1. OOPS: Foundations & Pillars

Why OOPS?

Before OOPS, **Procedural Programming** focused on functions and logic. However, as software grew, managing global data became difficult. OOPS was introduced to:

- **Mirror the Real World:** Objects have states (attributes) and behaviors (methods).
- **Enhance Reusability:** Through inheritance.
- **Improve Security:** Through data hiding.

Key Concepts

- **Class:** A blueprint or template (e.g., `Car` design).
- **Object:** An instance of a class (e.g., your specific Toyota Camry).

2. OOPS Scenario: The Banking System (Encapsulation)

Problem: In a banking app, if the `balance` variable is public, any part of the program can change it without verification (e.g., `account.balance = -1000`).

Solution: Encapsulation Encapsulation is the bundling of data (variables) and methods (functions) that operate on that data into a single unit (Class), and restricting direct access to some components.

Implementation Logic:

1. **Private Data:** Mark `balance` as private.
2. **Public Methods:** Use `get_balance()` and `deposit()` / `withdraw()` .
3. **Validation:** Inside `withdraw()` , check if the amount is positive and less than the current balance.

> **Security Tip:** Data hiding prevents "accidental corruption" and ensures "integrity" because data can only be modified via authorized methods.

3. DSA: Finding the Second Largest Element

Finding the second largest element in an array requires handling edge cases (like all elements being the same or arrays with fewer than two elements).

Efficient Approach (Single Pass)

Instead of sorting (which takes $O(n \log n)$), we can do this in $O(n)$ time.

Algorithm:

1. Initialize `largest = -∞` and `second_largest = -∞` .

2. Traverse the array:
    - If current element $x > largest$:
        - Update `second_largest = largest`
        - Update `largest = x`
    - Else if $x > second\_largest$ AND $x \neq largest$:
        - Update `second_largest = x`

Edge Cases:

- Array size < 2: Return -1 or error.
- All elements same: Second largest does not exist.

4. SQL: INNER JOIN

The `INNER JOIN` keyword selects records that have matching values in both tables.

The Logic

Think of it as the **Intersection** of two sets in a Venn Diagram.

Syntax

```
SELECT orders.order_id, customers.customer_name
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;
```

Key Takeaways:

- **Requirement:** A common column (usually a Primary Key in one table and a Foreign Key in another).
- **Result:** Only returns rows where there is a match in **both** tables. If a customer has no orders, they won't appear in the result above.

Summary Table

Topic	Focus	Key takeaway
OOPS	Encapsulation	Protects data integrity via Private access modifiers.
DSA	Second Largest	$O(n)$ is better than sorting; watch out for duplicates.
SQL	INNER JOIN	Retrieves rows only when there is a match in both tables.