C-DAC Mumbai

Date 26/09/2024

Subject: Algorithm and Data Structure
Assignment 2

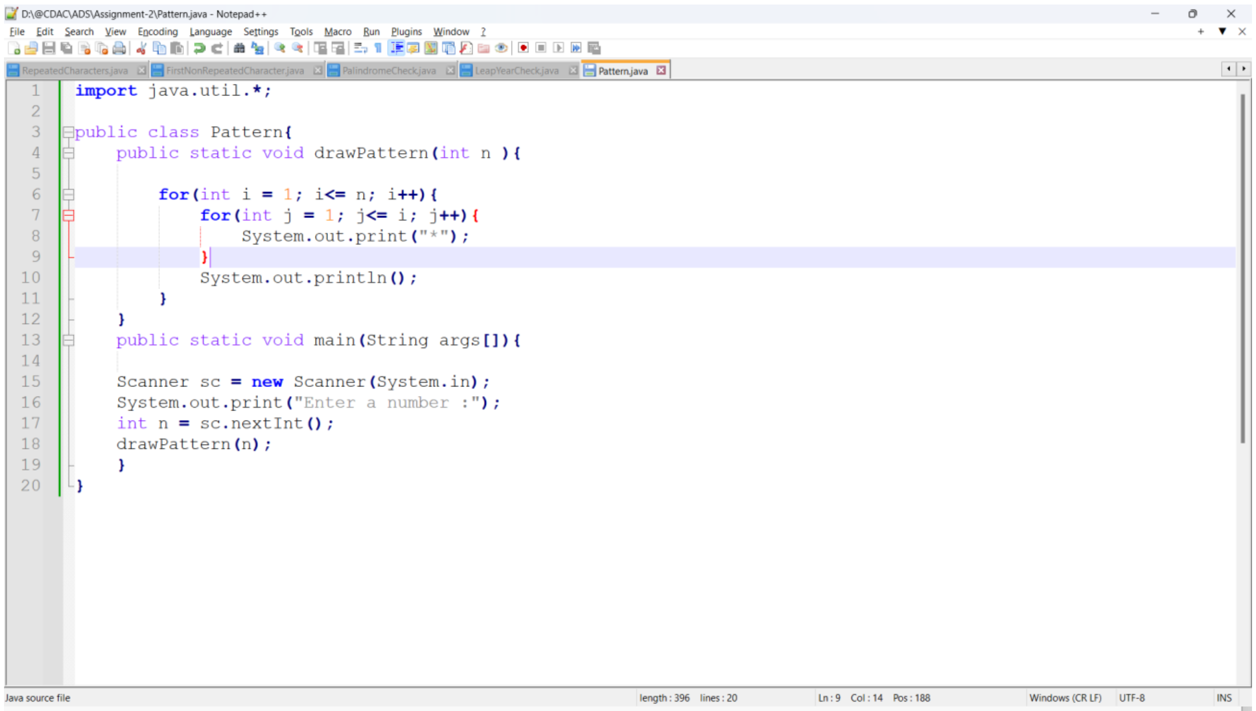
Solve the assignment with following thing to be added in each question.

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

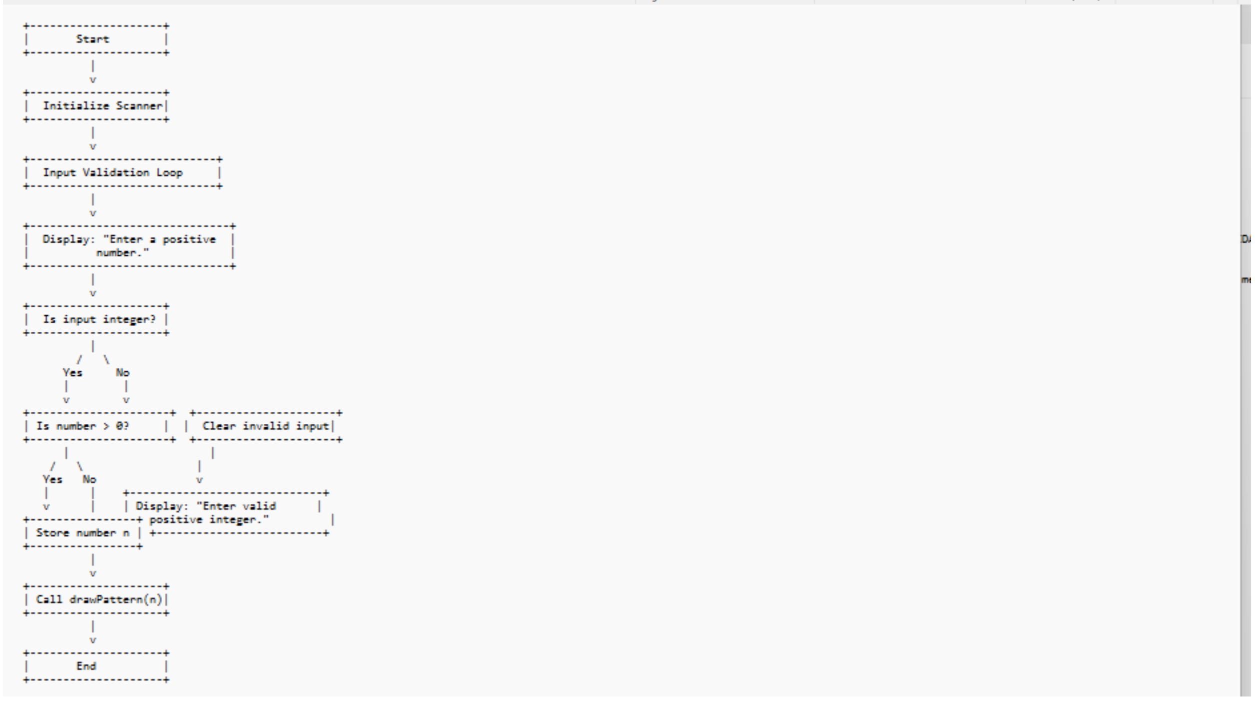
1. Printing Patterns
Problem: Write a Java program to print patterns such as a right triangle of stars.

Test Cases:

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


D:\@CDAC\ADS\Assignment-2\Pattern.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
RepeatedCharacters.java
FirstNonRepeatedCharacter.java
PalindromeCheck.java
LeapYearCheck.java
Pattern.java
Java source file
length : 396 lines : 20
Ln : 9 Col : 14 Pos : 188
Windows (CR LF) UTF-8 INS

```java
import java.util.*;

public class Pattern{
    public static void drawPattern(int n ){

        for(int i = 1; i<= n; i++){
            for(int j = 1; j<= i; j++){
                System.out.print("*");
            }
            System.out.println();
        }
    }
    public static void main(String args[]){

    Scanner sc = new Scanner(System.in);
    System.out.print("Enter a number :");
    int n = sc.nextInt();
    drawPattern(n);
    }
}
```



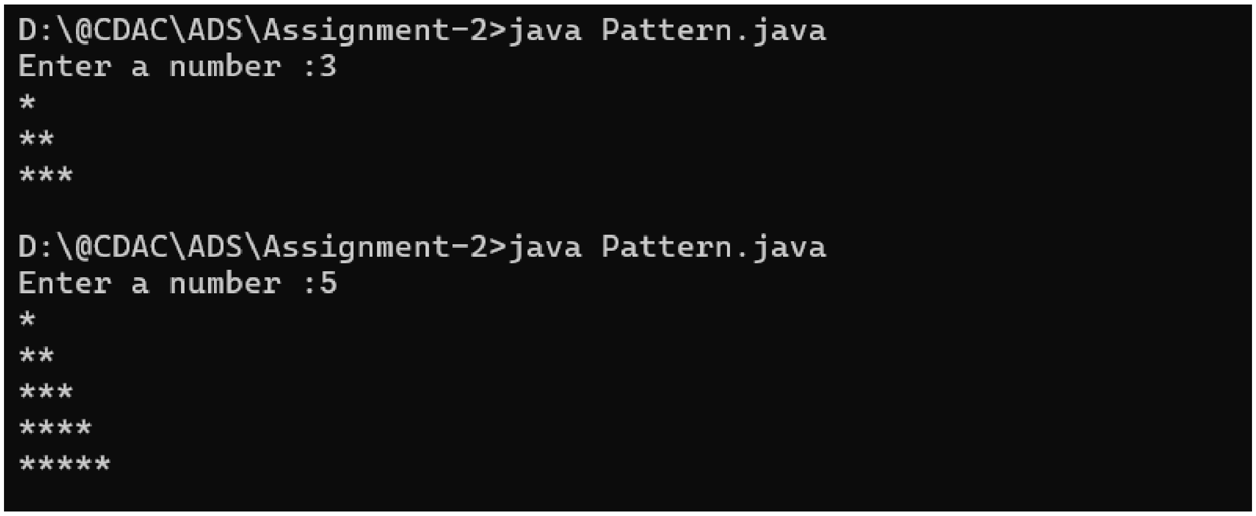
```
+-------------------+
|       Start       |
+-------------------+
          |
          v
+-------------------+
| Initialize Scanner|
+-------------------+
          |
          v
+---------------------------+
| Input Validation Loop     |
+---------------------------+
          |
          v
+-----------------------------+
| Display: "Enter a positive  |
|          number."           |
+-----------------------------+
          |
          v
+-------------------+
| Is input integer? |
+-------------------+
          |
        /   \
      Yes     No
       |       |
       v       v
+--------------------+  +--------------------+
| Is number > 0?     |  |  Clear invalid input|
+--------------------+  +--------------------+
       |                      |
     /   \                   |
   Yes   No                  v
    |     |     +---------------------------+
    v     |     | Display: "Enter valid     |
+---------------+ positive integer."          |
| Store number n | +------------------------+
+---------------+
          |
          v
+-------------------+
| Call drawPattern(n)|
+-------------------+
          |
          v
+-------------------+
|       End         |
+-------------------+
```

☐ **Start**: Begin the program.
☐ **Initialize Scanner**: Create a Scanner object for user input.
☐ **Input Validation Loop**:

- Display a prompt: "Enter a positive number."
- Check if the input is an integer.
    - **Yes**: Check if the integer is greater than 0.
        - **Yes**: Store the number and exit the loop.
        - **No**: Display an error message.

- **No**: Clear invalid input and display an error message.

- **Call drawPattern(n)**: Call the method to draw the pattern.
- **End**: Close the Scanner and end the program.



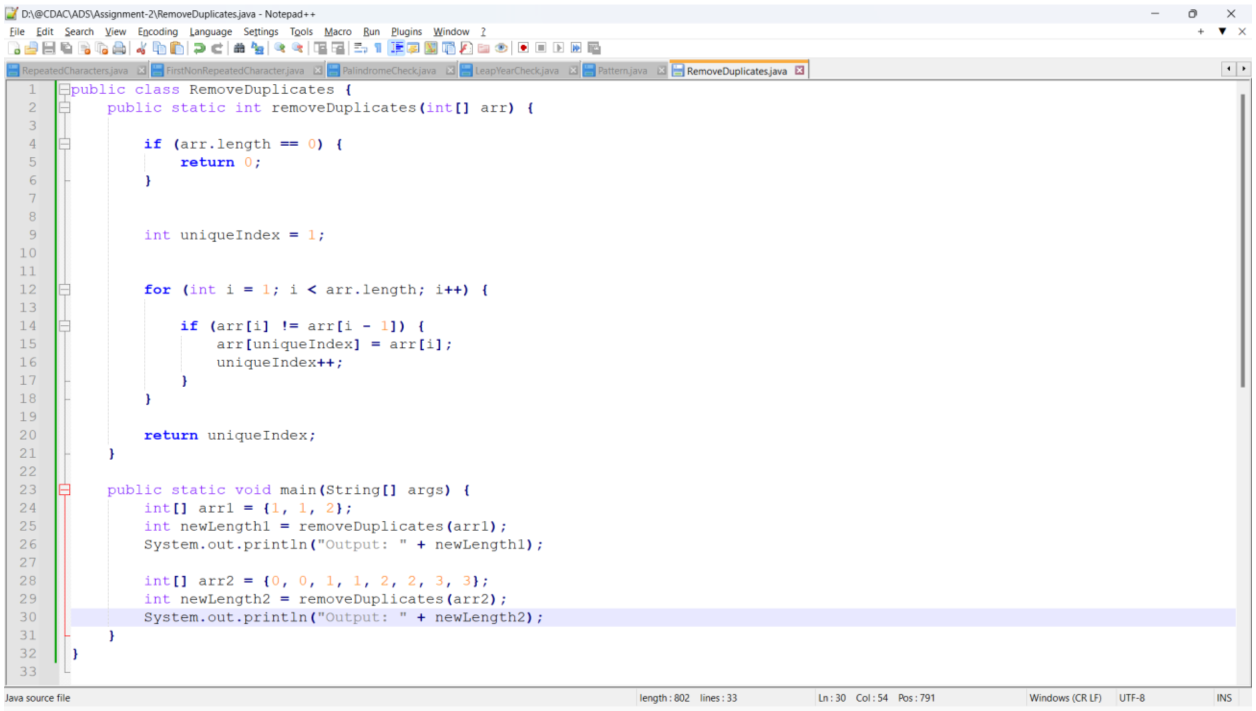
```
D:\@CDAC\ADS\Assignment-2>java Pattern.java
Enter a number :3
*
**
***

D:\@CDAC\ADS\Assignment-2>java Pattern.java
Enter a number :5
*
**
***
****
*****
```

- **Time Complexity**: O(n2)
- **Space Complexity**: O(1)

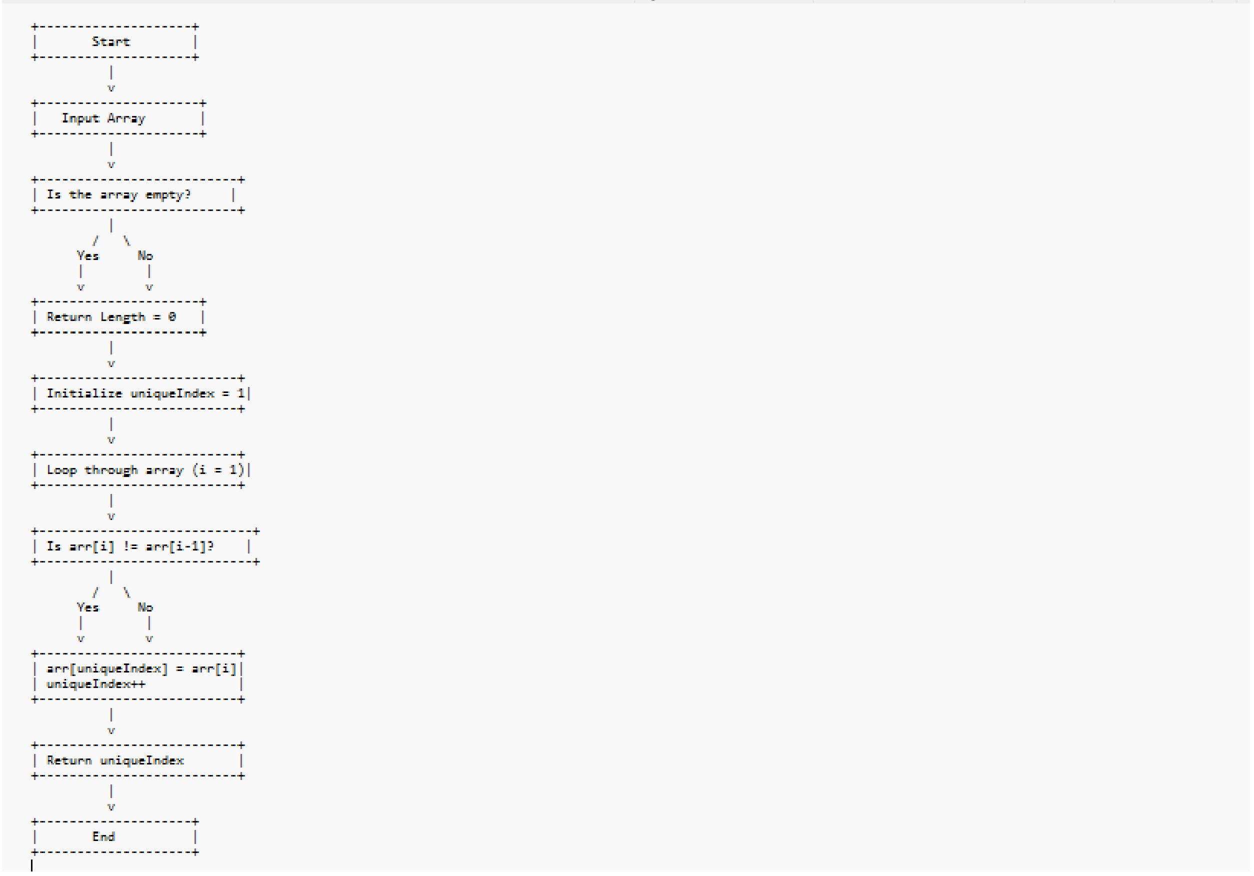
2. Remove Array Duplicates
Problem: Write a Java program to remove duplicates from a sorted array and return the new length of the array.

Test Cases:

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



```java
public class RemoveDuplicates {
    public static int removeDuplicates(int[] arr) {

        if (arr.length == 0) {
            return 0;
        }


        int uniqueIndex = 1;


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

            if (arr[i] != arr[i - 1]) {
                arr[uniqueIndex] = arr[i];
                uniqueIndex++;
            }
        }

        return uniqueIndex;
    }

    public static void main(String[] args) {
        int[] arr1 = {1, 1, 2};
        int newLength1 = removeDuplicates(arr1);
        System.out.println("Output: " + newLength1);

        int[] arr2 = {0, 0, 1, 1, 2, 2, 3, 3};
        int newLength2 = removeDuplicates(arr2);
        System.out.println("Output: " + newLength2);
    }
}
```



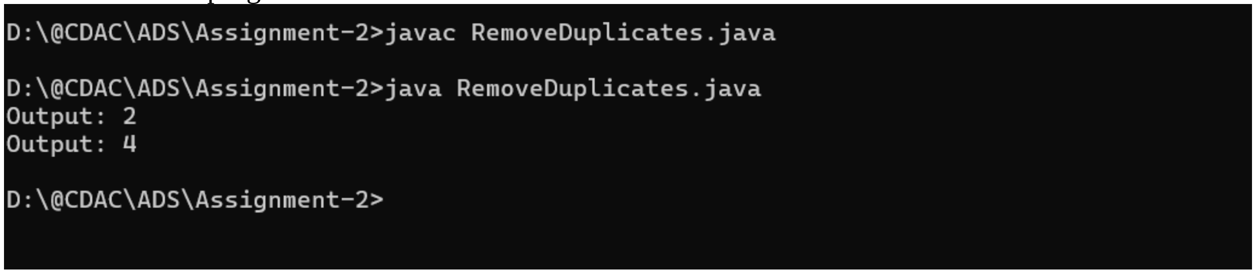
```
+--------------------+
|       Start        |
+--------------------+
          |
          v
+---------------------+
|   Input Array       |
+---------------------+
          |
          v
+--------------------------+
| Is the array empty?     |
+--------------------------+
          |
        /   \
      Yes     No
       |       |
       v       v
+---------------------+
| Return Length = 0   |
+---------------------+
          |
          v
+--------------------------+
| Initialize uniqueIndex = 1|
+--------------------------+
          |
          v
+--------------------------+
| Loop through array (i = 1)|
+--------------------------+
          |
          v
+----------------------------+
| Is arr[i] != arr[i-1]?     |
+----------------------------+
          |
        /   \
      Yes     No
       |       |
       v       v
+--------------------------+
| arr[uniqueIndex] = arr[i]|
| uniqueIndex++            |
+--------------------------+
          |
          v
+--------------------------+
| Return uniqueIndex       |
+--------------------------+
          |
          v
+--------------------+
|       End          |
+--------------------+
```

- **Start**: Begin the program.
- **Input Array**: Take the sorted array as input.

 Check if Array is Empty:
- If **Yes**, return length 0.
- If **No**, continue to the next step.

 Initialize uniqueIndex to 1: Set the unique index for unique elements.
 Loop through Array (from index 1 to end):
- Check if the current element is different from the previous one:
    - If **Yes**:
        - Assign current element to arr[uniqueIndex].
        - Increment uniqueIndex.
    - If **No**, continue to the next iteration.

 Return uniqueIndex: The length of the array after removing duplicates.
 End: End the program.



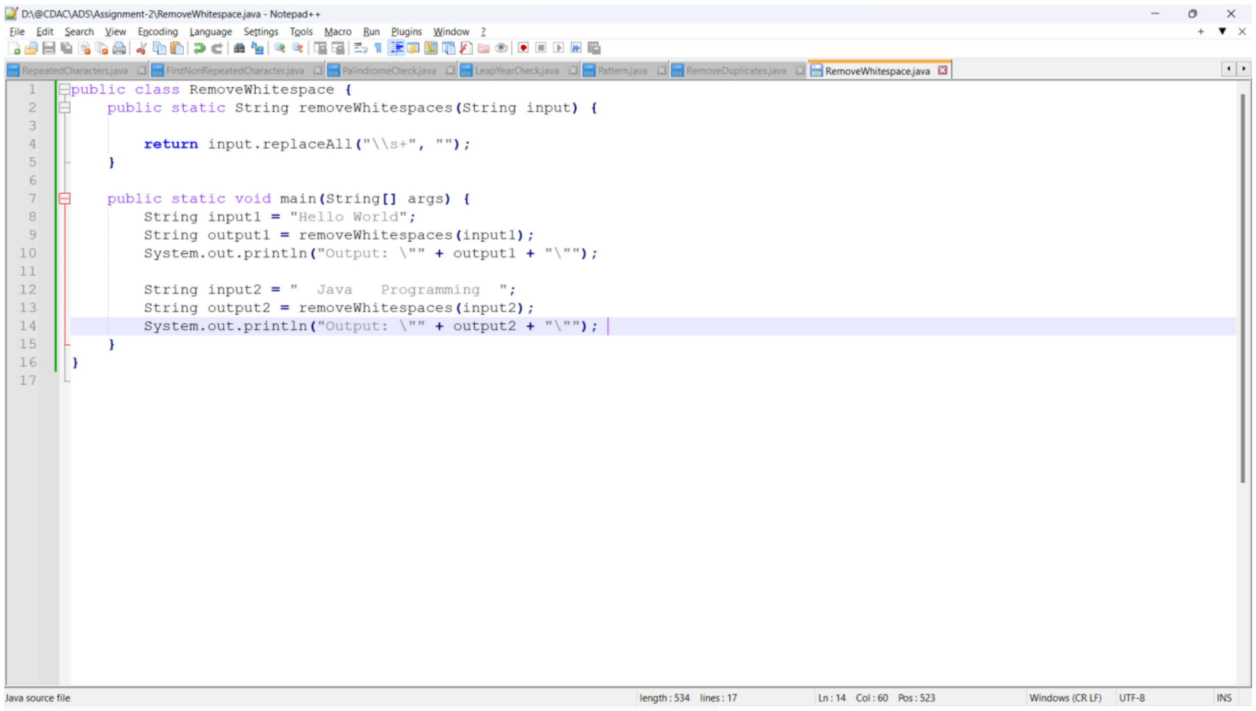
```
D:\@CDAC\ADS\Assignment-2>javac RemoveDuplicates.java

D:\@CDAC\ADS\Assignment-2>java RemoveDuplicates.java
Output: 2
Output: 4

D:\@CDAC\ADS\Assignment-2>
```

 Time Complexity: O(n), where nnn is the length of the input array. We traverse the array once.
 Space Complexity: O(1) since we are modifying the array in-place and using a constant amount of extra space.

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

Test Cases:

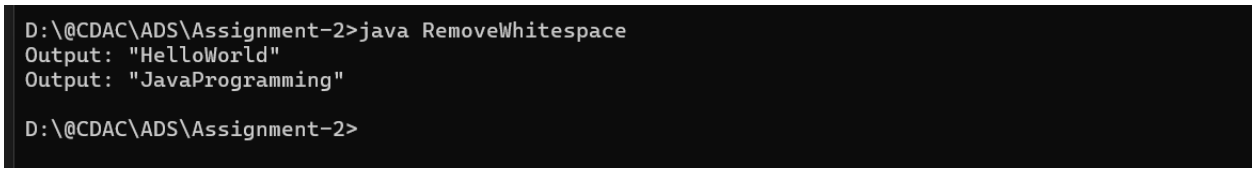
Input: "Hello World"
Output: "HelloWorld"
Input: " Java   Programming "
Output: "JavaProgramming"


D:\@CDAC\ADS\Assignment-2\RemoveWhitespace.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
RepeatedCharacters.java
FirstNonRepeatedCharacter.java
PalindromeCheck.java
LeapYearCheck.java
Pattern.java
RemoveDuplicates.java
RemoveWhitespace.java
Java source file
length : 534 lines : 17
Ln : 14 Col : 60 Pos : 523
Windows (CR LF) UTF-8 INS

```java
public class RemoveWhitespace {
    public static String removeWhitespaces(String input) {

        return input.replaceAll("\\s+", "");
    }

    public static void main(String[] args) {
        String input1 = "Hello World";
        String output1 = removeWhitespaces(input1);
        System.out.println("Output: \"" + output1 + "\"");

        String input2 = "  Java   Programming  ";
        String output2 = removeWhitespaces(input2);
        System.out.println("Output: \"" + output2 + "\"");
    }
}
```

```
+---------------------+
|        Start        |
+---------------------+
           |
           v
+----------------------+
|    Input String      |
+----------------------+
           |
           v
+------------------------------+
| Call removeWhitespaces(input)|
+------------------------------+
           |
           v
+----------------------------+
|   Output: Display Result   |
+----------------------------+
           |
           v
+---------------------+
|         End         |
+---------------------+
```

- **Start**: Begin the program.
- **Input String**: Take the input string from the user.
- **Call removeWhitespaces(input)**:
  - This method uses replaceAll("\\s+", "") to remove all whitespace from the string.
- **Display Output**: Print the resulting string after removing whitespace.
- **End**: End the program.



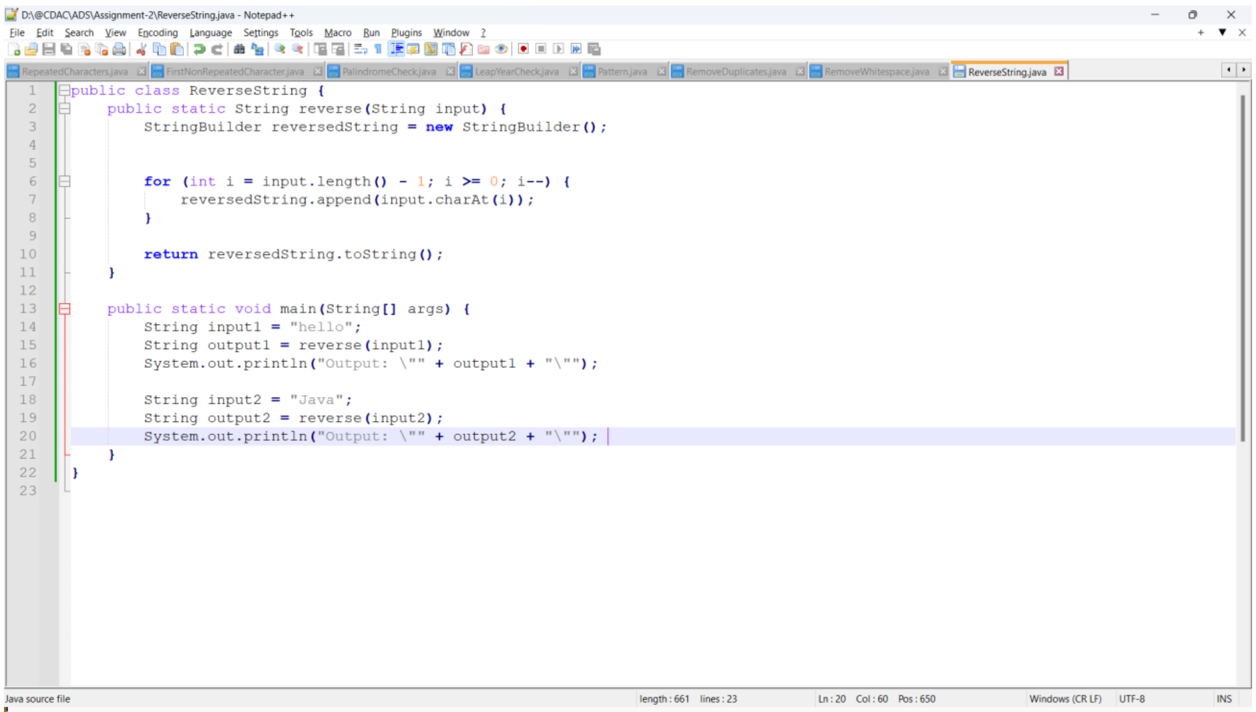
```
D:\@CDAC\ADS\Assignment-2>java RemoveWhitespace
Output: "HelloWorld"
Output: "JavaProgramming"

D:\@CDAC\ADS\Assignment-2>
```

- **Time Complexity**: O(n)
- **Space Complexity**: O(n)

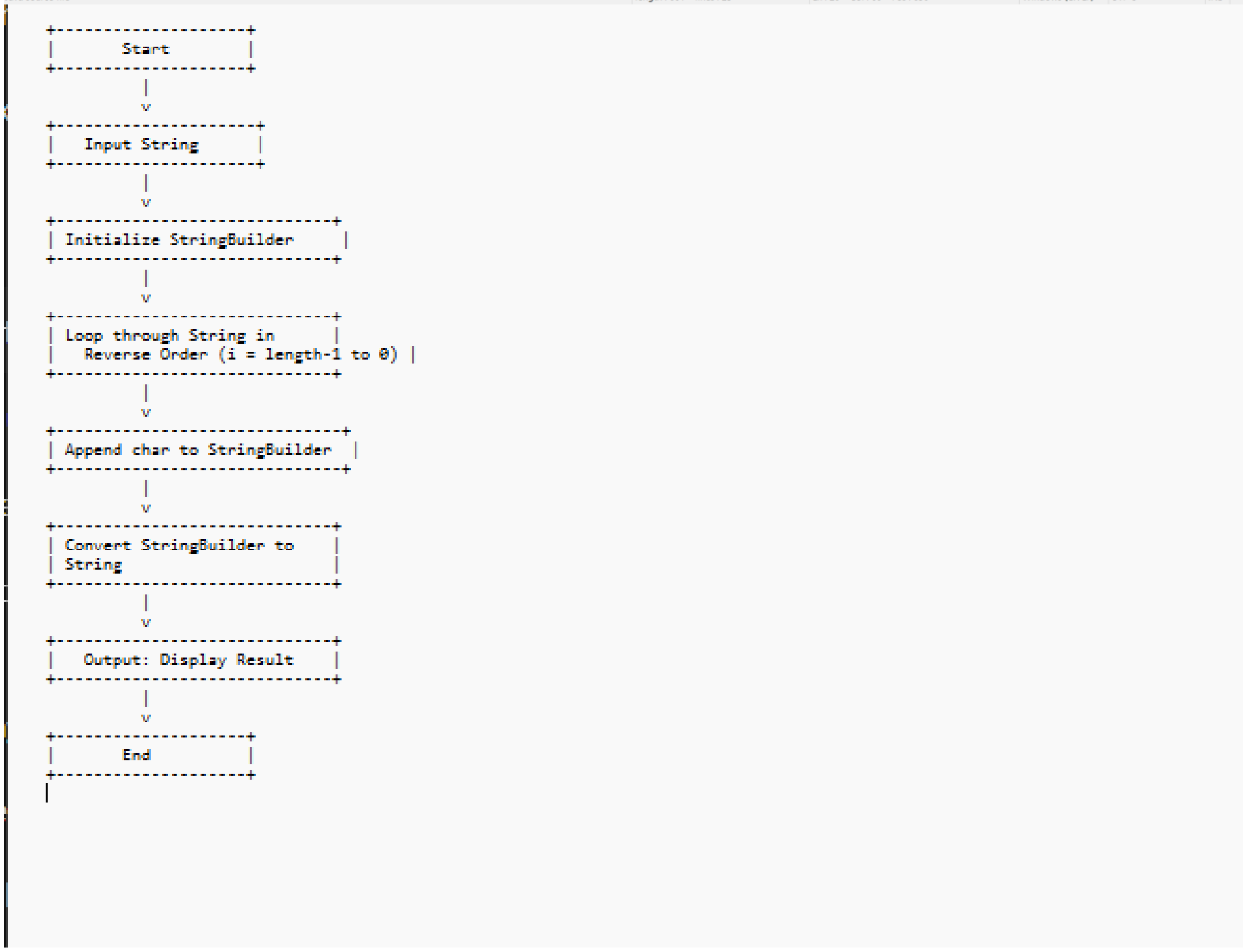
4. Reverse a String
Problem: Write a Java program to reverse a given string.

Test Cases:

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


D:\@CDAC\ADS\Assignment-2\ReverseString.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
RepeatedCharacters.java
FirstNonRepeatedCharacter.java
PalindromeCheck.java
LeapYearCheck.java
Pattern.java
RemoveDuplicates.java
RemoveWhitespace.java
ReverseString.java
Java source file
length : 661 lines : 23
Ln : 20 Col : 60 Pos : 650
Windows (CR LF) UTF-8 INS

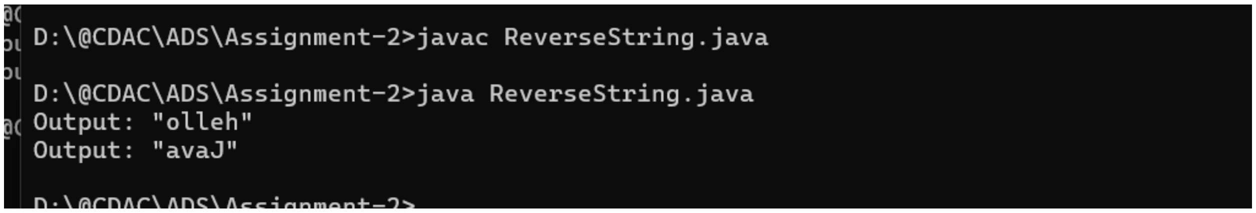
```java
public class ReverseString {
    public static String reverse(String input) {
        StringBuilder reversedString = new StringBuilder();


        for (int i = input.length() - 1; i >= 0; i--) {
            reversedString.append(input.charAt(i));
        }

        return reversedString.toString();
    }

    public static void main(String[] args) {
        String input1 = "hello";
        String output1 = reverse(input1);
        System.out.println("Output: \"" + output1 + "\"");

        String input2 = "Java";
        String output2 = reverse(input2);
        System.out.println("Output: \"" + output2 + "\"");
    }
}
```



```
+---------------------+
|        Start        |
+---------------------+
          |
          v
+----------------------+
|   Input String       |
+----------------------+
          |
          v
+-----------------------------+
| Initialize StringBuilder    |
+-----------------------------+
          |
          v
+-----------------------------+
| Loop through String in      |
|   Reverse Order (i = length-1 to 0) |
+-----------------------------+
          |
          v
+------------------------------+
| Append char to StringBuilder |
+------------------------------+
          |
          v
+-----------------------------+
| Convert StringBuilder to    |
| String                      |
+-----------------------------+
          |
          v
+-----------------------------+
|   Output: Display Result    |
+-----------------------------+
          |
          v
+---------------------+
|        End          |
+---------------------+
```

- **Start**: Begin the program.

- **Input String**: Take the input string from the user.
- **Initialize StringBuilder**: Create a StringBuilder to store the reversed string.
- **Loop through String in Reverse**:
  - Iterate from the last character of the string to the first.
  - Append each character to the StringBuilder.
- **Convert StringBuilder to String**: Convert the StringBuilder to a String to get the reversed string.
- **Display Output**: Print the resulting reversed string.
- **End**: End the program.



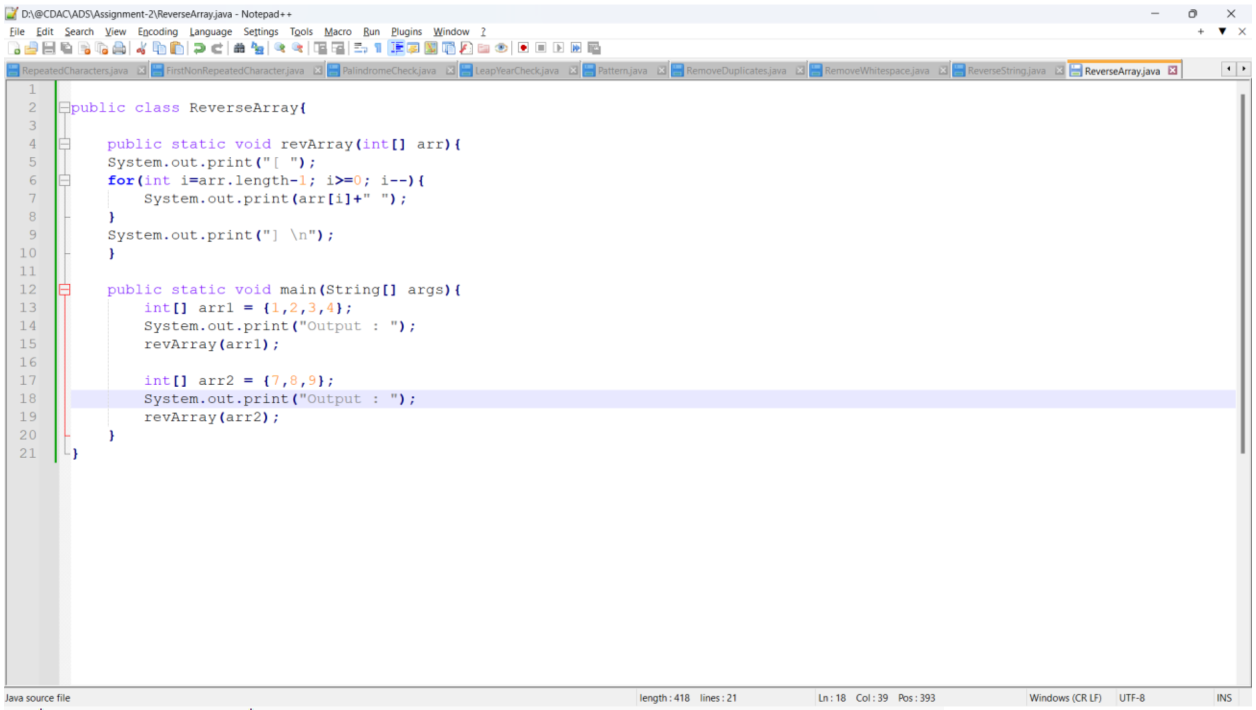
```
D:\@CDAC\ADS\Assignment-2>javac ReverseString.java

D:\@CDAC\ADS\Assignment-2>java ReverseString.java
Output: "olleh"
Output: "avaJ"

D:\@CDAC\ADS\Assignment-2>
```

- **Time Complexity**: O(n) where nnn is the length of the input string. The loop iterates through each character exactly once.
- **Space Complexity**: O(n) in the worst case, as a new string (via StringBuilder) is created to store the reversed string.

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]



D:\@CDAC\ADS\Assignment-2\ReverseArray.java - Notepad++

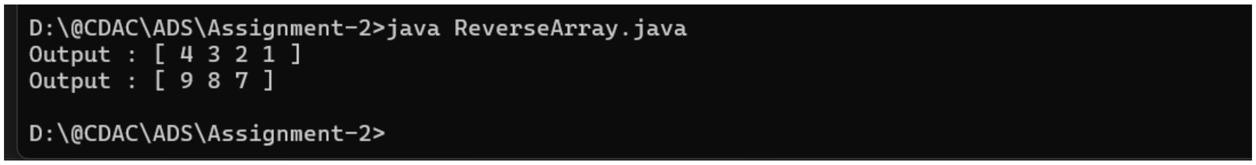
```java

public class ReverseArray{

    public static void revArray(int[] arr){
    System.out.print("[ ");
    for(int i=arr.length-1; i>=0; i--){
        System.out.print(arr[i]+" ");
    }
    System.out.print("] \n");
    }

    public static void main(String[] args){
        int[] arr1 = {1,2,3,4};
        System.out.print("Output : ");
        revArray(arr1);

        int[] arr2 = {7,8,9};
        System.out.print("Output : ");
        revArray(arr2);
    }
}
```

Java source file length : 418 lines : 21 Ln : 18 Col : 39 Pos : 393 Windows (CR LF) UTF-8 INS

```
+--------------------+
|       Start        |
+--------------------+
          |
          v
+---------------------+
|   Define Input      |
|      Array          |
+---------------------+
          |
          v
+---------------------+
|   Print "Output:"   |
+---------------------+
          |
          v
+---------------------+
|  Call revArray(arr) |
+---------------------+
          |
          v
+-------------------------------+
|     In revArray Method        |
|                             |
|   Print "[ "                  |
|                             |
|   For i = arr.length - 1    |
|     to 0                      |
|       Print arr[i] + " "     |
|   Print "]"                  |
+-------------------------------+
          |
          v
+---------------------+
|   Return to Main    |
+---------------------+
          |
          v
+--------------------+
|       End          |
+--------------------+
```

- **Start**: Begin the program.
- **Input**: Define the input array (e.g., arr1 and arr2).
- **Output Message**: Print the "Output: " message.

- **Call revArray Method**: Pass the array to the revArray method.
- **In revArray Method**:
  - Print "[".
  - Loop through the array from the last index to the first.
  - Print each element followed by a space.
  - Print "]" to close the output.
- **Return to Main**: After the array is printed, return to the main method.
- **End**: End the program.



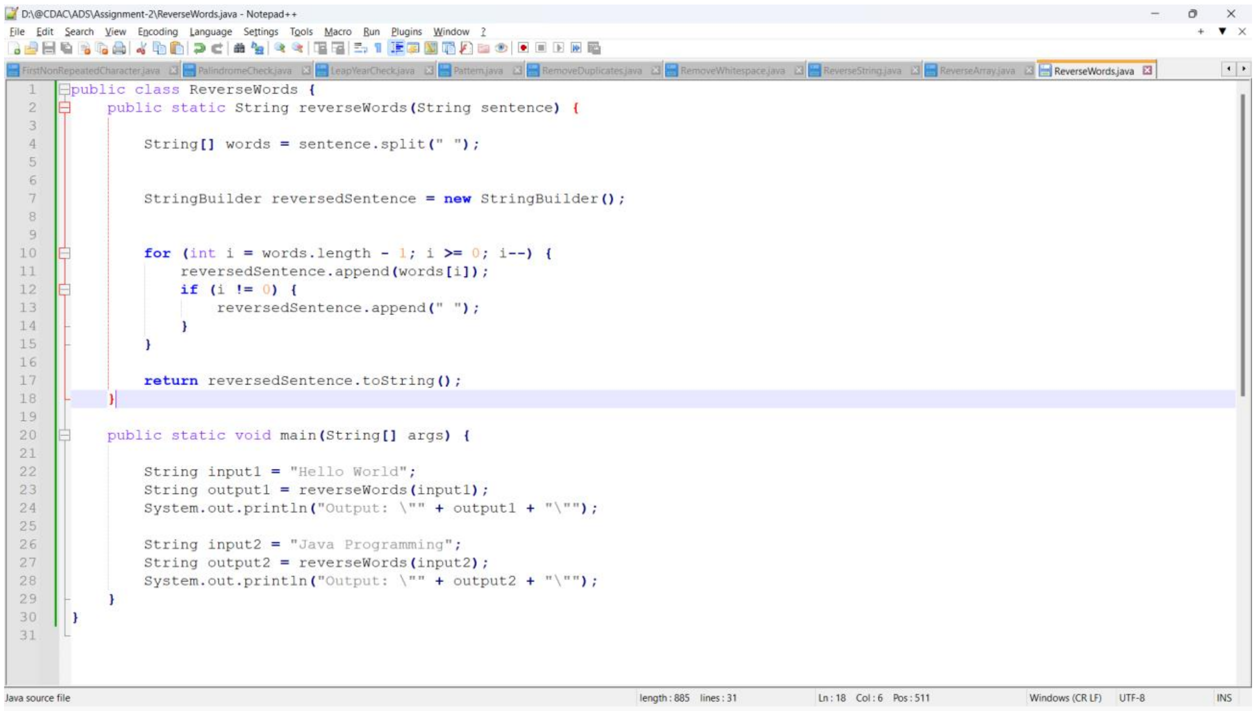
```
D:\@CDAC\ADS\Assignment-2>java ReverseArray.java
Output : [ 4 3 2 1 ]
Output : [ 9 8 7 ]

D:\@CDAC\ADS\Assignment-2>
```

- **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"


D:\@CDAC\ADS\Assignment-2\ReverseWords.java - Notepad++

```java
public class ReverseWords {
    public static String reverseWords(String sentence) {

        String[] words = sentence.split(" ");


        StringBuilder reversedSentence = new StringBuilder();


        for (int i = words.length - 1; i >= 0; i--) {
            reversedSentence.append(words[i]);
            if (i != 0) {
                reversedSentence.append(" ");
            }
        }

        return reversedSentence.toString();
    }

    public static void main(String[] args) {

        String input1 = "Hello World";
        String output1 = reverseWords(input1);
        System.out.println("Output: \"" + output1 + "\"");

        String input2 = "Java Programming";
        String output2 = reverseWords(input2);
        System.out.println("Output: \"" + output2 + "\"");
    }
}
```

Java source file   length : 885   lines : 31   Ln : 18   Col : 6   Pos : 511   Windows (CR LF)   UTF-8   INS

```
+--------------------+
|       Start        |
+--------------------+
          |
          v
+---------------------+
|   Read Input Sentence|
+---------------------+
          |
          v
+---------------------+
|   Split Sentence    |
|      into Words     |
|  (words = sentence.split(" ")) |
+---------------------+
          |
          v
+---------------------+
| Initialize StringBuilder |
|    (reversedSentence)    |
+---------------------+
          |
          v
+----------------------------------+
| For i = words.length - 1 down to 0 |
+----------------------------------+
          |
          v
+---------------------+
| Append words[i] to StringBuilder |
+---------------------+
          |
          v
+---------------------+
|  If i != 0, append " "   |
+---------------------+
          |
          v
+---------------------+
|   End For Loop       |
+---------------------+
          |
          v
+---------------------+
|   Return reversedSentence as String  |
+---------------------+
          |
          v
+---------------------+
|   Print Output      |
+---------------------+
          |
          v
+--------------------+
|       End          |
+--------------------+
```

- **Start**: Indicates the beginning of the program.

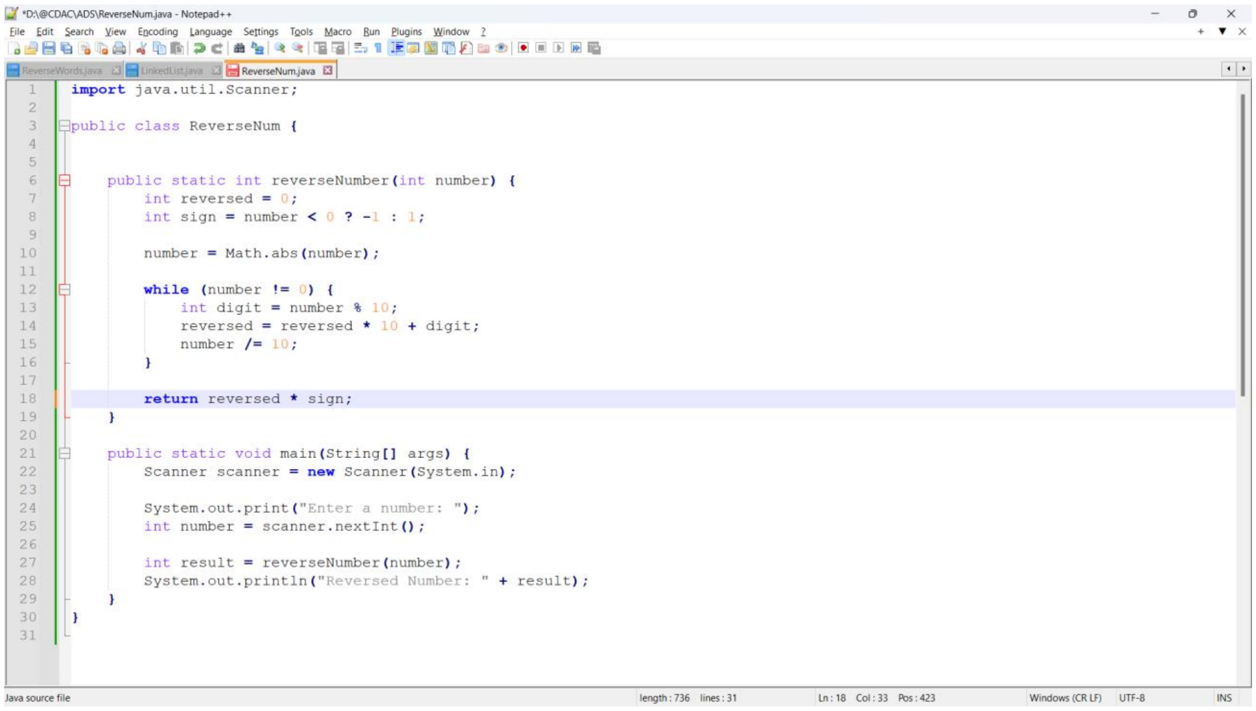
- **Read Input Sentence**: Step where the program reads the input sentence (in the main method).
- **Split Sentence into Words**: Uses the split method to break the sentence into words and store them in an array.
- **Initialize StringBuilder**: Initializes a StringBuilder to construct the reversed sentence.
- **For Loop**: Iterates through the words array in reverse order.
    - **Append words[i]**: Appends the current word to the StringBuilder.
    - **Conditional Check**: Checks if the current index is not zero to append a space after the word.
- **End For Loop**: Marks the completion of the loop.
- **Return String**: Converts the StringBuilder back to a string and returns it.
- **Print Output**: Displays the final reversed sentence in the main method.
- **End**: Indicates the end of the program.

Time Complexity: O(n)
Space Complexity: O(n)

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

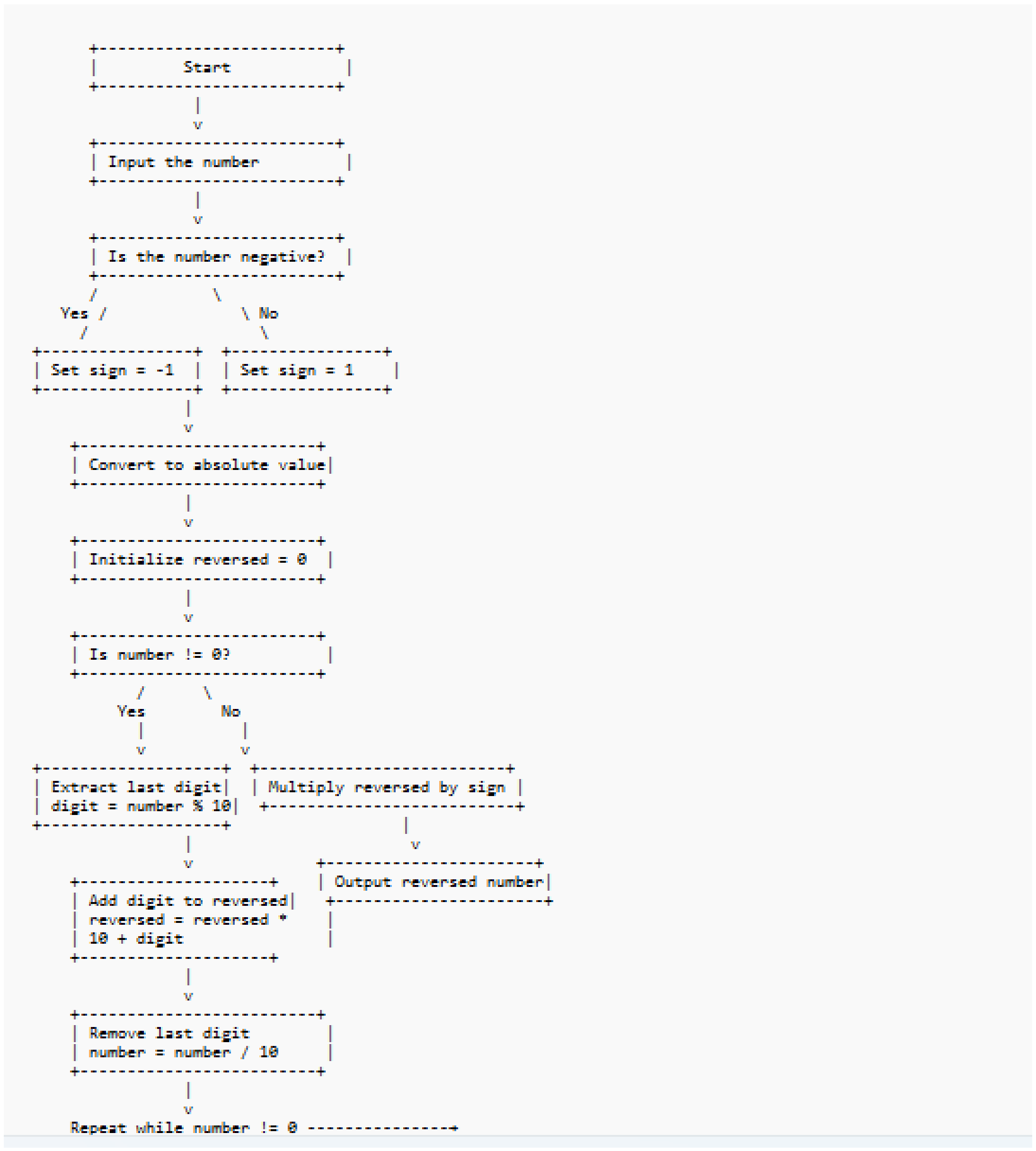
Test Cases:

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


*D:\@CDAC\ADS\ReverseNum.java - Notepad++

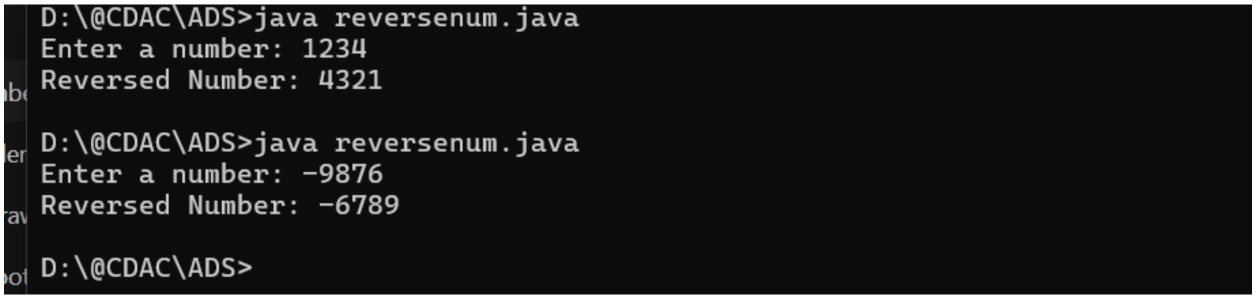
```java
import java.util.Scanner;

public class ReverseNum {


    public static int reverseNumber(int number) {
        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;
        }

        return reversed * sign;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int number = scanner.nextInt();

        int result = reverseNumber(number);
        System.out.println("Reversed Number: " + result);
    }
}
```

Java source file   length : 736   lines : 31   Ln : 18   Col : 33   Pos : 423   Windows (CR LF)   UTF-8   INS



```
       +--------------------------+
       |          Start           |
       +--------------------------+
                   |
                   v
       +--------------------------+
       | Input the number         |
       +--------------------------+
                   |
                   v
       +--------------------------+
       | Is the number negative?  |
       +--------------------------+
       /             \
   Yes /               \ No
     /                  \
+----------------+  +----------------+
| Set sign = -1  |  | Set sign = 1   |
+----------------+  +----------------+
                  |
                  v
     +--------------------------+
     | Convert to absolute value|
     +--------------------------+
                  |
                  v
     +--------------------------+
     | Initialize reversed = 0  |
     +--------------------------+
                  |
                  v
     +--------------------------+
     | Is number != 0?          |
     +--------------------------+
             /       \
          Yes         No
             |          |
             v          v
+--------------------+  +---------------------------+
| Extract last digit|  | Multiply reversed by sign |
| digit = number % 10|  +---------------------------+
+--------------------+              |
                  |                  v
                  v         +----------------------+
     +---------------------+  | Output reversed number|
     | Add digit to reversed|  +----------------------+
     | reversed = reversed *   |
     | 10 + digit              |
     +---------------------+
                  |
                  v
     +--------------------------+
     | Remove last digit        |
     | number = number / 10     |
     +--------------------------+
                  |
                  v
     Repeat while number != 0 ---------------+
```

- **Start**
- **Input** the number.
- Check if the number is negative:
  - If yes, set sign to -1.
  - If no, set sign to 1.
- Convert the number to its absolute value (Math.abs()).
- Initialize reversed = 0.
- While the number is not equal to 0:
  - Extract the last digit (number % 10).
  - Add the digit to reversed (reversed * 10 + digit).
  - Remove the last digit from the number (number / 10).
- Multiply reversed by sign to restore the original sign.
- **Output** the reversed number.
- **End**

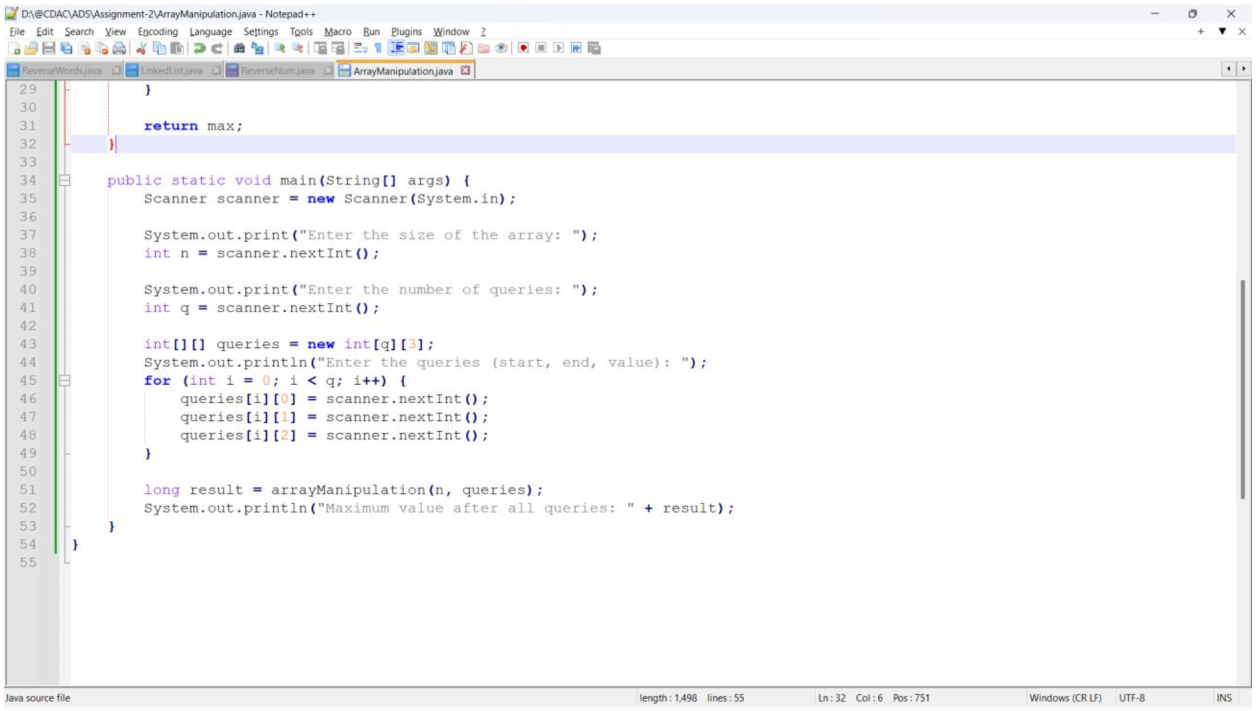

```
D:\@CDAC\ADS>java reversenum.java
Enter a number: 1234
Reversed Number: 4321

D:\@CDAC\ADS>java reversenum.java
Enter a number: -9876
Reversed Number: -6789

D:\@CDAC\ADS>
```

- **Time Complexity: O(n)**
- **Space Complexity: O(1)**

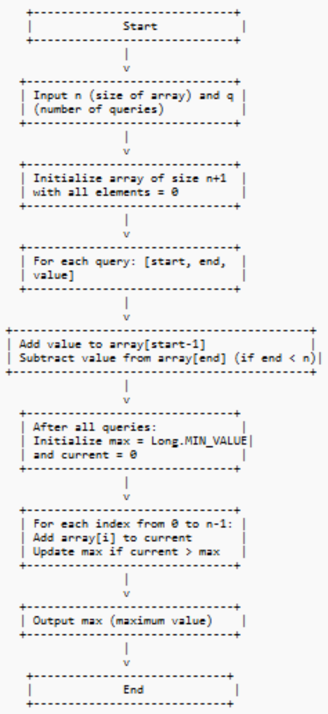
8. Array Manipulation
Problem: Perform a series of operations to manipulate an array based on range update queries. Each query adds a value to a range of indices.

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


D:\@CDAC\ADS\Assignment-2\ArrayManipulation.java - Notepad++
ReverseWords.java
LinkedList.java
ReverseNum.java
ArrayManipulation.java
Java source file
length : 1,498 lines : 55
Ln : 32 Col : 6 Pos : 751
Windows (CR LF)
UTF-8
INS

```java
        }

        return max;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the array: ");
        int n = scanner.nextInt();

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

        int[][] queries = new int[q][3];
        System.out.println("Enter the queries (start, end, value): ");
        for (int i = 0; i < q; i++) {
            queries[i][0] = scanner.nextInt();
            queries[i][1] = scanner.nextInt();
            queries[i][2] = scanner.nextInt();
        }

        long result = arrayManipulation(n, queries);
        System.out.println("Maximum value after all queries: " + result);
    }
}
```

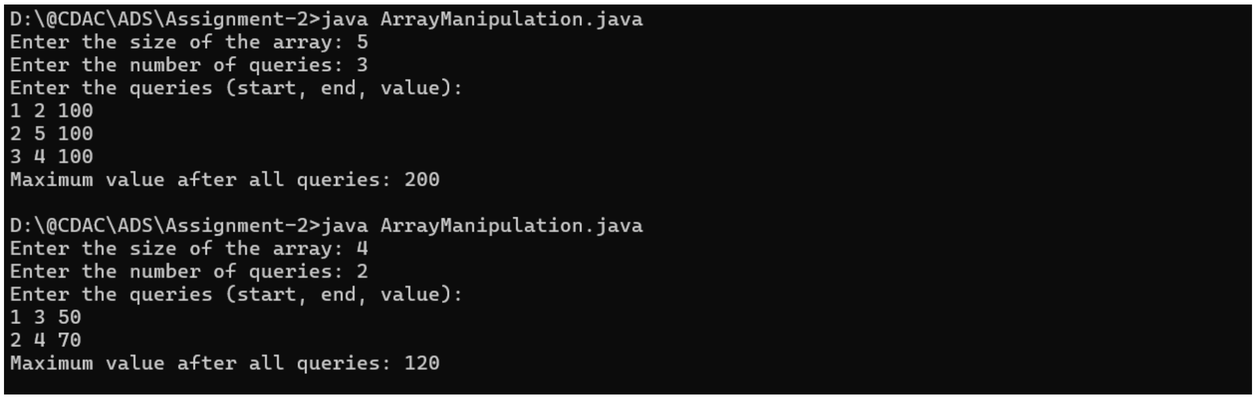


```
  +-----------------------------+
  |            Start            |
  +-----------------------------+
               |
               v
 +-------------------------------+
 | Input n (size of array) and q |
 | (number of queries)           |
 +-------------------------------+
               |
               v
 +-------------------------------+
 | Initialize array of size n+1  |
 | with all elements = 0         |
 +-------------------------------+
               |
               v
 +-------------------------------+
 | For each query: [start, end,  |
 | value]                        |
 +-------------------------------+
               |
               v
+--------------------------------------------+
| Add value to array[start-1]                |
| Subtract value from array[end] (if end < n)|
+--------------------------------------------+
               |
               v
 +-------------------------------+
 | After all queries:            |
 | Initialize max = Long.MIN_VALUE|
 | and current = 0               |
 +-------------------------------+
               |
               v
 +-------------------------------+
 | For each index from 0 to n-1: |
 | Add array[i] to current       |
 | Update max if current > max   |
 +-------------------------------+
               |
               v
 +-------------------------------+
 | Output max (maximum value)    |
 +-------------------------------+
               |
               v
  +-----------------------------+
  |             End             |
  +-----------------------------+
```

- **Start**
- **Input** n (size of array) and q (number of queries)
- Initialize an array array of size n+1 with all elements set to 0.
- **For** each query [start, end, value]:
  - Add value to array[start - 1] (convert to 0-based indexing).
  - Subtract value from array[end] (to mark the end of the effect).
- After processing all queries, initialize max = Long.MIN_VALUE and current = 0.
- **For** each index i from 0 to n-1:

- Add array[i] to current (compute prefix sum).
- Update max if current is greater than max.

 Output the maximum value (max).
 End



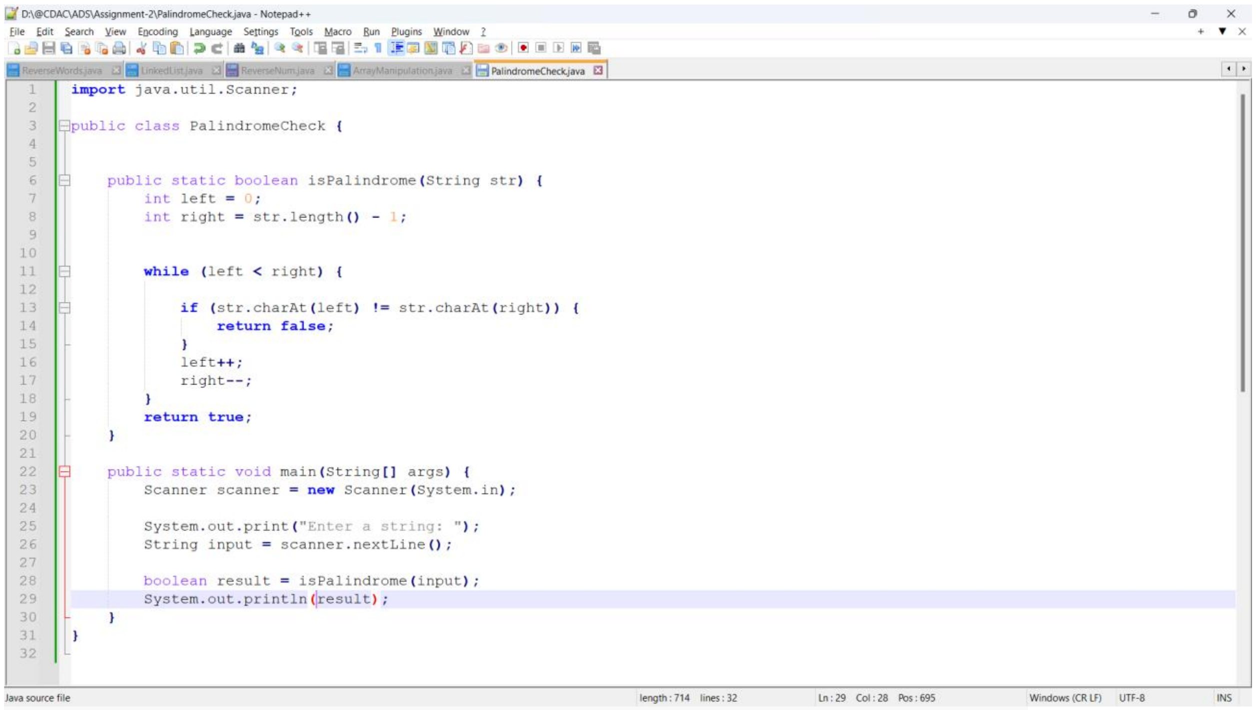
```
D:\@CDAC\ADS\Assignment-2>java ArrayManipulation.java
Enter the size of the array: 5
Enter the number of queries: 3
Enter the queries (start, end, value):
1 2 100
2 5 100
3 4 100
Maximum value after all queries: 200

D:\@CDAC\ADS\Assignment-2>java ArrayManipulation.java
Enter the size of the array: 4
Enter the number of queries: 2
Enter the queries (start, end, value):
1 3 50
2 4 70
Maximum value after all queries: 120
```

Time complexity : **O(n + q)**
Space complexity : **O(n)**.

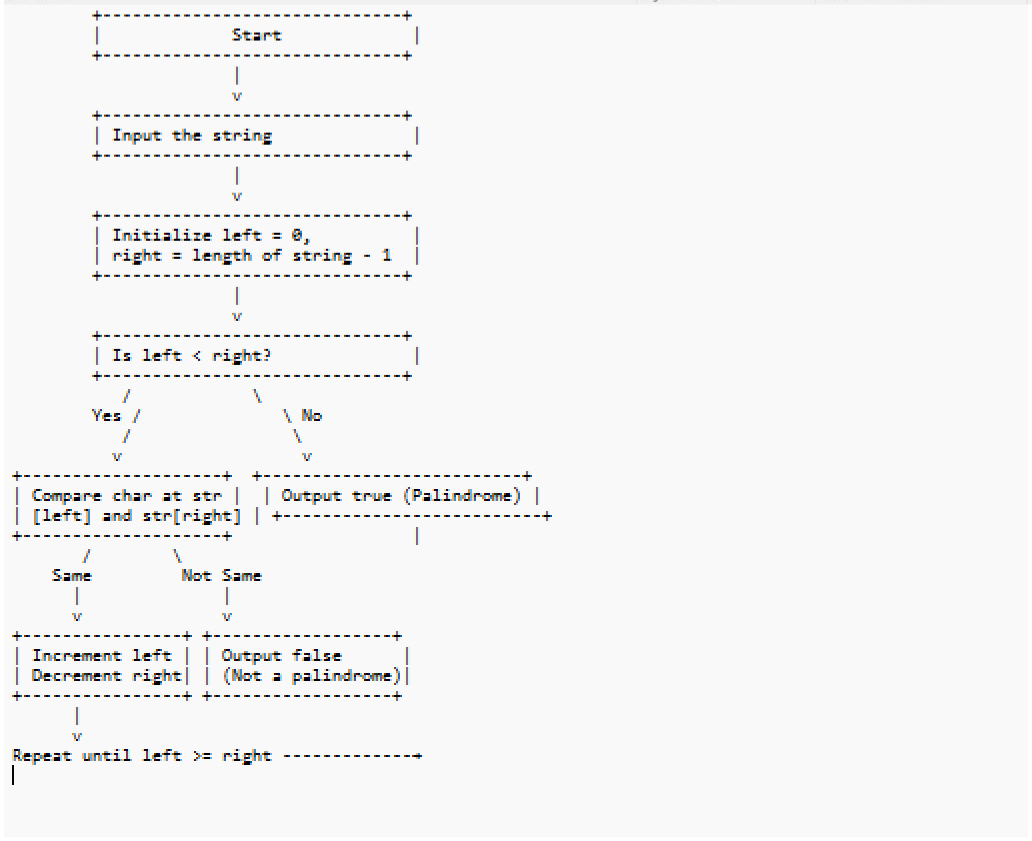
9. String Palindrome
Problem: Write a Java program to check if a given string is a palindrome.

Test Cases:

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


D:\@CDAC\ADS\Assignment-2\PalindromeCheck.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ReverseWords.java
LinkedList.java
ReverseNum.java
ArrayManipulation.java
PalindromeCheck.java
Java source file
length : 714 lines : 32
Ln : 29 Col : 28 Pos : 695
Windows (CR LF)
UTF-8
INS

```java
import java.util.Scanner;

public class PalindromeCheck {


    public static boolean isPalindrome(String str) {
        int left = 0;
        int right = str.length() - 1;


        while (left < right) {

            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        boolean result = isPalindrome(input);
        System.out.println(result);
    }
}
```



```
        +-------------------------------+
        |             Start             |
        +-------------------------------+
                      |
                      v
        +-------------------------------+
        | Input the string              |
        +-------------------------------+
                      |
                      v
        +-------------------------------+
        | Initialize left = 0,          |
        | right = length of string - 1  |
        +-------------------------------+
                      |
                      v
        +-------------------------------+
        | Is left < right?              |
        +-------------------------------+
           /            \
      Yes /               \ No
         /                 \
        v                   v
+--------------------+ +--------------------------+
| Compare char at str |  | Output true (Palindrome) |
| [left] and str[right] | +--------------------------+
+--------------------+                |
       /        \
    Same         Not Same
      |             |
      v             v
+----------------+ +------------------+
| Increment left | | Output false     |
| Decrement right| | (Not a palindrome)|
+----------------+ +------------------+
      |
      v
Repeat until left >= right -------------+
|
```

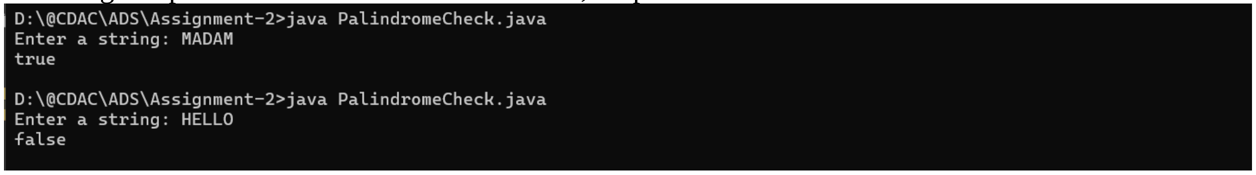
- **Start**: The program starts and takes the input string from the user.
- **Initialize**: Set two pointers: left to 0 (the start of the string) and right to the length of the string minus 1 (the end of the string).
- **Check if left < right**:

- If left is still less than right, continue checking characters.
- If left is greater than or equal to right, all characters have been compared, and the string is a palindrome, so output true.

 Character Comparison: Compare the characters at the positions left and right:

- If they are the same, increment left and decrement right, and repeat the process.
- If they are different, the string is not a palindrome, so output false.

 Output: After all comparisons, if the loop completes without mismatches, output true indicating that the string is a palindrome. If a mismatch is found, output false.



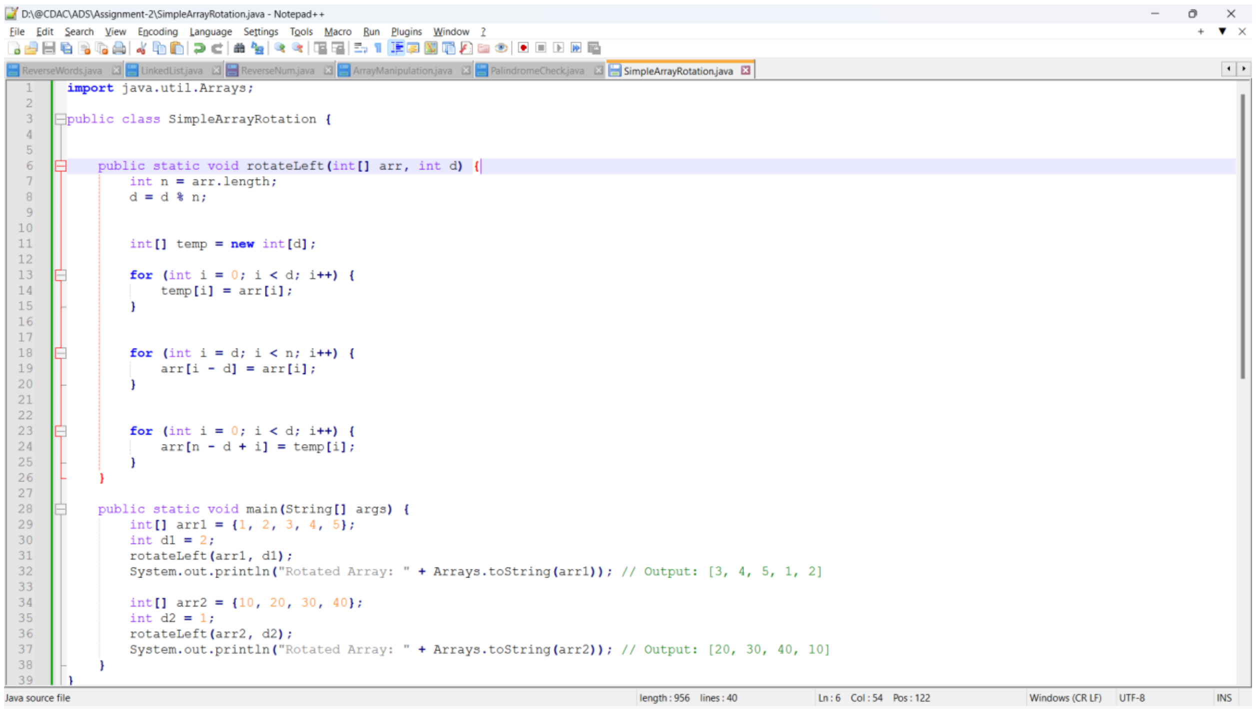
```
D:\@CDAC\ADS\Assignment-2>java PalindromeCheck.java
Enter a string: MADAM
true

D:\@CDAC\ADS\Assignment-2>java PalindromeCheck.java
Enter a string: HELLO
false
```

Time Complexity: O(n)
Space Complexity: O(1).

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

Test Cases:

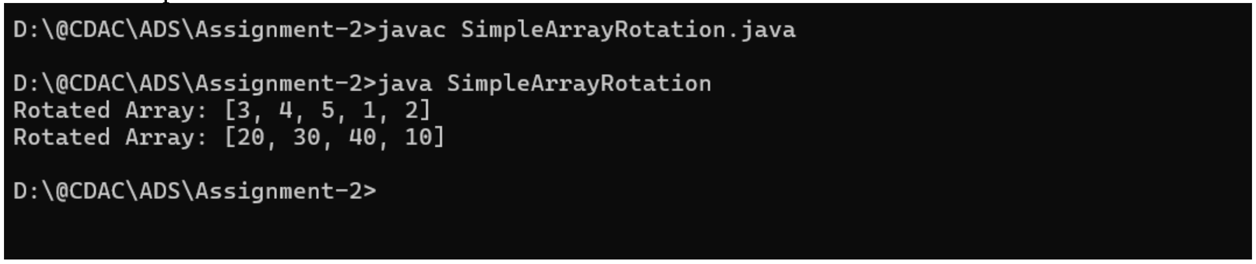
Input: arr = [1, 2, 3, 4, 5], d = 2
Output: [3, 4, 5, 1, 2]
Input: arr = [10, 20, 30, 40], d = 1
Output: [20, 30, 40, 10]



```java
import java.util.Arrays;

public class SimpleArrayRotation {


    public static void rotateLeft(int[] arr, int d) {
        int n = arr.length;
        d = d % n;


        int[] temp = new int[d];

        for (int i = 0; i < d; i++) {
            temp[i] = arr[i];
        }


        for (int i = d; i < n; i++) {
            arr[i - d] = arr[i];
        }


        for (int i = 0; i < d; i++) {
            arr[n - d + i] = temp[i];
        }
    }

    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        int d1 = 2;
        rotateLeft(arr1, d1);
        System.out.println("Rotated Array: " + Arrays.toString(arr1)); // Output: [3, 4, 5, 1, 2]

        int[] arr2 = {10, 20, 30, 40};
        int d2 = 1;
        rotateLeft(arr2, d2);
        System.out.println("Rotated Array: " + Arrays.toString(arr2)); // Output: [20, 30, 40, 10]
    }
}
```

```
+--------------------------------+
|             Start              |
+--------------------------------+
               |
               v
+--------------------------------+
| Input array and d (positions)|
+--------------------------------+
               |
               v
+--------------------------------+
| Calculate d = d % n (handle    |
| cases where d >= n)            |
+--------------------------------+
               |
               v
+--------------------------------+
| Create temporary array temp[]|
| of size d                      |
+--------------------------------+
               |
               v
+--------------------------------+
| Copy first d elements to temp|
| (i.e. temp[i] = arr[i])        |
+--------------------------------+
               |
               v
+--------------------------------+
| Shift remaining elements of    |
| arr[] left by d positions      |
+--------------------------------+
               |
               v
+--------------------------------+
| Copy temp[] elements to the    |
| end of arr[]                   |
| (i.e. arr[n-d+i] = temp[i])    |
+--------------------------------+
               |
               v
+--------------------------------+
| Output the rotated array       |
+--------------------------------+
               |
               v
+--------------------------------+
|              End               |
+--------------------------------+
```

 Start: The program begins by taking the input array and the number of positions d to rotate the array.
 Handle Edge Case: Calculate d % n to ensure the number of positions is within the bounds of the array length.
 Create Temporary Array: A temporary array temp is created to store the first d elements of the original array.
 Copy Elements to Temp: Copy the first d elements of the input array into the temporary array.
 Shift Remaining Elements: Shift the remaining elements of the input array to the left by d positions.
 Copy Temp to End: Copy the elements of temp back to the end of the original array.
 Output the Result: The rotated array is displayed.
 End: The process finishes.



```
D:\@CDAC\ADS\Assignment-2>javac SimpleArrayRotation.java

D:\@CDAC\ADS\Assignment-2>java SimpleArrayRotation
Rotated Array: [3, 4, 5, 1, 2]
Rotated Array: [20, 30, 40, 10]

D:\@CDAC\ADS\Assignment-2>
```

Time complexity is **O(n)**.
Space complexity is **O(d)**