Apex Super Specialist Track Viva Questions with Answers

1. Apex Fundamentals

Q1. What is Apex? How is it different from Java? Apex is a strongly typed, object-oriented programming language used in Salesforce for business logic. It's similar to Java but runs in a multitenant environment and is optimized for Salesforce data manipulation.

Q2. Explain the execution order of Apex code. Execution order: System validation rules -> Before triggers -> Custom validation rules -> After triggers -> Assignment rules -> Auto-response rules -> Workflow rules -> Processes -> Escalation rules -> Roll-up summary -> Criteria-based sharing.

Q3. What are the data types supported in Apex? Primitive (Integer, String, Boolean), Collections (List, Set, Map), Enums, sObjects, Objects, and User-defined classes.

Q4. What is the difference between `public`, `global`, and `private`? - `private`: accessible only within the class. - `public`: accessible within the namespace. - `global`: accessible across namespaces, required for web services.

Q5. What are static and final variables? - `static`: belongs to the class, not instance. - `final`: constant value, cannot be reassigned.

2. Triggers

Q1. What is the difference between `before` and `after` triggers? - `before`: used to validate or modify values before saving to DB. - `after`: used to access field values set by the system or write related records.

Q2. How do you prevent recursion in triggers? Using static Boolean flags or custom classes like `TriggerHandler` to control execution.

Q3. What is Trigger.new and Trigger.old context? - `Trigger.new`: new versions of records (for insert/update). - `Trigger.old`: old versions of records (for update/delete).

Q4. Can we perform DML operations inside a loop in a trigger? No, it causes governor limit issues. Use collections and perform bulk DML after the loop.

Q5. How do you handle bulk records in triggers? Use collections (Map/List), avoid SOQL/DML in loops, and follow best practices.

3. Governor Limits

Q1. What are governor limits in Salesforce? Limits enforced by Salesforce to ensure resource use is contained in multi-tenant environment.

Q2. How do you monitor and avoid SOQL limits? Use Limits class (`Limits.getQueries()`), bulk patterns, and avoid SOQL in loops.

Q3. How do collections help in handling governor limits? They allow bulk processing and reduce the number of DML/queries.

Q4. What is the maximum number of records returned by SOQL in one transaction? 50,000 records.

4. Asynchronous Apex

Q1. What is Asynchronous Apex? Code that runs separately from the main thread. Used for long-running operations.

Q2. Differences between @future, Batch Apex, Queueable, and Scheduled? - `@future`: simple async, no chaining. - `Batch`: handles large data in chunks. - `Queueable`: async with chaining, more flexible. - `Scheduled`: runs at specific times.

Q3. When to use Queueable over Batch? For simpler async logic with chaining and finer control over execution.

Q4. Can we call a future method from a trigger? Yes, but not from another future method or Queueable.

Q5. How do you monitor async jobs? Via Apex Jobs under Setup or using `AsyncApexJob` object.

5. Apex Testing

Q1. What is the purpose of test classes in Apex? To ensure code works correctly and meets code coverage requirements for deployment.

Q2. Best practices for test classes? Use `@isTest`, cover positive/negative cases, assert results, no hardcoding IDs.

Q3. What is `@isTest(SeeAllData=true)`? Allows access to org data. Avoid using it unless necessary.

Q4. How do you test exceptions in Apex? Use `try-catch` and assert the exception is thrown.

Q5. Difference between test.startTest() and test.stopTest()? Limits reset inside the block. Ensures async code gets executed.

6. SOQL and SOSL

Q1. Difference between SOQL and SOSL? - SOQL: queries specific objects. - SOSL: searches across multiple objects.

Q2. When to use SOSL? When searching across different objects for a keyword.

Q3. How to avoid "Too many SOQL queries" error? Bulkify code, move SOQLs out of loops.

Q4. Relationship queries in SOQL? - Parent-to-child: Subquery using child relationship name. - Child-to-parent: Dot notation.

Q5. Can you perform DML inside a SOQL loop? Not recommended. Causes limit errors. Use collections.

7. Integration in Apex

Q1. Types of Apex callouts? REST and SOAP.

Q2. How to perform a REST callout in Apex? Use `Http`, `HttpRequest`, `HttpResponse` classes.

Q3. Purpose of HttpRequest/HttpResponse? - `HttpRequest`: set endpoint/method/body. - `HttpResponse`: get response from external system.

Q4. How to handle callout exceptions? Use try-catch and check status codes.

Q5. What is `Continuation` in Apex? Used to handle long-running callouts (up to 120 sec).

8. Apex Design Patterns

Q1. What is Trigger Handler Pattern? A design that centralizes trigger logic into handler classes for readability and reusability.

Q2. Explain Singleton and Factory pattern in Apex. - Singleton: Ensures one instance (commonly for caching). - Factory: Creates object instances dynamically.

Q3. How do you structure Apex code for large apps? Layered approach (Controller -> Service -> DAO), modular, reusable classes.

Q4. Use of interfaces and abstract classes? To define standard behavior across unrelated classes and allow polymorphism.

9. Security in Apex

Q1. Enforce FLS and OLS in Apex? Use `Schema.sObjectType.Contact.fields.Email.isAccessible()` or `Security.stripInaccessible()`.

Q2. What is with sharing vs without sharing? Controls whether sharing rules apply to Apex class. `with sharing` respects user access.

Q3. When to use stripInaccessible? When querying/inserting/updating records while respecting field-level access.

Q4. How do you secure Apex REST services? Use authentication, `with sharing`, and field-level checks.

10. Error Handling and Debugging

Q1. How to handle exceptions in Apex? Use try-catch-finally. Log errors or show user-friendly messages.

Q2. What is a custom exception class? User-defined exception type extending `Exception` class.

Q3. How to debug Apex code? Use `System.debug()`, debug logs, and Developer Console.

Q4. Efficient use of `System.debug()`? Use conditionally, and avoid excessive logging in production.

11. Apex in LWC & VF Context

Q1. How do you use Apex with LWC? Expose methods using `@AuraEnabled` and call via `import` in JS.

Q2. How do you expose an Apex method to LWC? Annotate with `@AuraEnabled(cacheable=true)` for read-only data.

Q3. What is @AuraEnabled(cacheable=true)? Allows client-side caching of Apex results. Improves performance.

Q4. How to handle server-side errors in LWC? Use `.catch()` block in JS and show error toast or message.

12. Scenario-Based

Q1. Design trigger framework for multiple objects? Use base handler class, event-based methods, and delegate logic using interfaces.

Q2. Handle 5M records from external API? Use external service with batch + callout or Platform Event + Async.

Q3. Partial failures in batch job? Implement error handling in `execute()`, log failed records, and continue processing.

Q4. Retry mechanism for failed callouts? Log failed calls, use custom object/flag for retry logic via Scheduled Job or future method.