Jr. Developer Technical Test - Practice Questions & Step-by-step Solutions

Core Java - Solutions

1. OOP Principles (with short examples)

• Encapsulation: Grouping data and methods in a class and restricting direct access. Example: private fields + public getters/setters.
• Inheritance: A class (subclass) acquires properties of another (superclass). Example: class Manager extends Employee.
• Polymorphism: Same method behaves differently. Example: method overloading and overriding.
• Abstraction: Hiding complex implementation behind a simple interface or abstract class.

2. Interface vs Abstract Class

Interface: All methods (prior to Java 8 default methods aside) are abstract; a class can implement multiple interfaces. Abstract class: can have implemented methods and fields; single inheritance.

3. Exception handling (try/catch/finally, throw vs throws)

Use try-catch to handle exceptions. finally executes always. 'throw' throws an exception instance; 'throws' declares that a method may throw exceptions.

4. Collections - ArrayList vs LinkedList

ArrayList: backed by array — fast random access (O(1)), slower inserts/removals in middle (O(n)). LinkedList: fast insert/remove at ends but slower random access (O(n)).
HashMap vs HashSet: HashMap stores key-value pairs; HashSet stores unique values and uses a HashMap internally.

5. String immutability / StringBuilder / StringBuffer

String is immutable. StringBuilder is mutable and not synchronized (fast). StringBuffer is synchronized (thread-safe, slower). Use StringBuilder for repeated modifications in single-threaded code.

6. JVM vs JDK vs JRE

JDK: Development kit (compiler, tools). JRE: Runtime environment (JVM + class libraries). JVM: Java Virtual Machine — runs bytecode.

7. Stack vs Heap

Stack: stores method call frames and local primitive variables (LIFO). Heap: runtime memory for objects (shared, GC-managed).

8. Java: Reverse a string (no built-ins)

Solution (step-by-step): convert string to char array and swap characters from ends moving inward.

```
public String reverse(String s) {
    char[] a = s.toCharArray();
    int i = 0, j = a.length - 1;
    while (i < j) {
        char tmp = a[i];
        a[i] = a[j];
        a[j] = tmp;
        i++; j--;
    }
    return new String(a);
}
```

9. Java: Factorial (recursive & iterative)

```
// Recursive
long fact(int n) {
    if (n <= 1) return 1;
    return n * fact(n-1);
}
// Iterative
long factIter(int n) {
    long f = 1;
    for (int i = 2; i <= n; i++) f *= i;
    return f;
}
```

10. Java: Check prime (efficient)

```
boolean isPrime(int n) {
    if (n <= 1) return false;
    if (n <= 3) return true;
    if (n % 2 == 0) return false;
    for (int i = 3; i * i <= n; i += 2) if (n % i == 0) return false;
    return true;
}
```

11. Java: Palindrome check (string)

```
boolean isPalindrome(String s) {
    int i = 0, j = s.length() - 1;
    while (i < j) {
        if (s.charAt(i++) != s.charAt(j--)) return false;
    }
    return true;
}
```

12. Java: Fibonacci series (iterative)

```
void fib(int n) {
    int a = 0, b = 1;
    for (int i = 0; i < n; i++) {
        System.out.print(a + " ");
        int t = a + b;
        a = b;
        b = t;
    }
}
```

C Language - Solutions

1. malloc vs calloc
malloc(size) allocates memory but contents are uninitialized. calloc(n, size) allocates and initializes memory to zero.

2. struct vs union
struct: all members have their own memory; size = sum of members (plus padding). union: all members share same memory; size = size of largest member. Use union when fields are mutually exclusive.

3. Storage classes
auto: default for local variables. static: preserves value across calls; has internal linkage for file-level static. extern: declares variable defined elsewhere. register: hints storing in CPU register (mostly ignored by modern compilers).

4. Pointers basics
Pointer to pointer: int **p; Pointer arithmetic: incrementing int* moves by sizeof(int). Always ensure pointers are initialized before dereference.

SQL - Solutions

1. DELETE vs TRUNCATE vs DROP

DELETE: removes rows (can use WHERE), is transactional and can be rolled back. TRUNCATE: removes all rows, faster, usually non-transactional. DROP: removes the table definition and data.

2. Primary Key vs Foreign Key

Primary Key: unique identifier for table rows. Foreign Key: column that references PK of another table to enforce referential integrity.

3. Second-highest salary (simple SQL)

Method 1 (using subquery):

```
SELECT MAX(salary) AS SecondHighest
FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);
```

Method 2 (using window functions):

```
SELECT salary FROM (
  SELECT salary, DENSE_RANK() OVER (ORDER BY salary DESC) r
  FROM employees
) t WHERE r = 2;
```

4. Find duplicate records

```
SELECT col1, col2, COUNT(*)
FROM my_table
GROUP BY col1, col2
HAVING COUNT(*) > 1;
```

5. Count employees per department where count > 5

```
SELECT department, COUNT(*) cnt
FROM employees
GROUP BY department
HAVING COUNT(*) > 5;
```

6. JOIN example

```
SELECT e.name, d.name AS dept_name
FROM employees e
JOIN departments d ON e.dept_id = d.id;
```

HTML / CSS / JavaScript - Solutions

HTML/CSS

1. Inline, block, inline-block: Inline doesn't start new line (e.g., span), block starts new line and stretches full width (e.g., div), inline-block behaves like inline but can have width/height.

2. Semantic HTML tags: header, nav, main, article, section, footer, aside — used to improve structure and accessibility.

3. CSS specificity: order of precedence — inline styles > IDs > classes/attributes/pseudo-classes > element selectors.

4. CSS position types:
• static: default. • relative: relative to its normal position. • absolute: positioned relative to nearest positioned ancestor. • fixed: relative to viewport. • sticky: toggles between relative and fixed depending on scroll.

JavaScript

1. var vs let vs const: var is function-scoped and hoisted; let/const are block-scoped. const cannot be reassigned (object contents can change).
2. == vs ===: == compares after type coercion; === strict equality (no coercion).
3. Hoisting: Declarations (var, function) are hoisted. var initialized to undefined at runtime before execution; let/const are hoisted but in temporal dead zone until declaration.
4. Arrow functions vs normal: Arrow functions do not have their own 'this' (lexical this). They cannot be used as constructors.
5. Closure (simple explanation): A closure is when an inner function remembers variables from its outer scope even after the outer function has finished. Example:

```
function outer() {
    let x = 5;
    return function() { return x + 1; };
}
```

JS Coding: Reverse array

```
function reverseArr(a) {
    let i = 0, j = a.length - 1;
    while (i < j) {
        const tmp = a[i]; a[i] = a[j]; a[j] = tmp;
        i++; j--;
    }
    return a;
}
```

JS Palindrome

```
function isPalindrome(s) {
    s = s.replace(/\W/g,'').toLowerCase();
    return s === s.split('').reverse().join('');
}
```

JS Factorial

```
function fact(n) {
    let f = 1;
    for (let i = 2; i <= n; i++) f *= i;
    return f;
}
```

React - Solutions

1. Components (class vs functional): Class components use lifecycle methods and state (older). Functional components use hooks (useState, useEffect) and are preferred now.

2. JSX: JavaScript syntax extension that looks like HTML and compiles to React.createElement calls.

3. Props vs State: Props are read-only inputs passed from parent; state is internal and changeable via setState/useState.

4. useState & useEffect (example)

```
function Counter() {
  const [count, setCount] = React.useState(0);
  React.useEffect(() => { document.title = `Count: ${count}`; }, [count]);
  return <button onClick={() => setCount(c => c + 1)}>{count}</button>;
}
```

5. Controlled vs Uncontrolled: Controlled components use React state to manage form inputs. Uncontrolled uses refs to access DOM directly.

Logical Reasoning / Aptitude - Solutions

Q1: Sequence: 2, 6, 12, 20, ?
Step 1: Recognize pattern: these equal n*(n+1) for n=1,2,3,4: 1*2=2, 2*3=6, 3*4=12, 4*5=20. Step 2: Next is 5*6 = 30.
Answer: 30

Q2: Cats & Mice (Given earlier)
Given: 3 cats catch 3 mice in 3 minutes. => One cat catches 1 mouse in 3 minutes => rate per cat = 1/3 mouse per minute. Need: 100 mice in 100 minutes => required rate = 1 mouse per minute. Let x cats: x * (1/3) = 1 => x = 3.
Answer: 3 cats

Q3: Train 180 m crosses a pole in 6 sec, find speed
Step 1: speed = distance/time = 180/6 = 30 m/s. Step 2: convert to km/h: multiply by 18/5 => 30 * 18/5 = 108 km/h.
Answer: 30 m/s = 108 km/h

Q4: Blood relation example
Example puzzle: 'A is B's brother. B is C's sister.' How is A related to C? Step 1: B is sibling to both A and C (since B is C's sister). Therefore A and C are siblings. A is C's brother (assuming A is male).
Answer: A is C's brother

Q5: Probability - tossing 2 coins, probability of at least one head
Sample space: {HH, HT, TH, TT}. Only TT has no head. So probability(at least one head) = 3/4 = 0.75.
Answer: 3/4

Q6: Percentage: student scored 180 out of 300
Percentage = (180/300)*100 = 60%

Quick Tips for the Test

• Revise Core Java programs: string operations, arrays, recursion, OOP definitions. • Practice SQL queries (GROUP BY, JOINs, subqueries). • Practice JS basics: closures, hoisting, DOM basics if asked. • For aptitude, practice 10-15 quick puzzles: sequences, rates, time-speed-distance, percentages, basic probability. • In GD/interview: communicate clearly, show thought process, and be honest about what you don't know.