Assignment 3

Hitarth Patel  
150096724046  
Jensen Huang

1. Explain all the loops in Java (Explanation, syntax, example program).

→

Loops in Java

In Java, loops are fundamental constructs that allow the execution of a block of code multiple times. There are three primary types of loops: for, while, and do-while. Each loop serves different use cases based on the requirements of the iteration.

## 1. For Loop

The for loop is typically used when the number of iterations is known beforehand. It consists of three main components: initialization, condition, and increment/decrement.

## Syntax

for (initialization; termination condition; increment/decrement) {

*// Set of statements to be executed repeatedly*

}

## Example

class ForLoopDemo {

public static void **main**(String[] *args*) {

for (int i = 1; i <= 10; i++) {

if (i % 2 == 0) {

System.out.println(i);

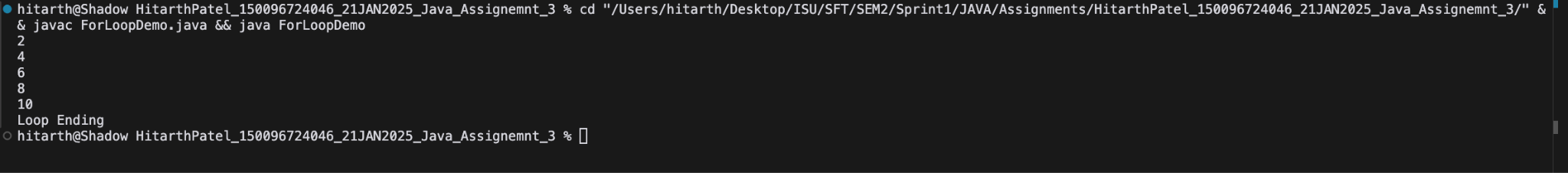
}

}

System.out.println("Loop Ending");

}

}



## 2. While Loop

The while loop is used when the number of iterations is not known in advance and depends on a condition that may change during execution. The loop continues as long as the specified condition evaluates to true.

## Syntax

while (condition) {

*// Set of statements to be executed repeatedly*

}

## Example

class WhileLoopDemo {

public static void **main**(String[] *args*) {

int i = 1;

while (i <= 5) {

System.out.println(i);

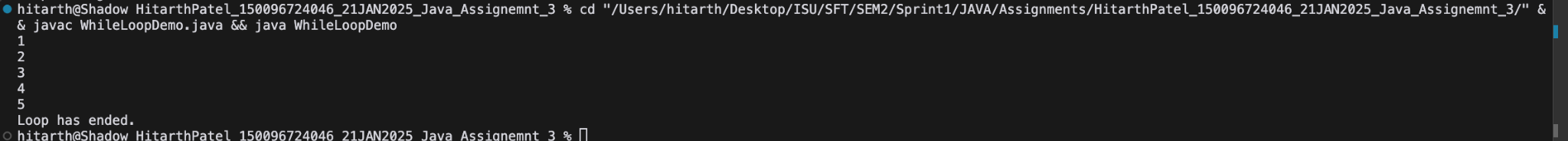
i++;

}

System.out.println("Loop has ended.");

}

}



## 3. Do-While Loop

The do-while loop is similar to the while loop but guarantees that the block of code will execute at least once before checking the condition.

## Syntax

do {

*// Set of statements to be executed repeatedly*

} while (condition);

## Example

class DoWhileLoopDemo {

public static void **main**(String[] *args*) {

int i = 1;

do {

System.out.println(i);

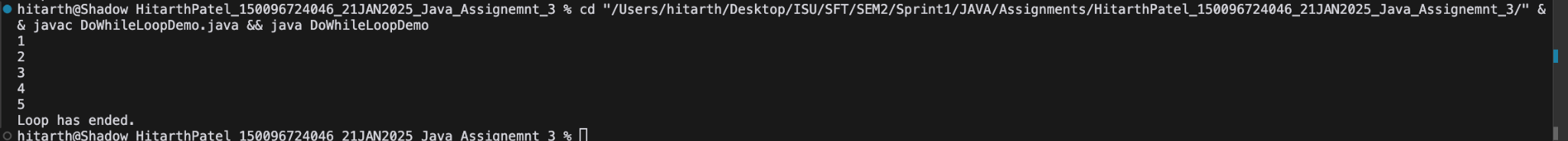
i++;

} while (i <= 5);

System.out.println("Loop has ended.");

}

}



2. Explain arrays in Java (1D array: syntax, initialization, example, 2D array:syntax, initialization, example, Jagged array: syntax, initialization, example).

→

Arrays in Java

Arrays in Java are data structures that allow you to store multiple values of the same type in a single variable. They provide a way to group related data together, making it easier to manage and manipulate. Java supports one-dimensional arrays, multi-dimensional arrays (like two-dimensional arrays), and jagged arrays.

## 1. One-Dimensional Array

A one-dimensional array is a linear data structure that holds a fixed number of values of the same type.

## Syntax

dataType arrayName[] = {value1, value2, ..., valueN};

## Initialization

You can initialize a one-dimensional array either by specifying its size or by directly providing the values.

## Example

class OneDimensionalArrayExample {

public static void **main**(String[] *args*) {

int[] numbers = new int[5];

int[] predefinedNumbers = {10, 20, 30, 40, 50};

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

numbers[i] = (i + 1) \* 10;

}

System.out.println("Numbers:");

for (int num : numbers) {

System.out.print(num + " ");

}

System.out.println("\nPredefined Numbers:");

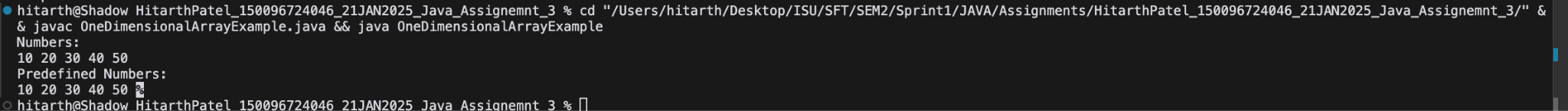
for (int num : predefinedNumbers) {

System.out.print(num + " ");

}

}

}



## 2. Two-Dimensional Array

A two-dimensional array is essentially an array of arrays, allowing you to create a matrix-like structure.

## Syntax

dataType arrayName[][] = {{value1, value2}, {value3, value4}, ...};

## Initialization

You can initialize a two-dimensional array by specifying the number of rows and columns or by directly providing the values.

## Example

class TwoDimensionalArrayExample {

public static void **main**(String[] *args*) {

int[][] matrix = new int[2][3];

int[][] predefinedMatrix = {{1, 2, 3}, {4, 5, 6}};

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

for (int j = 0; j < matrix[i].length; j++) {

matrix[i][j] = (i + j + 1);

}

}

System.out.println("Matrix:");

for (int[] row : matrix) {

for (int val : row) {

System.out.print(val + " ");

}

System.out.println();

}

System.out.println("Predefined Matrix:");

for (int[] row : predefinedMatrix) {

for (int val : row) {

System.out.print(val + " ");

}

System.out.println();

}

}

}

## 

## 3. Jagged Array

A jagged array is an array of arrays where each sub-array can have different lengths. This allows for more flexibility in storing data.

## Syntax

dataType[][] jaggedArrayName = new dataType[numberOfRows][];

## Initialization

You need to initialize each sub-array separately.

## Example

class JaggedArrayExample {

public static void **main**(String[] *args*) {

int[][] jaggedArray = new int[3][];

jaggedArray[0] = new int[2];

jaggedArray[1] = new int[3];

jaggedArray[2] = new int[1];

jaggedArray[0][0] = 1;

jaggedArray[0][1] = 2;

jaggedArray[1][0] = 3;

jaggedArray[1][1] = 4;

jaggedArray[1][2] = 5;

jaggedArray[2][0] = 6;

System.out.println("Jagged Array:");

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

for (int j = 0; j < jaggedArray[i].length; j++) {

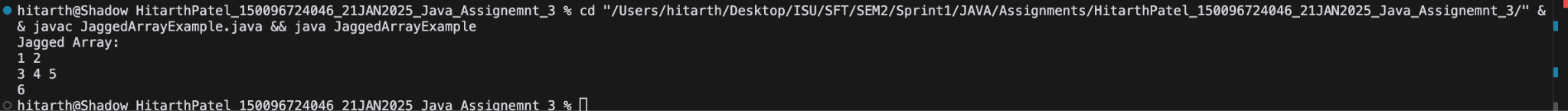
System.out.print(jaggedArray[i][j] + " ");

}

System.out.println();

}

}

}

3. WAP to compute sum and average of 5 double elements of array.

→

import java.util.Scanner;

class SumAndAverage {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

double[] array = new double[5];

double sum = 0;

for (int i = 0; i < 5; i++) {

System.out.print("Enter element " + (i+1) + ": ");

array[i] = scanner.nextDouble();

sum += array[i];

}

double average = sum / 5;

System.out.println("Sum: " + sum);

System.out.println("Average: " + average);

scanner.close();

}

}



4. WAP to perform linear search in java.

→

import java.util.Scanner;

class LinearSearch {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] array = new int[size];

System.out.println("Enter the elements of the array:");

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

array[i] = scanner.nextInt();

}

System.out.print("Enter the number to search: ");

int number = scanner.nextInt();

boolean found = false;

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

if (array[i] == number) {

System.out.println("Number found at index " + i);

found = true;

break;

}

}

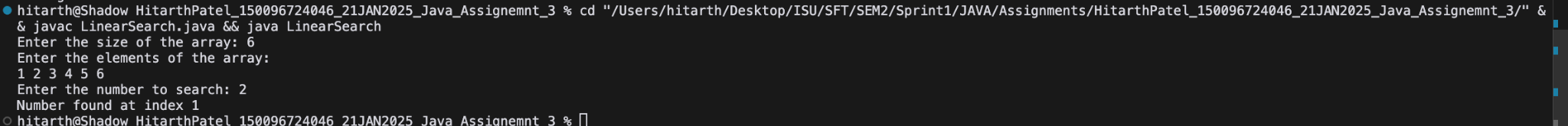
if (!found) {

System.out.println("Number not found in the array.");

}

scanner.close();

}

}

5. WAP to perform binary search in java.

→

import java.util.Scanner;

public class BinarySearch {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] array = new int[size];

System.out.println("Enter the elements of the array in sorted order:");

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

array[i] = scanner.nextInt();

}

System.out.print("Enter the number to search: ");

int number = scanner.nextInt();

int result = binarySearch(array, number);

if (result == -1) {

System.out.println("Element not found in the array");

} else {

System.out.println("Element found at index " + result);

}

scanner.close();

}

public static int **binarySearch**(int[] *array*, int *target*) {

int left = 0;

int right = *array*.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (*array*[mid] == *target*) {

return mid;

} else if (*array*[mid] < *target*) {

left = mid + 1;

} else {

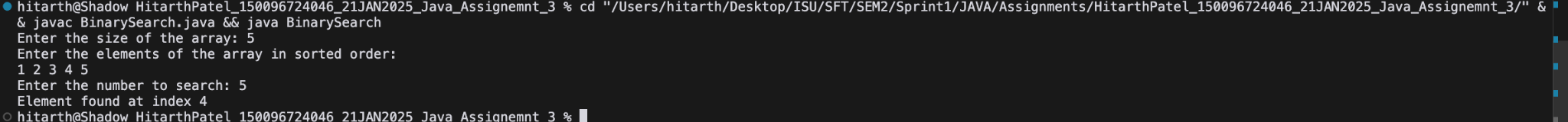
right = mid - 1;

}

}

return -1;

}



6. Write a Java program to insert an element (specific position) into an array.

→

import java.util.Scanner;

public class SpecificPositionAdd {

public static void **insertElement**(int[] *array*, int *position*, int *element*) {

int[] newArray = new int[*array*.length + 1];

System.arraycopy(*array*, 0, newArray, 0, *position*);

newArray[*position*] = *element*;

System.arraycopy(*array*, *position*, newArray, *position* + 1, *array*.length - *position*);

*array* = newArray;

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

System.out.print(*array*[i] + " ");

}

}

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the size of the array:");

int size = scanner.nextInt();

int[] array = new int[size];

System.out.println("Enter the elements of the array:");

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

array[i] = scanner.nextInt();

}

System.out.println("Enter the position where you want to insert the element:");

int position = scanner.nextInt();

System.out.println("Enter the element you want to insert:");

int element = scanner.nextInt();

System.out.println("Original array:");

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

System.out.print(array[i] + " ");

}

System.out.println("\nArray after inserting element at position " + position + ":");

insertElement(array, position, element);

scanner.close();

}

}



7. Write a Java program to remove a specific element from an array.

→

import java.util.Scanner;

public class SpecificPositionRemove {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] array = new int[size];

System.out.println("Enter the elements of the array: ");

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

array[i] = scanner.nextInt();

}

System.out.print("Enter the index of the element to remove (0-" + (size - 1) + "): ");

int removeIndex = scanner.nextInt();

array = removeElement(array, removeIndex);

System.out.println("Array after removing element at index " + removeIndex + ":");

for (int i : array) {

System.out.print(i + " ");

}

scanner.close();

}

public static int[] **removeElement**(int[] *array*, int *index*) {

if (*index* < 0 || *index* >= *array*.length) {

System.out.println("Invalid index. Returning original array.");

return *array*;

}

int[] newArray = new int[*array*.length - 1];

System.arraycopy(*array*, 0, newArray, 0, *index*);

System.arraycopy(*array*, *index* + 1, newArray, *index*, *array*.length - *index* - 1);

return newArray;

}

}



8. Write a Java program to find the maximum and minimum value of an array.

→

import java.util.Scanner;

public class MinAndMaxArray {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of elements in the array: ");

int n = scanner.nextInt();

int[] array = new int[n];

System.out.println("Enter the elements of the array: ");

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

array[i] = scanner.nextInt();

}

int max = array[0];

int min = array[0];

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

if(array[i] > max){

max = array[i];

}else if(array[i] < min){

min = array[i];

}

}

System.out.println("Maximum Value is : " + max);

System.out.println("Minimum Value is : " + min);

scanner.close();

}

}



9. Write a Java program to reverse an array of integer values.

→

import java.util.Scanner;

public class ReverseArray {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] array = new int[size];

System.out.println("Enter the elements of the array: ");

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

array[i] = scanner.nextInt();

}

System.out.println("Original array: ");

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

System.out.print(array[i] + " ");

}

reverseArray(array);

System.out.println("\nReversed array: ");

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

System.out.print(array[i] + " ");

}

scanner.close();

}

public static void **reverseArray**(int[] *array*) {

int left = 0;

int right = *array*.length - 1;

while (left < right) {

int temp = *array*[left];

*array*[left] = *array*[right];

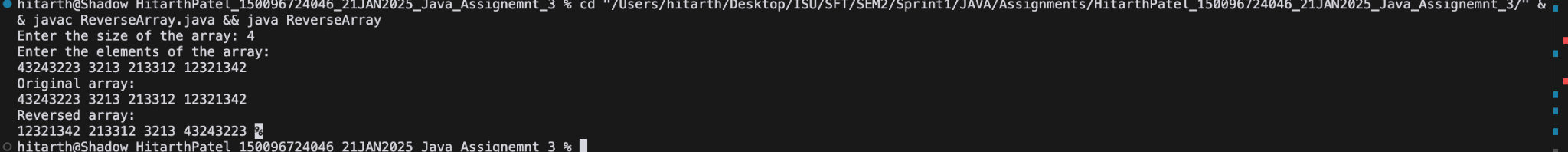
*array*[right] = temp;

left++;

right--;

}

}

}

10.Write a Java program to find duplicate values in an array of integer values.

→

import java.util.Scanner;

public class DuplicateValueArray {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] array = new int[size];

System.out.println("Enter the elements of the array: ");

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

array[i] = scanner.nextInt();

}

System.out.println("Duplicate values in the array are: ");

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

for (int j = i + 1; j < size; j++) {

if (array[i] == array[j]) {

System.out.print(array[i] + " ");

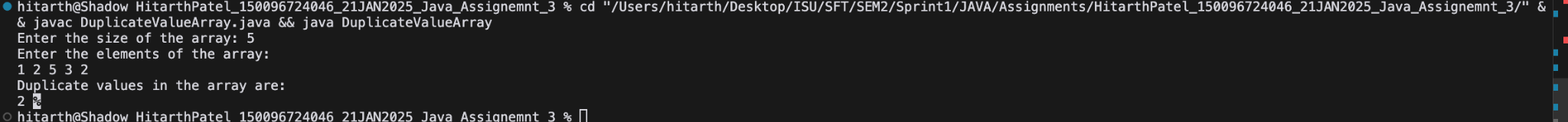
}

}

}

scanner.close();

}

}

11.Write a Java program to find duplicate values in an array of string values.

→

import java.util.Scanner;

public class DuplicateArrayFind {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

String[] array = new String[size];

System.out.println("Enter the elements of the array: ");

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

array[i] = scanner.next();

}

System.out.println("Duplicate values in the array are: ");

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

for (int j = i + 1; j < size; j++) {

if (array[i].equals(array[j])) {

System.out.print(array[i] + " ");

}

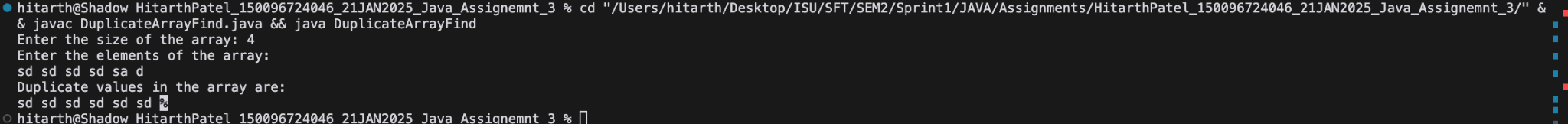
}

}

scanner.close();

}

}



12.WAP to perform matrix addition in java.

→

public class MatrixAddition {

public static void **main**(String[] *args*) {

int[][] matrix1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int[][] matrix2 = {{10, 11, 12}, {13, 14, 15}, {16, 17, 18}};

int[][] result = addMatrices(matrix1, matrix2);

System.out.println("Matrix 1:");

printMatrix(matrix1);

System.out.println("Matrix 2:");

printMatrix(matrix2);

System.out.println("Result:");

printMatrix(result);

}

public static int[][] **addMatrices**(int[][] *matrix1*, int[][] *matrix2*) {

int rows = *matrix1*.length;

int cols = *matrix1*[0].length;

int[][] result = new int[rows][cols];

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

for (int j = 0; j < cols; j++) {

result[i][j] = *matrix1*[i][j] + *matrix2*[i][j];

}

}

return result;

}

public static void **printMatrix**(int[][] *matrix*) {

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

for (int j = 0; j < *matrix*[0].length; j++) {

System.out.print(*matrix*[i][j] + " ");

}

System.out.println();

}

}

}



13.WAP to perform matrix multiplication in java.

→

public class MatrixMultiplication {

public static void **main**(String[] *args*) {

int[][] matrix1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int[][] matrix2 = {{10, 11, 12}, {13, 14, 15}, {16, 17, 18}};

int[][] result = multiplyMatrices(matrix1, matrix2);

System.out.println("Matrix 1:");

printMatrix(matrix1);

System.out.println("Matrix 2:");

printMatrix(matrix2);

System.out.println("Result:");

printMatrix(result);

}

public static int[][] **multiplyMatrices**(int[][] *matrix1*, int[][] *matrix2*) {

int rows = *matrix1*.length;

int cols = *matrix2*[0].length;

int[][] result = new int[rows][cols];

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

for (int j = 0; j < cols; j++) {

for (int k = 0; k < *matrix1*[0].length; k++) {

result[i][j] += *matrix1*[i][k] \* *matrix2*[k][j];

}

}

}

return result;

}

public static void **printMatrix**(int[][] *matrix*) {

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

for (int j = 0; j < *matrix*[0].length; j++) {

System.out.print(*matrix*[i][j] + " ");

}

System.out.println();

}

}

}



14.Explain String class and specify different ways to create String objects.

→

The String class in Java is a fundamental part of the language, representing a sequence of characters. Strings are immutable, meaning once a String object is created, its value cannot be changed. This immutability allows for efficient memory management and thread safety.

## Characteristics of the String Class

* Immutability: Once a String object is created, it cannot be modified. Any operation that seems to modify a string actually creates a new string.
* String Pool: Java maintains a pool of strings to optimize memory usage. When you create a string literal, Java checks this pool first to see if an identical string already exists.

## Ways to Create String Objects

There are several ways to create String objects in Java:

1. Using String Literals:
   * You can create a string using double quotes. If the same string literal is created again, it will reference the existing object in the string pool.
   * String str1 = "Hello, World!";
2. Using the new Keyword:
   * This method explicitly creates a new String object in heap memory, regardless of whether an identical string exists in the pool.
   * String str2 = new String("Hello, World!")
3. Using Character Arrays:
   * You can also create a string from an array of characters.
   * char[] chars = {'H', 'e', 'l', 'l', 'o'};
   * String str3 = new String(chars);
4. Using Byte Arrays:
   * Strings can be created from byte arrays using the default character encoding or specified encoding.
   * byte[] bytes = {72, 101, 108, 108, 111}; *// ASCII values for 'Hello'*
   * String str4 = new String(bytes);
   * String str5 = new String(bytes, "UTF-8");*// Using specific charset*
5. Using StringBuilder
   * For mutable strings (strings that can change), you can use StringBuilder .These classes allow for dynamic modification of strings.
   * StringBuilder sb = new StringBuilder();
   * sb.append("Hello");
   * sb.append(", World!");
   * String str6 = sb.toString(); *// Converts to String*

15. WAP to demonstrate the functionality of all the methods of String class.

1. int length()

2. char charAt(int index)

3. int indexOf(int ch)

4. int indexOf(int ch, int fromIndex)

5. int indexOf(String substring)

6. int indexOf(String substring, int fromIndex)

7. lastIndexOf( )

8. String substring(int beginIndex)

9. String substring(int beginIndex, int endIndex)

10.boolean contains(CharSequence s)

11.String concat(String s)

12.boolean equals(Object o)

13.boolean equalsIgnoreCase(String s)

14.boolean isEmpty()

15.boolean equals(Object o)

16.boolean equalsIgnoreCase(String s)

17.String toLowerCase()

18.String toUpperCase()

19.int compareTo( String anotherString)

20.int compareToIgnoreCase( String anotherString)

21.String trim()

22. String replace (char oldChar, char newChar)

23.char[] toCharArray()

24. boolean startsWith(String s)

25.boolean endswith(String s)

26.static String join(CharSequence delim, CharSequence . . . strs)

→

import java.util.Arrays;

public class Stringclass {

public static void **main**(String[] *args*) {

String str = "Hello, World!";

System.out.println("Length of the string: " + str.length());

System.out.println("Character at index 4: " + str.charAt(4));

System.out.println("Index of 'o': " + str.indexOf('o'));

System.out.println("Index of 'o' after index 5: " + str.indexOf('o', 5));

System.out.println("Index of 'World': " + str.indexOf("World"));

System.out.println("Index of 'World' after index 5: " + str.indexOf("World", 5));

System.out.println("Last index of 'o': " + str.lastIndexOf('o'));

System.out.println("Substring from index 7: " + str.substring(7));

System.out.println("Substring from index 7 to 12: " + str.substring(7, 12));

System.out.println("Contains 'World': " + str.contains("World"));

System.out.println("Concatenation with '!' : " + str.concat("!"));

System.out.println("Equals 'Hello, World!': " + str.equals("Hello, World!"));

System.out.println("Equals ignore case 'hello, world!': " + str.equalsIgnoreCase("hello, world!"));

System.out.println("Is empty: " + str.isEmpty());

System.out.println("To lower case: " + str.toLowerCase());

System.out.println("To upper case: " + str.toUpperCase());

System.out.println("Compare to 'Hello, World!': " + str.compareTo("Hello, World!"));

System.out.println("Compare to ignore case 'hello, world!': " + str.compareToIgnoreCase("hello, world!"));

System.out.println("Trim: " + str.trim());

System.out.println("Replace 'o' with '0': " + str.replace('o', '0'));

System.out.println("To char array: " + Arrays.toString(str.toCharArray()));

System.out.println("Starts with 'Hello': " + str.startsWith("Hello"));

System.out.println("Ends with 'World!': " + str.endsWith("World!"));

System.out.println("Join with ', ': " + String.join(", ", "Hello", "World!"));

}

}

Q16. Write a program to compare two strings lexicographically, ignoring case differences.

→

public class TwoString {

public static void **main**(String[] *args*) {

String str1 = "Hello";

String str2 = "hello";

int result = str1.compareToIgnoreCase(str2);

if (result < 0) {

System.out.println(str1 + " is lexicographically smaller than " + str2);

} else if (result > 0) {

System.out.println(str1 + " is lexicographically larger than " + str2);

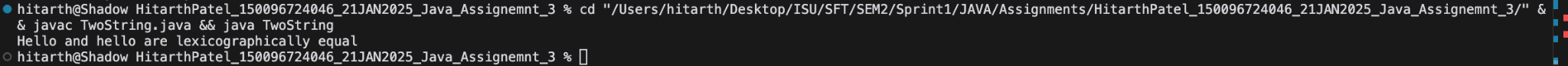
} else {

System.out.println(str1 + " and " + str2 + " are lexicographically equal");

}

}

}



Q17. Write a program to check whether two String objects contain the same data.

→

public class TwoStringCheck {

public static void **main**(String[] *args*) {

String str1 = "Hello";

String str2 = "Hello";

if (str1.equals(str2)) {

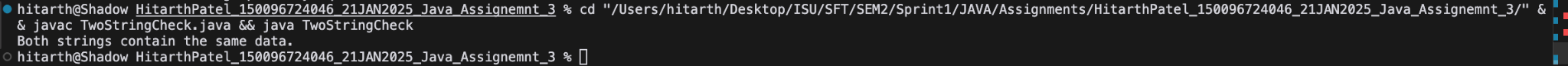
System.out.println("Both strings contain the same data.");

} else {

System.out.println("Both strings do not contain the same data.");

}

}

}

Q18. Write a program to replace each substring of a given string.

→

public class ReplaceSubstring {

public static void **main**(String[] *args*) {

String str = "Hello, World!";

String oldSubStr = "World";

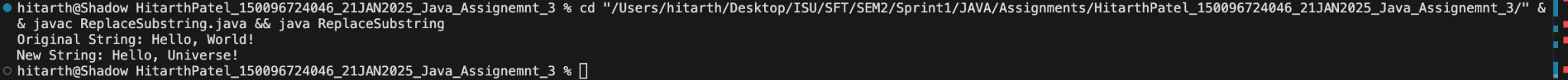
String newSubStr = "Universe";

String newStr = str.replace(oldSubStr, newSubStr);

System.out.println("Original String: " + str);

System.out.println("New String: " + newStr);

}

}

Q19. Write a Java program to check whether a given string starts with the contents of another string.

→

public class CheckString {

public static void **main**(String[] *args*) {

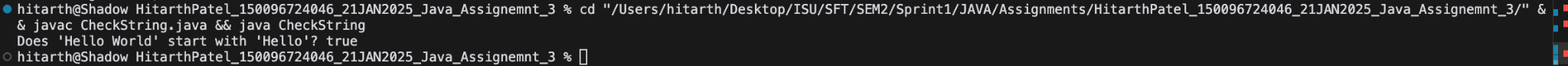
String mainString = "Hello World";

String subString = "Hello";

System.out.println("Does '" + mainString + "' start with '" + subString + "'? " + mainString.startsWith(subString));

}

}



Q20. Write a Java program to create a new string repeating every character twice of a given string.

→

public class NewString {

public static void **main**(String[] *args*) {

String str = "hello";

String newStr = "";

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

newStr += str.charAt(i);

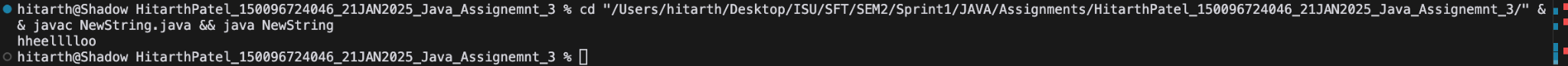
newStr += str.charAt(i);

}

System.out.println(newStr);

}

}



Q21. Write a Java program to return the sum of the digits present in the given string. If there is no digits the sum return is 0.

→

import java.util.Scanner;

public class SumDigits {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a string:");

String input = scanner.nextLine();

int sum = 0;

for (char c : input.toCharArray()) {

if (Character.isDigit(c)) {

sum += Integer.parseInt(String.valueOf(c));

}

}

System.out.println("Sum of digits in the string: " + sum);

scanner.close();

}

}



Q22. Write a Java program to Count words in Given String.

→

public class CountWord {

public static void **main**(String[] *args*) {

String str = "This is a sample string";

int count = 1;

for (int i = 0; i < str.length() - 1; i++) {

if ((str.charAt(i) == ' ') && (str.charAt(i + 1) != ' ')) {

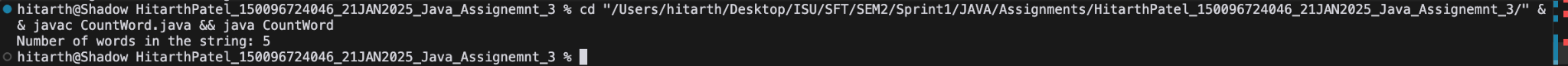
count++;

}

}

System.out.println("Number of words in the string: " + count);

}

}

Q23. Write a Java program to Swap Two Strings.

→

public class SwapString {

public static void **main**(String[] *args*) {

String str1 = "Hello";

String str2 = "World";

System.out.println("Before swapping:");

System.out.println("String 1: " + str1);

System.out.println("String 2: " + str2);

String temp = str1;

str1 = str2;

str2 = temp;

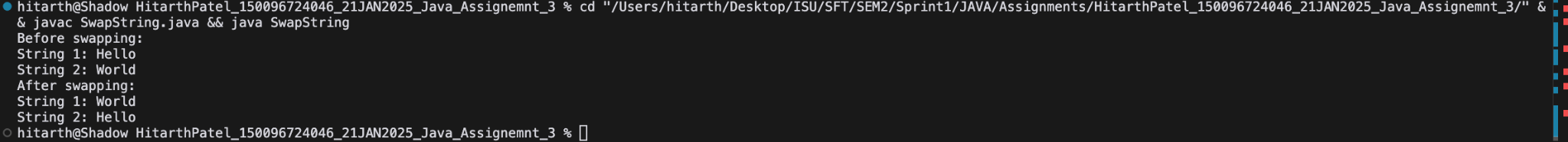
System.out.println("After swapping:");

System.out.println("String 1: " + str1);

System.out.println("String 2: " + str2);

}

}



Q24. Write a Java program to Swap Two Strings without Third String Variable.

→

public class SwapTwoString {

public static void **main**(String[] *args*) {

String str1 = "Hello";

String str2 = "World";

System.out.println("Before swapping:");

System.out.println("String 1: " + str1);

System.out.println("String 2: " + str2);

str1 = str1 + str2;

str2 = str1.substring(0, str1.length() - str2.length());

str1 = str1.substring(str2.length());

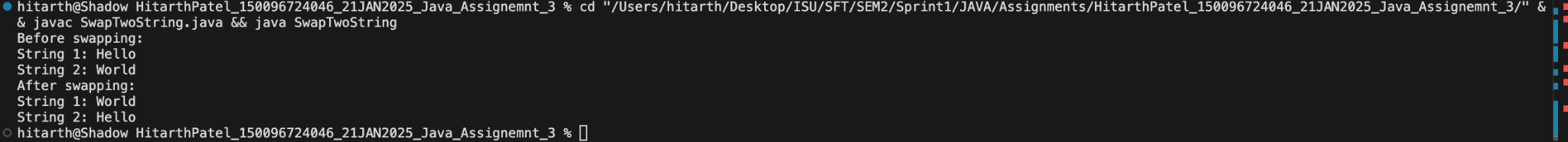
System.out.println("After swapping:");

System.out.println("String 1: " + str1);

System.out.println("String 2: " + str2);

}

}



Q25. Write a Java program to Reverse Each Word of a String.

→

class Reverseword {

public static void **main**(String[] *args*) {

String str = "Hello World";

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

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

String word = words[i];

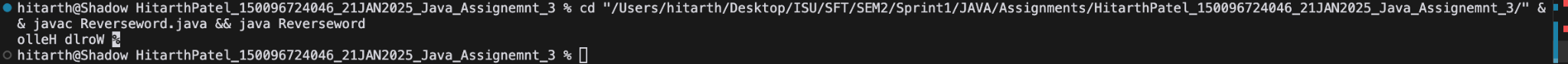
String reversedWord = new StringBuilder(word).reverse().toString();

System.out.print(reversedWord + " ");

}

}

}



Q26. Java String program to check whether a string is a Palindrome.

→

public class palindrome {

public static void **main**(String[] *args*) {

String str = "radar";

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

if (str.equals(reversed)) {

System.out.println(str + " is a palindrome");

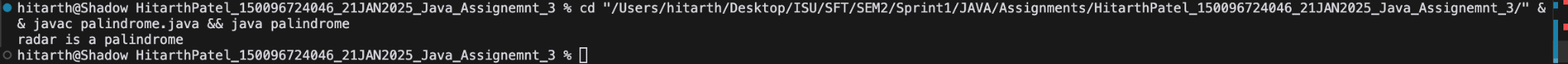
} else {

System.out.println(str + " is not a palindrome");

}

}

}



Q27. Program to Check Two Strings Are Anagram Of Each Other in Java.

→

public class Anagram {

public static boolean **areAnagrams**(String *str1*, String *str2*) {

if (*str1*.length() != *str2*.length()) {

return false;

}

char[] charArray1 = *str1*.toLowerCase().toCharArray();

char[] charArray2 = *str2*.toLowerCase().toCharArray();

java.util.Arrays.sort(charArray1);

java.util.Arrays.sort(charArray2);

return java.util.Arrays.equals(charArray1, charArray2);

}

public static void **main**(String[] *args*) {

System.out.println(areAnagrams("listen", "silent"));

System.out.println(areAnagrams("hello", "world"));

}

}



Q28. Write a Java program to Count Number of Uppercase and Lowercase letters.

→

import java.util.Scanner;

public class UpperLowerCaseCount {

public static void **main**(String[] *args*) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a string: ");

String str = scanner.nextLine();

int upper = 0, lower = 0;

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

char ch = str.charAt(i);

if (Character.isUpperCase(ch)) {

upper++;

} else if (Character.isLowerCase(ch)) {

lower++;

}

}

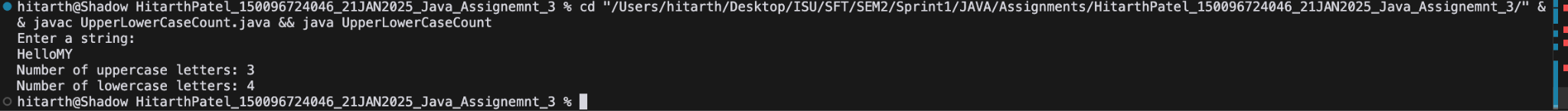
System.out.println("Number of uppercase letters: " + upper);

System.out.println("Number of lowercase letters: " + lower);

scanner.close();

}

}



Q29. Write a Java program to Given string Convert Uppercase to Lowercase.

→

public class UpperToLower {

public static void **main**(String[] *args*) {

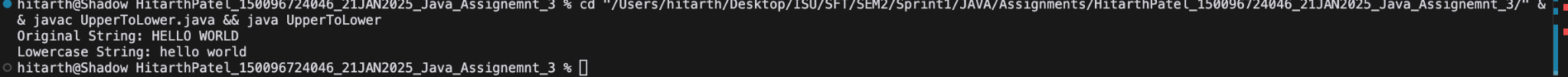
String str = "HELLO WORLD";

System.out.println("Original String: " + str);

System.out.println("Lowercase String: " + str.toLowerCase());

}

}



Q30. Write a Java program to check if the letter 'e' is present in the word 'Hello World'.

→

public class Replace {

public static void main(String[] *args*) {

String word = "Hello World";

if (word.contains("e")) {

System.out.println("The letter 'e' is present in the word.");

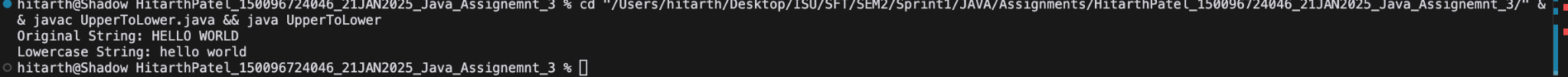
} else {

System.out.println("The letter 'e' is not present in the word.");

}

}

}



Q31. Differentiate between String and StringBuilder.

→

| **Feature** | **String** | **StringBuilder** |
| --- | --- | --- |
| Mutability | Immutable (cannot be changed once created) | Mutable (can be modified) |
| Performance | Slower for modifications due to immutability | Faster for frequent modifications |
| Memory Storage | Stored in the String Pool | Stored in the heap |
| Thread Safety | Thread-safe due to immutability | Not thread-safe (not synchronized) |
| Concatenation | Each concatenation creates a new object | Uses append() method for efficient concatenation |
| Usage Scenarios | Best for fixed text or when thread safety is required | Ideal for dynamic text changes in single-threaded environments |

Q32. Explain StringBuilder class. What is the initial capacity of StringBuilder object. Explain

constructors. Explain the following methods of String Builder class along with examples:

i. compareTo()

ii. delete()

iii. deleteCharAt()

iv. ensureCapacity()

v. getChars()

vi. insert()

vii. indexOf()

viii. lastIndexOf()

ix. setCharAt()

x. setLength()

xi. append()

xii. charAt()

xiii. reverse()

xiv. length()

xv. capacity()

xvi. substring()

xvii. replace()

→

## StringBuilder Class in Java

The StringBuilder class in Java is part of the java.lang package and provides a mutable sequence of characters. Unlike the String class, which is immutable, StringBuilder allows for modifications without creating new objects, making it more efficient for string manipulation.

## Initial Capacity

The default initial capacity of a StringBuilder object is 16 characters. If the number of characters exceeds this capacity, the capacity is increased dynamically by (old capacity \* 2) + 2.

## Constructors of StringBuilder

1. StringBuilder()
   * Creates an empty StringBuilder with a default initial capacity of 16.
   * StringBuilder sb1 = new StringBuilder();
2. StringBuilder(int capacity)
   * Creates a StringBuilder with the specified initial capacity.
   * StringBuilder sb2 = new StringBuilder(50);
3. StringBuilder(String str)
   * Creates a StringBuilder initialized with the contents of the specified string.
   * StringBuilder sb3 = new StringBuilder("Hello");
4. StringBuilder(CharSequence seq)
   * Creates a StringBuilder initialized with the contents of the given CharSequence.
   * CharSequence cs = "World";
   * StringBuilder sb4 = new StringBuilder(cs);

## Methods of StringBuilder

Here are explanations and examples for various methods of the StringBuilder class:

1. compareTo(String anotherString)
   * Compares two strings lexicographically.
   * StringBuilder sb1 = new StringBuilder("abc");
   * System.out.println(sb1.compareTo("abd"));*// Output: -1*
2. delete(int start, int end)
   * Removes characters from the specified start index to end index (exclusive).
   * StringBuilder sb2 = new StringBuilder("Hello World");
   * sb2.delete(5, 11); *// Removes " World"*
   * System.out.println(sb2); *// Output: "Hello"*
3. deleteCharAt(int index)
   * Removes the character at the specified index.
   * StringBuilder sb3 = new StringBuilder("Hello");
   * sb3.deleteCharAt(0); *// Removes 'H'*
   * System.out.println(sb3); *// Output: "ello"*
4. ensureCapacity(int minimumCapacity)
   * Ensures that the capacity is at least equal to the specified minimum.
   * StringBuilder sb4 = new StringBuilder();
   * sb4.ensureCapacity(50);
   * System.out.println(sb4.capacity()); *// Output: 50*
5. getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
   * Copies characters from this string builder into the destination character array.
   * StringBuilder sb5 = new StringBuilder("Hello");
   * char[] chars = new char[5];
   * sb5.getChars(0, 5, chars, 0);
   * System.out.println(chars); *// Output: Hello*
6. insert(int offset, String str)
   * Inserts the specified string at the given position.
   * StringBuilder sb6 = new StringBuilder("Hello");
   * sb6.insert(5, " World");
   * System.out.println(sb6); *// Output: "Hello World"*
7. indexOf(String str)
   * Returns the index within this string builder of the first occurrence of the specified substring.
   * StringBuilder sb7 = new StringBuilder("Hello World");
   * System.out.println(sb7.indexOf("World")); *// Output: 6*
8. lastIndexOf(String str)
   * Returns the index within this string builder of the last occurrence of the specified substring.
   * System.out.println(sb7.lastIndexOf("o")); *// Output: 7*
9. setCharAt(int index, char ch)
   * Replaces the character at the specified index with a new character.
   * sb6.setCharAt(0, 'h'); *// Changes 'H' to 'h'*
   * System.out.println(sb6); *// Output: "hello World"*
10. setLength(int newLength)
    * Sets the length of this string builder to the specified value.
    * sb6.setLength(5); *// Trims to "hello"*
    * System.out.println(sb6); *// Output: "hello"*
11. append(String str)
    * Appends the specified string to this string builder.
    * sb6.append(" Everyone");
    * System.out.println(sb6); *// Output: "hello Everyone"*
12. charAt(int index)
    * Returns the character at the specified index.
    * System.out.println(sb6.charAt(1)); *// Output: 'e'*
13. reverse()
    * Reverses the sequence of characters in this string builder.
    * sb6.reverse();
    * System.out.println(sb6); *// Output: "enoyrevE olleh"*
14. length()
    * Returns the number of characters in this string builder.
    * System.out.println(sb6.length()); *// Output: 12 (length after reversing)*
15. capacity()
    * Returns the current capacity of this string builder.
    * System.out.println(sb6.capacity()); *// Output may vary based on operations performed*
16. substring(int start, int end)
    * Returns a substring from this string builder.
    * System.out.println(sb7.substring(0, 5)); *// Output: "Hello"*
17. replace(int start, int end, String str)
    * Replaces characters in a substring with the specified string.
    * sb7.replace(0, 5, "Hi");
    * System.out.println(sb7); *// Output: "Hi World"*