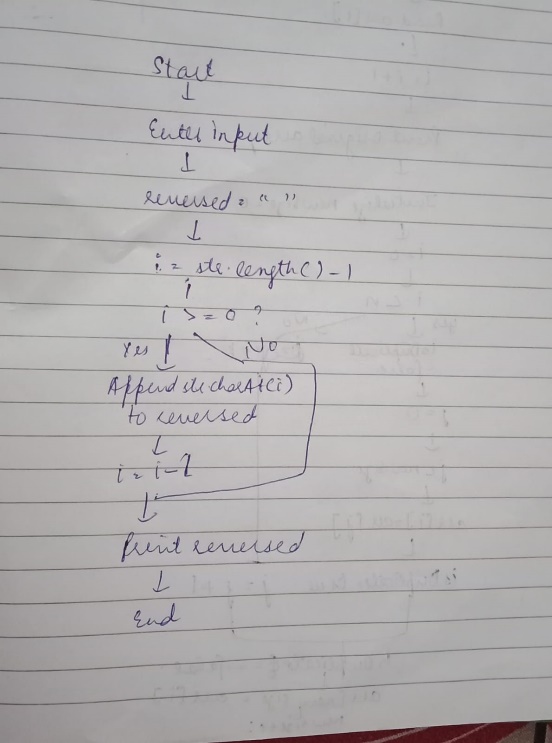
Output: "olleh"

Input: "Java"

Output: "avaJ"



**Flowchart-**

****

**Explanation -** Firstly, I’ve created a public class named ReverseStr. Inside the main method I’ve taken a string as input with the help of Scanner class. An empty String variable named reversed is initialized. There is a for loop that iterates from the last character of str to the first. str.charAt(i) method retrieves the character at position i. The retrieved character is appended to the reversed string.

**Time Complexity** – O(n^2)

**Space Complexity** – O(n)

5. Reverse Array in Place

Problem: Write a Java program to reverse an array in place.

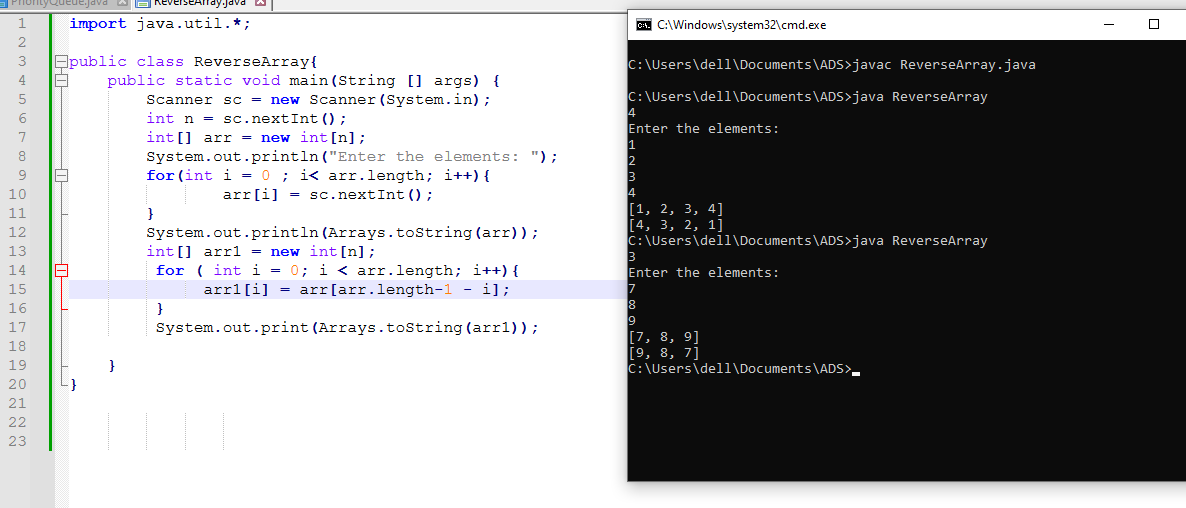
Test Cases:

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

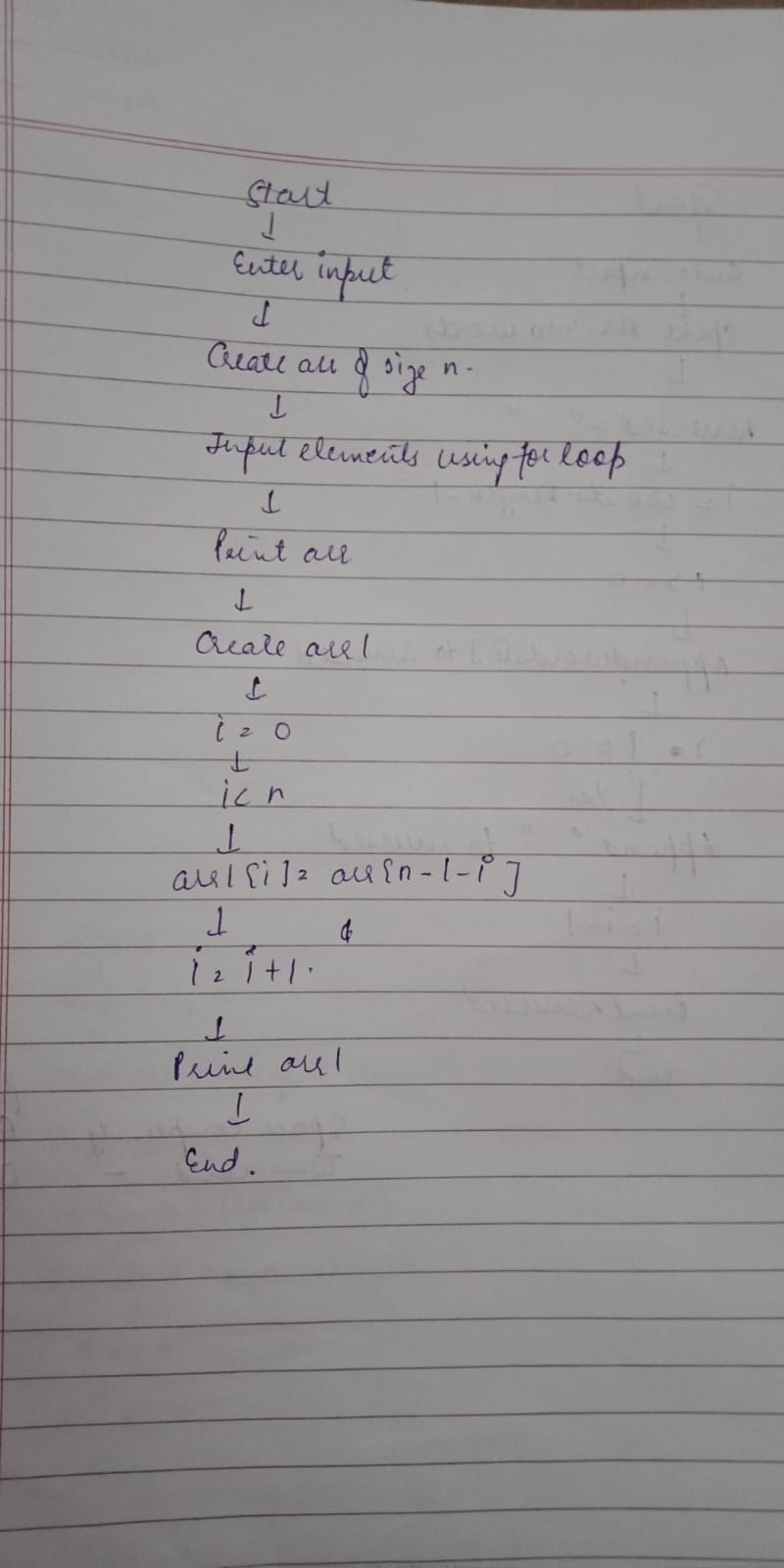
Output: [4, 3, 2, 1]

Input: arr = [7, 8, 9]

Output: [9, 8, 7]



**Flowchart-**



**Explanation –** Firstly, I’ve created a public class named ReverseArray. . Inside the main method I’ve declared an integer variable n and assign it the value read from the user input using sc.nextInt() and initialized an integer array named arr with a size of n. Then I printed the array. Created a new array arr1 of the same size n to store the reversed elements of arr. The for loop iterates through each element of arr. arr.length - 1 - i calculates the index of the corresponding element in the original array arr in reverse order. Printed the arr1 array in reversed order.

**Time Complexity** – O(n)

**Space Complexity** – O(n)

6. Reverse Words in a String

Problem: Write a Java program to reverse the words in a given sentence.

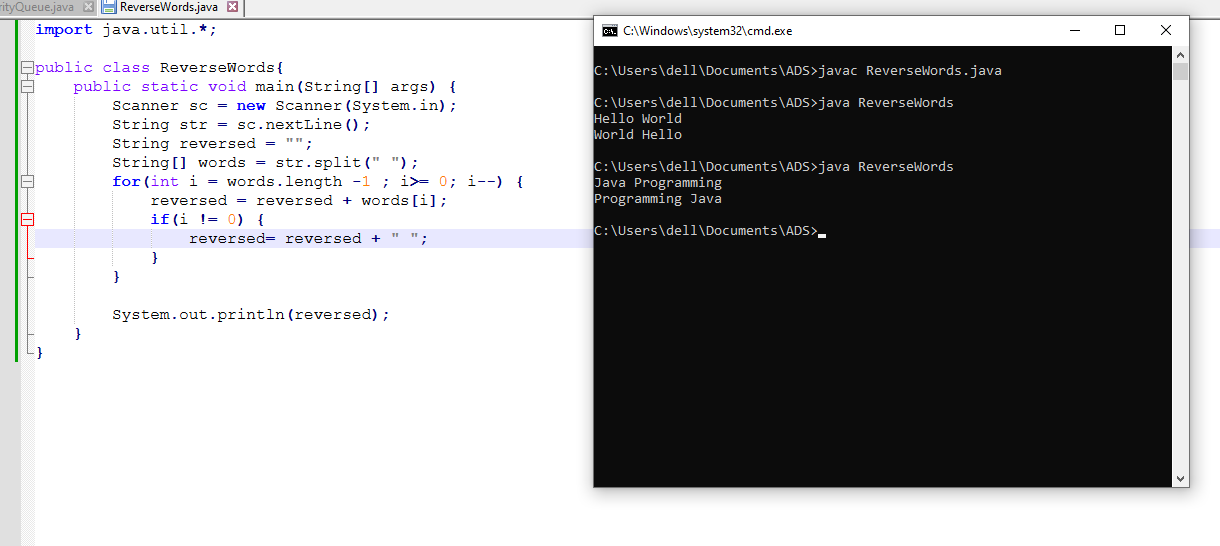
Test Cases:

Input: "Hello World"

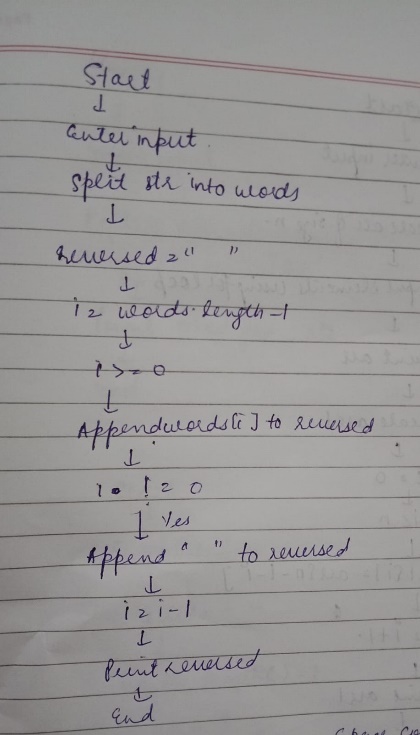
Output: "World Hello"

Input: "Java Programming"

Output: "Programming Java"



**Flowchart-**

****

**Explanation -** Firstly, I’ve created a public class named ReverseWords. Inside the main method I’ve taken a string as input with the help of Scanner class. An empty String variable named reversed is initialized. The split(" ") method splits the input string str into an array of substrings. The for loop iterates from the last element of words to the first element. Appended each word from the words array to the reversed string. Printed the reversed string.

**Time Complexity** – O(n^2)

**Space Complexity** – O(n)

7. Reverse a Number

Problem: Write a Java program to reverse a given number.

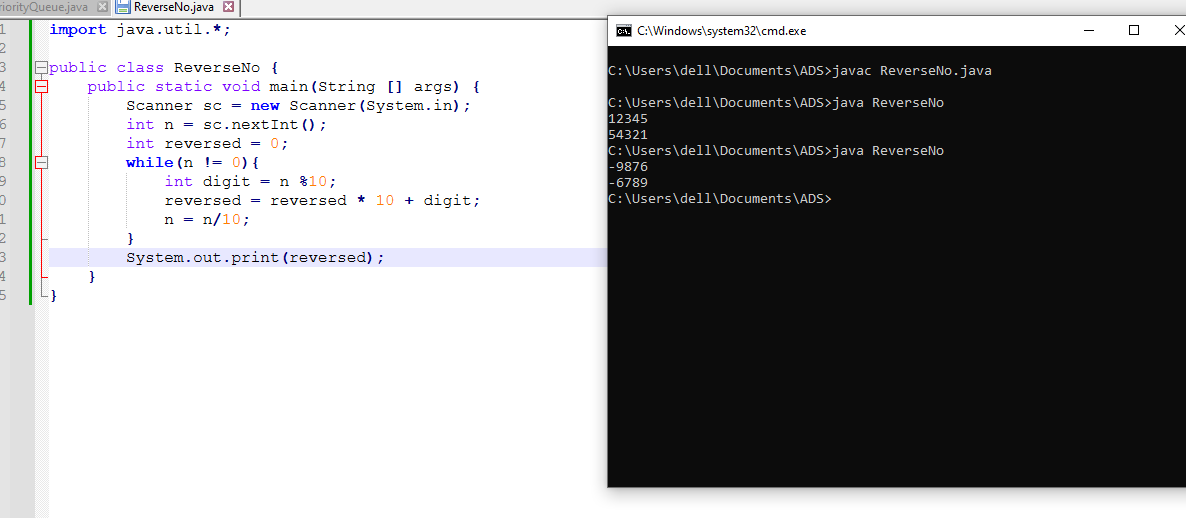
Test Cases:

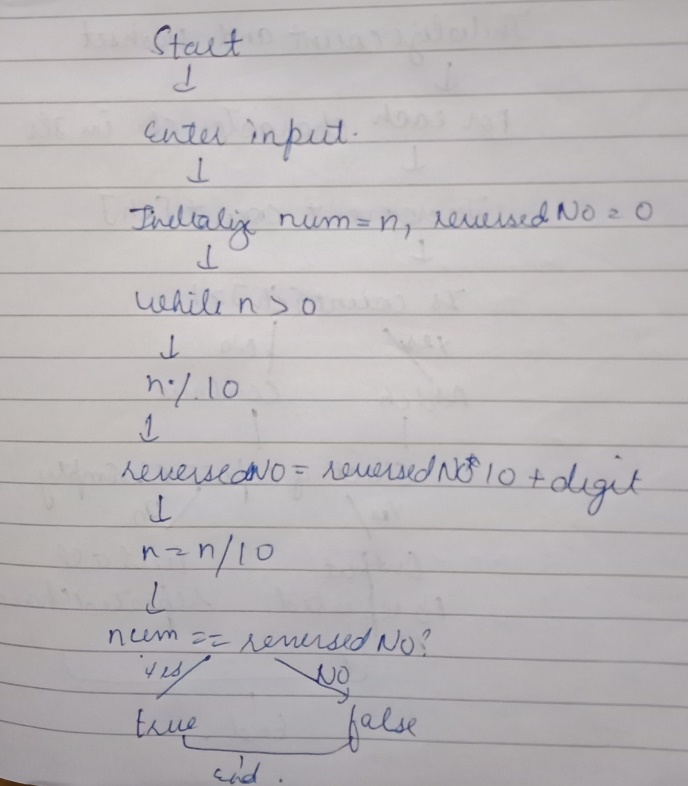
Input: 12345

Output: 54321

Input: -9876

Output: -6789



****

**Explanation -** Firstly, I have created a class of ReverseNo . Inside the main method I’ve taken a integer as input with the help of Scanner class. An empty int variable named reversed is initialized. Then inside the while loop which continues as long as n is not equal to 0 the last digit of n is extracted using the mod operation. The reversed is updated by shifting its digits left and adding the extracted digit. The last digit is removed from n by performing integer division by 10. At last reversed is printed.

**Time Complexity –**  O (logn)

**Space Complexity-** O (1)

9. String Palindrome

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

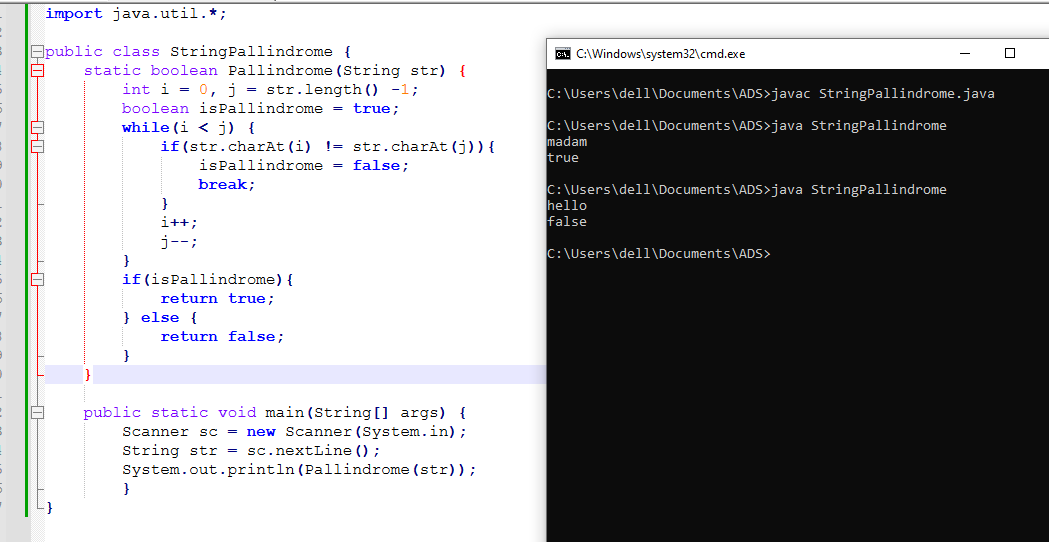
Test Cases:

Input: "madam"

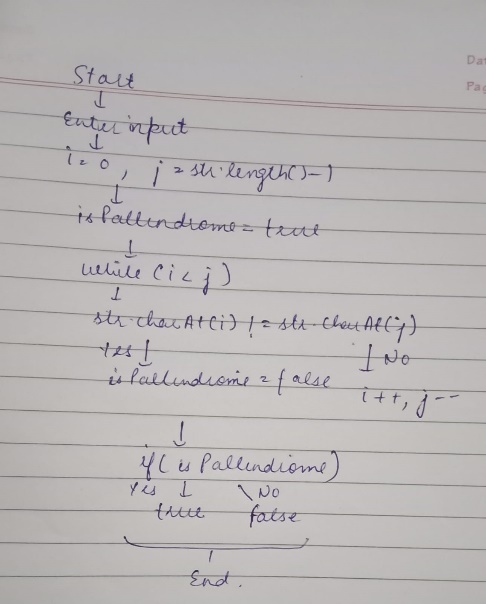
Output: true

Input: "hello"

Output: false



**Flowchart-**

****

**Explanation -** Firstly, I’ve created a public class named StringPallindrome. Inside it I’ve created a method named as Pallindrome of Boolean return type. i is initialized to 0 to point to the start of the string and j is initialized to str.length() - 1 to point to the end of the string. A boolean variable isPallindrome is initialized to true. A while loop that continues as long as i is less than j. If the characters at I and j are not equal, isPallindrome is set to false, and the loop is exited using break. This indicates that the string is not a palindrome. Moves i one step forward (i++) and j one step backward (j--) to compare the next pair of characters. If isPallindrome remains true , the method returns true.Otherwise, it returns false.

**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.

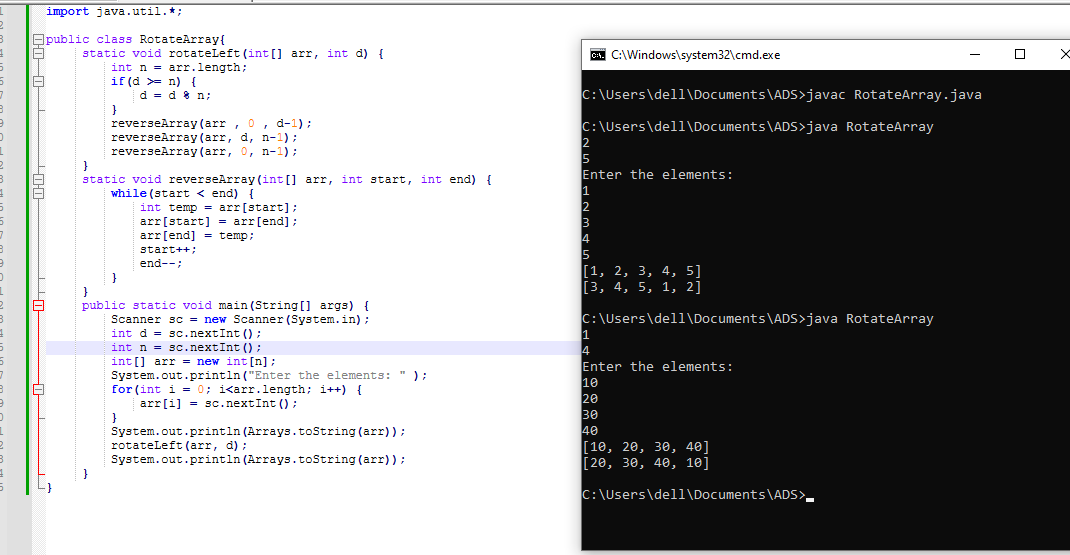
Test Cases:

Input: arr = [1, 2, 3, 4, 5], d = 2

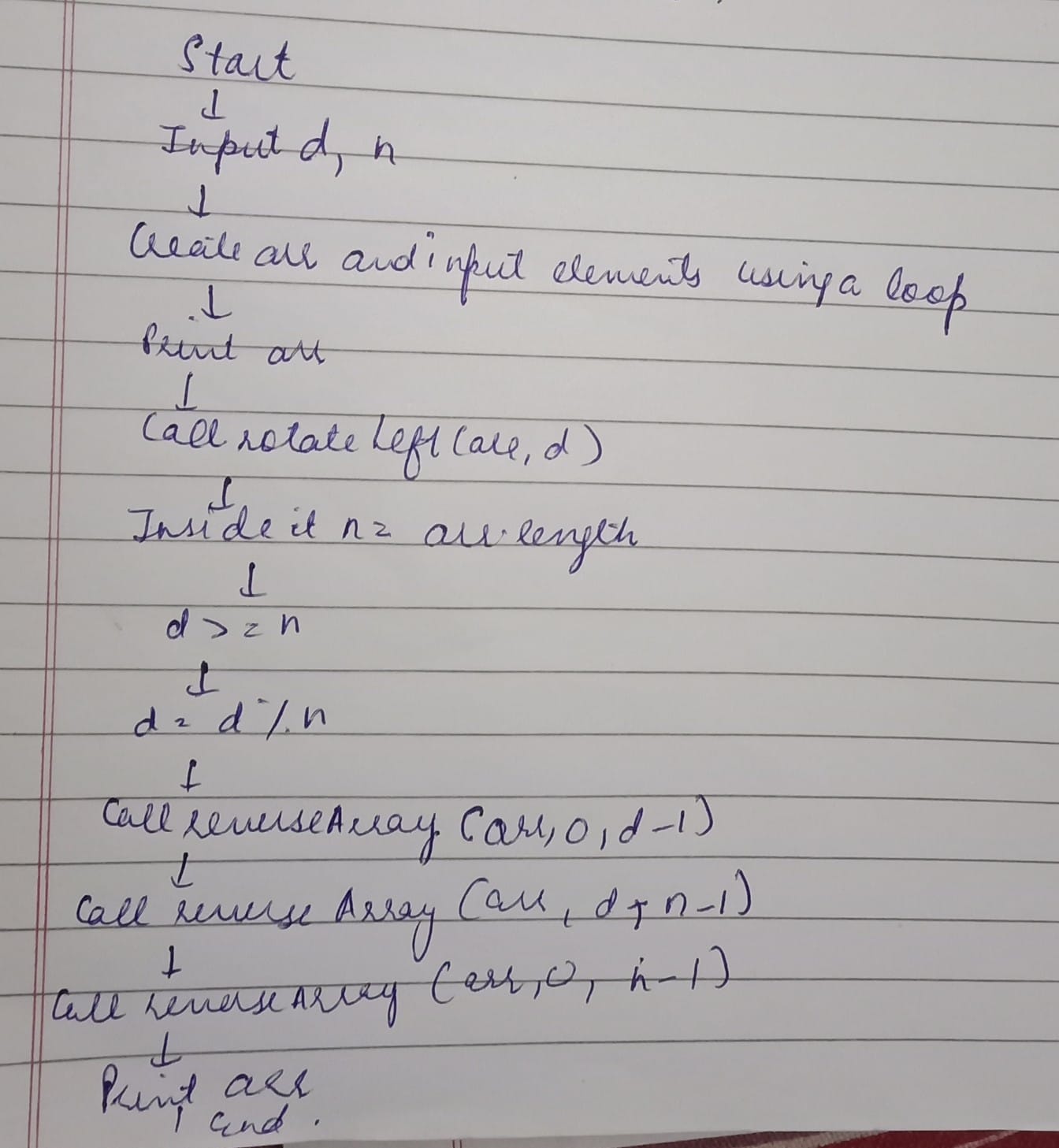
Output: [3, 4, 5, 1, 2]

Input: arr = [10, 20, 30, 40], d = 1

Output: [20, 30, 40, 10]



**Flowchart –**

****

**Explanation -** Firstly, I’ve created a public class named RotateArray. Inside it I’ve created a method named as rotateLeft of void return type. In the rotateLeft method, the program first determines the length n of the array and handles cases where d is greater than or equal to n by computing d % n. The reverseArray method is used to reverse elements. The array is first reversed from index 0 to d-1, then from d to n-1, and finally, the entire array from 0 to n-1 is reversed, resulting in the desired left-rotated array. The main method handles user input, where d and n are read, followed by n array elements, and then calls the rotateLeft method to perform the rotation. The program prints the original array and the array after left rotation.

**Time Complexity** – O(n)

**Space Complexity** – O(1)