# Java Programming Assignment

## Section 1: Java Data Types

1. **What are the different primitive data types available in Java?**

In Java, there are 8 primitive data types used to store simple values.

byte stores small whole numbers from -128 to 127.

short stores larger whole numbers than byte

int used for whole numbers

long stores very large whole numbers.

float stores decimal numbers with less precision. (e.g., 7.89f).

double stores decimal numbers with more precision (e.g, 5.7894).

char Stores a single character

boolean Stores only two values: True or False.

1. **Explain the difference between primitive and non-primitive data types in Java.**

Primitive data types in Java are the most basic types like int, char, boolean, and double. They store simple values directly in memory and are not objects. They are fast and memory-efficient.

Non-primitive data types, also known as reference types, include classes, arrays, and interfaces. They store references to objects rather than the actual data.

1. **Write a Java program that demonstrates the use of all primitive data types.**

public class Primitive\_types {

public static void main(String[] args) {

byte a = 100;

short b = 10000;

int c= 100000;

long d = 10000000000L;

float e = 5.75f;

double f = 19.99;

char g = 'A';

boolean h= true;

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

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

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

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

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

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

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

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

}

}

1. **What is type casting? Provide an example of implicit and explicit casting in Java.**

**Type casting** means **converting one data type into another**, there are two types:

Implicit Casting (Widening): Converts smaller type to larger type.

Ex:int num = 10;

double result = num;

System.out.println(result);

Explicit Casting (Narrowing): converts larger type to smaller type.

Ex:double value = 9.99;

int result = (int) value;

System.out.println(result);

1. **What is the default value of each primitive data type in Java?**

| **Data Type** | **Default Value** |
| --- | --- |
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| boolean | false |

## Section 2: Java Control Statements

1. **What are control statements in Java? List the types with examples.**

Control statements in Java are used to control the flow of execution in a program — based on conditions or loops.

**i)Conditional / Decision-making Statements**

Used to execute code only if a condition is true.

a)**IF statement-**if condition is true it will execute

b) **if-else** – Chooses between two blocks based on a condition.

c) **else-if ladder** – Checks multiple conditions one by one

D) **switch** – Selects one block of code to run from many options

ii) **Looping (Iteration) Statements:**

**a)for** – Repeats a block of code a fixed number of times.

**b)while** – Repeats a block of code while a condition is true.

**c)do-while** – Runs code once, then repeats while a condition is true

**iii)** **Jumping Statements**

**a)break –** Exits from a loop or switch block early

**b)continue –** Skips the current loop iteration and goes to the next

**c)return –** Exits from a method and optionally returns a value

**2. Write a Java program to demonstrate the use of if-else and switch-case statements.**

import java.util.Scanner;

public class C1{

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter your age:");

int age = scanner.nextInt();

if (age >= 18) {

System.out.println("Adult");

} else {

System.out.println("Not adult");

}

// switch-case

System.out.print("Enter day (1-7): ");

int day = scanner.nextInt();

switch (day) {

case 1:

System.out.println("Monday");

break;

default:

System.out.println("other day");

}

}

}

**3.What is the difference between break and continue statements?**

 **break** is used to **exit** a loop or switch block completely.and break stop the loop early.

 **continue** is used to **skip the current iteration** and move to the next one in a loop.and it skips one step but keeps the loop running.

1. **Write a Java program to print even numbers between 1 to 50 using a for loop.**

public class EvenNumbers {

public static void main(String[] args) {

System.out.println("Even numbers from 1 to 50:");

for (int i = 1; i <= 50; i++) {

if (i % 2 == 0) {

System.out.println(i);

}

}

}

}

1. **Explain the differences between while and do-while loops with examples.**

 **A while loop** checks the condition **before** running the loop.

Ex: int i = 5;

while (i < 5) {

System.out.println(i);

}

 **A do-while loop** runs the loop **at least once**, and checks the condition **after**.

EX:int i = 5;

do {

System.out.println(i);

} while (i < 5);

## Section 3: Java Keywords and Operators

1. **What are keywords in Java? List 10 commonly used keywords.**

Keywords are reserved words in Java that have special meaning and are used to define the structure of a program. You cannot use them as variable names.

Keywords:int, class, public, static, void, if, else, for, return, new

1. **Explain the purpose of the following keywords: static, final, this, super.**
   * **static** used to define a variable or method that belongs to the class, not the object.
   * **final** used to declare constants or prevent method/variable from being changed.
   * **this** refers to the **current object** of the class.
   * **super** refers to the **parent class** and is used to access its methods or constructor.
2. **What are the types of operators in Java?**

Arithmetic, Relational, Logical, Assignment, Unary, Bitwise, Ternary.

1. **Write a Java program demonstrating the use of arithmetic, relational, and logical operators.**

public class OperatorsDemo {

public static void main(String[] args) {

// Arithmetic Operators

int a = 10;

int b = 5;

System.out.println("Arithmetic Operators:");

System.out.println("a + b = "+(a+b));

System.out.println("a - b="+(a-b));

System.out.println("a \* b ="+(a\*b));

System.out.println("a / b = "+(a / b));

System.out.println("a % b="+ (a %b));

System.out.println("\nRelational Operators:");

System.out.println("a == b:"+(a ==b));

System.out.println("a != b: " + (a != b));

System.out.println("a > b: " + (a > b));

System.out.println("a < b: " + (a < b));

System.out.println("a >= b:"+(a >= b));

System.out.println("a <= b:"+(a <=b));

boolean x = true;

boolean y = false;

System.out.println("\nLogical Operators:");

System.out.println("x && y: " + (x && y));

System.out.println("x || y: " + (x || y));

System.out.println("!x: " + (!x));

}

}

1. **What is operator precedence? How does it affect the outcome of expressions?**

Operator precedence is the rule that defines the order in which operators are evaluated in an expression.  
Operators with higher precedence are evaluated before those with lower precedence, which can change the result of the expression if not grouped properly.

# Additional Questions

## Java Data Types

1. **What is the size and range of each primitive data type in Java?**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size** | **Range** |
| byte | 1 byte(8-bit) | -128 to 127 |
| short | 2 bytes (16-bit) | -32,768 to 32,767 |
| int | 4 bytes (32-bit) | -2,14to 2,14 |
| long | 8 bytes (64-bit) | -9,223, to 9,223 |
| float | 4 bytes (32-bit) | ±3.4e−3 to ±3.4e+3 |
| double | 8 bytes (64-bit) | ±1.7e−3 to ±1.7e+3 |
| char | 2 bytes (16-bit) | 0 to 65,53 |
| boolean | 1 bit (logical) | true or false |

1. **How does Java handle overflow and underflow with numeric types?**

Java does not give an error for overflow or underflow.  
Instead, the value wraps around to the minimum or maximum value of that data type.

1. **Write a program to convert a double value to an int without data loss.**

public class DoubleToIntConversion {

public static void main(String[] args) {

double number = 25.0;

if (number % 1 == 0) {

int converted = (int) number;

System.out.println("Converted int value: " + converted);

} else {

System.out.println("Cannot convert without data loss.");

}

}

}

1. **What is the difference between char and String in Java?**
   * char is a **primitive data type** that stores a **single character** (e.g., 'A').
   * String is a **class** that stores a **sequence of characters** (e.g., "Hello").
   * char uses single quotes ' ', while String uses double quotes " ".
2. **Explain wrapper classes and their use in Java.**

**Wrapper classes** are used to convert primitive data types into objects (e.g., int → Integer, char → Character).  
They are useful when working with collections, generics, and methods that require objects instead of primitives.

## Java Control Statements

1. **Write a Java program using nested if statements.**

public class NestedIfExample {

public static void main(String[] args) {

int age = 20;

boolean hasID = true;

if (age >= 18) {

if (hasID) {

System.out.println("You are allowed to enter.");

} else {

System.out.println("You need an ID to enter.");

}

} else {

System.out.println("You are not allowed to enter.");

}

}

}

1. **Write a Java program to display the multiplication table of a number using a loop.**

import java.util.Scanner;

public class MultiplicationTable {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num = scanner.nextInt();

System.out.println("Multiplication table of " + num);

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

System.out.println(num + " x " + i + " = " (num \* i));

}

scanner.close();

}

}

1. **How do you exit from nested loops in Java?**

To exit a nested loop in Java, you can use the **break** statement.  
To exit multiple levels of nested loops at once, use a labeled break.

1. **Compare and contrast for, while, and do-while loops.**

* **for loop** is used when you know the number of repetitions in advance.
* **while loop** checks the condition before the loop runs and may not run at all.
* **do-while loop** runs at least once because it checks the condition after the loop body.

1. **Write a program that uses a switch-case to simulate a basic calculator.**

import java.util.Scanner;

public class Calculator {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number 1: ");

int a = sc.nextInt();

System.out.print("Enter number 2: ");

int b = sc.nextInt();

System.out.print("Enter operator (+, -, \*, /): ");

char op = sc.next().charAt(0);

switch (op) {

case '+':

System.out.println("Result:"+(a+b));

break;

case '-':

System.out.println("Result: " + (a - b));

break;

case '\*':

System.out.println("Result: "+(a\*b));

break;

case '/':

if (b != 0)

{

System.out.println("Result: " + (a / b));}

else System.out.println("Cannot divide by zero.");

break;

default:

System.out.println("Invalid operator.");

}

sc.close();

}

}

## Java Keywords and Operators

1. **What is the use of the `instanceof` keyword in Java?**

Instanceof keyword is used to check if an object belongs to a specific class or subclass.  
It returns true or false and helps avoid class cast errors during type conversion.

1. **Explain the difference between `==` and `.equals()` in Java.**

* **==** compares whether two references point to the same memory location (used for primitives and object references).
* **.equals()** compares the actual content or values of two objects (commonly used with String, etc.).

1. **Write a program using the ternary operator.**

import java.util.Scanner;

public class TernaryExample {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int num = sc.nextInt();

String result = (num % 2 == 0) ? "Even" : "Odd";

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

sc.close();

}

}

1. **What is the use of `this` and `super` in method overriding?**

* **this** refers to the **current class's object**,used to call **current class methods or variables**.
* **super** refers to the **parent class** and is used to call **superclass methods or constructors**.
* In method overriding, super is often used to call the **overridden method of the parent class**.
* **this** can be used if the method needs to refer to its own class version

1. **Explain bitwise operators with examples.**

* **AND (**&**)** r eturns 1 if both bits are 1.
* **OR (**|**)** returns 1 if at least one bit is 1.
* **XOR (**^**)** returns 1 if bits are different.
* **NOT (**~**)** Inverts all bits (1’s complement).
* **Left Shift (**<<**)** Shifts bits to the left, filling with 0s.
* **Right Shift (**>>**)** shifts bits to the right, preserving sign.

public class BitwiseDemo {

public static void main(String[] args) {

int a = 5, b = 3;

System.out.println("a & b="+(a & b));

System.out.println("a | b =" +(a|b));

System.out.println("a ^ b="+(a ^ b));

System.out.println("~a = " + (~a));

System.out.println("a <<1 ="+ (a << 1))

System.out.println("a >> 1=" +(a >>1))

}

}