1. public class PrimitiveDataTypesDemo {

public static void main(String[] args) {

// Integer data types

byte byteValue = 10;

short shortValue = 1000;

int intValue = 100000;

long longValue = 1000000000L; // Note the 'L' suffix to denote a long literal

// Floating-point data types

float floatValue = 3.14f; // Note the 'f' suffix to denote a float literal

double doubleValue = 3.14159265359;

// Character data type

char charValue = 'A';

// Boolean data type

boolean booleanValue = true;

// Output values

System.out.println("Byte: " + byteValue);

System.out.println("Short: " + shortValue);

System.out.println("Int: " + intValue);

System.out.println("Long: " + longValue);

System.out.println("Float: " + floatValue);

System.out.println("Double: " + doubleValue);

System.out.println("Char: " + charValue);

System.out.println("Boolean: " + booleanValue);

}

}

2. Below is a Java program demonstrating data type conversion including implicit conversion (widening), explicit conversion (narrowing), and scenarios of overflow and underflow:

public class DataTypeConversionDemo {

public static void main(String[] args) {

// Implicit conversion (widening) from smaller data types to larger ones

int intValue = 10;

long longValue = intValue; // int to long

float floatValue = 3.14f;

double doubleValue = floatValue; // float to double

// Explicit conversion (narrowing) from larger data types to smaller ones

double largeDoubleValue = 123456789.123456789;

float smallFloatValue = (float) largeDoubleValue; // double to float (explicit casting)

long largeLongValue = 999999999999L;

int smallIntValue = (int) largeLongValue; // long to int (explicit casting)

// Overflow and underflow scenarios

int overflowInt = Integer.MAX\_VALUE + 1; // Overflow scenario

int underflowInt = Integer.MIN\_VALUE - 1; // Underflow scenario

// Output results

System.out.println("Implicit Conversion:");

System.out.println("int to long: " + longValue);

System.out.println("float to double: " + doubleValue);

System.out.println("\nExplicit Conversion:");

System.out.println("double to float: " + smallFloatValue);

System.out.println("long to int: " + smallIntValue);

System.out.println("\nOverflow and Underflow Scenarios:");

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

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

}

}

This program demonstrates both implicit and explicit data type conversion in Java, along with scenarios of overflow and underflow.

3.Below is a Java program demonstrating some common String methods and constructors:

java

public class StringMethodsAndConstructorsDemo {

public static void main(String[] args) {

// String constructors

String str1 = new String(); // Empty string

String str2 = new String("Hello"); // Initialized with a string literal

char[] charArray = {'W', 'o', 'r', 'l', 'd'};

String str3 = new String(charArray); // Initialized with a char array

byte[] byteArray = {72, 101, 108, 108, 111};

String str4 = new String(byteArray); // Initialized with a byte array

// String methods

String str = "Hello, World!";

// length() method

int length = str.length();

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

// charAt() method

char charAtIndex = str.charAt(7);

System.out.println("Character at index 7: " + charAtIndex);

// substring() method

String substring = str.substring(7);

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

// indexOf() method

int index = str.indexOf('W');

System.out.println("Index of 'W': " + index);

// concat() method

String newString = str.concat(" How are you?");

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

// replace() method

String replacedString = str.replace('o', 'x');

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

// toUpperCase() method

String upperCaseString = str.toUpperCase();

System.out.println("String in upper case: " + upperCaseString);

// toLowerCase() method

String lowerCaseString = str.toLowerCase();

System.out.println("String in lower case: " + lowerCaseString);

// trim() method

String stringWithSpaces = " Hello, World! ";

String trimmedString = stringWithSpaces.trim();

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

}

}

This program demonstrates various constructors to create strings and also illustrates several common methods available in the String class in Java.

4.public class StringBuilderStringBufferDemo {

public static void main(String[] args) {

// Example of StringBuilder

StringBuilder stringBuilder = new StringBuilder();

stringBuilder.append("Hello");

stringBuilder.append(" ");

stringBuilder.append("World");

String resultStringBuilder = stringBuilder.toString();

System.out.println("StringBuilder Result: " + resultStringBuilder);

// Example of StringBuffer

StringBuffer stringBuffer = new StringBuffer();

stringBuffer.append("Hello");

stringBuffer.append(" ");

stringBuffer.append("World");

String resultStringBuffer = stringBuffer.toString();

System.out.println("StringBuffer Result: " + resultStringBuffer);

// Performance comparison

int iterations = 1000000; // Number of iterations for appending strings

long startTimeStringBuilder = System.currentTimeMillis();

StringBuilder stringBuilderPerformance = new StringBuilder();

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

stringBuilderPerformance.append("Hello");

}

long endTimeStringBuilder = System.currentTimeMillis();

long elapsedTimeStringBuilder = endTimeStringBuilder - startTimeStringBuilder;

System.out.println("Time taken by StringBuilder: " + elapsedTimeStringBuilder + " milliseconds");

long startTimeStringBuffer = System.currentTimeMillis();

StringBuffer stringBufferPerformance = new StringBuffer();

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

stringBufferPerformance.append("Hello");

}

long endTimeStringBuffer = System.currentTimeMillis();

long elapsedTimeStringBuffer = endTimeStringBuffer - startTimeStringBuffer;

System.out.println("Time taken by StringBuffer: " + elapsedTimeStringBuffer + " milliseconds");

}

}

4.public class MatrixCreation {

public static void main(String[] args) {

// Matrix A: storing numbers from 1 to 9 in the first row

int[][] matrixA = new int[2][9];

for (int i = 0; i < 9; i++) {

matrixA[0][i] = i + 1;

}

// Matrix B: storing numbers from 9 to 1 in the second row

int value = 9;

for (int i = 0; i < 9; i++) {

matrixA[1][i] = value;

value--;

}

// Printing Matrix A

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

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 9; j++) {

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

}

System.out.println();

}

// Printing Matrix B

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

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 9; j++) {

System.out.print(matrixA[1-i][j] + " ");

}

System.out.println();

}

}

}

5. public class ConfusionMatrix {

public static void main(String[] args) {

// Example confusion matrix values

int[][] confusionMatrix = {

{50, 10}, // Actual Positive

{5, 80} // Actual Negative

};

// Calculate TP, TN, FP, FN

int TP = confusionMatrix[0][0];

int FP = confusionMatrix[0][1];

int FN = confusionMatrix[1][0];

int TN = confusionMatrix[1][1];

// Calculate precision, recall, and F1-score

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

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

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

// Print the confusion matrix

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

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

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

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

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

// Print precision, recall, and F1-score

System.out.println("\nPrecision: " + precision);

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

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

}

}

6.

import java.util.Arrays;

public class MatrixCreation {

public static void main(String[] args) {

int rows = 3;

int cols = 4;

// Create a 2D matrix

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

// Fill the matrix with values

int value = 1;

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

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

matrix[i][j] = value;

value++;

}

}

// Print the matrix using deepToString method from Arrays class

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

System.out.println(Arrays.deepToString(matrix));

}

}

7.

import java.util.Arrays;

public class EqualLengthArrays {

public static void main(String[] args) {

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

int[] array2 = {4, 5};

// Check if arrays have equal length

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

// Determine the smallest length among the two arrays

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

// Resize arrays using Arrays.copyOf()

array1 = Arrays.copyOf(array1, minLength);

array2 = Arrays.copyOf(array2, minLength);

}

// Print the arrays

System.out.println("Array 1: " + Arrays.toString(array1));

System.out.println("Array 2: " + Arrays.toString(array2));

}

}

8.

public class GreatestOfThree {

public static void main(String[] args) {

int num1 = 10;

int num2 = 20;

int num3 = 15;

// Using ternary operators to find the greatest number

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

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

}

}

9.

public class GreatestIfStatement {

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 = " + num1);

} else {

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

}

} else {

if (num2 > num3) {

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

} else {

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

}

}

}

}

11 public class AgeCalculations {

public static void main(String[] args) {

int agePerson = 28;

int ageYoungerBrother = 24;

// Age of the older brother

int ageOlderBrother = ageYoungerBrother + (agePerson - ageYoungerBrother);

System.out.println("Age of the older brother: " + ageOlderBrother);

// Age of the person when older brother is 56

int ageWhenOlderBrotherIs56 = ageOlderBrother + (ageOlderBrother - ageYoungerBrother);

System.out.println("Age of the person when older brother is 56: " + ageWhenOlderBrotherIs56);

// Age of the younger brother when older brother was 5

int ageYoungerBrotherWhenOlderBrotherWas5 = ageYoungerBrother - (ageOlderBrother - 5);

System.out.println("Age of the younger brother when older brother was 5: " + ageYoungerBrotherWhenOlderBrotherWas5);

}

}

12.

public class BirthYearCalculation {

public static void main(String[] args) {

int currentYear = 2073;

int eligibilityAge = 21;

// Calculating birth year

int birthYear = currentYear - eligibilityAge;

int medianYear = (birthYear + 2023) / 2;

System.out.println("Median of birth year and 2023: " + medianYear);

}

}

13

public class GreatestLoop {

public static void main(String[] args) {

int loop1 = 10;

int loop2 = 11;

int loop3 = 12;

if (loop1 > loop2 && loop1 > loop3) {

System.out.println("Loop 1 has the greatest value.");

} else if (loop2 > loop3) {

System.out.println("Loop 2 has the greatest value.");

} else {

System.out.println("Loop 3 has the greatest value.");

}

}

}

14.

public class MissingNumbersSeries {

public static void main(String[] args) {

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

int missingNumber = 0;

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

if (series[i + 1] - series[i] != 4) {

missingNumber = series[i] + 4;

System.out.println("Missing number in the series: " + missingNumber);

}

}

}

}

15.

public class PrimeNumberCount {

public static void main(String[] args) {

int[] numbers = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}; // Sample array

int primeCount = 0;

for (int num : numbers) {

boolean isPrime = true;

if (num <= 1) {

isPrime = false;

} else {

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

if (num % i == 0) {

isPrime = false;

break;

}

}

}

if (isPrime) {

primeCount++;

}

}

System.out.println("Number of prime numbers: " + primeCount);

}

}

16.

public class MedianMeetingPoint {

public static void main(String[] args) {

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

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

int median = -1;

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

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

median = normalOrder[i];

break;

}

}

System.out.println("Median where they meet: " + median);

}

}

17.

public class LuckyGuess {

public static void main(String[] args) {

int guess = 7; // Sample guess

String result;

switch (guess) {

case 7:

result = "Lucky guess!";

break;

case 13:

result = "Unlucky guess!";

break;

default:

result = "Normal guess";

break;

}

System.out.println(result);

}

}

18.

public class ShortestPath {

public static void main(String[] args) {

int shortestPath = 2; // Sample shortest path

String result;

switch (shortestPath) {

case 1:

result = "First for loop is shortest path";

break;

case 2:

result = "Second for loop is shortest path";

break;

default:

result = "Both for loops have the same length";

break;

}

System.out.println(result);

}

}

19.

public class LogicGates {

public static void main(String[] args) {

boolean a = true;

boolean b = false;

// OR gate

boolean orResult = a || b;

System.out.println("OR gate result: " + orResult);

// AND gate

boolean andResult = a && b;

System.out.println("AND gate result: " + andResult);

}

}

20.

public class ShiftValues {

public static void main(String[] args) {

int[] A = {1, 2, 3, 4, 5}; // Sample array

int[] B = {6, 7, 8, 9, 10}; // Sample array

if (A.length > B.length) {

// Shift values from left to right

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

A[i] = A[i + 1];

}

A[A.length - 1] = 0; // Set the last element to 0

} else if (A.length < B.length) {

// Shift values from right to left

for (int i = B.length - 1; i > 0; i--) {

B[i] = B[i - 1];

}

B[0] = 0; // Set the first element to 0

}

// Printing the arrays after shifting

System.out.print("A: ");

for (int num : A) {

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

}

System.out.println();

System.out.print("B: ");

for (int num : B) {

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

}

}

}