## **Section 1: Java Data Types**

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

Java has 8 primitive data types:

* byte
* short
* int
* long
* float
* double
* char
* boolean

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

Primitive data types are built into Java and store simple values directly in memory. They include byte, short, int, long, float, double, char, and boolean. Non-primitive data types are created by programmers and store references to objects in memory. Examples include String, Array, Class, and Interface.

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

class PrimitiveDataTypes {

public static void main(String[] args) {

byte b = 127;

short s = 32000;

int i = 100000;

long l = 9999999999L;

float f = 3.14f;

double d = 3.14159265359;

char c = 'A';

boolean bool = true;

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

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

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

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

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

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

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

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

}

}

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

Type casting is converting one data type to another. Implicit casting happens automatically when converting smaller to larger data types. Explicit casting requires manual conversion using parentheses.

class TypeCasting {

public static void main(String[] args) {

int i = 100;

long l = i;

double d = 9.78;

int x = (int) d;

System.out.println("Implicit casting: " + l);

System.out.println("Explicit casting: " + x);

}

}

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

* 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 control the flow of program execution. Types include:

Selection statements: if, if-else, switch Iteration statements: for, while, do-while Jump statements: break, continue, return

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

class ControlStatements {

public static void main(String[] args) {

int number = 85;

if (number >= 90) {

System.out.println("Grade A");

} else if (number >= 80) {

System.out.println("Grade B");

} else {

System.out.println("Grade C");

}

int day = 3;

switch (day) {

case 1:

System.out.println("Monday");

break;

case 2:

System.out.println("Tuesday");

break;

case 3:

System.out.println("Wednesday");

break;

default:

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

}

}

}

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

Break statement terminates the loop completely and transfers control to the statement after the loop. Continue statement skips the current iteration and moves to the next iteration of the loop.

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

class EvenNumbers {

public static void main(String[] args) {

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

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

}

}

}

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

While loop checks condition first then executes. Do-while loop executes first then checks condition.

class LoopDifference {

public static void main(String[] args) {

int i = 1;

while (i <= 3) {

System.out.println("While: " + i);

i++;

}

int j = 1;

do {

System.out.println("Do-while: " + j);

j++;

} while (j <= 3);

}

}

## **Section 3: Java Keywords and Operators**

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

Keywords are reserved words with predefined meanings in Java. Common keywords: class, public, static, void, int, if, else, for, while, return

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

Static: belongs to class rather than instance Final: prevents modification or inheritance This: refers to current object Super: refers to parent class

**3. What are the types of operators in Java?**

* Arithmetic operators: +, -, \*, /, %
* Relational operators: ==, !=, >, <, >=, <=
* Logical operators: &&, ||, !
* Assignment operators: =, +=, -=, \*=, /=
* Unary operators: ++, --, +, -
* Bitwise operators: &, |, ^, ~, <<, >>

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

class Operators {

public static void main(String[] args) {

int a = 10, b = 5;

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

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

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

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

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

boolean x = true, y = false;

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

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

}

}

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

Operator precedence determines the order of operations in expressions. Higher precedence operators are evaluated first. For example, multiplication has higher precedence than addition, so 2 + 3 \* 4 equals 14, not 20.

## **Additional Questions**

### **Java Data Types**

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

* byte: 1 byte, -128 to 127
* short: 2 bytes, -32,768 to 32,767
* int: 4 bytes, -2,147,483,648 to 2,147,483,647
* long: 8 bytes, -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
* float: 4 bytes, approximately ±3.40282347E+38F
* double: 8 bytes, approximately ±1.79769313486231570E+308
* char: 2 bytes, 0 to 65,535
* boolean: 1 bit, true or false

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

Java wraps around values when overflow or underflow occurs. For integers, the value wraps to the opposite end of the range. For floating-point numbers, the result may be positive or negative infinity.

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

class ConvertDouble {

public static void main(String[] args) {

double d = 123.45;

int wholePart = (int) d;

double fractionalPart = d - wholePart;

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

System.out.println("Whole part: " + wholePart);

System.out.println("Fractional part: " + fractionalPart);

}

}

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

Char is a primitive data type that stores a single character using 2 bytes. String is a non-primitive data type that stores a sequence of characters and is actually an object of the String class.

**10. Explain wrapper classes and their use in Java.**

Wrapper classes provide object equivalents of primitive data types. They include Byte, Short, Integer, Long, Float, Double, Character, and Boolean. They're used when objects are needed instead of primitives, such as in collections or when null values are required.

### **Java Control Statements**

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

class NestedIf {

public static void main(String[] args) {

int age = 25;

boolean hasLicense = true;

if (age >= 18) {

if (hasLicense) {

System.out.println("Can drive");

} else {

System.out.println("Cannot drive without license");

}

} else {

System.out.println("Too young to drive");

}

}

}

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

class MultiplicationTable {

public static void main(String[] args) {

int number = 5;

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

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

}

}

}

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

Use labeled break statement to exit from nested loops.

class ExitNestedLoop {

public static void main(String[] args) {

outer: for (int i = 1; i <= 3; i++) {

for (int j = 1; j <= 3; j++) {

if (i == 2 && j == 2) {

break outer;

}

System.out.println(i + " " + j);

}

}

}

}

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

For loop: best when number of iterations is known, has initialization, condition, and increment in one line While loop: best when number of iterations is unknown, checks condition before execution Do-while loop: executes at least once, checks condition after execution

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

class Calculator {

public static void main(String[] args) {

int num1 = 10, num2 = 5;

char operator = '+';

int result = 0;

switch (operator) {

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

case '\*':

result = num1 \* num2;

break;

case '/':

result = num1 / num2;

break;

default:

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

return;

}

System.out.println(num1 + " " + operator + " " + num2 + " = " + result);

}

}

### **Java Keywords and Operators**

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

The instanceof keyword tests whether an object is an instance of a specific class or interface. It returns true if the object is an instance, false otherwise.

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

== compares references for objects and values for primitives. .equals() compares the actual content or value of objects. For strings, == checks if both references point to the same object, while .equals() checks if the content is the same.

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

class TernaryOperator {

public static void main(String[] args) {

int a = 10, b = 20;

int max = (a > b) ? a : b;

System.out.println("Maximum: " + max);

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

System.out.println("Number is: " + result);

}

}

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

In method overriding, this refers to the current object and calls the overridden method in the current class. Super refers to the parent class and calls the original method from the parent class, allowing access to the overridden method.

**10. Explain bitwise operators with examples.**

class BitwiseOperators {

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));

}

}