**1. Program to Demonstrate Primitive Data Types in Java**

Pseudocode:

BEGIN

DECLARE variables of types: byte, short, int, long, float, double, char, and boolean

ASSIGN appropriate values to each variable

PRINT all the values of the variables

END

**Java Code:**

public class PrimitiveDataTypesDemo {

public static void main(String[] args) {

// Integer data types

byte byteValue = 10;

short shortValue = 100;

int intValue = 1000;

long longValue = 10000L;

// Floating-point data types

float floatValue = 10.5f;

double doubleValue = 20.99;

// Character data type

char charValue = 'A';

// Boolean data type

boolean booleanValue = true;

// Print all 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. Program Demonstrating Data Type Conversion in Java**

# a) Implicit Conversion (Widening)

Pseudocode:

BEGIN

DECLARE a smaller data type variable (int)

CONVERT the smaller data type to a larger one (long) automatically

PRINT the converted value

END

Java Code:

java

public class ImplicitConversion {

public static void main(String[] args) {

int intValue = 100; // Smaller data type

long longValue = intValue; // Implicit conversion (widening)

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

}

}

# b) Explicit Conversion (Narrowing)

Pseudocode:

BEGIN

DECLARE a larger data type variable (double)

CONVERT the larger data type to a smaller one (int) explicitly

PRINT the converted value

END

Java Code:

java

public class ExplicitConversion {

public static void main(String[] args) {

double doubleValue = 100.45; // Larger data type

int intValue = (int) doubleValue; // Explicit conversion (narrowing)

System.out.println("Explicit Conversion (double to int): " + intValue);

}

}

# c) Overflow and Underflow Scenarios

Pseudocode:

BEGIN

DECLARE a byte variable and assign a value exceeding its range

PRINT the resulting overflow value

DECLARE a byte variable and assign a negative value less than its minimum

PRINT the resulting underflow value

END

Java Code:

java

public class OverflowUnderflowDemo {

public static void main(String[] args) {

// Overflow scenario

byte byteValueOverflow = (byte) 130; // Exceeds byte range

System.out.println("Overflowed byte value: " + byteValueOverflow);

// Underflow scenario

byte byteValueUnderflow = (byte) -130; // Below byte range

System.out.println("Underflowed byte value: " + byteValueUnderflow);

}

}

### 3. Java Program for String Methods and String Constructors

Pseudocode:

BEGIN

CREATE string objects using various constructors

DEMONSTRATE common string methods (length(), substring(), charAt(), toUpperCase(), concat())

PRINT the results of string manipulations

END

Java Code:

java

public class StringMethodsDemo {

public static void main(String[] args) {

// String constructors

String str1 = new String("Hello");

String str2 = "World";

// String methods

System.out.println("Length of str1: " + str1.length());

System.out.println("Substring of str1: " + str1.substring(1, 4));

System.out.println("Character at index 2 in str1: " + str1.charAt(2));

System.out.println("Uppercase of str2: " + str2.toUpperCase());

System.out.println("Concatenation of str1 and str2: " + str1.concat(str2));

}

}

---

### 4. StringBuilder and StringBuffer Program Differences

Pseudocode:

BEGIN

CREATE StringBuilder and StringBuffer objects

APPEND strings using both

COMPARE performance by appending a large number of strings in a loop

MEASURE and print the time taken for both

END

Java Code:

java

public class StringBuilderVsStringBuffer {

public static void main(String[] args) {

// Simple example of appending strings

StringBuilder sb = new StringBuilder("Hello");

sb.append(" World!");

System.out.println("StringBuilder: " + sb);

StringBuffer sbf = new StringBuffer("Hello");

sbf.append(" World!");

System.out.println("StringBuffer: " + sbf);

// Performance comparison

int iterations = 100000;

long startTime, endTime;

// StringBuilder performance

startTime = System.nanoTime();

StringBuilder sbPerf = new StringBuilder();

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

sbPerf.append("A");

}

endTime = System.nanoTime();

System.out.println("StringBuilder Time: " + (endTime - startTime) + " ns");

// StringBuffer performance

startTime = System.nanoTime();

StringBuffer sbfPerf = new StringBuffer();

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

sbfPerf.append("A");

}

endTime = System.nanoTime();

System.out.println("StringBuffer Time: " + (endTime - startTime) + " ns");

}

}

---

### 5. Java Array for Printing Two Loops (1 to 9 and 9 to 1) and Storing in Matrix A and B

Pseudocode:

BEGIN

CREATE two matrices A and B

FILL matrix A with numbers from 1 to 9

FILL matrix B with numbers from 9 to 1

PRINT both matrices

END

Java Code:

java

public class MatrixLoops {

public static void main(String[] args) {

int[] A = new int[9];

int[] B = new int[9];

// Filling matrix A from 1 to 9

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

A[i] = i + 1;

}

// Filling matrix B from 9 to 1

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

B[i] = 9 - i;

}

// Printing matrix A

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

// Printing matrix B

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

}

}

---

### 6. Java Program to Create Confusion Matrix and Calculate TP, TN, FP, FN, and F1-Score

Pseudocode:

BEGIN

CREATE confusion matrix

CALCULATE True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN)

CALCULATE Precision, Recall, and F1-Score

PRINT all the calculated values

END

Java Code:

java

public class ConfusionMatrix {

public static void main(String[] args) {

// Confusion matrix (for binary classification)

int[][] confusionMatrix = {

{50, 10}, // TP, FN

{5, 35} // FP, TN

};

int TP = confusionMatrix[0][0];

int FN = confusionMatrix[0][1];

int FP = 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 results

System.out.println("TP: " + TP + ", TN: " + TN + ", FP: " + FP + ", FN: " + FN);

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

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

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

}

}

---

### 7. Write a Program Using Arrays Class in Java for Creating a 2D Matrix

Pseudocode:

BEGIN

CREATE a 2D matrix using the Arrays class

FILL the matrix with user input values

PRINT the matrix

END

Java Code:

java

import java.util.Arrays;

import java.util.Scanner;

public class MatrixUsingArrays {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int rows = sc.nextInt();

int cols = sc.nextInt();

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

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

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

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

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

}

}

// Print the matrix

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

for (int[] row : matrix) {

System.out.println(Arrays.toString(row));

}

}

}

### 8. Java Program to Find If Two Arrays Have Equal Length and Adjust Them to the Smallest One Using `Arrays.copyOf()`

\*\*Pseudocode:\*\*

```

BEGIN

TAKE two arrays as input

CHECK if both arrays have the same length

IF not, copy elements from both arrays up to the length of the smaller array using Arrays.copyOf()

PRINT the new arrays

END

```

\*\*Java Code:\*\*

```java

import java.util.Arrays;

public class EqualLengthArrays {

public static void main(String[] args) {

int[] arr1 = {1, 2, 3, 4, 5};

int[] arr2 = {6, 7, 8};

// Compare lengths and adjust to the smaller one

if (arr1.length != arr2.length) {

int minLength = Math.min(arr1.length, arr2.length);

arr1 = Arrays.copyOf(arr1, minLength);

arr2 = Arrays.copyOf(arr2, minLength);

}

// Print the adjusted arrays

System.out.println("Array 1: " + Arrays.toString(arr1));

System.out.println("Array 2: " + Arrays.toString(arr2));

}

}

```

---

### 9. Find the Greatest of Three Numbers in Java Without Using `if` (Using Ternary Operators)

\*\*Pseudocode:\*\*

```

BEGIN

TAKE three numbers as input

USE nested ternary operators to find the greatest number

PRINT the greatest number

END

```

\*\*Java Code:\*\*

```java

public class GreatestOfThree {

public static void main(String[] args) {

int a = 10, b = 25, c = 15;

// Find the greatest number using ternary operators

int greatest = (a > b) ? (a > c ? a : c) : (b > c ? b : c);

// Print the greatest number

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

}

}

```

---

### 10. Use Nested `if` to Find Which `if` Statement Contains the Greatest Value

\*\*Pseudocode:\*\*

```

BEGIN

TAKE three numbers as input

USE nested `if` statements to find the greatest value

PRINT which `if` statement has the greatest value

END

```

\*\*Java Code:\*\*

```java

public class NestedIfGreatest {

public static void main(String[] args) {

int x = 12, y = 45, z = 30;

// Use nested if statements to find the greatest value

if (x > y) {

if (x > z) {

System.out.println("x is the greatest: " + x);

} else {

System.out.println("z is the greatest: " + z);

}

} else {

if (y > z) {

System.out.println("y is the greatest: " + y);

} else {

System.out.println("z is the greatest: " + z);

}

}

}

}

```

---

### 11. Age Calculation

\*\*Pseudocode:\*\*

```

BEGIN

SET the current age of the older brother and younger brother

CALCULATE the older brother's age when he is 56 using the age difference

IF the age difference is more than 3, calculate the younger brother's age when the older brother was 5

PRINT the results

END

```

\*\*Java Code:\*\*

```java

public class AgeCalculation {

public static void main(String[] args) {

int olderBrotherAge = 28;

int youngerBrotherAge = 24;

int ageDifference = olderBrotherAge - youngerBrotherAge;

// Older brother's age when he is 56

int olderBrotherFutureAge = 56;

int youngerBrotherFutureAge = youngerBrotherAge + (olderBrotherFutureAge - olderBrotherAge);

// Print the future age of the younger brother

System.out.println("When the older brother is 56, the younger brother will be " + youngerBrotherFutureAge);

// Calculate younger brother's age when the older brother was 5, if age difference is more than 3

if (ageDifference > 3) {

int youngerBrotherAtFive = 5 - ageDifference;

System.out.println("When the older brother was 5, the younger brother was " + youngerBrotherAtFive);

}

}

}

```

### 12. Find a Person's Birth Year Based on Their Voting Eligibility

\*\*Pseudocode:\*\*

```

BEGIN

SET current year = 2073

SET eligibility age = 21

CALCULATE birth year = current year - eligibility age

IF birth year > 2060 THEN

FIND the years between vote-eligible year and 2023

ELSE

FIND the median of birth year and 2023

END IF

PRINT the result

END

```

\*\*Java Code:\*\*

```java

public class BirthYearEligibility {

public static void main(String[] args) {

int currentYear = 2073;

int eligibilityAge = 21;

int birthYear = currentYear - eligibilityAge;

if (birthYear > 2060) {

int voteEligibleYear = birthYear + eligibilityAge;

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

} else {

int median = (birthYear + 2023) / 2;

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

}

}

}

```

---

### 13. Use Nested `if` to Find Which Loop Contains the Greatest Value

\*\*Pseudocode:\*\*

```

BEGIN

SET loop values for two loops

USE nested `if` to compare the values

PRINT which loop contains the greatest value

END

```

\*\*Java Code:\*\*

```java

public class NestedIfLoops {

public static void main(String[] args) {

int loop1 = 50;

int loop2 = 30;

if (loop1 > loop2) {

System.out.println("Loop 1 has the greatest value: " + loop1);

} else {

System.out.println("Loop 2 has the greatest value: " + loop2);

}

}

}

```

---

### 14. Find Missing Numbers in the Series: 1, 5, 11, 19

\*\*Pseudocode:\*\*

```

BEGIN

SET the series with known numbers

FIND the difference between numbers

CALCULATE missing numbers in the series using the pattern

PRINT missing numbers

END

```

\*\*Java Code:\*\*

```java

public class MissingNumbersSeries {

public static void main(String[] args) {

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

int diff = 4;

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

int next = series[i] + diff \* (i + 1);

System.out.println("Next number in the series: " + next);

}

}

}

```

---

### 15. Iterate Through Array Elements Using Enhanced For Loop and Find Prime Numbers

\*\*Pseudocode:\*\*

```

BEGIN

CREATE an array of numbers

ITERATE through the array using an enhanced for loop

CHECK each number if it's prime

PRINT prime numbers

END

```

\*\*Java Code:\*\*

```java

public class PrimeNumbersInArray {

public static void main(String[] args) {

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

System.out.println("Prime numbers in the array:");

for (int num : arr) {

if (isPrime(num)) {

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

}

}

}

static boolean isPrime(int num) {

if (num <= 1) return false;

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

if (num % i == 0) return false;

}

return true;

}

}

```

---

### 16. Iterate Two For Loops with One in Normal Order and Another in Reverse Order, and Find the Median Where They Meet

\*\*Pseudocode:\*\*

```

BEGIN

CREATE two arrays

ITERATE one array in normal order

ITERATE the second array in reverse order

FIND the median where both loops meet

END

```

\*\*Java Code:\*\*

```java

public class MedianTwoArrays {

public static void main(String[] args) {

int[] arr1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int[] arr2 = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

for (int i = 0; i < arr1.length; i++) {

if (arr1[i] == arr2[i]) {

System.out.println("Median where they meet: " + arr1[i]);

}

}

}

}

```

---

### 17. Java Program Using `switch` Case to Find Luck Guess

\*\*Pseudocode:\*\*

```

BEGIN

TAKE a guess from user

USE `switch` to check guess value

PRINT if the guess is lucky or not

END

```

\*\*Java Code:\*\*

```java

import java.util.Scanner;

public class LuckGuess {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter your lucky guess: ");

int guess = sc.nextInt();

switch (guess) {

case 7:

case 13:

case 21:

System.out.println("You have guessed a lucky number!");

break;

default:

System.out.println("Try again!");

}

}

}

```

---

### 18. Java `switch` Case to Find Which For Loop Is Shortest Path First

\*\*Pseudocode:\*\*

```

BEGIN

CREATE two loops

USE `switch` case to find which loop is shortest path first

PRINT the result

END

```

\*\*Java Code:\*\*

```java

public class ShortestPath {

public static void main(String[] args) {

int path1 = 5;

int path2 = 3;

switch (path1 < path2 ? 1 : 2) {

case 1:

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

break;

case 2:

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

break;

}

}

}

```

---

### 19. Java Program to Implement OR Gate and AND Gate

\*\*Pseudocode:\*\*

```

BEGIN

TAKE two boolean inputs

IMPLEMENT OR gate using logical OR operator

IMPLEMENT AND gate using logical AND operator

PRINT the results

END

```

\*\*Java Code:\*\*

```java

public class LogicGates {

public static void main(String[] args) {

boolean A = true;

boolean B = false;

// OR Gate

System.out.println("A OR B: " + (A || B));

// AND Gate

System.out.println("A AND B: " + (A && B));

}

}

```

---

### 20. Program to Shift Values from Left to Right if `A > B` Else Shift Right to Left Using Logical Shift Operators

\*\*Pseudocode:\*\*

```

BEGIN

TAKE two arrays A and B

COMPARE values of A and B

IF A > B, shift values from left to right

ELSE shift values from right to left

PRINT shifted arrays

END

```

\*\*Java Code:\*\*

```java

import java.util.Arrays;

public class ArrayShift {

public static void main(String[] args) {

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

int[] B = {2, 3, 1, 0};

if (Arrays.compare(A, B) > 0) {

// Shift left to right

int temp = A[A.length - 1];

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

A[i] = A[i - 1];

}

A[0] = temp;

} else {

// Shift right to left

int temp = B[0];

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

B[i] = B[i + 1];

}

B[B.length - 1] = temp;

}

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

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

}

}

```