Section 1: Java Data Types

Q. What are the different primitive data types available in Java?
A. Java has eight primitive data types: byte, short, int, long, float, double, char, and boolean. They store simple values like whole numbers, decimals, characters, and true/false.

Q. Explain the difference between primitive and non-primitive data types in Java.
A. Primitive types store simple values directly (for example int or char). Non-primitive types are objects or references (for example String, arrays, or classes). Primitive types have fixed size and are faster, non-primitive can have methods and can be null.

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

```
 class PrimitiveDemo {

   public static void main(String[] args) {

      byte b = 10;

      short s = 1000;

      int i = 50000;

      long l = 100000L;

      float f = 5.5f;

      double d = 10.123456;

      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);

   }

}
```

Output:

byte: 10

short: 1000

int: 50000

long: 100000

float: 5.5

double: 10.123456

char: A

boolean: true

Q. What is type casting? Provide an example of implicit and explicit casting in Java.
A. Type casting is converting a value from one data type to another. Implicit casting (widening) happens automatically when converting small to large type (for example int to double). Explicit casting (narrowing) is done manually when converting large to small type (for example double to int).

Example:

```
class CastingExample {
   public static void main(String[] args) {
      int a = 10;
      double b = a; // implicit casting (int -> double)

      double x = 9.78;
      int y = (int) x; // explicit casting (double -> int)

      System.out.println("Implicit casting: " + b);
      System.out.println("Explicit casting: " + y);
   }
}
```

Output:

Implicit casting: 10.0

Explicit casting: 9

Q. What is the default value of each primitive data type in Java?
A. byte = 0
short = 0
int = 0
long = 0L
float = 0.0f
double = 0.0d

char = '\u0000' (empty character)
boolean = false

Section 2: Java Control Statements

Q. What are control statements in Java? List the types with examples.
A. Control statements change the flow of the program. Types: decision-making (if, if-else, switch), looping (for, while, do-while), and jump statements (break, continue).

Q. Write a Java program to demonstrate the use of if-else and switch-case statements.
A .class IfElseSwitchExample {

```
  public static void main(String[] args) {
    int num = 5;
    if (num % 2 == 0) {
      System.out.println(num + " is even");
    } else {
      System.out.println(num + " is odd");
    }

    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("Invalid day");
    }
  }
}
```

Output

5 is odd

Wednesday

Q. What is the difference between break and continue statements?
A. break completely exits the loop. continue skips only the current iteration and continues with the next iteration of the loop.

Q. Write a Java program to print even numbers between 1 to 50 using a for loop.
A. class EvenNumbers {

```
  public static void main(String[] args) {
      for (int i = 1; i <= 50; i++) {
         if (i % 2 == 0) {
            System.out.println(i);
         }
      }
   }
}
```

Output

```
2
4
6
...
50
```

Q. Explain the differences between while and do-while loops with examples.
A. while checks the condition before running the loop body, so it may not run at all if the condition is false. do-while runs the loop body once first, then checks the condition, so it always runs at least once.

Example:

```
class WhileDoWhileExample {
   public static void main(String[] args) {
      int i = 0;
      while (i > 0) {
         System.out.println("While loop");
         i--;
      }

      int j = 0;
      do {
         System.out.println("Do-while loop");
         j--;
      } while (j > 0);
```

```
    }
}
```

Output:

Do-while loop

Section 3: Java Keywords and Operators

Q. What are keywords in Java? List 10 commonly used keywords.
A. Keywords are reserved words with special meaning in Java. Ten common keywords: class, public, static, void, int, double, if, else, for, return.

Q. Explain the purpose of the following keywords: static, final, this, super.
A. static marks class-level members (shared by all objects). final makes a variable constant or prevents method overriding or class extension. this refers to the current object. super refers to the parent class (used to access parent methods or constructors).

Q. What are the types of operators in Java?
A. Main types: arithmetic (+, -, *, /, %), relational (>, <, ==, !=), logical (&&, ||, !), assignment (=, +=), unary (++, --), bitwise (&, |, ^, ~, <<, >>), and ternary (? :).

Q. Write a Java program demonstrating the use of arithmetic, relational, and logical operators.
A.
```
class OperatorsDemo {
    public static void main(String[] args) {
    int a = 10, b = 5;
    System.out.println("Addition: " + (a + b));       // arithmetic
    System.out.println("a > b? " + (a > b));         // relational
    System.out.println("a>0 && b>0? " + (a > 0 && b > 0)); // logical
  }
}
```

Output:

Addition: 15

a > b? true

a>0 && b>0? true

Q. What is operator precedence? How does it affect the outcome of expressions?
A. Operator precedence is the order in which operators are evaluated. Operators with higher precedence run first (for example * and / before + and -). This changes results when multiple operators are in one expression.

Additional Questions — Java Data Types

Q. What is the size and range of each primitive data type in Java?
A. 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 (approx 6-7 decimal digits precision)
double: 8 bytes (approx 15-16 decimal digits precision)
char: 2 bytes, 0 to 65,535 (Unicode)
boolean: 1 bit (true or false)

Q. How does Java handle overflow and underflow with numeric types?
A. Java does not throw exceptions for numeric overflow/underflow. Values wrap around (e.g., int max + 1 becomes int min). For floating point, overflow may produce Infinity and underflow may produce 0.0 or very small numbers.

Q. Write a program to convert a double value to an int without data loss.
A. Exact no-loss conversion is only possible if the double has no fractional part and it fits in int range. Example using a value with no fractional part:

```
class DoubleToInt {

   public static void main(String[] args) {

      double d = 20.0; // no fractional part

      int i = (int) d;

      System.out.println(i);

   }

}
```

Output:

20

Q. What is the difference between char and String in Java?
A. char stores a single character and uses single quotes like 'A'. String stores a sequence of characters and uses double quotes like "Hello". String is an object with methods.

Q. Explain wrapper classes and their use in Java.
A. Wrapper classes are object versions of primitive types (for example Integer for int, Double for double). They let you use primitive values where objects are needed (for example in collections like ArrayList) and provide useful methods.

Additional Questions — Java Control Statements

Q. Write a Java program using nested if statements.
A.
```
class NestedIf {

   public static void main(String[] args) {

      int num = 10;

      if (num > 0) {

         if (num % 2 == 0) {

            System.out.println("Positive even");
```

```
        } else {
            System.out.println("Positive odd");
        }
    } else {
        System.out.println("Zero or Negative");
    }
  }
}
```

Output:

Positive even

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

```
class MultiplicationTable {
  public static void main(String[] args) {
    int num = 5;
    for (int i = 1; i <= 10; i++) {
      System.out.println(num + " x " + i + " = " + (num * i));
    }
  }
}
```

Output:

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

...

5 x 10 = 50

Q. How do you exit from nested loops in Java?
A. Use a labeled break to exit from outer loop directly.

Example:

```
class ExitNestedLoops {
  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);
            }
        }
    }
}
```

Output:

1 1

1 2

1 3

2 1

Q. Compare and contrast for, while, and do-while loops.
A. for loop is good when you know how many times to run. while checks the condition first and may not run at all. do-while runs the body once before checking the condition, so it always runs at least once.

Q. Write a program that uses a switch-case to simulate a basic calculator.
A. (Basic syllabus version using fixed numbers)

```
class BasicCalculator {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;
        char op = '+'; // change to '+', '-', '*', '/'
        int result = 0;

        switch (op) {
            case '+':
                result = a + b;
                break;
            case '-':
                result = a - b;
```

```
            break;
        case '*':
            result = a * b;
            break;
        case '/':
            result = a / b;
            break;
        default:
            System.out.println("Invalid operator");
            return;
      }
      System.out.println("Result: " + result);
   }
}
```

Output:

Result: 15

Additional Questions — Java Keywords and Operators

Q. What is the use of the instanceof keyword in Java?
A. instanceof checks whether an object is an instance of a particular class or interface. It returns true or false.

Q. Explain the difference between == and .equals() in Java.
A. == checks if two reference variables point to the same object (same memory address). .equals() checks logical equality (content) — for objects, .equals() can be overridden to compare values (for example String's .equals()).

Q. Write a program using the ternary operator.
A.
```
class TernaryExample {
   public static void main(String[] args) {
      int a = 5, b = 10;
      int max = (a > b) ? a : b;
      System.out.println("Max: " + max);
   }
}
```

Output:

Max: 10

Q. What is the use of this and super in method overriding?
A. In method overriding, this refers to the current class object (useful to access current object's fields/methods). super is used to call the parent class version of a method or the parent constructor.

Q. Explain bitwise operators with examples.
A. Bitwise operators work on bits of integer types. For example &: AND, |: OR, ^: XOR, ~: NOT, <<: left shift, >>: right shift.

Example:

```
class BitwiseExample {
    public static void main(String[] args) {
        int a = 5;  // binary 0101
        int b = 3;  // binary 0011
        System.out.println("a & b: " + (a & b)); // 0001 = 1
        System.out.println("a | b: " + (a | b)); // 0111 = 7
        System.out.println("a ^ b: " + (a ^ b)); // 0110 = 6
    }
}
```

Output:

```
a & b: 1
a | b: 7
a ^ b: 6
```