Nucleus Software Exports

💻 Round 3 – Java Coding & Output MCQs

Excellent ✅
Now you're entering Round 3 — Coding & Output Test, which Nucleus Software and similar IT firms (like TCS, Infosys, Persistent) use to check your Java coding logic, debugging, and output-prediction skills.

This section usually combines short code snippets, logic building, and algorithmic reasoning.

1.
int a = 5, b = 2;
System.out.println(a / b * b);

A) 4 B) 5 C) 2 D) 0

✅ Answer: A) 4
Explanation: a/b = 2 (integer division) → 2 × 2 = 4.

2.
System.out.println(10 + 20 + "Tech");
System.out.println("Tech" + 10 + 20);

A) 30Tech, Tech30
B) 30Tech, Tech1020
C) 1020Tech, Tech30
D) Error

✅ Answer: B) 30Tech and Tech1020
Explanation: + with String performs concatenation once a String appears.

3.

```
int x = 0;
for (int i = 1; i <= 5; i++) {
    x += i;
}
System.out.println(x);
```

A) 10 B) 15 C) 5 D) 20

✅ Answer: B) 15
Explanation: Sum = 1 + 2 + 3 + 4 + 5 = 15.

4.

```
String s1 = "Java";
String s2 = "Java";
String s3 = new String("Java");
System.out.println(s1 == s2);
System.out.println(s1 == s3);
```

✅ Answer:
true
false
Explanation: String literals share the same pool reference; new creates a new object.

5.

```
int arr[] = {1, 2, 3, 4, 5};
System.out.println(arr[2] + arr[4]);
```

A) 5 B) 7 C) 8 D) 9

✅ Answer: C) 8 (3 + 5)

6.

What will happen?

```
int a = 10;
System.out.println(++a * a++);
```

✅ Answer: 121
Explanation: ++a = 11, then a++ returns 11 but increments to 12 → 11 × 11 = 121.

7.

```
int a = 5;
System.out.println(a++ + ++a + a++);
```

✅ Answer: 20
Explanation: (5) + (7) + (7) = 19 – double-check → a sequence = 5→6, ++a=7, a++=7→8 → 5+7+7 = 19.

✅ Correct Answer: 19

8.

Output of:

```
for(int i=1;i<=3;i++){
    for(int j=1;j<=i;j++){
        System.out.print("*");
    }
    System.out.println();
}
```

✅ Answer:

```
*
**
```

***

Explanation: Nested loop prints incremental pattern.

9.

Predict output:

```
int a = 2;
switch(a) {
    case 1: System.out.print("One "); break;
    case 2: System.out.print("Two ");
    case 3: System.out.print("Three "); break;
    default: System.out.print("Default");
}
```

✅ **Answer: Two Three**
Explanation: No break after case 2 → fall-through.

10.

```
int i = 0;
while (i < 3) {
    System.out.print(i);
    i++;
}
```

✅ **Answer: 012**
Explanation: Prints i from 0 to 2.

11.

```
class Test {
    public static void main(String[] args) {
        System.out.println(fun(5));
    }
```

```
    static int fun(int n){
        if(n==1) return 1;
        return n * fun(n-1);
    }
}
```

✅ **Answer: 120**
Explanation: Recursive factorial of 5.

12.

```
class Parent {
    void show() { System.out.println("Parent"); }
}
class Child extends Parent {
    void show() { System.out.println("Child"); }
}
public class Main {
    public static void main(String[] args) {
        Parent p = new Child();
        p.show();
    }
}
```

✅ **Answer: Child**
Explanation: Runtime polymorphism → method overriding.

13.

What will be output?

```
try {
   int a = 10/0;
   System.out.println("A");
} catch(Exception e){
   System.out.println("B");
} finally {
```

```
   System.out.println("C");
}
```

✅ **Answer: B C**
Explanation: Exception caught → catch then finally executes.

14.

```
List<Integer> list = new ArrayList<>();
list.add(1);
list.add(2);
list.add(3);
for(Integer i : list)
    if(i == 2) list.remove(i);
System.out.println(list);
```

A) [1, 3] B) [1, 2, 3] C) ConcurrentModificationException D) [1, 3, 2]

✅ **Answer: C) ConcurrentModificationException**
Explanation: Can't modify list while iterating with enhanced for loop.

15.

Small debugging logic — find correct output:

```
String s = null;
try {
   System.out.println(s.length());
} catch(NullPointerException e){
   System.out.println("Null Error");
}
```

✅ **Answer: Null Error**
Explanation: NullPointerException caught.

⚙ Round 3 Summary

Category	No.	Focus
Core Java Logic + Operators	5	Increment/decrement, loops
OOP (Polymorphism + Inheritance)	2	Overriding, runtime binding
Strings & Collections	3	Mutability, for-each issues
Exception Handling	2	try-catch-finally
Pattern / Logic Building	3	Nested loops, recursion

💼 Round 4 – Technical Interview (Concept + Coding + Database)

Perfect ✅
Now you're entering Round 4 – Technical Interview (Face-to-Face / Virtual) for Nucleus Software Exports – Assistant Software Engineer.

This round tests conceptual clarity, real-world problem-solving, and your approach to clean, optimized Java code — typically a mix of Java + OOP + DBMS + Frameworks (Spring, Hibernate).

🧠 Core Java & OOPS (1–10)

---

1. What are the four main OOPS concepts in Java?

✅ **Answer:**

- **Encapsulation**
- **Inheritance**
- **Polymorphism**
- **Abstraction**

---

2. Explain the difference between Overloading and Overriding.

✅ **Answer:**

- **Overloading: Same method name, different parameters (compile-time polymorphism).**
- **Overriding: Same method name and parameters in subclass (run-time polymorphism).**

---

3. What is Encapsulation?

✅ **Answer:**
Wrapping data (variables) and methods into a single unit (class) and restricting direct access using private fields + getters/setters.

---

4. Difference between == and .equals() in Java.

✅ **Answer:**

- **== compares references.**
- **.equals() compares content.**

---

5. What is an abstract class and how is it different from an interface?

✅ **Answer:**

- **Abstract class can have both abstract and concrete methods.**
- **Interface contains only abstract (and default/static) methods.**
- **A class can extend only one abstract class but implement multiple interfaces.**

---

6. Explain final, finally, and finalize().

✅ **Answer:**

- **final: keyword (variable/method/class can't be changed/overridden).**
- **finally: block that executes always after try-catch.**
- **finalize(): method called by garbage collector before object destruction.**

---

7. What is the difference between Stack and Heap memory?

✅ **Answer:**

- **Stack: stores local variables and function calls.**
- **Heap: stores objects created with new.**

---

8. Explain the lifecycle of a Java object.

✅ **Answer:**
Creation → Usage → Dereferencing → Garbage Collection.

---

9. What are access modifiers in Java?

✅ **Answer:**
public, private, protected, (default) — define visibility and accessibility.

---

10. What is the difference between ArrayList and LinkedList?

✅ **Answer:**

- **ArrayList: better for searching, random access.**
- **LinkedList: better for insertion/deletion.**

⚙ Java Advanced / Collections / Threads (11–15)

---

11. What is the difference between HashMap and Hashtable?

✅ **Answer:**

- **HashMap is non-synchronized, allows null keys.**
- **Hashtable is synchronized, no null keys.**

---

12. Explain synchronization in Java.

✅ **Answer:**
Synchronization ensures that only one thread accesses a shared resource at a time — using `synchronized` keyword.

---

13. What is the difference between process and thread?

✅ **Answer:**

- **Process = independent program in execution.**
- **Thread = lightweight subprocess inside a process.**

---

14. What is the purpose of `volatile` keyword?

✅ **Answer:**
It ensures that a variable's value is always read from main memory, not from a thread's local cache.

---

15. How does garbage collection work in Java?

✅ **Answer:**
GC automatically reclaims memory of unreachable objects from the heap.

❄️ Spring Framework & Hibernate (16–20)

---

16. What is Spring Framework?

✅ **Answer:**
A lightweight Java framework that provides Dependency Injection (DI) and Inversion of Control (IoC) to build enterprise applications.

---

17. Difference between `@Controller` and `@RestController` in Spring.

✅ **Answer:**

- **`@Controller` → returns views (HTML/JSP).**
- **`@RestController` → returns data (JSON/XML).**

---

18. What is Dependency Injection?

✅ **Answer:**
A design pattern where Spring automatically provides the required class objects (dependencies) instead of creating them manually.

---

19. What are Spring Boot starters?

✅ **Answer:**
Pre-configured templates that simplify adding dependencies (e.g., `spring-boot-starter-web`, `spring-boot-starter-data-jpa`).

---

20. What is ORM in Hibernate?

✅ **Answer:**
Object-Relational Mapping — it maps Java objects to database tables to avoid manual SQL handling.

🛢️ Database / SQL / PL-SQL (21–25)

---

21. Difference between DELETE, TRUNCATE, and DROP.

✅ **Answer:**

- **DELETE: removes selected rows, can use WHERE.**
- **TRUNCATE: removes all rows, keeps structure.**
- **DROP: removes table structure entirely.**

---

22. What is a Primary Key?

✅ **Answer:**
A column (or set) that uniquely identifies each row; cannot be NULL.

---

23. What is a Foreign Key?

✅ **Answer:**
A column that references the primary key of another table — establishes a relationship between tables.

---

24. Write a query to get 2nd highest salary from employee table.

✅ **Answer:**

```
SELECT MAX(salary)
```

```
FROM employee
WHERE salary < (SELECT MAX(salary) FROM employee);
```

---

25. Difference between INNER JOIN and LEFT JOIN.

✅ **Answer:**

- **INNER JOIN: returns only matching rows.**
- **LEFT JOIN: returns all from left table + matched from right.**

💬 Practical / Debug / Scenario Questions (26–30)

---

26. If you find a NullPointerException in your code, what steps will you take?

✅ **Answer:**

- **Identify which variable is null.**
- **Add null checks or optional wrappers.**
- **Ensure proper object initialization.**

---

27. How do you improve performance in Java applications?

✅ **Answer:**

- **Use efficient data structures.**
- **Minimize object creation.**
- **Use StringBuilder for concatenation.**
- **Enable caching, batch DB operations.**

---

28. How does JDBC connect Java with a database?

✅ **Answer:**
Through DriverManager, Connection, Statement, and ResultSet interfaces.

---

29. What happens when you call `new Thread().start()` vs `new Thread().run()`?

✅ **Answer:**

- **`start()` → starts a new thread.**
- **`run()` → executes in the current thread (no multithreading).**

---

30. What is the difference between compile-time and runtime polymorphism?

✅ **Answer:**

- **Compile-time: Method overloading.**
- **Runtime: Method overriding.**

🎒 Round 4 – Quick Recap

Section	Topics	Difficulty	Focus
Core Java & OOP	10 Qs	Easy–Medium	Fundamentals
Advanced Java / Threads	5 Qs	Medium	Multithreading & Collections
Spring / Hibernate	5 Qs	Medium	Annotations, DI
SQL / DBMS	5 Qs	Medium	Queries, Keys, Joins

Section	Topics	Difficulty	Focus
Debug / Real-World	**5 Qs**	**Medium**	**Problem solving & code thinking**