Certainly! Let's go through each of the interview questions in detail:

\*\*Second Technical Round (Enzigma Software)\*\*:

1. \*\*Intro yourself\*\*:

- Provide a brief introduction, highlighting your educational background, work experience, and any relevant skills or achievements.

2. \*\*Code - Find Second Highest and Second Lowest Element\*\*:

- Given the integer array `arr`, you need to find the second highest and second lowest elements without using built-in functions. Here's a sample code in Java:

```java

int arr[] = {23, 48, 4, 55, 67, 85, 4, 5, 3, 90, 34, 27};

int highest = Integer.MIN\_VALUE, secondHighest = Integer.MIN\_VALUE;

int lowest = Integer.MAX\_VALUE, secondLowest = Integer.MAX\_VALUE;

for (int num : arr) {

if (num > highest) {

secondHighest = highest;

highest = num;

} else if (num < highest && num > secondHighest) {

secondHighest = num;

}

if (num < lowest) {

secondLowest = lowest;

lowest = num;

} else if (num > lowest && num < secondLowest) {

secondLowest = num;

}

}

System.out.println("Second Highest: " + secondHighest);

System.out.println("Second Lowest: " + secondLowest);

```

3. \*\*Inheritance\*\*:

- Inheritance is an object-oriented programming concept where a new class (subclass or derived class) is created by inheriting properties and behaviors (attributes and methods) from an existing class (superclass or base class).

4. \*\*Types of Inheritance\*\*:

- There are several types of inheritance, including single inheritance (one subclass inherits from one superclass), multiple inheritance (one subclass inherits from multiple superclasses), multilevel inheritance, hierarchical inheritance, and hybrid inheritance.

5. \*\*Multiple Inheritance in Java\*\*:

- Java supports multiple inheritance through interfaces but not through classes. Multiple inheritance with classes can lead to the diamond problem, which Java avoids.

6. \*\*Diamond Problem\*\*:

- The diamond problem occurs in languages that support multiple inheritance with classes. It's a situation where a class inherits from two classes that have a common ancestor. It can lead to ambiguity in method resolution.

7. \*\*Reducing the Diamond Problem\*\*:

- In Java, the diamond problem is avoided by allowing multiple inheritance through interfaces. Interfaces provide a way to implement multiple inheritance without the complications of the diamond problem.

8. \*\*Interface and Abstraction\*\*:

- An interface in Java is a contract that defines a set of abstract methods (methods without implementation). It's used to achieve abstraction by specifying what methods a class must implement without providing the implementation itself.

9. \*\*Unimplemented Interface Methods\*\*:

- In interfaces, methods are declared but not implemented. Classes that implement the interface must provide the implementation for these methods.

10. \*\*Interface and Data Abstraction\*\*:

- Interfaces can also be used to achieve data abstraction by specifying a set of data fields (constants) that implementing classes must use.

11. \*\*Difference Between Delete, Drop, and Truncate\*\*:

- In the context of databases:

- Delete: Removes specific rows from a table.

- Drop: Deletes an entire table or database object.

- Truncate: Removes all rows from a table but retains the table structure.

\*\*L2 Round (Enzigma Software)\*\*:

This round appears to focus on assessing your skills, qualities, and improvement areas. You may be asked scenario-based questions, project-related queries, and questions about your technical skills and communication abilities.

\*\*1st Round (Datametica Pvt Ltd)\*\*:

1. \*\*Object-Oriented Programming (OOP)\*\*:

- Explain the principles and concepts of OOP, such as encapsulation, inheritance, and polymorphism.

2. \*\*Exception Handling\*\*:

- Describe the concept of exception handling in Java and how it's used to manage errors and unexpected situations.

3. \*\*Java 8 Features\*\*:

- Discuss the new features introduced in Java 8, such as lambdas, streams, and the Stream API.

4. \*\*Stream API\*\*:

- Provide details about the Stream API in Java 8, including its purpose and how it can be used for processing collections of data.

5. \*\*Polymorphism\*\*:

- Explain both compile-time and runtime polymorphism in Java with examples.

6. \*\*Exception vs. Error\*\*:

- Differentiate between exceptions and errors in Java and their use cases.

7. \*\*XOR Operation\*\*:

- Explain the XOR (exclusive OR) operation and its significance in bitwise operations.

8. \*\*Functional Programming\*\*:

- Define functional programming and discuss its advantages in software development.

9. \*\*Data Structures\*\*:

- Describe different types of data structures, such as arrays, linked lists, stacks, and queues.

10. \*\*Stack and Queue\*\*:

- Explain the concepts of stacks and queues with real-life examples, highlighting their use cases.

11. \*\*LinkedList\*\*:

- Provide an overview of linked lists and how they differ from arrays in terms of data storage and manipulation.

12. \*\*DFS and BFS\*\*:

- Explain Depth-First Search (DFS) and Breadth-First Search (BFS) traversal algorithms and their applications.

13. \*\*Data Structures in Social Media Websites\*\*:

- Discuss which data structures are commonly used in social media websites and why they are chosen.

14. \*\*Normalization\*\*:

- Define normalization in the context of databases and its importance in data organization.

15. \*\*ACID Properties\*\*:

- Explain the ACID properties (Atomicity, Consistency, Isolation, Durability) in database management.

16. \*\*Group By vs. Having Clause\*\*:

- Differentiate between the GROUP BY and HAVING clauses in SQL and their usage.

17. \*\*Finding 2nd Highest Salary\*\*:

- Describe how to find the second-highest salary in a database table without using the `LIMIT` keyword.

18. \*\*Aggregation Function vs. Normal Function\*\*:

- Discuss the differences between aggregation functions (e.g., SUM, AVG) and normal functions in MySQL.

19. \*\*Project Explanation\*\*:

- Provide a detailed explanation of your project, including its objectives, technologies used, and your role in its development.

20. \*\*Optimizing Code in Coding Test\*\*:

- Explain how you optimized and improved the code written in your coding test.

These questions cover a range of technical topics, and your answers should demonstrate your knowledge and expertise in Java, data structures, SQL, and project-related aspects.