Technical Interview Cheat Sheet – Q1 to Q10

1. What is Object-Oriented Programming (OOP)?

Answer:

OOP is a programming paradigm centered around objects and classes. Its 4 main pillars are:

Encapsulation → Wrapping data and methods together. *(E.g., a `BankAccount` class with `balance` and `deposit()` methods)*.

Inheritance → Reusing parent class features. *(E.g., `Car` inherits from `Vehicle`)*.

Polymorphism → Same method behaves differently. *(E.g., `draw()` method in `Circle` vs `Square`)*.

Abstraction → Hiding unnecessary details. *(E.g., we use `car.drive()` without knowing how the engine works internally)*.

Real-world analogy: A car blueprint defines general properties (class). Each car produced is an object, with brand-specific variations.

---

2. Difference between Process and Thread?

Answer:

Process: Independent execution unit with its own memory. Heavier to create.

Thread: Lightweight execution unit inside a process, shares memory.

Example: Chrome browser runs each tab as a separate process. Inside each tab, multiple threads handle rendering, JavaScript, etc.

Feature	Process	Thread
Memory	Separate	Shared
Creation	Expensive	Cheaper
Isolation	High	Low

---

3. What is Normalization in DBMS?

Answer:

Normalization is structuring databases to minimize redundancy and ensure data integrity.

Example:

❌ Without Normalization:

```
Student(ID, Name, CourseName, InstructorName)
```

If an instructor changes name, you must update every row.

✅ With 3rd Normal Form (3NF):

Students(ID, Name, CourseID)

Courses(CourseID, CourseName, InstructorID)

Instructors(InstructorID, InstructorName)

This prevents anomalies and keeps data consistent.

---

4. Explain the OSI Model.

Answer:

The OSI model has **7 layers**:

Physical → Hardware, cables, signals

Data Link → MAC addresses, switches, error detection

Network → IP addressing, routing

Transport → TCP/UDP, error recovery

Session → Manages connections

Presentation → Data encryption, compression, translation

Application → User-facing apps (HTTP, FTP, SMTP)

Real-world analogy: Sending a letter → Writing, packaging, addressing, delivering, opening.

---

5. What is Recursion?

Answer:

Recursion is when a function calls itself until a base condition is met.

Example in Python:

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
```

```
    return n * factorial(n - 1)

print(factorial(5))  # 120
```

Real-world analogy: A set of nested boxes, each box contains a smaller one until the last (base case).

---

6. Difference between Primary Key and Foreign Key?

Answer:

Primary Key: Unique identifier for a record.

Foreign Key: References primary key in another table.

Example:

```
Customers(CustomerID, Name)
Orders(OrderID, CustomerID, Amount)
```

Here, `CustomerID` in Orders is a foreign key referencing Customers.

---

7. What is Deadlock in Operating Systems?

Answer:

Deadlock occurs when two or more processes wait indefinitely for each other's resources.

Conditions for deadlock (Coffman's):

Mutual exclusion

Hold and wait

No preemption

Circular wait

Example: Process A holds Printer, needs Scanner; Process B holds Scanner, needs Printer → Deadlock.

---

8. What is a REST API?

Answer:

REST (Representational State Transfer) is an architecture style for web services using HTTP.

Stateless: Each request is independent.

Resources: Identified by URLs.

Methods: GET, POST, PUT, DELETE.

Example:

`GET /users/1` → Fetch user with ID 1.

`POST /users` → Create new user.

---

9. What is the difference between SQL and NoSQL databases?

Answer:

SQL → Relational, structured, tables, fixed schema (MySQL, PostgreSQL).

NoSQL → Non-relational, flexible schema, supports JSON-like docs, key-value, graph (MongoDB, Cassandra).

Example:

SQL → Bank systems where transactions must be consistent.

NoSQL → Social media feeds with high scalability and flexibility.

---

10. What is Multithreading?

Answer:

Multithreading is running multiple threads concurrently within a single process.

Example in Java:

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running");
    }
}
public class Main {
    public static void main(String args[]) {
        MyThread t1 = new MyThread();
        t1.start();
    }
}
```

Use case: A web server handling multiple client requests simultaneously.

Technical Interview Cheat Sheet – Q11 to Q20

11. What is the difference between Compilation and Interpretation?

Answer:

Compilation → Converts entire source code into machine code before execution (C, C++).

Interpretation → Translates and runs line by line (Python, JavaScript).

Example:

Compilation: Like translating a whole book before reading.

Interpretation: Like translating each sentence while reading aloud.

---

12. What is an Index in SQL? Why is it used?

Answer:

An **index** improves query performance by speeding up search operations, like a book index.

Without index → Database scans all rows.

With index → Database jumps directly to relevant rows.

Example:

```
CREATE INDEX idx_customer_name ON Customers(Name);
```

Now searching by name becomes much faster.

⚠️ Downside: Indexes take extra space and slow down inserts/updates.

---

13. What is the difference between TCP and UDP?

Answer:

Feature	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Reliability	Reliable (acknowledgements, retransmissions)	Unreliable (no guarantees)
Speed	Slower	Faster
Connection	Connection-oriented	Connectionless
Use Cases	Web, Email, File Transfer	Gaming, Streaming, VoIP

Example:

TCP → Sending an email (must arrive correctly).

UDP → Online video call (better to skip a lost packet than delay).

---

14. What is Big-O Notation?

Answer:

Big-O describes algorithm efficiency in terms of input size **n**.

Common complexities:

O(1) → Constant (array access)

O(log n) → Logarithmic (binary search)

O(n) → Linear (loop through array)

O(n log n) → Merge sort

O(n²) → Nested loops

Example: Searching a phone number:

Scanning page by page → O(n).

Using alphabetical index → O(log n).

---

15. What is a JOIN in SQL?

Answer:

JOIN combines rows from multiple tables based on a condition.

Types:

INNER JOIN → Matches common records.

LEFT JOIN → All records from left + matching right.

RIGHT JOIN → All from right + matching left.

FULL JOIN → All records from both tables.

Example:

```
SELECT Customers.Name, Orders.Amount
FROM Customers
INNER JOIN Orders ON Customers.ID = Orders.CustomerID;
```

This gives customers who placed orders.

---

16. Explain the concept of Virtualization.

Answer:

Virtualization creates **virtual versions of servers, storage, or networks** using software.

Benefits: Efficient resource use, scalability, cost reduction.

Example:

Instead of running 10 physical servers, you run 10 **virtual machines (VMs)** on 1 server using VMware or VirtualBox.

Cloud providers (AWS, Azure) heavily rely on virtualization.

---

17. What is Load Balancing?

Answer:

Load balancing distributes network or application traffic across multiple servers.

Ensures no server is overloaded.

Increases reliability and uptime.

Example:

When you visit Amazon.com → Load balancer directs you to the nearest or least busy server.

Types:

Round Robin → Sequential distribution.

Least Connections → Server with fewest connections.

IP Hash → Based on client IP.

---

18. Difference between Stack and Queue?

Answer:

Stack → LIFO (Last In, First Out). Example: Undo feature in Word.

Queue → FIFO (First In, First Out). Example: Printer job queue.

Example in Python:

```
# Stack
stack = []
stack.append(1)
stack.append(2)
stack.pop()  # 2

# Queue
from collections import deque
queue = deque([1, 2])
queue.append(3)
queue.popleft()  # 1
```

---

19. What is a Hash Table?

Answer:

A **hash table** stores data as key-value pairs, using a hash function to map keys to indexes.

Fast lookups → O(1) average case.

Example:

Python dictionary →

```
user = {"id": 101, "name": "Alice"}
print(user["name"])  # Alice
```

Real-world analogy: A librarian uses an **index card system** to find a book instantly instead of scanning every shelf.

---

20. What is the difference between Monolithic and Microservices Architecture?

Answer:

Monolithic → Entire application is one unit (tightly coupled).

Microservices → Application broken into small independent services communicating via APIs.

Example:

Monolithic → An e-commerce app with payments, products, and users in one big codebase.

Microservices → Payments, products, and users are **separate services**, can be deployed/scaled independently.

⚡ Companies like Netflix & Amazon shifted from monolithic to microservices for scalability.

Technical Interview Cheat Sheet – Q21 to Q30

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

Answer:

Stack Memory → Stores local variables & function calls. Managed automatically (push/pop). Faster.

Heap Memory → Stores objects & dynamic memory allocation. Managed manually (malloc/free in C, garbage collector in Java/Python).

Example:

```
int x = 10;   // Stored in stack
int *p = malloc(sizeof(int)); // Stored in heap
```

Real-world analogy: Stack is like a **stack of plates** (LIFO), heap is like a **playground** where you can allocate/free space.

---

22. What is Cloud Computing?

Answer:

Cloud computing delivers computing services (servers, storage, databases, networking) over the internet.

Types of services:

IaaS → Infrastructure as a Service (AWS EC2).

PaaS → Platform as a Service (Google App Engine).

SaaS → Software as a Service (Gmail, Dropbox).

Benefits: Scalability, cost savings, accessibility.

Example: Instead of buying servers, startups deploy apps on AWS or Azure.

---

23. What is ACID in databases?

Answer:

ACID → Properties ensuring reliable transactions.

Atomicity → All or nothing.

Consistency → Database must stay valid.

Isolation → Transactions don't interfere.

Durability → Data is permanent after commit.

Example:

Transferring money → Deduct ₹100 from A and add ₹100 to B. Both must succeed or both fail.

---

24. Explain Garbage Collection in Java/Python.

Answer:

Garbage Collection (GC) automatically frees memory occupied by unused objects.

Java: Uses JVM's Garbage Collector (Mark & Sweep, Generational GC).

Python: Uses reference counting + cyclic garbage collector.

Example:

```
a = [1,2,3]
b = a
del a
# b still points to list, so GC won't remove it yet
```

Benefit: Prevents memory leaks.

---

25. What is DNS and how does it work?

Answer:

DNS = **Domain Name System**, converts human-readable names → IP addresses.

Steps:

User enters `www.google.com`.

Browser checks cache.

If not found → DNS resolver queries root server → TLD server → Authoritative server.

Returns IP address (e.g., `142.250.190.78`).

Real-world analogy: DNS is like a **phonebook** that translates names into numbers.

---

26. Explain CAP Theorem in Distributed Systems.

Answer:

CAP theorem says a distributed system can only guarantee **2 out of 3**:

Consistency (C): All nodes see the same data.

Availability (A): Every request gets a response.

Partition Tolerance (P): System continues working despite network issues.

Examples:

MongoDB → CP system.

Cassandra → AP system.

---

27. What is the difference between IPv4 and IPv6?

Answer:

Feature	IPv4	IPv6
Address size	32-bit (4 bytes)	128-bit (16 bytes)
Format	Dotted decimal (`192.168.0.1`)	Hexadecimal (`2001:0db8::1`)
Addresses	~4.3 billion	~340 undecillion

Reason for IPv6: IPv4 exhaustion due to internet growth.

---

28. What is Agile methodology?

Answer:

Agile is an iterative approach to software development emphasizing flexibility and customer feedback.

Principles: Individuals over tools, working software, customer collaboration, responding to change.

Frameworks: Scrum, Kanban, XP.

Example:

Scrum team works in **2-week sprints** → Continuous delivery of small increments.

---

29. What is Middleware?

Answer:

Middleware = Software layer between OS & applications enabling communication.

Examples:

Database middleware (ODBC, JDBC).

Message-oriented middleware (RabbitMQ, Kafka).

Web middleware (Express.js).

Real-world analogy: Middleware is like a **translator** between two people speaking different languages.

---

30. What is the difference between Overloading and Overriding?

Answer:

Overloading (Compile-time polymorphism): Same method name, different parameters.

Overriding (Run-time polymorphism): Subclass provides a new definition of parent method.

Example in Java:

```
class Calculator {
    int add(int a, int b) { return a + b; }     // Overloading
    double add(double a, double b) { return a + b; }
}

class Animal {
    void sound() { System.out.println("Animal sound"); }
}

class Dog extends Animal {
    void sound() { System.out.println("Bark"); }  // Overriding
}
```

Technical Interview Cheat Sheet – Q31 to Q40

31. What is a Firewall?

Answer:

A **firewall** is a network security system that monitors and filters incoming/outgoing traffic based on rules.

Types:

Packet filtering → Filters packets by IP, port, protocol.

Stateful inspection → Tracks connections.

Proxy firewall → Intermediary for requests.

Next-gen firewalls → Deep packet inspection, intrusion detection.

Example: A company firewall blocks traffic on port 21 (FTP) to prevent unauthorized file transfers.

---

32. What is Inheritance in OOP?

Answer:

Inheritance allows a class (child) to acquire properties and behaviors from another class (parent).

Benefits: Code reuse, hierarchy, extendibility.

Example in Java:

```
class Vehicle {
    void drive() { System.out.println("Vehicle is moving"); }
}
class Car extends Vehicle {
    void honk() { System.out.println("Beep!"); }
}
```

Here, `Car` inherits `drive()` from `Vehicle`.

---

33. Difference between Static and Dynamic Typing?

Answer:

Static Typing → Type checked at compile time (C, Java). Errors found early.

Dynamic Typing → Type checked at runtime (Python, JavaScript). Flexible but error-prone.

Example:

Java → `int x = 10;` (must declare type).

Python → `x = 10` (type inferred at runtime).

---

34. What is Docker?

Answer:

Docker is a platform that uses **containers** to run applications in isolated environments.

Benefits: Lightweight, portable, consistent across environments.

Example:

Instead of "works on my machine" issue, Docker ensures the same container runs on dev, test, and production.

```
docker run -d -p 8080:80 nginx
```

Runs NGINX server inside a container.

---

35. Explain Public Key and Private Key in Cryptography.

Answer:

Asymmetric encryption uses two keys:

Public Key → Shared openly to encrypt data.

Private Key → Kept secret, used to decrypt.

Example:

If Alice wants to send Bob a message:

Alice encrypts with Bob's **public key**.

Only Bob can decrypt with his **private key**.

Used in HTTPS, digital signatures, SSH.

---

36. What is a Semaphore in OS?

Answer:

Semaphore = Synchronization tool used to manage access to shared resources in concurrent programming.

Types:

Binary semaphore (mutex) → 0 or 1, like a lock.

Counting semaphore → Allows multiple accesses.

Example:

In a printing system with 3 printers, a counting semaphore initialized to 3 ensures max 3 processes print at once.

---

37. What is Continuous Integration (CI) and Continuous Deployment (CD)?

Answer:

CI: Developers frequently merge code into a shared repo. Automated builds/tests verify changes.

CD: Code changes are automatically deployed to production after passing tests.

Tools: Jenkins, GitHub Actions, GitLab CI/CD.

Example: A developer pushes code → Tests run → If green, app auto-deploys to AWS.

---

38. What is an API Gateway?

Answer:

API Gateway = Entry point that manages API requests in microservices.

Responsibilities:

Routing requests to appropriate services.

Authentication & authorization.

Rate limiting & monitoring.

Example: AWS API Gateway → Handles thousands of requests to different backend microservices in a single endpoint.

---

39. What is the difference between Black Box and White Box Testing?

Answer:

Black Box Testing → Tester doesn't know code internals. Tests functionality (UI, inputs/outputs).

White Box Testing → Tester knows internal code logic. Tests paths, conditions, loops.

Example:

Black box → Entering login credentials, checking if login works.

White box → Testing how password validation function handles edge cases.

---

40. What is Sharding in Databases?

Answer:

Sharding = Splitting large databases into smaller, faster, more manageable parts (shards).

Example:

A social media app with 1B users.

Instead of one DB, data is split by **user_id range** →

Shard 1: users 1–100M

Shard 2: users 100M–200M, etc.

Benefits: Scalability, performance.

Challenges: Complex queries across shards.

Technical Interview Cheat Sheet – Q41 to Q50

41. What is a Data Warehouse?

Answer:

A **data warehouse** is a centralized repository for storing structured data from multiple sources, optimized for analytics and reporting (not transactions).

ETL (Extract, Transform, Load): Moves data from source → warehouse.

OLTP vs OLAP:

OLTP → Online Transaction Processing (banking).

OLAP → Online Analytical Processing (data warehouse, BI).

Example: Retail chain stores sales data in a warehouse → Analysts run reports on best-selling products.

---

42. What is Latency and Throughput?

Answer:

Latency: Time taken to process a single request.

Throughput: Number of requests processed per unit time.

Example:

A messaging app:

Latency = time to deliver one message.

Throughput = number of messages sent per second across system.

---

43. Explain the difference between HTTP and HTTPS.
Answer:

HTTP (HyperText Transfer Protocol): Data is sent in plain text.

HTTPS (HTTP Secure): Uses SSL/TLS encryption for secure communication.

Example:

HTTP → `http://example.com` (insecure).

HTTPS → `https://example.com` (encrypted, padlock icon).

---

44. What is the difference between Machine Learning and Deep Learning?
Answer:

Machine Learning (ML): Algorithms learn patterns from data (Regression, Decision Trees, SVM).

Deep Learning (DL): Subset of ML using neural networks with multiple layers.

Example:

ML → Spam email detection using word frequency.

DL → Self-driving car vision system using Convolutional Neural Networks (CNN).

---

45. What is a CDN (Content Delivery Network)?

Answer:

A CDN is a geographically distributed network of servers that deliver content (videos, images, scripts) from the closest server to the user.

Benefits: Faster load times, reduced latency, better availability.

Example: Netflix uses CDNs to stream videos from the nearest edge server to your location.

---

46. What is a Race Condition?

Answer:

A **race condition** occurs when multiple threads/processes access shared resources and the final outcome depends on timing.

Example in Python:

```
import threading
counter = 0

def increment():
    global counter
    for _ in range(100000):
```

```
        counter += 1

t1 = threading.Thread(target=increment)
t2 = threading.Thread(target=increment)
t1.start(); t2.start()
t1.join(); t2.join()
print(counter)  # Expected 200000, but result may vary
```

Solution: Use locks/mutex to avoid race conditions.

---

47. What is a Proxy Server?

Answer:

A proxy server acts as an intermediary between client and internet.

Uses:

Anonymity (hides IP).

Caching for faster browsing.

Filtering/blocking content.

Example: Company network uses proxy to block social media sites.

---

48. What is the difference between Continuous Delivery and Continuous Deployment?

Answer:

Continuous Delivery: Code is always in a deployable state, but release requires manual approval.

Continuous Deployment: Every successful change automatically goes to production without manual steps.

Example:

Delivery → Jenkins pipeline prepares release; manager approves before going live.

Deployment → GitHub Actions pushes directly to production.

---

49. What is a Graph Database?

Answer:

A **graph database** stores data as **nodes** (entities) and **edges** (relationships).

Examples: Neo4j, Amazon Neptune.

Use cases: Social networks, fraud detection, recommendation engines.

Example:

Node: User → "Alice"

Node: User → "Bob"

Edge: "Alice follows Bob"

---

50. What are Design Patterns in Software Engineering?

Answer:

Design patterns = Reusable solutions to common software design problems.

Categories:

Creational: Singleton, Factory, Builder.

Structural: Adapter, Decorator, Composite.

Behavioral: Observer, Strategy, Command.

Example:

Singleton Pattern: Only one instance of a class. Used in logging, config managers.

```
class Singleton {
    private static Singleton instance;
    private Singleton() {}
    public static Singleton getInstance() {
        if (instance == null)
            instance = new Singleton();
        return instance;
    }
}
```