Key Abbrev

Friday, 1 November 2024 8:02 PM

1. SOLID Principles

- **Single Responsibility Principle (SRP)**: Each class should have only one reason to change, meaning it should handle one responsibility or functionality.
- **Open/Closed Principle (OCP)**: Software entities (classes, modules, functions) should be open for extension but closed for modification.
- **Liskov Substitution Principle (LSP)**: Objects of a superclass should be replaceable with objects of subclasses without affecting the correctness of the program.
- **Interface Segregation Principle (ISP)**: Clients shouldn't be forced to depend on interfaces they don't use; split large interfaces into smaller, more specific ones.
- **Dependency Inversion Principle (DIP)**: High-level modules should not depend on low-level modules but rather on abstractions.

2. ACID Compliance

- Used in database transactions to ensure reliability:
  - **Atomicity**: Transactions are all-or-nothing; if one part fails, the entire transaction fails.
  - **Consistency**: Transactions take the database from one valid state to another, maintaining all rules.
  - **Isolation**: Transactions are isolated from each other, preventing concurrent interference.
  - **Durability**: Once a transaction is committed, it remains so, even in the case of a system failure.

3. CAP Theorem

- Defines trade-offs in distributed systems:
  - **Consistency**: All nodes see the same data at the same time.
  - **Availability**: Every request receives a response, without guaranteeing the most recent data.
  - **Partition Tolerance**: The system continues to operate even if there is a network split.
- CAP theorem states that a distributed database can only achieve two out of the three properties simultaneously.

4. Other Key Concepts

- **DRY (Don't Repeat Yourself)**: Avoid redundancy by centralizing code logic.
- **KISS (Keep It Simple, Stupid)**: Design should be as simple as possible, avoiding unnecessary complexity.
- **YAGNI (You Aren't Gonna Need It)**: Only add functionality when it's necessary.
- **Idempotency**: Performing an operation multiple times should have the same effect as doing it once (important in APIs).

These concepts help build robust, scalable, and maintainable software systems and are common discussion points in interviews.