Tell me about your self

Hello, my name is saikumar, i have 2.5 years of experience as a java developer. During this time, I worked extensively with technologies such java, sql and various database management systems such as MySql, Oracle and PostgreSQl. I have strong expertise in java based application development, focusing on client management, reporting systems and integrating third-party services such as payment gateways, Email/SMS APIs and mobile applications

In my recent roles, I worked on projects like JDA for Walmart and Imaging&workflow for DBSbank. At walmart, I handled serviceNow tickets, resolved data-related issues, optimized sql queries for application integration and worked closely with cross-functions teams. At DBSbank, I contributed to designing and developing robust java-based components for imaging&workflow processs, worked on tools like Heidisql and jira for optimizing the database and task management.

In terms of tools, I'm proficient with version control systems like GitHub and Bitbucket. I'm experienced using DevOps tools such as jenkins and splunk for seamless integration and monitoring. I'm skilled at writing optimized sql queries and performed unit and integration tests using Junit and Mockito, ensuring system stability and performance under production conditions.

1. What is the Stream API in Java?

The Stream API, introduced in Java 8, is used to process sequences of elements (usually collections) in a functional style. It allows operations like filter, map, reduce, and collect in a declarative way.

2. What is Thread scheduler in Java?

It is part of JVM that decides which thread should run. There is no guarantee that which runnable thread will be chosen to run by the thread scheduler. Only one thread can run at a time in a single process.

3. What is daemon Thread?

It is a low priority thread that runs in background to perform tasks such as garbage collection. JVM terminates the daemon thread when all user threads finish their execution. JVM does not care whether the Daemon thread is running or not.

4. Can you explain wait(), notify() and notifyAll()methods?

wait(): Causes the current thread to wait until notified.

Notify(): it is used send a notification and wakes up a single thread that is waiting on the object’s monitor.

NitifyAll(): It is used to send notifications and wakes up all threads which are waiting on the object’s monitor.

5. What is shutdown hook ?

A shutdown hook is simply a thread that is invoked implicitly before JVM shuts down. It is one of the most important features of JVM because, it perform clean-up operations, like closing resources (files, network connections, etc.) or save application status before JVM shuts down.

6. What is deadlock in java?

It describes a situation where 2 or more threads are blocked forever, waiting for each other.

Or

It occurs in a situation when a thread is waiting for object's lock, i.e. acquired by the another thread and 2nd thread is waiting for object lock, i.e. acquired by 1st thread. Since both threads are waiting for each other to release the lock, the condition is called dead lock.

7. Can you explain the Life cycle of a thread?

MyThread t = new MyThread(); //instantiation of a thread

i. Once we create a thread object then the thread is said to be in **New** or **Born** state

ii. Once we call start() method then the thread will be entered into the **Ready** or **Runnable** state and waiting the CPU allocation.

iii. If the thread schedular allocates the CPU then thread will be entered to **Running** state

iv. Then it calls run() method, after completion of run() method thread will be entered to dead state.

8. What causes a Nullpointerexception in Java?

The NullPointerException occurs due to a situation in application code where an uninitialized object is attempted to be accessed or modified. Essentially, this means the object reference does not point anywhere and has a null value.

9. What is exception handling & Explain Exception and Error?

It is a powerful mechanism to handle runtime errors so that the normal flow of application is maintained.

Exception: An event that interrupts normal flow of a program. It is an object which is thrown at runtime. If there is any exception occurred, we can provide a solution programatically.

Ex: FileNotFoundException and NullPointerException.

Error: It is an illegal operation performed by the user which results in abnormal working of program. We can not provide a solution programmatically. It raises due to the lack of system resources.

Ex: LinkageError and StackOverFlowError.

10. What is user defined exception?

Exceptions which are created by the user based on the application requirement to handle to the specific error conditions that are not convered by standard java exceptions is called userdefined exceptions.

Ex: InvalidDepositException

11. What is the difference between throw and throws keywords?

Throw: Sometimes we will create exception objects explicitly and handover to JVM manually using 'throw' statement.

Or

The throw keyword allows us to throw an exception object to interrupt the normal flow of the program. This is most commonly used when a program fails to satisfy a given condition:

Throws: If there is any checked exception raised in our program we must & should handle that exception by using 'throws' statement or 'try-catch block.

12. Can you explain final, finally and finilize in java?

Final:

- It is a modifier applicable for classes, methods and variables
- If a class is declared as final then child class creation is not possible
- If a method is declared as final then overriding of that method is not possible
- If a variable is declared as final then reassignment is not possible

Finally:

- Used to maintain the cleanup code
- It is always executed, whether the exception handled or not
- It is always associated with try, catch blocks to maintain clean up code which is always executed

Finalize:

- It is a method which is called by the garbage collector just before destroying an object to perform cleanup activities.

13 What are the main characteristics of List, Set, and Map in Java?

- **List**: An ordered collection that allows duplicate elements. Elements are indexed and can be accessed by position. Examples: ArrayList, LinkedList.
- **Set**: A collection that does not allow duplicates and does not guarantee any specific order. Examples: HashSet, TreeSet.
- **Map**: A collection that stores key-value pairs, where each key is unique. It does not allow duplicate keys but allows duplicate values. Examples: HashMap, TreeMap.

14. What is a constructor?

- It is a special function member of a class
- Used to initialize an object
- These are executed under the object creation, so it is known as lazy loading.
- It has the same name as class name.

Types of Constructors: default, parameterized and no-argument constructors

15. Can you explain the concepts of method overloading & method overriding?

Method Overloading: Two methods have same name with different parameters in a single class is called “method overloading. It reduces the complexity of programming & increases the readability of program.

Method Overriding: Two methods have same name with same parameters, one of the method is present in Parent class & another method is present in child class, called “method overriding”.

16. What are static and Final keywords in java?

Static keyword means the value is same for every instance of the class.

Final keyword means once the value is assigned, value can never be changed.

17. Can you explain the main principles of Object Oriented Programming(OOP)?

i. Abstraction

Hiding the essential information and highlight some set of services is called abstraction. We can implement this using abstract classes & interfaces

Ex: GUI ATM Machine

Where bank people hide the internal implementation & highlight some services like banking, withdrawl, mini stmt etc.

The adv.of abstraction is

It provide the flexibility to the end user to use the system

It gives security because it will hide the internal

implementation It provides the maintainability of app.

ii. *Encapsulation*

The process of grouping variables and its associated methods in a single unit is called encapsulation. A class is said to be encapsulation,if supports data hiding & abstraction

iii. *Inheritence*

It means one class can extends to another class so that codes can be reused from one class to another class. Existing class is called 'Parent' class, derived class is called 'child' class.

Single level inheritence: Parent class has only one child class

Multi level inheritence: A class which is derived from super class and that super class also derived from another super class.

Hierarchical inheritence: A super class contains multiple or more than 1 base class.

iv. Polymorphism

The ability of an object to represent in different forms is called Polymorphism

A polymorphism which exhibits at compile time is called static/compile time polymorphism.

Ex: Method overloading

A polymorphism which exhibits at runtime is called dynamic/runtime polymorphism.

Ex: Method overriding

18. Can we override static method in java?

No, we can't override static method in java.

Static methods are part of the class itself and do not belong to instances of the class. Overriding requires runtime resolution of the method based on the object, which is not possible with static methods, as they are bound to the class and resolved during compile time.

19. Can we overload a main() method in java?

Yes, it is +ble to overload main() method in java but JVM always excutes the main() method which contains String arguments then it will execute int or float etc. main methods

20. Write a java program to find 2nd largest string in given array

Ex: {"apple", "banana", "cherry", "blueberry", "kiwi"};

```
public class LargestAndSecondLargestFinder {
  public static void main(String[] args) {
    // Example array of strings
    String[] strings = {"apple", "banana", "cherry", "blueberry", "kiwi"};
    // Call the method to find the largest and second-largest strings
    String[] results = findLargestAndSecondLargest(strings);
```

```
        // Print the results
        System.out.println("The largest string is: " + results[0]);
        System.out.println("The second-largest string is: " + results[1]);
    }
 public static String[] findLargestAndSecondLargest(String[] array) {
        // Check if the array is null or has fewer than 2 elements
        if (array == null || array.length < 2) {
            return new String[] {null, null}; // Not enough elements to find both
        }
// Initialize variables to store the largest and second-largest strings
        String largest = null;
        String secondLargest = null;
        // Iterate through the array
        for (String str : array) {
            if (str != null) {
                // Update largest and second-largest if necessary
                if (largest == null || str.length() > largest.length()) {
                    secondLargest = largest; // Update second-largest before updating largest
                    largest = str;
                } else if (str.length() > (secondLargest != null ? secondLargest.length() : 0) &&
                        !str.equals(largest)) {
                    secondLargest = str;
                }
            }
        }
        return new String[] {largest, secondLargest};
    }
}
```

21. Write a java program to find given word is palindrome or not?

Ex: Madam

```
public class StringPalindrome {
        public static void main(String[] args) {
                Scanner s = new Scanner(System.in);
                System.out.println("Enter String");
                String n= s.nextLine();
                if(isPalindrome(n)){
                        System.out.println(n+"  palindrome");
                } else {
                        System.out.println(n+" not palindrome");
                }
        }
        public static boolean isPalindrome(String n){
                if(n==null||n.isEmpty()){
                        return false;
                }
                int left =0;
                int right = n.length()-1;
                while(left<right){
                        if(n.charAt(left)!=n.charAt(right)){
                                return false;
```

```
                    }
                    left++;
                    right--;
                }
                return true;
        }
}
```
22. Write a Java program to findout the given number is prime or not

```
public class Prime {
        public static void main(String[] args) {
                Scanner s= new Scanner(System.in);
                System.out.println("Enter number");
                int number = s.nextInt();
                if(isPrime(number))
                                System.out.println(number + " is prime number");
                  else {
                     System.out.println(number + " is not prime number");
                     }
        }
        public static boolean isPrime(int n){
                if(n <=1){
                        return false;
                }
                if(n <=3){
                        retrun true;
               }
                if(n%2 == 0 || n%3 == 0){
                        return false;
                }
                for(int i=5; i*i <=n; i+=6){
                        if(n%i ==0 || n%(i+2)==0){
                                return false;
                        }
                }
        return true;
        }
}
```

23. What is @SpringBootApplication annotation?

@SpringBootApplication is a convenience annotation that combines three important annotations:

- @EnableAutoConfiguration: Tells Spring Boot to automatically configure your application based on the libraries present in the classpath.
- @ComponentScan: Enables Spring's component scanning mechanism, so Spring can find and register components (like beans and services) within the application context.
- @Configuration: Marks the class as a configuration class, meaning it can define Spring beans through @Bean methods.

24. What is dependency injection and how does Spring Boot handle it?

Dependency injection is a design pattern where dependencies are injected into a class by an external entity, promoting loose coupling. Spring Boot uses the **@Autowired** annotation to manage dependencies automatically, supported by Spring's IoC container. This enables easy management and testing of dependencies in Spring applications.

25. What is Spring Boot Actuator and how do you use it?

Spring Boot Actuator is a set of built-in tools that provides production-ready features like monitoring, metrics, health checks, and application management. It exposes various endpoints to gather information about the application's health, beans, environment properties, and more.

To use it, you need to add the spring-boot-starter-actuator dependency:

Key features of Spring Boot Actuator:

- **Health Checks**: /actuator/health endpoint returns the health status of the application, including database, disk space, etc.
- **Metrics**: /actuator/metrics exposes performance metrics like memory usage, request count, and more.
- **Environment Info**: /actuator/env shows environment properties, system properties, and configuration properties.
- **Application Info**: /actuator/info provides arbitrary information about the application, like version, build date, etc.
- **Logging**: /actuator/loggers allows dynamic adjustment of logging levels at runtime.

26. How do you handle database transactions in Spring Boot?

We use @Transactional annotation to manage database transactions, ensuring data consistency. This helps to roll back transactions in case of failures, which is crucial in maintaining database integrity, especially in complex multi-step operations

27. What are HTTP Methods used in REST APIs ?

Restful APIs use standard HTTP methods to perform CRUD operations on resources. The main HTTP methods are
GET:
Purpose: Retrieves data from the server (Read operation). When you want to fetch data from a resource.
Example: GET /users (Get a list of users) or GET /users/ID
POST:
Purpose: Sends data to the server to create a new resource (Create operation). When you need to create a new resource on the server. Example: POST /users (Create a new user).
PUT:

Purpose: Updates an existing resource on the server (Update operation). When you need to update an entire resource. Example: PUT /users/ ID
PATCH:
Purpose: Partially updates an existing resource (Partial Update operation). When you want to update a specific part of a resource. Example: PATCH /users/ ID
DELETE:
Purpose: Deletes a resource from the server (Delete operation).When you want to remove a resource from the server. Example: DELETE /users/ID

28. What is join & explain types of joins?

JOIN in SQL is used to combine rows or records from two or more tables in a database based on a related column between them.

Inner Join returns only the matching rows in both tables. If there is no match, the row is excluded from the result set.

LEFT JOIN returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values are returned for columns from the right table.

RIGHT JOIN returns all rows from the right table and the matched rows from the left table. If there is no match, NULL values are returned for columns from the left table.

FULL OUTER JOIN returns all rows when there is a match in **either** the left or the right table. If there is no match, the result will contain NULL values for the missing side.

29. How do you optimize SQL queries to handle large datasets efficiently?

To optimize SQL queries for large datasets, we have to follow some strategies such as

- Use EXPLAIN to analyze Queries - helps to understand how indexes are being used and which parts of query need optimization.
- Avoid SELECT * (Select Only Required Columns)
- Avoid Complex Joins if Possible
- Optimize Subqueries and Nested Queries
- Limit the Use of DISTINCT and GROUP BY- these will slowdown the query
- Use stored procedures when appropriate
- Avoid Overuse of Triggers and Views

30. What is view & explain types of views ?

view: view is a virtual table, logical representation of data from one or more tables.

They are different types

Simple view: when view is created using one base table it is called as Single view.

Complex view: When a view is created using 2 or more tables it is called Complex view.

Read only view: view that cannot be modified directly. You can only query the view for data.we can not perform DML operations(insert,update or delete)

With check option view: These views will allow DML operation only when where condition is satisfied.

Materialized view: A view whose result is physically stored (i.e., it materializes the data) in the database. Materialized views will help improving the performance of select statement on view.To create materialized view, the base table should have primary key.

31. What is an Abstract class and an Interface?

Abstract class can have abstract and non-abstract methods. The abstract keyword is used to declare abstract class. It can extend another Java class and implement multiple Java interfaces. It can be extended using keyword "extends". it can have statici,non-static,final and non-final variables.

Interface can have only abstract methods. Since Java 8, it can have default and static methods also. Interface has only static and final variables. The interface keyword is used to declare interface. An interface can be implemented using keyword "implements".

32. What is the difference between length & length() method?

Length: it's a final variable which is applicable for only arrays. It will return the length of an array.

Length() : it's a final method which is applicable for string objects. It will return the no.of characters present in string.

33. What is the use case of "==" & .equals() methods?

==: it is a comparison operator which returns boolean either true or false. It is used for reference comparison or address comparison.

.equals(): it is a method present in object class and string classs which returns boolean either true or false. It is used for content comparison

34. What is the use of printStackTrace() method?

Used to print the detailed information of exception occurred such as name of exception, description of exception and stack trace (line no. of exception).

35. What are the types of exceptions in java?

Checked Exception: Occurs at compile time

Ex: SQL Exception, IOException and ClassNotFoundException

Unchecked Exception: Occurs at runtime(JVM is responsible).

Ex: NumberFormatException, NullPointerException and ArrayIndexOutofBoundException and Arithmetic Exception

36. Can you explain static and instance methods in java?

Static method: If we use static keyword in a method declaration then the method is said to be static method. Static method is associated with a class.It can be called using the class name only without creating an instance of a class.

Instance method: is a method that belongs to an instance of a class.The instance method is associated with an object. It can be called on a specific instance of a class using the object reference.

37. What is a functional interface?

A functional interface in Java is an interface that contains exactly one abstract method. They can have multiple default or static methods. Functional interfaces can be used as the assignment target for lambda expressions or method references. An example of a functional interface is Runnable.

38. What are the main features of Java 8?

Lambda Expressions,Streams API,Default Methods in Interfaces,Optional Class.New Date and Time API

39. How do you create a thread in Java?

There are two main ways to create a thread in Java:

i) Extending the Thread Class

ii) Implementing the Runnable Interface

40. What is the purpose of 'super' and 'this' keywords in Java?

"super" keyword refers to the superclass and can be used to access superclass methods and constructors from a subclass.

"this" keyword is user to refer the current class object. It can be used to access the current class methods and constructors.

41. Can you explain a challenging scenario that you encountered in Spring Boot and how you resolved it.

I encountered a challenging bug while working with Spring Boot and JWT authentication. The problem was that valid JWT tokens were being rejected with a 401 Unauthorized error, even though the token was correct.To fix this, I manually stored the user's authentication details in the security context. I also realized that the JWT filter needed to be placed earlier in the filter chain to ensure proper handling before other authentication filters. After making these changes, the issue was resolved. This experience reinforced the importance of understanding filter order and context management in spring security

42. Can you explain your previous project?

I worked on a comprehensive banking project that aimed to streamline and enhance core banking operations.

I was involved in the design and development of various modules within the banking system, including:

Customer Account Management: Designing the structure to manage customer accounts, including account creation, balatnce tracking, transaction history, etc.

Transaction Processing: Implementing and optimizing transaction workflows such as deposits, withdrawals, fund transfers, and loan disbursements using Java and related frameworks.

Developed and maintained APIs for third-party integrations like payment gateways, SMS/Email services, and mobile apps.

Worked with relational databases like MySQL to design and optimize tables, stored procedures, and queries. This ensured that customer data was stored efficiently and could be queried quickly, even with large volumes of transaction history.

Identified bottlenecks in transaction processing and implemented improvements using multi-threading to handle large volumes of real-time transactions efficiently. Improved database queries, leveraging indexing to maintain high throughput in the system.

Participated in the deployment of the system using Jenkins for continuous integration/continuous deployment (CI/CD) pipelines.Utilized monitoring tools like Grafana to track system performance and uptime, ensuring high availability and reliability for end-users.