**Day 1 Analyticals Questions**

**Java Fundamentals and Analytical questions**

|  |  |
| --- | --- |
| **Subject code** | **CSA0918** |
| **Subject Name** | **Proramming in Java for Query Processing** |
| **Year & Sem** | **III YEAR** |
| **Ac Year / Slot** | **2024 / A Slot** |
| **Name of Faculty** | **Dr. K. Anbazhagan** |

1. Write a program to demonstrate primitive datatypes in java. Below are the datatypes Integer data types: byte, short, int, and long. Floating-point data types: float and double. Character data type: char.

Boolean data type: Boolean.

**Program**

import java.util.Scanner;

public class PrimitiveDataTypes {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a byte value: ");

byte b = scanner.nextByte();

System.out.print("Enter a short value: ");

short s = scanner.nextShort();

System.out.print("Enter an int value: ");

int i = scanner.nextInt();

System.out.print("Enter a long value: ");

long l = scanner.nextLong();

System.out.print("Enter a float value: ");

float f = scanner.nextFloat();

System.out.print("Enter a double value: ");

double d = scanner.nextDouble();

System.out.print("Enter a character: ");

char c = scanner.next().charAt(0);

System.out.print("Enter a boolean value (true or false): ");

boolean bool = scanner.nextBoolean();

System.out.println("\nEntered values:");

System.out.println("byte: " + b);

System.out.println("short: " + s);

System.out.println("int: " + i);

System.out.println("long: " + l);

System.out.println("float: " + f);

System.out.println("double: " + d);

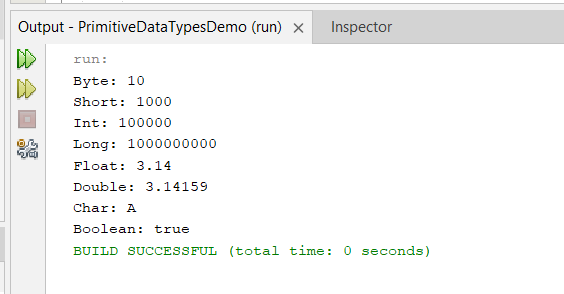
System.out.println("char: " + c);

System.out.println("boolean: " + bool);

}

}

**Output**



2. Example program demonstrating data type conversion in Java for all primitive data types:

a) Implicit conversion (widening) from

smaller data types to larger ones.

b) Explicit conversion (narrowing) from larger

data types to smaller ones.

c) Overflow and underflow scenarios where

the value exceeds the range of the target

data type.

**Program**

import java.util.Scanner;

public class DataTypeConversion {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter an integer value: ");

int intVal = sc.nextInt();

long longVal = intVal;

float floatVal = longVal;

double doubleVal = floatVal;

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

System.out.println(intVal);

System.out.println(longVal);

System.out.println(floatVal);

System.out.println(doubleVal);

double doubleVal2 = 100.12345;

float floatVal2 = (float)doubleVal2;

long longVal2 = (long)floatVal2;

int intVal2 = (int)longVal2;

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

System.out.println(doubleVal2);

System.out.println(floatVal2);

System.out.println(longVal2);

System.out.println(intVal2);

int intMax = Integer.MAX\_VALUE;

int overflow = intMax + 1;

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

System.out.println(intMax);

System.out.println(overflow);

int intMin = Integer.MIN\_VALUE;

int underflow = intMin - 1;

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

System.out.println(intMin);

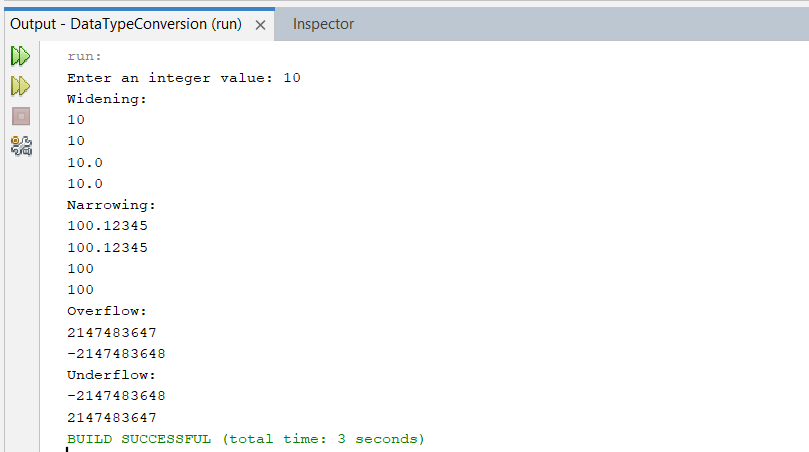
System.out.println(underflow);

sc.close();

}

}

**Output**



3. Java program for String methods and String

constructors

**Program**

import java.util.Scanner;

public class string {

public static void main(String[] args) {

// String constructors

String str1 = new String();

String str2 = new String("Hello, World!");

char[] charArray = {'J', 'a', 'v', 'a'};

String str3 = new String(charArray);

byte[] byteArray = {65, 66, 67, 68};

String str4 = new String(byteArray);

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

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

System.out.println("str3: " + str3);

System.out.println("str4: " + str4);

String originalString = "Java Programming";

int length = originalString.length();

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

char charAtIndex = originalString.charAt(2);

System.out.println("Character at index 2: " + charAtIndex);

String substring = originalString.substring(5, 16);

System.out.println("Substring from index 5 to 15: " + substring);

int indexOf = originalString.indexOf("Programming");

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

String concatString = originalString.concat(" is fun!");

System.out.println("Concatenated string: " + concatString);

String replacedString = originalString.replace("Java", "Python");

System.out.println("String after replacement: " + replacedString);

String upperCaseString = originalString.toUpperCase();

System.out.println("Uppercase string: " + upperCaseString);

String lowerCaseString = originalString.toLowerCase();

System.out.println("Lowercase string: " + lowerCaseString);

String stringWithSpaces = " Trim Example ";

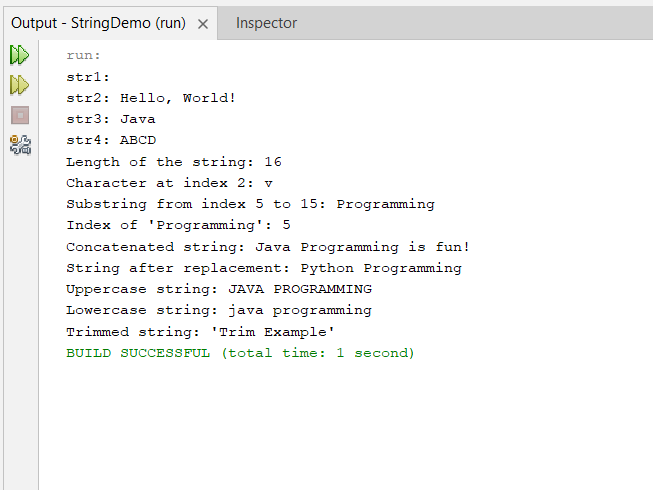
String trimmedString = stringWithSpaces.trim();

System.out.println("Trimmed string: '" + trimmedString + "'");

}

}

**Output**



4.String builder and String buffer program differences with example

Demonstrate simple examples of StringBuilder and StringBuffer appending strings. Then compare the performance of StringBuilder

and StringBuffer by appending a large number of strings in a loop (iterations times) and measuring the time taken.

**Note:**

The key differences between StringBuilder and StringBuffer in Java are:

Thread Safety:

StringBuffer: Synchronized and thread-safe. This means multiple threads can access and modify it concurrently without data corruption. However, this synchronization comes at a performance cost.

StringBuilder: Non-synchronized and not thread-safe. Only one thread can access and modify it at a time. If multiple threads access it without proper synchronization, it can lead to errors.

Performance:

StringBuffer: Due to synchronization overhead, it's generally slower than StringBuilder when thread safety is not required.

StringBuilder: Its lack of synchronization makes it faster than StringBuffer in single-threaded scenarios or when synchronization is managed properly.

Other Differences:

Methods: Both have the same public methods for string manipulation (append, insert, delete, etc.). However, StringBuffer has a few extra methods related to capacity and sub-sequences, which are mostly redundant with methods available in the String class.

Memory: Both allocate memory dynamically as needed but StringBuffer may allocate slightly more memory initially due to synchronization overhead.

When to use which:

Use StringBuffer: When your application is multithreaded and multiple threads might access and modify the string concurrently.

Use StringBuilder: In single-threaded environments or when you manage synchronization yourself. It's often the preferred choice for better performance when thread safety is not a concern.

Additional considerations:

In complex scenarios, profile your application's string operations to measure the impact of thread safety and specific string manipulations on your performance.

Remember that using either StringBuilder or StringBuffer is more efficient than string concatenation using the + operator, which creates new string objects in each iteration.

**Program**

import java.util.Scanner;

public class StringBuilderVsStringBuffer1 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of iterations for string appending: ");

int iterations = scanner.nextInt();

long startTime = System.currentTimeMillis();

StringBuilder sb = new StringBuilder();

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

sb.append("Iteration " + i + "\n");

}

String sbResult = sb.toString();

long endTime = System.currentTimeMillis();

long sbTime = endTime - startTime;

startTime = System.currentTimeMillis();

StringBuffer sbuffer = new StringBuffer();

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

sbuffer.append("Iteration " + i + "\n");

}

String sbufferResult = sbuffer.toString();

endTime = System.currentTimeMillis();

long sbufferTime = endTime - startTime;

System.out.println("StringBuilder results:\n" + sbResult);

System.out.println("StringBuilder time taken: " + sbTime + " milliseconds");

System.out.println("StringBuffer results:\n" + sbufferResult);

System.out.println("StringBuffer time taken: " + sbufferTime + " milliseconds");

String conclusion = "For " + iterations + " iterations, StringBuilder performed " + (sbufferTime - sbTime) +

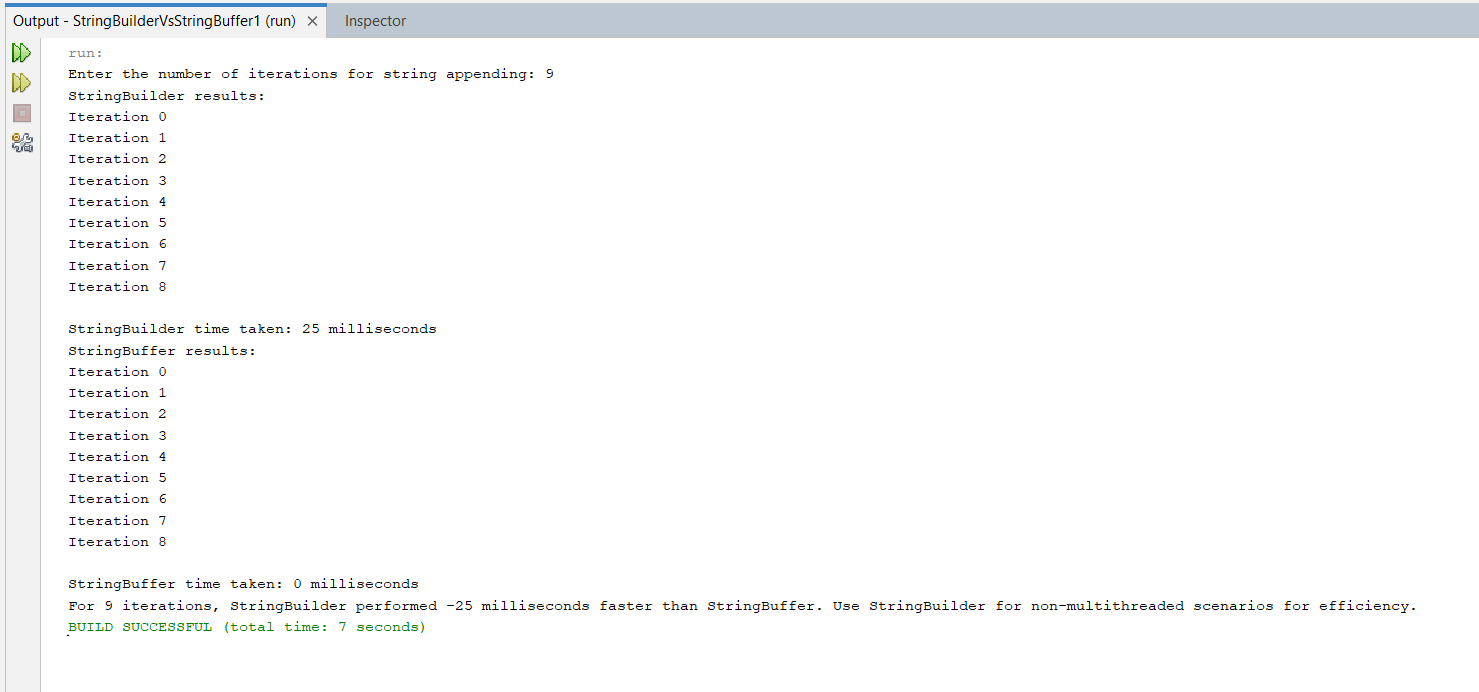
" milliseconds faster than StringBuffer. Use StringBuilder for non-multithreaded scenarios for efficiency.";

System.out.println(conclusion);

}

}

**Output**



5. Java Array for printing first loop from 1 to 9 and second loop from 9 to 1 and store them in matrix A and matrix B

**Program**

import java.util.Scanner;

public class MatrixExample {

public static void main(String[] args) {

int n = 3;

int[][] A = new int[n][n];

// Assigning values 1 to 9 for A matrix

int counter = 1;

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

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

A[i][j] = counter;

counter++;

}

}

int[][] B = new int[n][n];

// Assigning value 9 to 1 for B matrix

counter = 9;

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

for (int j = n - 1; j >= 0; j--) {

B[i][j] = counter;

counter--;

}

}

// Printing matrix A

System.out.println("Matrix A (1 to 9):");

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

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

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

}

System.out.println();

}

// Printing matrix B

System.out.println("\nMatrix B (9 to 1):");

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

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

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

}

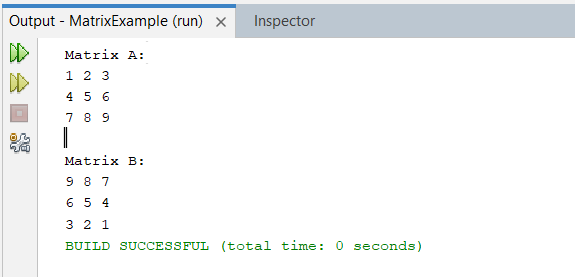
System.out.println();

}

}

}

**Output**



1. Java program to create confusion matrix and calculate TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative), and F1- score:

**Program:**

import java.util.Scanner;

public class ConfusionMatrix {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input confusion matrix elements from the user

System.out.println("Enter the Confusion Matrix Elements:");

System.out.print("True Positives (TP): ");

int TP = scanner.nextInt();

System.out.print("True Negatives (TN): ");

int TN = scanner.nextInt();

System.out.print("False Positives (FP): ");

int FP = scanner.nextInt();

System.out.print("False Negatives (FN): ");

int FN = scanner.nextInt();

// Calculate True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN)

double precision = (double) TP / (TP + FP);

double recall = (double) TP / (TP + FN);

double f1Score = 2 \* (precision \* recall) / (precision + recall);

// Output the results

System.out.println("\nResults:");

System.out.println("True Positives (TP): " + TP);

System.out.println("True Negatives (TN): " + TN);

System.out.println("False Positives (FP): " + FP);

System.out.println("False Negatives (FN): " + FN);

System.out.println("Precision: " + precision);

System.out.println("Recall: " + recall);

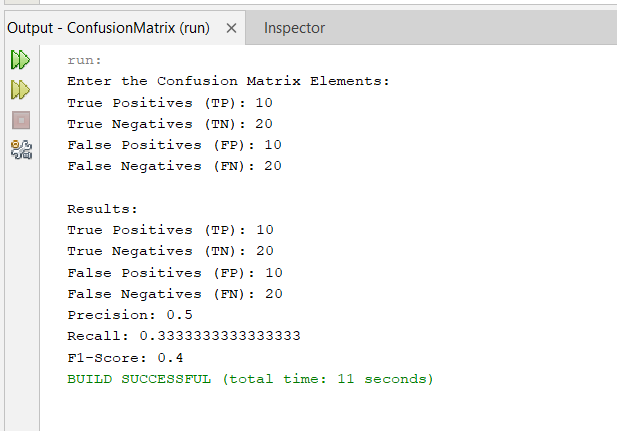
System.out.println("F1-Score: " + f1Score);

scanner.close();

}

}

**Output:**



7. Write a program using Arrays class in java for

creating 2D matrix

**Program:**

import java.util.Scanner;

public class Matrix {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of rows: ");

int rows = scanner.nextInt();

System.out.print("Enter the number of columns: ");

int columns = scanner.nextInt();

int[][] matrix = new int[rows][columns];

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

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

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

System.out.print("Enter element at position [" + i + "][" + j + "]: ");

matrix[i][j] = scanner.nextInt();

}

}

System.out.println("Entered Matrix:");3

printMatrix(matrix);

scanner.close();

}

private static void printMatrix(int[][] matrix) {

for (int[] row : matrix) {

for (int element : row) {

System.out.print(element + "\t");

}

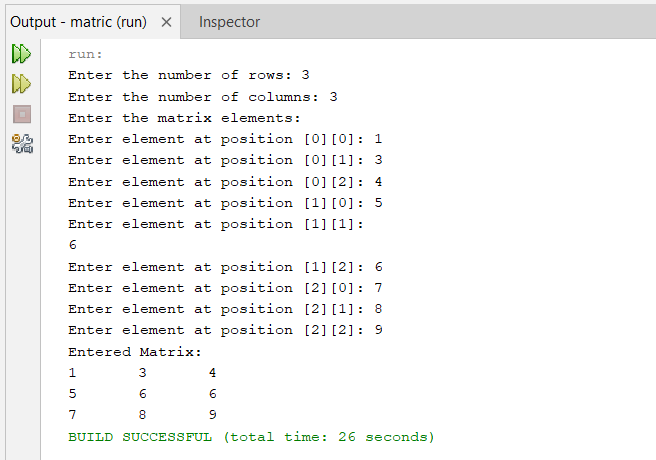
System.out.println();

}

}

}

**Output:**



1. Java program to find if 2 arrays have equal length, if not make it equal according to the smallest array among the both, use Arrays class and copyOf() function

**Program**

import java.util.Arrays;

public class EqualizeArrays {

public static void main(String[] args) {

int[] array1 = {1, 2, 3, 4, 5};

int[] array2 = {1, 2, 3};

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

System.out.println("Array1: " + Arrays.toString(array1));

System.out.println("Array2: " + Arrays.toString(array2));

// Check if arrays have equal length

if (array1.length != array2.length) {

// Make them equal according to the smallest array length

int minSize = Math.min(array1.length, array2.length);

if (array1.length > minSize) {

array1 = Arrays.copyOf(array1, minSize);

} else {

array2 = Arrays.copyOf(array2, minSize);

}

}

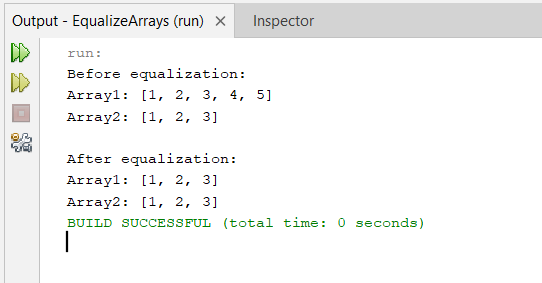
System.out.println("\nAfter equalization:");

System.out.println("Array1: " + Arrays.toString(array1));

System.out.println("Array2: " + Arrays.toString(array2));

} }

**Output**



9. Find greatest of 3 numbers in java, without using If statement

Hint: Ternary operators

**Program**

Import java.util.Scanner;

public class GreatestOfThree {

public static void main(String[] args) {

int num1 = 10;

int num2 = 20;

int num3 = 15;

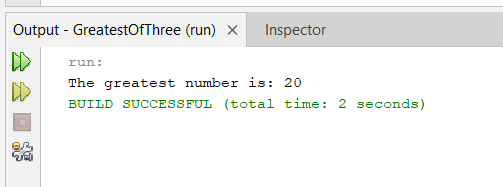
int greatest = num1 > num2 ? (num1 > num3 ? num1 : num3) : (num2 > num3 ?

num2 : num3);

System.out.println("The greatest number is: " + greatest);

} }

**Output**



1. Use nested if to find which if statement contains the greatest value

**Program**

import java.util.Scanner;

public class GreatestInNestedIf {

public static void main(String[] args) {

int num1 = 10;

int num2 = 20;

int num3 = 15;

if (num1 > num2) {

if (num1 > num3) {

System.out.println("The greatest value is in the first if statement: " + num1);

} else {

System.out.println("The greatest value is in the third if statement: " + num3);

}

} else {

if (num2 > num3) {

System.out.println("The greatest value is in the second if statement: " + num2);

} else {

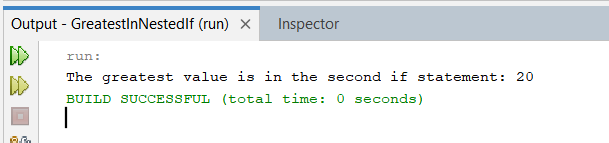
System.out.println("The greatest value is in the third if statement: " + num3);

}

}

} }

**Output**



11. If a person is age 28 when his younger brother is 24, what will be his age when the older brother is 56. If age difference is more than 3 then find the age of younger brother when older brother was 5.

**Program**

import java.util.Scanner;

public class AgeCalculation {

public static void main(String[] args) {

int personAgeAt28 = 28;

int youngerBrotherAgeAt28 = 24;

// Find the age difference

int ageDifference = personAgeAt28 - youngerBrotherAgeAt28;

// Calculate the age of the person when the older brother is 56

int personAgeAt56 = 56 + ageDifference;

// Check if the age difference is more than 3

if (ageDifference > 3) {

// Calculate the age of the younger brother when the older brother was 5

int youngerBrotherAgeAt5 = youngerBrotherAgeAt28 - ageDifference + 5;

System.out.println("The age of the younger brother when the older brother was 5: " + youngerBrotherAgeAt5);

}

else {

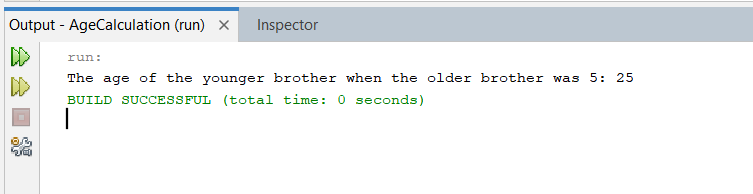
// Print the age of the person when the older brother is 56

System.out.println("The age of the person when the older brother is 56: " + personAgeAt56);

}

} }

**Output**



12. Find a person's birth year based on their eligibility to vote, given that the current year is 2073 and the eligibility age is 21 years, we first need to calculate the birth year. If the birth year is more than 2060, we then find the years between the person's vote-eligible year and 2023 else find median of birth year and 2023

**Program**

public class BirthYear {

public static void main(String[] args) {

int currentYear = 2073;

int eligibilityAge = 21;

int voteEligibleYear = currentYear - eligibilityAge;

if (voteEligibleYear > 2060) {

System.out.println("Birth year is more than 2060.");

System.out.println("Years between vote-eligible year and 2023:");

for (int year = voteEligibleYear; year >= 2023; year--) {

System.out.println(year);

}

} else {

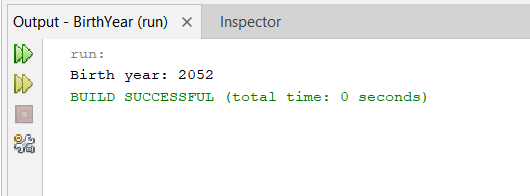
int birthYear = currentYear - eligibilityAge;

System.out.println("Birth year: " + birthYear);

}

} }

**Output**



13. Use nested if to find which loop contains the greatest value

**Program**

public class GreatestValueInLoops {

public static void main(String[] args) {

int[] loop1 = {10, 15, 8, 20, 12};

int[] loop2 = {25, 18, 22, 30, 28};

int maxInLoop1 = findMax(loop1);

int maxInLoop2 = findMax(loop2);

if (maxInLoop1 > maxInLoop2) {

System.out.println("Loop 1 contains the greatest value: " + maxInLoop1);

} else if (maxInLoop2 > maxInLoop1) {

System.out.println("Loop 2 contains the greatest value: " + maxInLoop2);

} else {

System.out.println("Both loops contain the same greatest value: " + maxInLoop1);

}

}

private static int findMax(int[] loop) {

int max = loop[0];

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

if (loop[i] > max) {

max = loop[i];

}

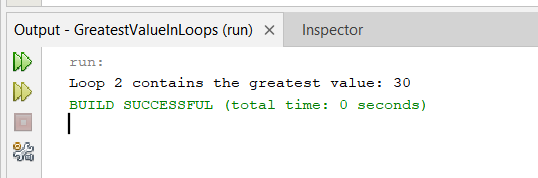
}

return max;

}

}

**Output**



1. Find missing numbers in the series 1,5,11,19 using java.

**Program**

public class MissingNumbersInSeries {

public static void main(String[] args) {

int[] series = {1, 5, 11, 19};

findMissingNumbers(series);

}

private static void findMissingNumbers(int[] series) {

System.out.print("Missing numbers in the series are: ");

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

int diff = series[i + 1] - series[i];

if (diff > 1) {

for (int j = 1; j < diff; j++) {

int missingNumber = series[i] + j;

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

}

}

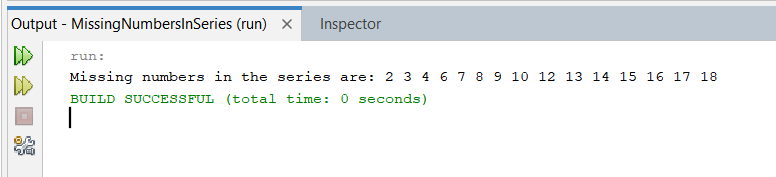
}

System.out.println();

}

}

**Output**



15. write java program to iterate through array elements using enhanced for loop. And find no. of prime numbers

**Program**

public class PrimeNumbersInArray {

// Function to check if a number is prime

public static boolean isPrime(int n) {

if (n <= 1) {

return false;

}

for (int i = 2; i <= Math.sqrt(n); i++) {

if (n % i == 0) {

return false;

}

}

return true;

}

public static void main(String[] args) { int[] array = {7, 8, 11, 15, 17, 20, 23};

int primeCount = 0;

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

for (int num : array) {

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

if (isPrime(num)) {

primeCount++;

}

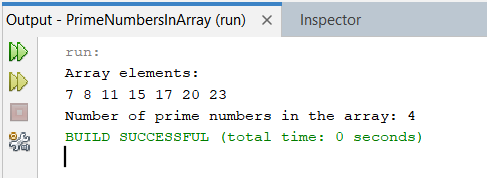
}

System.out.println("\nNumber of prime numbers in the array: " + primeCount);

}

}

**Output**



16 .Iterate to two for loops, both contains 10 array elements, one is normal order, another in reverse order, find the median where they will meet

**Program**

public class MedianAtMeetingPoint {

public static void main(String[] args) {

int[] normalOrder = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int[] reverseOrder = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

int median = -1; // Initialize median to a default value

// Iterate through both arrays simultaneously

for (int i = 0, j = reverseOrder.length - 1; i < normalOrder.length && j >= 0; i++, j--) {

// If the current element from normalOrder array is greater than or equal to the current element

// from reverseOrder array, we have reached or passed the meeting point of the median

if (normalOrder[i] >= reverseOrder[j]) {

median = normalOrder[i]; // Set the median value

break; // Exit the loop

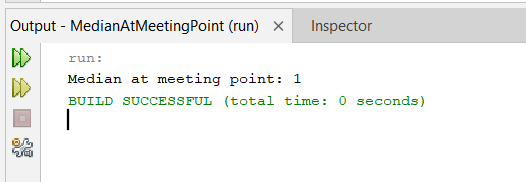
}

}

System.out.println("Median at meeting point: " + median);

} }

**Output**

\

17. Write java program using switch case to find luck guess

**Program**

import java.util.Scanner;

public class LuckyGuess {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Welcome to the Lucky Guess game!");

System.out.println("Guess a number between 1 and 5:");

int luckyNumber = (int) (Math.random() \* 5) + 1;

// Generate a random number between 1 and 5

int guess = scanner.nextInt(); // Get user's guess

switch (guess) {

case 1:

case 2:

case 3:

case 4:

case 5:

if (guess == luckyNumber) {

System.out.println("Congratulations! You guessed the lucky number: " +

luckyNumber);

} else {

System.out.println("Sorry! Better luck next time. The lucky number was: " +

luckyNumber);

}

break;

default:

System.out.println("Invalid guess! Please guess a number between 1 and 5.");

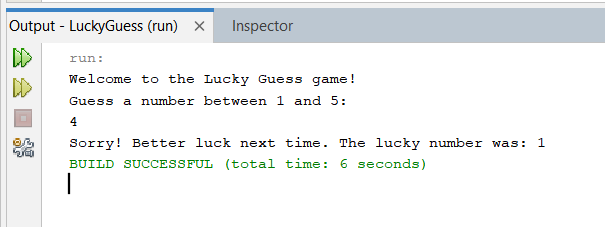
}

scanner.close();

}

}

**Output**



18. Java switch case to find which for loop is shortest path first

**Program**

public class ShortestPath {

public static void main(String[] args) {

int[] loop1 = {1, 2, 3, 4, 5};

int[] loop2 = {10, 20, 30, 40, 50, 60};

int loop1Iterations = loop1.length;

int loop2Iterations = loop2.length;

switch (Integer.compare(loop1Iterations, loop2Iterations)) {

case -1:

System.out.println("Loop 1 is the shortest path.");

break;

case 0:

System.out.println("Both loops have the same number of iterations.");

break;

case 1:

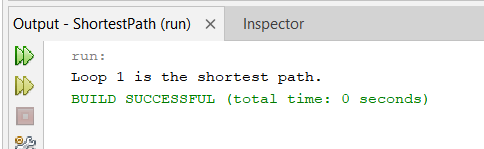
System.out.println("Loop 2 is the shortest path.");

break;

}

} }

**Output**



19. Write a java program to implement OR gate and AND gate

**Program**

public class LogicalGates {

// OR gate implementation

public static boolean OR(boolean input1, boolean input2) {

return input1 || input2;

}

// AND gate implementation

public static boolean AND(boolean input1, boolean input2) {

return input1 && input2;

}

public static void main(String[] args) {

// Testing OR gate

boolean input1 = true;

boolean input2 = false;

System.out.println("OR gate:");

System.out.println(input1 + " OR " + input2 + " = " + OR(input1, input2));

// Testing AND gate

input1 = true;

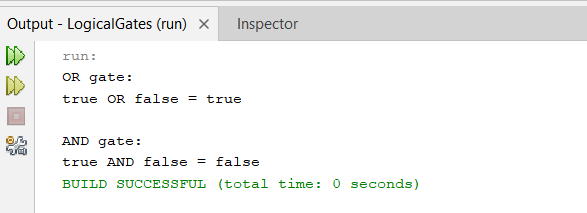
input2 = false;

System.out.println("\nAND gate:");

System.out.println(input1 + " AND " + input2 + " = " + AND(input1, input2));

} }

**Output**



20. Write a program to shift values from left to right if A>B else shift right to left using logical shift operators. Where A and B are arrays

**Program**

public class ArrayShift {

public static void main(String[] args) {

int[] A = {5, 8, 3, 9, 2};

int[] B = {3, 6, 2, 7, 4};

if (A.length != B.length) {

System.out.println("Arrays A and B must have the same length.");

return;

}

// Check each pair of elements in arrays A and B

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

if (A[i] > B[i]) {

// Shift elements from left to right

for (int j = 0; j < A.length - 1; j++) {

A[j] = A[j + 1];

B[j] = B[j + 1]; }

A[A.length - 1] = 0;

B[B.length - 1] = 0;

} else {

// Shift elements from right to left

for (int j = A.length - 1; j > 0; j--) {

A[j] = A[j - 1];

B[j] = B[j - 1];

}

A[0] = 0;

B[0] = 0;

}

}

// Print the shifted arrays

System.out.println("Shifted arrays:");

System.out.println("A: " + java.util.Arrays.toString(A));

System.out.println("B: " + java.util.Arrays.toString(B));

}

}

**Output**

