Java & Spring Boot Interview Notes

Core Java Concepts

Core Java Concepts (Step-by-Step):

1. Basics of Java:
- Data Types (int, float, boolean, char)
- Variables, Operators, Type Casting
- Control Statements (if, else, switch)
- Loops (for, while, do-while)
- Arrays and Strings

2. OOPs Concepts:
- Class & Object
- Inheritance
- Polymorphism (Overloading & Overriding)
- Abstraction (abstract class & interface)
- Encapsulation

3. Exception Handling:
- try, catch, finally
- throw, throws
- Custom Exceptions

4. Collections Framework:
- List, Set, Map, Queue
- ArrayList, LinkedList, HashSet, TreeSet, HashMap
- Iterator, Comparator, Comparable

5. Multithreading:
- Thread class and Runnable interface
- Synchronization
- Inter-thread communication

6. File I/O:
- FileReader, FileWriter, BufferedReader
- Serialization and Deserialization

7. Java 8+ Features:
- Lambda Expressions
- Stream API
- Functional Interfaces (Predicate, Consumer, etc.)
- Optional class

OOPs Concepts (Detailed)

OOPs Concepts in Java (Expanded):

Java & Spring Boot Interview Notes

1. Encapsulation:
- Wrapping data (variables) and code (methods) into a single unit (class).
- Use private variables + public getters/setters.

2. Inheritance:
- One class inherits another.
- "extends" keyword is used.
- Promotes code reusability.

3. Polymorphism:
- Method Overloading (compile-time)
- Method Overriding (runtime)

4. Abstraction:
- Hiding implementation using abstract classes or interfaces.
- Abstract class can have both abstract & non-abstract methods.
- Interface supports multiple inheritance.

5. Association, Aggregation, Composition:
- Association: General relationship between classes.
- Aggregation: "Has-a" relationship, weaker (can exist independently).
- Composition: Stronger aggregation (lifecycle is dependent).

6. Static vs Non-static:
- Static methods/variables belong to class, not instances.
- Accessed directly with class name.

7. Constructor & Constructor Overloading

Advanced Java Topics

Advanced Java Topics:

1. JDBC:
- Connect Java to DB using DriverManager
- Steps: Load driver, get connection, create statement, execute

2. Servlets:
- Java class that handles HTTP requests/responses
- doGet(), doPost()

3. JSP (Java Server Pages):
- Used to create dynamic web pages with HTML + Java

4. Filters and Listeners:
- Filters modify requests/responses.
- Listeners track lifecycle events (session, context)

Java & Spring Boot Interview Notes

5. JavaMail API, JSON Processing (Jackson), WebSockets

6. Design Patterns:
- Singleton, Factory, Builder, DAO, MVC

7. Maven & Gradle:
- Build tools for managing dependencies and build lifecycle

8. RESTful Web Services:
- JSON based API creation with Spring Boot

9. Unit Testing:
- JUnit, Mockito, TestRestTemplate

10. Security:
- Spring Security basics

Spring vs Spring Boot

Spring vs Spring Boot:

Spring:
- Requires XML or Java Config
- External web server (Tomcat)
- Manual dependency management
- Slower development

Spring Boot:
- Auto-configuration
- Embedded server (Tomcat/Jetty)
- Starter dependencies
- Faster development

Spring Boot Annotations Explained

Important Spring Boot Annotations:

1. @SpringBootApplication - Combines @Configuration, @EnableAutoConfiguration, @ComponentScan
2. @RestController - For REST APIs, combines @Controller + @ResponseBody
3. @Controller - Handles web pages (used with Thymeleaf, JSP)
4. @Repository - DAO classes, exception translation
5. @Component - Generic bean
6. @Service - Business logic layer
7. @RequestMapping - Map all HTTP methods
8. @GetMapping / @PostMapping / @PutMapping / @DeleteMapping - HTTP method-specific routes
9. @Bean - Manually define beans
10. @Configuration - Used for config classes

Java & Spring Boot Interview Notes

SQL Joins Summary

MySQL Join Types:

1. INNER JOIN - Only matched rows
2. LEFT JOIN - All from left + matches from right
3. RIGHT JOIN - All from right + matches from left
4. FULL JOIN - All records from both (use UNION)
5. CROSS JOIN - Cartesian product

Database Normalization

Database Normalization:

1NF: Atomic values, no repeating groups
2NF: No partial dependency (for composite keys)
3NF: No transitive dependency
BCNF: Advanced form of 3NF

Goal: Remove redundancy, maintain consistency.

Palindrome Programs in Java

Palindrome String in Java:
```
String str = "madam";
```
Check with two-pointer technique (i=0, j=str.length-1)

Palindrome Array:
```
int[] arr = {1,2,3,2,1};
```
Compare from both ends i and j.

Method:
```
while(i < j) {
  if(arr[i] != arr[j]) return false;
}
```

Java + Spring Boot Interview Questions

Java + Spring Boot Interview Questions (Common & Advanced):

Core Java:
- What are OOPs principles?
- Difference between abstract class and interface?
- What is method overloading vs overriding?

Java & Spring Boot Interview Notes

- What is the difference between == and .equals()?
- What is final, finally, finalize?
- Difference between String, StringBuilder, StringBuffer?
- Checked vs Unchecked Exception?
- What is JVM, JRE, JDK?
- How garbage collection works in Java?
- What are collections and their hierarchy?
- HashMap vs Hashtable vs ConcurrentHashMap

Advanced Java:

- Serialization & Deserialization
- Java Reflection API
- Thread lifecycle and synchronization
- Java 8 features: Lambda, Streams, Functional interfaces
- Optional, Method references
- CompletableFuture & Concurrency API
- JDBC basics

Spring Boot Questions:

- What is Spring Boot?
- Difference between Spring and Spring Boot?
- What are Spring Boot starters?
- How does auto-configuration work?
- What is dependency injection? Types?
- What is @Component, @Service, @Repository?
- What is @Bean, @Configuration?
- What is application.properties/yml used for?
- What is actuator in Spring Boot?
- How to handle exceptions in Spring Boot?
- What is REST API? How to create it in Spring Boot?
- @RestController vs @Controller?
- Difference between @RequestParam and @PathVariable?
- What is Spring Data JPA?
- What is JpaRepository vs CrudRepository?
- How to connect Spring Boot to MySQL?
- What are profiles in Spring Boot?
- What is AOP (Aspect-Oriented Programming)?

Best Practices:

- Folder structure in Spring Boot
- Use of DTOs and ModelMapper
- Validation using @Valid and @NotNull, etc.
- Logging using SLF4J
- Unit testing with JUnit + Mockito