**Advanced Level (15-20)**

15. public class MemoryAllocationDemo {

public static void main(String[] args) {

int primitiveA = 10;

int primitiveB = primitiveA;

primitiveB = 20;

System.out.println("Primitive Variables:");

System.out.println("primitiveA = " + primitiveA);

System.out.println("primitiveB = " + primitiveB);

int[] referenceA = {10, 20, 30};

int[] referenceB = referenceA;

referenceB[0] = 99;

System.out.println("\nReference Variables:");

System.out.println("referenceA[0] = " + referenceA[0]);

System.out.println("referenceB[0] = " + referenceB[0]);

}

}

16. public class Circle {

static final double PI = 3.14159;

public static double calculateArea(double radius) {

return PI \* radius \* radius;

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

double radius = 5.0;

double area = calculateArea(radius);

System.out.println("Radius: " + radius);

System.out.println("Area of the circle: " + area);

// PI = 3.14;

}

}

17. public class VarTypeInferenceDemo {

public static void main(String[] args) {

var number = 10;

var name = "Java";

var pi = 3.14159;

var isJavaFun = true;

var numbers = new int[]{1, 2, 3};

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

System.out.println("name (String): " + name);

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

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

System.out.print("numbers (int[]): ");

for (var n : numbers) {

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

}

var message = new StringBuilder("Hello");

message.append(" World!");

System.out.println("\nmessage (StringBuilder): " + message);

}

}

18. public class NumericOverflowDemo {

public static void main(String[] args) {

byte b = 127;

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

b++;

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

int i = Integer.MAX\_VALUE;

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

i++;

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

long l = Long.MAX\_VALUE;

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

l++;

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

}

}

19. public class LongToFloatCheck {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

long longValue = scanner.nextLong();

float floatValue = (float) longValue;

long convertedBack = (long) floatValue;

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

System.out.println("Long value after converting back from float: " + convertedBack);

if (longValue == convertedBack) {

System.out.println("The long value can be safely stored in a float.");

} else {

System.out.println(" Precision loss, The long value cannot be stored exactly in a float.");

}

scanner.close();

}

}

20. public class VolatileExample {

private static volatile boolean running = true;

public static void main(String[] args) {

Thread worker = new Thread(() -> {

System.out.println("Worker thread started.");

while (running) {

//simulating Work

}

System.out.println("Worker thread stopped."); });

worker.start();

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Main thread setting running = false");

running = false;

}

}