# Java Programming Assignment

## Section 1: Java Data Types

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

Ans: Java has eight primitive data types:

* **byte**: 8-bit signed integer (-128 to 127)
* **short**: 16-bit signed integer (-32,768 to 32,767)
* **int**: 32-bit signed integer (-2^31 to 2^31-1)
* **long**: 64-bit signed integer (-2^63 to 2^63-1)
* **float**: 32-bit floating-point (IEEE 754)
* **double**: 64-bit floating-point (IEEE 754)
* **char**: 16-bit Unicode character (0 to 65,535)
* **boolean**: true or false

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

Ans:

* Primitive data types are predefined by Java, while non-primitive data types are defined by the programmer or Java libraries.
* Primitive data types have a fixed size, whereas non-primitive data types have a variable size depending on the object.
* Primitive data types are stored in stack memory, while non-primitive data types are stored in heap memory.
* Primitive data types have default values (like 0 or false), but non-primitive data types have a default value of **null**.
* Examples of primitive data types include **int**, **char**, and **boolean**, while examples of non-primitive data types include **String**, **Array**, **Class**, and **Interface**.
* Primitive data types do not have methods, but non-primitive data types can have methods.

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

Ans:

**Code:**

public class PrimitiveDataTypesDemo {

public static void main(String[] args) {

byte myByte = 100;

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

short myShort = 5000;

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

int myInt = 100000;

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

long myLong = 15000000000L;

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

float myFloat = 5.75f;

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

double myDouble = 19.99d;

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

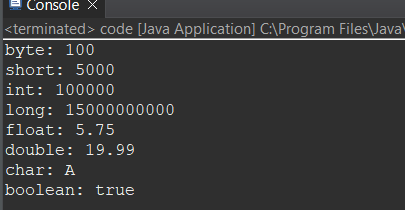
char myChar = 'A';

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

boolean myBoolean = true;

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

}

} Output: 

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

Ans:

**Type casting** is converting a value from one data type to another.

* **Implicit Casting (Widening)**: Automatically done by Java (smaller to larger type).
* **Explicit Casting (Narrowing)**: Requires manual casting (larger to smaller type).

For Example:

**Implicit casting (int to double)**

int numInt = 10;

double numDouble = numInt;

System.out.println("Implicit Casting: " + numDouble);

**Explicit casting (double to int)**

double anotherDouble = 9.78;

int anotherInt = (int) anotherDouble;

System.out.println("Explicit Casting: " + anotherInt);

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

Ans:

1. Byte = 0
2. Short = 0
3. Int = 0
4. Long = 0L
5. Float = 0.0f
6. Double = 0.0d
7. Char = '\u0000' (null)
8. Boolean = false

## Section 2: Java Control Statements

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

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

Types of control statements:

1. **Decision-making statements**:

* if statement
* if-else statement
* if-else-if ladder
* switch-case statement

1. **Loop statements**:

* for loop
* while loop
* do-while loop

1. **Jump statements**:

* break statement
* Continue statement
* Return statement

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

Ans: public class ControlDemo {

public static void main(String[] args) {

* **if-else example**

int num = 10;

if (num > 0) {

System.out.println("Number is positive");

} else if (num < 0) {

System.out.println("Number is negative");

} else {

System.out.println("Number is zero");

}

* **switch-case example**

char grade = 'B';

switch (grade) {

case 'A':

System.out.println("Excellent!");

break;

case 'B':

System.out.println("Good job!");

break;

case 'C':

System.out.println("Average");

break;

default:

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

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

Ans:

* **Break Statement:**

1. Terminates the loop or switch statement immediately
2. Control passes to the statement following the loop/switch
3. Example: if (i == 5) break;

* **Continue Statement**

1. Skips the current iteration and continues with the next iteration
2. Control passes to the next iteration of the loop
3. Example: if (i == 5) continue;

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

Ans: **Code:**

package Day\_2\_Assingment;

public class even {

public static void main(String[] args) {

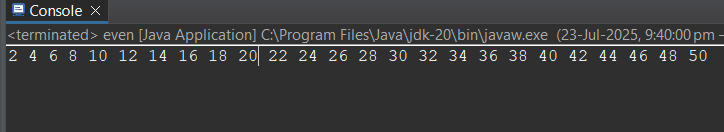
for (int i = 2; i <= 50; i += 2) {

System.out.println(i);

}

}

}

**Output:** 

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

Ans:

* **While Loop**

1. Checks condition before execution
2. May execute zero or more times
3. Syntax: while(condition) { ... }

* **Do While Loop**

1. Checks condition after execution
2. Executes at least once
3. Syntax: do { ... } while(condition);

## Section 3: Java Keywords and Operators

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

Ans: Keywords are reserved words in Java that have special meaning and cannot be used as identifiers (variable names, method names, class names, etc.). They are part of the Java language syntax.

**Java keywords:**

1. class
2. public
3. private
4. static
5. void
6. if
7. else
8. for
9. while
10. return

2. Explain the purpose of the following keywords: static, final, this, super.

Ans:

* static:
  + Used to create class-level variables and methods that belong to the class rather than any specific instance
  + Can be accessed without creating an object of the class
  + Only one copy exists regardless of how many objects are created
* final:
  + When applied to a variable: makes it constant
  + When applied to a method: prevents method overriding in subclasses
  + When applied to a class: prevents inheritance
* this:
  + Refers to the current object instance
  + Used to differentiate between instance variables and parameters with the same name
  + Can be used to call one constructor from another in the same class
* super:
  + Refers to the parent class
  + Used to call parent class constructors
  + Used to access parent class methods that are overridden in the child class

3. What are the types of operators in Java?

Ans:

Java has several types of operators:

1. **Arithmetic operators**: +, -, \*, /, %, ++, --
2. **Relational operators**: ==, !=, >, <, >=, <=
3. **Logical operators**: &&, ||, !
4. **Assignment operators**: =, +=, -=, \*=, /=, %=
5. **Bitwise operators**: &, |, ^, ~, <<, >>, >>>
6. **Ternary operator**: ? :
7. **Instanceof operator**: instanceof

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

Ans:

Code:

public class OperatorDemo { public static void main(String[] args) {

// Arithmetic operators

int a = 10, b = 5;

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

System.out.println("a + b = " + (a + b)); // 15

System.out.println("a - b = " + (a - b)); // 5

System.out.println("a \* b = " + (a \* b)); // 50

System.out.println("a / b = " + (a / b)); // 2

System.out.println("a % b = " + (a % b)); // 0

System.out.println("a++ = " + (a++)); // 10 (post-increment)

System.out.println("++b = " + (++b)); // 6(pre-increment)

// Relational operators  
 System.out.println("\nRelational Operators:");  
 System.out.println("a == b: " + (a == b)); // false  
 System.out.println("a != b: " + (a != b)); // true  
 System.out.println("a > b: " + (a > b)); // true  
 System.out.println("a < b: " + (a < b)); // false  
 System.out.println("a >= b: " + (a >= b)); // true  
 System.out.println("a <= b: " + (a <= b)); // false  
   
 // Logical operators  
 boolean x = true, y = false;  
 System.out.println("\nLogical Operators:");  
 System.out.println("x && y: " + (x && y)); // false  
 System.out.println("x || y: " + (x || y)); // true  
 System.out.println("!x: " + (!x)); // false  
   
 // Combining operators  
 System.out.println("\nCombined Expression:");  
 int result = (a + b) \* 2 / 3;  
 System.out.println("(a + b) \* 2 / 3 = " + result);

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

Ans: **Operator precedence** determines the order in which operators are evaluated in an expression when multiple operators are present. Operators with higher precedence are evaluated before operators with lower precedence.

**How it affects the outcome:**

* Expressions are evaluated according to precedence rules unless parentheses are used to override them
* Parentheses have the highest precedence and can be used to force evaluation order
* When operators have the same precedence, associativity rules determine the order (left-to-right or right-to-left)

**Common precedence order (highest to lowest):**

1. Postfix (expr++, expr--)
2. Unary (++expr, --expr, +expr, -expr, ~, !)
3. Multiplicative (\*, /, %)
4. Additive (+, -)
5. Shift (<<, >>, >>>)
6. Relational (<, >, <=, >=, instanceof)
7. Equality (==, !=)
8. Bitwise AND (&)
9. Bitwise XOR (^)
10. Bitwise OR (|)
11. Logical AND (&&)
12. Logical OR (||)
13. Ternary (? :)
14. Assignment (=, +=, -=, etc.)

# Additional Questions

## Java Data Types

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

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

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

9. What is the difference between char and String in Java?

10. Explain wrapper classes and their use in Java.

## Java Control Statements

6. Write a Java program using nested if statements.

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

8. How do you exit from nested loops in Java?

9. Compare and contrast for, while, and do-while loops.

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

## Java Keywords and Operators

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

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

8. Write a program using the ternary operator.

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

10. Explain bitwise operators with examples.