2022

Interview Booster

Arvind Singh

[JAVA INTERVIEW BOOSTER]

JAVA FRAMEWORK

BY

(ARVIND SINGH)

Quation 1 : What is Oops Concepts?

Answere 1

Data Hiding :

Our internal data should not go out directly that is outside person can't access

our internal data directly.

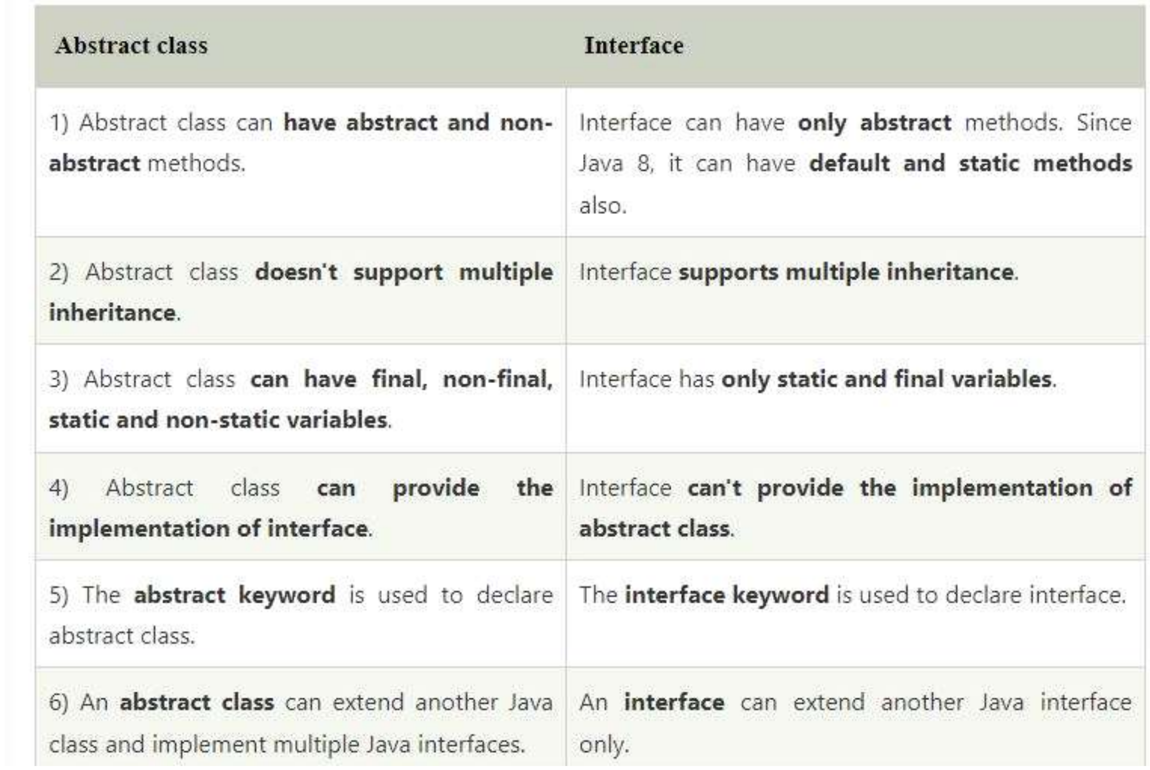
By using private modifier we can implement data hiding

Abstraction :

Hide internal implementation and just highlight the set of services, is called

abstraction.

By using abstract classes and interfaces we can implement abstraction



Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.

Encapsulation :

Binding of data and corresponding methods into a single unit is called

Encapsulation .

If any java class follows data hiding and abstraction such type of class is said to

be encapsulated class.

Encapsulation=Datahiding+Abstraction

Inheritance

Inheritance is a mechanism wherein a new class is derived from an existing class. In Java, classes may inherit or acquire the properties and methods of other classes. A class derived from

another class is called a subclass, whereas the class from which a subclass is derived is called a superclass.

Polymorphism:

Same name with different forms is the concept of polymorphism.

Example 1: We can use same abs() method for int type, long type, float type etc.

Example:

1. abs(int)

2. abs(long)

3. abs(float

Question 2: What Is Compile-time Polymorphism?

Answere :

Compile-time polymorphism is obtained through method overloading. The term method overloading allows us to have more than one method with the same name. Since this process is executed during compile time, that's why it is known as Compile-Time Polymorphism.

Run-Time Polymorphism: Whenever an object is bound with the functionality at run time, this is known as runtime polymorphism. The runtime polymorphism can be achieved by method overriding. Java virtual machine determines the proper method to call at the runtime, not at the compile time. It is also called dynamic or late binding. Method overriding says the child class has the same method as declared in the parent class. It means if the child class provides the specific implementation of the method that has been provided by one of its parent classes then it is known as method overriding.

Compile Time Polymorphism: Whenever an object is bound with its functionality at the compile time, this is known as the compile-time polymorphism. At compile-time, java knows which

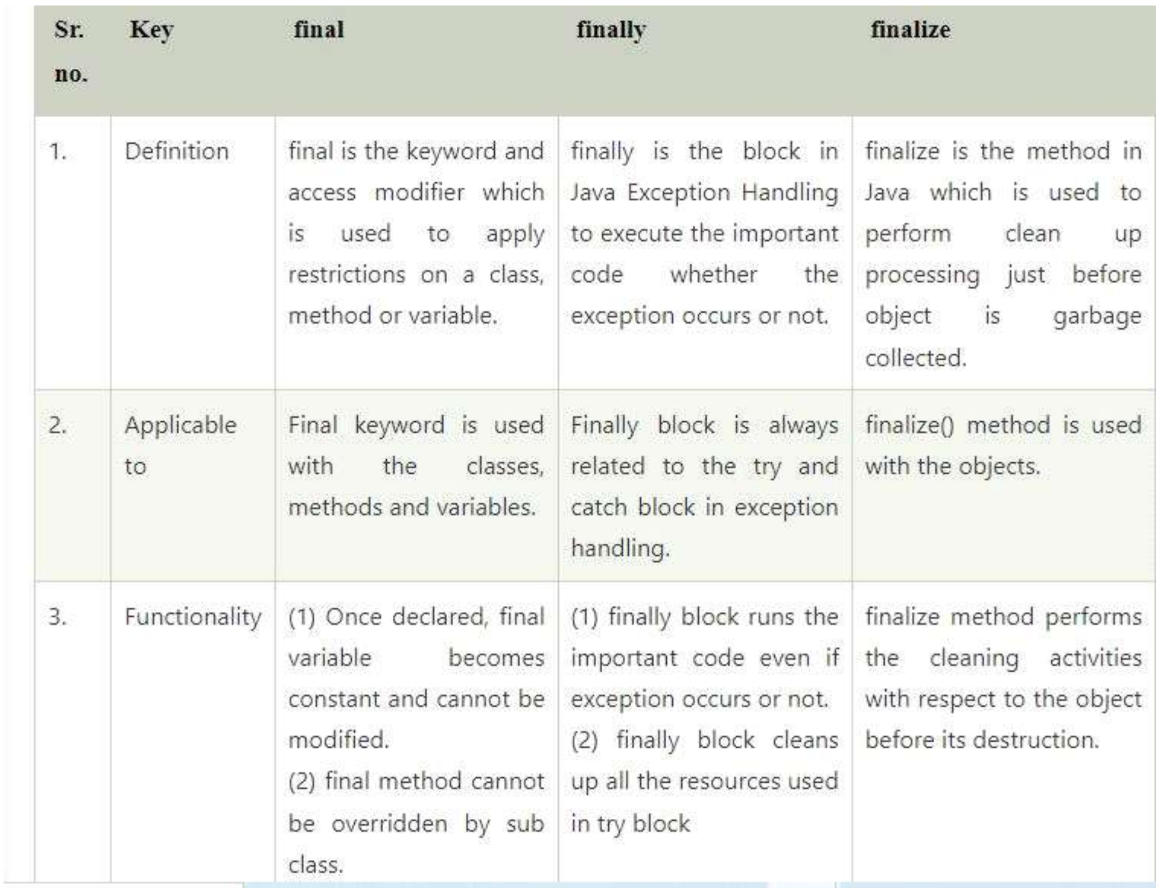
method to call by checking the method signatures. So this is called compile-time polymorphism or static or early binding. Compile-time polymorphism is achieved through method overloading. Method Overloading says you can have more than one function with the same name in one class having a different prototype. Function overloading is one of the ways to achieve polymorphism but it depends on technology and which type of polymorphism we adopt. In java, we achieve function overloading at compile-Time.

1) Inheritance talks about reusability.

2) Polymorphism talks about flexibility.

3) Encapsulation talks about security

Question 3: Difference between final, finally and finalize



Sr. no.	Key	final	finally	finalize
1.	Definition	final is the keyword and access modifier which is used to apply restrictions on a class, method or variable.	finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not.	finalize is the method in Java which is used to perform clean up processing just before object is garbage collected.
2.	Applicable to	Final keyword is used with the classes, methods and variables.	Finally block is always related to the try and catch block in exception handling.	finalize() method is used with the objects.
3.	Functionality	(1) Once declared, final variable becomes constant and cannot be modified. (2) final method cannot be overridden by sub class.	(1) finally block runs the important code even if exception occurs or not. (2) finally block cleans up all the resources used in try block	finalize method performs the cleaning activities with respect to the object before its destruction.

Question 4: What is String Pool in Java?

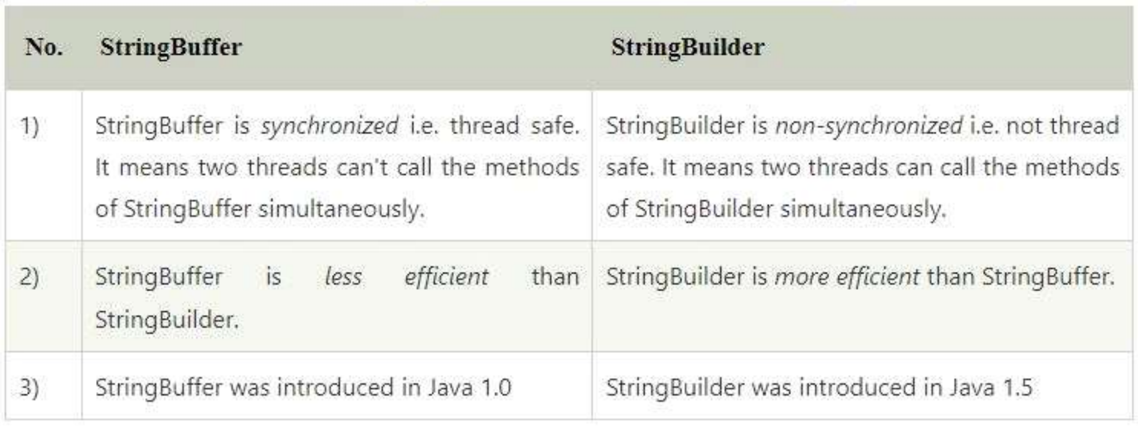
ANswere :

It is a Pool of Strings stored in the Heap Memory. All the strings created are stored in this pool if they do not already exist in the pool.

String class is immutable class and hence all string objects created using literals or new operators cannot be changed/modified.

Question: 5 What is Difference between StringBuffer and String Builder?

Answere



No.	StringBuffer	StringBuilder
1)	StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
2)	StringBuffer is *less efficient* than StringBuilder.	StringBuilder is *more efficient* than StringBuffer.
3)	StringBuffer was introduced in Java 1.0	StringBuilder was introduced in Java 1.5

Quation 6 What is Marker Interface ?

Answere :

In earlier versions of Java, Marker Interfaces were the only way to declare metadata about a class. For example, the Serializable Marker Interface lets the author of a class say that their class will behave correctly when serialized and deserialized.

In modern Java, marker interfaces have no place. They can be completely replaced by Annotations, which allow for a very flexible metadata capability. If you have information about a class, and that information never changes, then annotations are a very useful way to represent it.

Quation 7

How to create Immutable class in Java?

Answere :

Immutable class in java means that once an object is created, we cannot change its content. In Java, all the wrapper classes (like Integer, Boolean, Byte, Short) and String class is immutable. We can create our own immutable class as well.

1) The class must be declared as final so that child classes can’t be created.

2)Data members in the class must be declared private so that direct access is not allowed.

3)Data members in the class must be declared as final so that we can’t change the value of it after object creation.

4)A parameterized constructor should initialize all the fields performing a deep copy so that data members can’t be modified with an object reference.

5)Deep Copy of objects should be performed in the getter methods to return a copy rather than returning the actual object reference)

Quation 8 : What is Externalization in Java ?

Answere

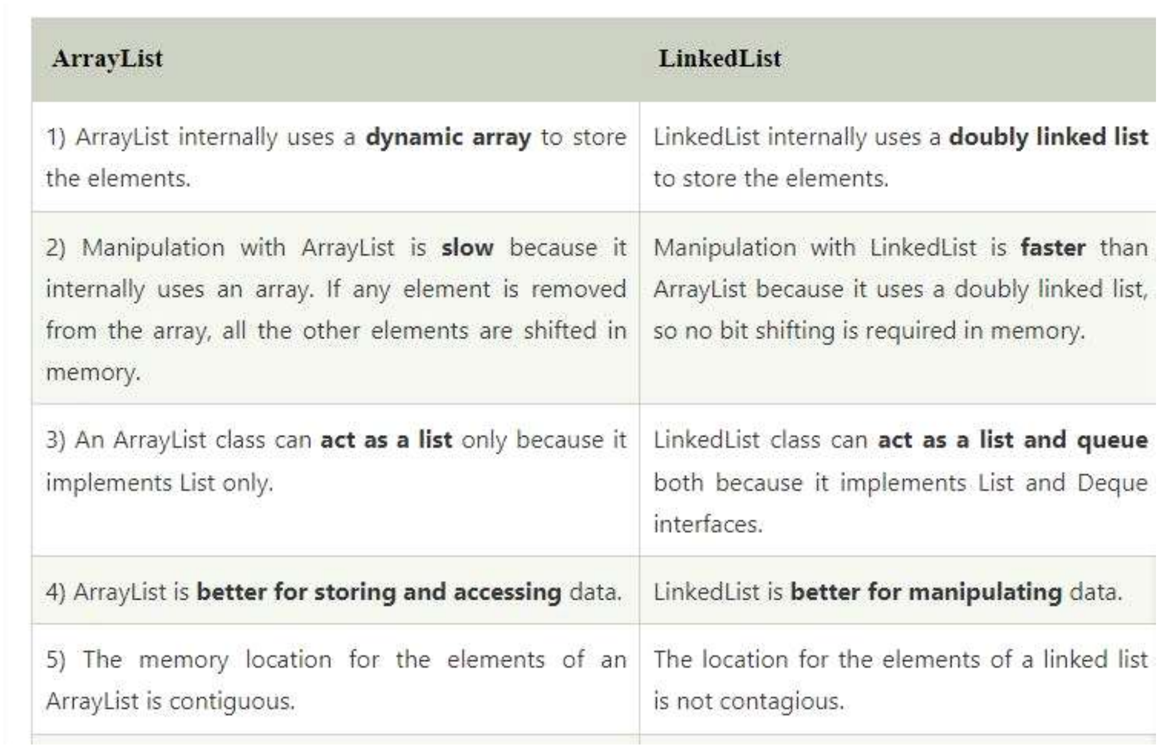
Externalization in Java is used to customize the serialization mechanism. Java serialization is not much efficient. When we have bloated objects that hold several attributes and properties, it is not good to serialize them. Here, the externalization will be more efficient.

Quation 9: What is Serialization?

Serialization is the process of converting an object into a stream of bytes to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called deserialization.

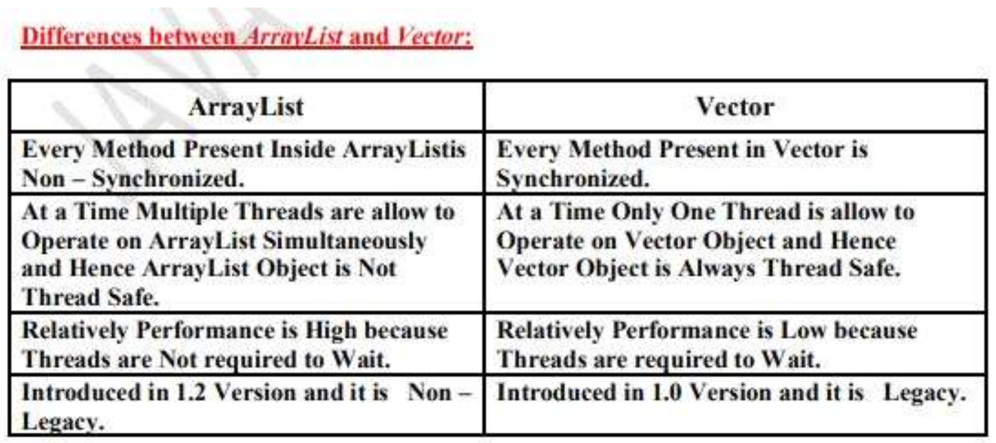
Question 10 : **Difference between ArrayList and LinkedList**

Answere :



ArrayList	LinkedList
1) ArrayList internally uses a **dynamic array** to store the elements.	LinkedList internally uses a **doubly linked list** to store the elements.
2) Manipulation with ArrayList is **slow** because it internally uses an array. If any element is removed from the array, all the other elements are shifted in memory.	Manipulation with LinkedList is **faster** than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
3) An ArrayList class can **act as a list** only because it implements List only.	LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces.
4) ArrayList is **better for storing and accessing** data.	LinkedList is **better for manipulating** data.
5) The memory location for the elements of an ArrayList is contiguous.	The location for the elements of a linked list is not contagious.

Question 11 : **Difference beween Arraylist anad Vector**



Differences between *ArrayList* and *Vector*:

ArrayList	Vector
Every Method Present Inside ArrayListis Non – Synchronized.	Every Method Present in Vector is Synchronized.
At a Time Multiple Threads are allow to Operate on ArrayList Simultaneously and Hence ArrayList Object is Not Thread Safe.	At a Time Only One Thread is allow to Operate on Vector Object and Hence Vector Object is Always Thread Safe.
Relatively Performance is High because Threads are Not required to Wait.	Relatively Performance is Low because Threads are required to Wait.
Introduced in 1.2 Version and it is Non – Legacy.	Introduced in 1.0 Version and it is Legacy.

Question 12 When we go for Comparable and When we go for Comparator:

Comparable Vs Comparator:

Answere

 For Predefined Comparable Classes (Like String) Default **Natural Sorting Order is Already**

Available. If we are Not satisfied with that we can Define Our Own Sorting by Comparator

Object.

 For Predefine Non- Comparable Classes (Like StringBuffer) **Default Natural Sorting Order**

is Not Already Available. If we want to Define Our Own Sorting we can Use Comparator

Object.

 For Our Own Classes (Like Employee) the Person who is **writing Employee Class he is**

Responsible to Define Default Natural Sorting Order by implementing Comparable

Interface.

** The Person who is using Our Own Class if he is Not satisfied with Default Natural Sorting**

Order he can Define his Own Sorting by using Comparator Object.

** If he is satisfied with Default Natural Sorting Order then he can Use Directly Our Class.**

Question 13 : What is Internal working of Hash Map?

Answere

1. **HashMap** has multiple buckets or bins which contain a head reference to a singly linked list. That means there would be as many linked lists as there are buckets. Initially, it has a bucket size of 16 which grows to 32 when the number of entries in the map crosses the 75%. (That means after inserting in 12 buckets bucket size becomes 32)

2. HashMap uses its static inner class **Node<K,V>** for storing map entries. That means each entry in hashMap is a **Node**. Internally HashMap uses a **hashCode** of the key Object and this hashCode is further used by the **hash function** to find the

index of the bucket where the new entry can be added.

3. HashMap uses multiple buckets and each bucket points to a **Singly Linked List** where the entries (nodes) are stored.

4. Once the bucket is identified by the hash function using hashcode, then hashCode is used to check if there is already a key with the same hashCode or not in the bucket(singly linked list).

5. If there already exists a key with the **same hashCode**, then the **equals()** method is used on the **keys.** If the equals method returns **true**, that means there is already a node with the same key and hence the value against that key is **overwritten** in the entry(node), otherwise, a new node is created and added to this Singly Linked List of that bucket.

6. If there is no key with the same hashCode in the bucket found by the hash function then the new Node is added into the bucket found.

Re-Hashing :

Whenever the number of entries in the hashmap crosses the ***threshold value*** then the bucket size of the hashmap is doubled and rehashing is performed and all already existing entries(nodes) of the map are copied and new entries are added to this increased hashmap.

Threshold value = Bucket size * Load factor

Eg. If bucket size is 16 and the load factor is 0.75 then the threshold value is 12.

Question 14 : What is the General Contract of the Hashcode

Answere

The general contract of hashCode is:

(1)During the execution of the application, if hashCode() is invoked more than once on the same Object then it must consistently return the same Integer value,

(2)If two Objects are equal, according to the **equals(Object)** method, then hashCode() method must produce the same Integer on each of the two Objects.

(3) If two Objects are unequal, according to the **equals(Object)** method, It is not necessary the Integer value produced by hashCode() method on each of the two Objects will be distinct.

Question 16 What is the difference between HashMap and ConcurrentHashMap ?

Answere :

- HashMap is non-Synchronized in nature i.e. HashMap is not Thread-safe whereas ConcurrentHashMap is Thread-safe in nature.
- HashMap performance is relatively high because it is non-synchronized in nature and any number of threads can perform simultaneously. But ConcurrentHashMap performance is low sometimes because sometimes Threads are required to wait on ConcurrentHashMap.
- While one thread is Iterating the HashMap object, if other thread try to add/modify the contents of Object then we will get Run-time exception saying **ConcurrentModificationException.** Whereas In ConcurrentHashMap we wont get any exception while performing any modification at the time of Iteration.
  Using HashMap

In HashMap, null values are allowed for key and values, whereas in ConcurrentHashMap null value is not allowed for key and value, otherwise we will get Run-time exception saying **NullPointerException.**

Question 17: What is WeakHashMap?

Answere :

It is Exactly Same as HashMap Except the following Difference.

In Case of HashMap, HashMap Dominates Garbage Collector. That is if Object doesn't

have any Reference Still it is Not Eligible for Garbage Collector if it is associated with

HashMap.

 But In Case of WeakHashMap if an Object doesn't contain any References then it is Always

Eligible for GC Even though it is associated with WeakHashMap. That is Garbage Collector Dominates WeakHashMap

Question 18 : What is Difference between ConCorrentHash Map andSynchronizedMap() and Hash Table

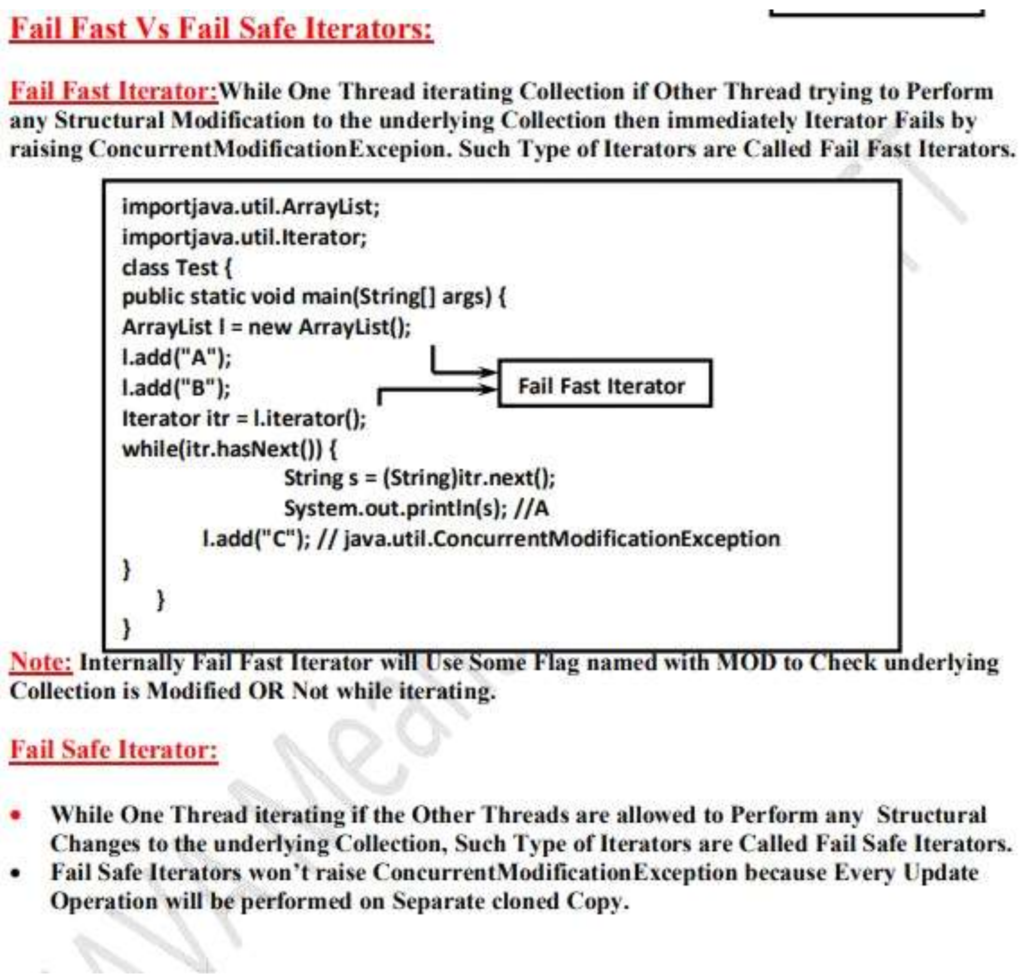
Answere

Difference between ConcurrentHashMap, synchronizedMap() and Hashtable

ConcurrentHashMap	synchronizedMap()	Hashtable
We will get Thread Safety without locking Total Map Object Just with Bucket Level Lock.	We will get Thread Safety by locking Whole Map Object.	We will get Thread Safety by locking Whole Map Object.
At a Time Multiple Threads are allowed to Operate on Map Object in Safe Manner.	At a Time Only One Thread is allowed to Perform any Operation on Map Object.	At a Time Only One Thread is allowed to Operate on Map Object.
Read Operation can be performed without Lock but write Operation can be performed with Bucket Level Lock.	Every Read and Write Operations require Total Map Object Lock.	Every Read and Write Operations require Total Map Object Lock.
While One Thread iterating Map Object, the Other Threads are allowed to Modify Map and we won't get ConcurrentModificationException.	While One Thread iterating Map Object, the Other Threads are Not allowed to Modify Map. Otherwise we will get ConcurrentModificationException	While One Thread iterating Map Object, the Other Threads are Not allowed to Modify Map. Otherwise we will get ConcurrentModificationException
Iterator of ConcurrentHashMap is Fail-Safe and won't raise ConcurrentModificationException.	Iterator of synchronizedMap is Fail-Fast and it will raise ConcurrentModificationException.	Iterator of synchronizedMap is Fail-Fast and it will raise ConcurrentModificationException.
null is Not allowed for Both Keys and Values.	null is allowed for Both Keys and Values.	null is Not allowed for Both Keys and Values.
Introduced in 1.5 Version.	Introduced in 1.2 Version.	Introduced in 1.0 Version.

Question 19 : What is Failsafe and FailFast

Answere



Fail Fast Vs Fail Safe Iterators:

Fail Fast Iterator: While One Thread iterating Collection if Other Thread trying to Perform any Structural Modification to the underlying Collection then immediately Iterator Fails by raising ConcurrentModificationExcepion. Such Type of Iterators are Called Fail Fast Iterators.

```
importjava.util.ArrayList;
importjava.util.Iterator;
class Test {
public static void main(String[] args) {
ArrayList l = new ArrayList();
l.add("A");
l.add("B");
Iterator itr = l.iterator();
while(itr.hasNext()) {
                    String s = (String)itr.next();
                    System.out.println(s); //A
          l.add("C"); // java.util.ConcurrentModificationException
}
     }
}
```

Note: Internally Fail Fast Iterator will Use Some Flag named with MOD to Check underlying Collection is Modified OR Not while iterating.

Fail Safe Iterator:

- While One Thread iterating if the Other Threads are allowed to Perform any Structural Changes to the underlying Collection, Such Type of Iterators are Called Fail Safe Iterators.
- Fail Safe Iterators won't raise ConcurrentModificationException because Every Update Operation will be performed on Separate cloned Copy.

Question 20 : What are some of the important features that are introduced in Java 8?

Answere:

There are many features important for development using Java that were implemented in Java 8. Some of these features are as follows:

- **Lambda expressions**
- **Default methods for interfaces**
- **Functional interface**
- **Stream API**
- **Date API**

Question 21: What is a lambda expression?

The lambda expression was added into Java 8 to provide users with an anonymous function that can be called without using the function name but has a functional set of parameters with a lambda entity.

Describe the syntax of a lambda expression.

Lambda expressions can be divided into three parts as shown in the below syntax:

```
//Lambda expression
```

```
(int a, int b) -> { System.out.println(a+b); return a+b;}
```

1. **Arguments:** A lambda expression can have zero or more arguments at any point in time.
2. **Array token:** It is used to point to the body of the expression.
3. **Body:** The body consists of expressions and statements. Braces are not required if it has only a single statement.

Question 22 : What are Functional Interfaces?

Answer: Functional Interface is an interface that has only one abstract method. The implementation of these interfaces is provided using a Lambda Expression which means that to use the Lambda Expression, you need to create a new functional interface or you can use the predefined functional interface of Java 8.

The annotation used for creating a new Functional Interface is "**@FunctionalInterface**".

Question 23 What is an optional class?

Answer: Optional class is a special wrapper class introduced in Java 8 which is used to avoid NullPointerExceptions. This final class is present under java.util package. NullPointerExceptions occurs when we fail to perform the Null checks.

Question 24 : What are the default methods?

Answer: Default methods are the methods of the Interface which has a body. These methods, as the name suggests, use the default keywords. The use of these default methods is "Backward Compatibility" which means if JDK modifies any Interface (without default method) then the classes which implement this Interface will break.

On the other hand, if you add the default method in an Interface then you will be able to provide the default implementation. This won't affect the implementing classes.

Question 25 : What are static methods in Interfaces?

Static methods, which contains method implementation is owned by the interface and is invoked using the name of the interface, it is suitable for defining the utility methods and cannot be overridden.

Question 26 : How can you create a Functional Interface?

Answer: Although Java can identify a Functional Interface, you can define one with the annotation

@FunctionalInterface

Once you have defined the functional interface, you can have only one abstract method. Since you have only one abstract method, you can write multiple static methods and default methods.

predefined functional interfaces and its description introduced in Java 8.

Runnable: use to execute the instances of a class over another thread with no arguments and no return value.

Callable: use to execute the instances of a class over another thread with no arguments and it either returns a value or throws an exception.

Comparator: use to sort different objects in a user-defined order

Comparable: use to sort objects in the natural sort order

Question 27 : What is a Predicate? State the difference between a Predicate and a Function?

Answer: Predicate is a pre-defined Functional Interface. It is under java.util.function.Predicate package. It accepts only a single argument which is in the form as shown below,

Predicate<T>

Predicate	Function
It has the return type as Boolean.	It has the return type as Object.
It is written in the form of **Predicate< T>** which accepts a	It is written in the form of **Function< T, R>** whi also accepts a single argument.

single argument.	
It is a Functional Interface which is used to evaluate Lambda Expressions. This can be used as a target for a Method Reference.	It is also a Functional Interface which is used evaluate Lambda Expressions. In Function< T, R>, for input type and R is for the result type. Thi also be used as a target for a Lambda Expression Method Reference.

Question 27 : What is MetaSpace in Java 8?

Answer: In Java 8, a new feature was introduced to store classes. The area where all the classes that are stored in Java 8 are called MetaSpace. MetaSpace has replaced the PermGen.

Till Java 7, PermGen was used by Java Virtual Machine to store the classes. Since MetaSpace is dynamic as it can grow dynamically and it does not have any size limitation, Java 8 replaced PermGen with MetaSpace.

Question 28 : What is the Consumer Functional Interface?

Answer: Consumer Functional Interface is also a single argument interface (like Predicate<T> and Function<T, R>). It comes under java.util.function.Consumer. This does not return any value.

Question 29 : What is the Supplier Functional Interface?

Answer: Supplier Functional Interface does not accept input parameters. It comes under java.util.function.Supplier. This returns the value using the get method.

Question 30 : What is a SAM Interface?

Answer: Java 8 has introduced the concept of FunctionalInterface that can have only one abstract method. Since these Interfaces specify only one abstract method, they are sometimes called as SAM Interfaces. SAM stands for “Single Abstract Method”.

Question 31: What is Stream Api?

Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result

Different Operations On Streams-

Intermediate Operations:

1. **map:** The map method is used to returns a stream consisting of the results of applying the given function to the elements of this stream.
```
List number = Arrays.asList(2,3,4,5);
List square = number.stream().map(x->x*x).collect(Collectors.toList());
```
2. **filter:** The filter method is used to select elements as per the Predicate passed as argument.
```
List names = Arrays.asList("Reflection","Collection","Stream");
List result = names.stream().filter(s-
>s.startsWith("S")).collect(Collectors.toList());
```
3. **sorted:** The sorted method is used to sort the stream.
```
List names = Arrays.asList("Reflection","Collection","Stream");
List result = names.stream().sorted().collect(Collectors.toList());
```

Terminal Operations:

1. **collect:** The collect method is used to return the result of the intermediate operations performed on the stream.
```
List number = Arrays.asList(2,3,4,5,3);
Set square = number.stream().map(x->x*x).collect(Collectors.toSet());
```
2. **forEach**: The forEach method is used to iterate through every element of the stream.
```
List number = Arrays.asList(2,3,4,5);
number.stream().map(x->x*x).forEach(y->System.out.println(y));
```
3. **reduce:** The reduce method is used to reduce the elements of a stream to a single value.
   The reduce method takes a BinaryOperator as a parameter.
```
List number = Arrays.asList(2,3,4,5);
int even = number.stream().filter(x->x%2==0).reduce(0,(ans,i)-> ans+i);
```

   Here ans variable is assigned 0 as the initial value and i is added to it .

Question 32: Given a list of employees, you need to filter all the employee whose age is greater than 20 and print the employee names.(Java 8 APIs only)

Answer:

```
List<String> employeeFilteredList = employeeList.stream()
                  .filter(e->e.getAge()>20)
                  .map(Employee::getName)
                  .collect(Collectors.toList());
```

Question 33: Given the list of employees, count number of employees with age 25?

Answere

```
List<Employee> employeeList = createEmployeeList();
long count = employeeList.stream()
.filter(e->e.getAge()>25)
.count();
System.out.println("Number of employees with age 25 are : "+count);
```

Question 34: Given the list of employees, find the employee with name "Mary".

Answere

```
List<Employee> employeeList = createEmployeeList();
```

```
      Optional<Employee> e1 = employeeList.stream()
                .filter(e->e.getName().equalsIgnoreCase("Mary")).findAny();

      if(e1.isPresent())
          System.out.println(e1.get());
```

▪

Question 35: Given a list of employee, find maximum age of employee?

Answer:

```
List<Employee> employeeList = createEmployeeList();
        OptionalInt max = employeeList.stream().
                          mapToInt(Employee::getAge).max();

        if(max.isPresent())
            System.out.println("Maximum age of Employee: 
"+max.getAsInt());
```

Question 36: Given a list of employees, sort all the employee on the basis of age? Use java 8 APIs only

Answere :

```
List<Employee> employeeList = createEmployeeList();
        employeeList.sort((e1,e2)->e1.getAge()-e2.getAge());
        employeeList.forEach(System.out::println);
```

Question 37: Get the details of highest paid employee in the organization?

Answere

```
Optional<Employee> highestPaidEmployeeWrapper=
employeeList.stream().collect(Collectors.maxBy(Comparator.comparingDou
ble(Employee::getSalary)));
```

Question 38: Get the names of all employees who have joined after 2015?

Answere

```
employeeList.stream()
            .filter(e -> e.getYearOfJoining() > 2015)
            .map(Employee::getName)
            .forEach(System.out::println);
```

Question 39: Get the details of youngest male employee in the product development department?

Answere

```
Optional<Employee> youngestMaleEmployeeInProductDevelopmentWrapper=
employeeList.stream()
            .filter(e -> e.getGender()=="Male" &&
e.getDepartment()=="Product Development")
            .min(Comparator.comparingInt(Employee::getAge));
```

Question 40: Who has the most working experience in the organization?

Answere

```
Optional<Employee> seniorMostEmployeeWrapper=
employeeList.stream().sorted(Comparator.comparingInt(Employee::getYearOfJoini
ng)).findFirst();
```

```
Employee seniorMostEmployee = seniorMostEmployeeWrapper.get();
```

Question 41: Find The Employee who name Start with s

Answere

```
List<Employee> basedOnStartName=employeeList.stream().filter(e ->
e.getFastName().startsWith("R")).collect(Collectors.toList());

        basedOnStartName.forEach(System.out::println);
```

Question 42: What is Method Reference?

Answer: In Java 8, a new feature was introduced known as Method Reference. This is used to refer to the method of functional interface. It can be used to replace Lambda Expression while referring to a method.

For Example: If the Lambda Expression looks like

```
num -> System.out.println(num)
```

Then the corresponding Method Reference would be,

```
System.out::println
```

where "::" is an operator that distinguishes class name from the method name.

Question 43: Explain the following Syntax

String:: Valueof Expression

Answer: It is a static method reference to the ***ValueOf*** method of the **String** class. System.out::println is a static method reference to println method of out object of System class.

It returns the corresponding string representation of the argument that is passed. The argument can be Character, Integer, Boolean, and so on.

Question 44: Explain the Map and FlatMap

Map : Java 8 Stream's map method is intermediate operation and consumes single element forom input Stream and produces single element to output Stream.

It simply used to convert Stream of one type to another.

```
<R> Stream<R>  map(Function<? super T,? extends R>mapper)
```

Map applies the mapper function on input Stream and generates the output Stream.

Here mapper function is functional interface which takes one input and provides one output.

FlatMap: In Java 8 Streams, the flatMap() method applies operation as a mapper function and provides a stream of element values. It means that in each iteration of each element the map() method creates a separate new stream. By using the flattening mechanism, it merges all streams into a single resultant stream. In short, it is used to convert a Stream of Stream into a list of values.

Syntax:

1. <R> Stream<R> flatMap(Function<? **super** T,? **extends** Stream<? **exten ds** R>> mapper)

The method takes a function as an argument. It accepts T as a parameter and returns a stream of R.

Question 45: What is a session object?

Answere

Session is a class provided as part of hibernate api, always a Session classobject holds the connection to the database and contains mapping and configuration information using which it perform the database operation.

The entire logic for accessing/persisting the data into relational database management interms of objects has been abstracted as part of the Session class.Let the operation we want to perform on the database could be

- accessing an object from db
- save
- update
- delete
- access by using sql query

let it be any operation, that can be carried by the Session class.

Question 46: What is a SessionFactory?

Answere

The name would itself suggest SessionFactory acts an factory for creating the object of Session. but there are several reasons for using the SessionFactory.

1. it acts as an context object in storing configuration and mapping information, so that we can avoid repeatedly reading these hibernate.cfg.xml andentity mapping files while creating and using the session object.
2. each time to create a session object we need connection to perform databaseoperation. everytime opening a connection and closing a connect impacts performance, instead session factory holds the connection pool by default using which it pulls connection from its connection pool and creates the object of Session.
3. Finally it acts as factory for creating the object of Session, so whilecreating the Session object it passing connection, mapping and configuration information to the Session object.

One SessionFactory object represents one database configuration and their tablemapping information.

SessionFactory is an heavy weight object because it holds the pool of connectionswithin it and it is always recommended to close the SessionFactory at the end of the application.

Question 47: get() vs load() method differences

Answere

get(): whenever we call the session.get(Class, Object), it immediately goes to the database, queries the data and populates into entity object and returns tous. This indicates get() method supports eager loading/initialization

- load(): whenever we call session.load(Class, Object), it creates an Proxy class at runtime within the jvm memory (using javaassist proxy lib) by populatingpk value into it and returns the Proxy object to us.

Whenever we tried accessing the data by calling the non-primary key column represented attributes accessor methods, then the proxy object has the logic forfetching the data from the database of the pk value and returns to us. This indicates load() method support lazy loading/initialization

get(): always supports eager loading/initialization only

load(): by default it supports lazy loading, but we can turn off the lazy loading behaviour by writing lazy=false in mapping file on entity class level

```
<hibernate-mapping>

  <class name="Branch" table="branch" lazy="false">

  </class>

</hibernate-mapping>
```

in this case load() also just works like get() method only

#4 get(): only supports eager loading

load(): if the entity class has
been marked as finalfinal class
Branch implements Serializable {

```
Branch branch = session.load(Branch.class, 1);
```

it returns branch object by populating data directly (witout lazy loading);,because load cannot create proxy on the final class, it justs works as get() method only.

Question 48: What is Entity Manager and Entity Manager Factory ?

Answere

EntityManager = Session class, which will takes care of performing persistenceoperations in Jpa

Note:- 1 object per one operation

EntityManagerFactory = SessionFactory class, holds the configuraiton and mappinginformation with which instantiates the object of EntityManager

Note:- 1 object of EntityManagerFactory per 1 database

Persistence = Configuration class, responsible for reading mapping and

configuration information with which it creates the object ofEntityManagerFactory

write persistence.xml = hibernate.cfg.xml, carrying the persistence framework configuration, mapping information and database details. The location of the filemust be under classpath of the application under META-INF/persistence.xml and name of the file should be persistence.xml only

Question 49: What is first-level cache in hibernate, what is the purpose of it and how does itworks?

Answere

First-Level cache in hibernate is also called as Session-Level Cache, the cache will be maintained at the Session object and will be destroyed when we close theSession object, so it is called "Session-Level" Cache, we can say the cache is local to the Session object.

The First-level cache is enabled by default in Hibernate Application, and thereis no way to turn it off. All the operations we perform in our application through Session object first goes through the Cache.

Question 50: Why do we need First-Level cache, why is it mandatory in hibernate?
Cache is meant for storing the data temporarily within the memory, so that we canfetch the data back when we need it without going back to the source location, which saves the performance of the application.

Question 51: What is second-level cache in hibernate, what is the purpose of it and how does itworks?

Answere

second-level cache

For storing the long-lived data or moderate data, the first-level doesnt help as the data should be stored longer amount of time within the Cachewe need 2nd level cache.

Second Level Cache is also called as SessionFactory Level Cache. By default 2nd level cache is not turned on as part of Hibernate Framework. We need to enable 2nd level cache and should tag which entities has to becached at 2nd level cache to Hibernate Framework.

1.2nd level cache is not enabled by default, we should enable the 2ndlevel cache explicitly
2.We need to mark which entities has to be cached unless otherwise nodata goes to 2nd level cache
3.Every Entity will be stored in it own cache. (should be configuredwith cache policies)
4.Hibernate integrates with 3rd party cache frameworks in the market toimplement 2nd level cache.

Question 52: What is N+1 problem in hibernate framework

Answere

The N+1 problem arises when we are using the fetch="select". While fetching the parent objects using HQL query the hibernate will not querythe child associations, now while iterating each of the parent and fetchchild object of them it query the database in fetching the child.

If we have N members to fetch N members one query

and for each member to fetch their associated task one separate query permember level will triggered. which indicates 1 = parent, N queries for child N + 1 problem

This will degrades the performance of your application while iteratingand accessing parent objects.

How to solve N+1 problem?

In addition to the fetch="select", we need to even apply batch-size="N".

```
<set name="tasks" inverse="true" fetch="select" batch-
size="5">

  <key column="member_no"/>

  <one-to-many class="Task"/>

</set>
```

```
Query query =
session.createQuery("from
Member");List<Member> members
= query.list();
```

```
for(Member
  member :
  members) {
  tasks =
  member.get
  Task();

}
```

when we tried accessing child associated objects for a member tasks, instead of query task for each member hibernate goes to the database andfetch tasks of 5 members and loads into the parent objects.

when we use batch-size="N" , it indicates how many child associated collections should be loaded from the database at one shot instead of foreach parent.

if we have 100 members in the database

1 = members

20 = 100 member tasks

Design patterns

Question 52: What is Singleton Design pattern

Singleton Design pattern : Singleton pattern restricts the instantiation of a class and ensures that only one instance of the class exists in the java virtual machine. The singleton class must provide a global access point to get the instance of the class. Singleton pattern is used for logging, driver objects, caching and thread pool

How to create Singleton class in optimized way:

First of all constructor should be private.

Reason:

Because we are restricting other classes to creating the object of a class.

To create an object default constructor is mandatory if we make it as private then other people unable to create the object of the class.

Take a static method

Reason:

Because without creating object we can able to call the method of that class by using static method.

Take a static Singleton class type variable

Reason:

To store the object in this variable so that we can able to check whether the object is existed or not, we take static variable because we are using that in side static method.

The code that we are writing it will not work in multi threaded environment it will create problem.

In **eager initialization,** the instance of Singleton Class is created at the time of class loading, this is the easiest method to create a singleton class

Lazy initialization method to implement Singleton pattern creates the instance in the global access method.

Reflection can be used to destroy all the above singleton implementation approaches.

To overcome this situation with Reflection, Joshua Bloch suggests the use of Enum to implement Singleton design pattern as Java ensures that any enum value is instantiated only once in a Java program. Since Java Enum values are globally accessible, so is the singleton

Serialization destroys the singleton pattern, to overcome this scenario all we need to do it provide the implementation of readResolve() method.

Question 53: Wwhat is Factory Design Pattern

Answere

Factory pattern is one of most used design pattern in Java. This type of design pattern comes under creational pattern, to avoid complexity of creation object we should go for Factory Design pattern as this pattern provides one of the best ways to create an object.

In factory design pattern we can create object without exposing the creation logic to the client and refer to newly created object using a common interface. As we used factory class for making our classes completely loosely coupled

Question 54: What is Abstract Factory Pattern

(1) Abstract Factory pattern provides approach to code for interface rather than implementation.

(2) Abstract Factory pattern is "factory of factories" and can be easily extended to accommodate more products, for example we can add another sub-class Laptop and a factory LaptopFactory.

(3) Abstract Factory pattern is robust and avoid conditional logic of Factory pattern.

Question 55: What is Caching Design pattern

Answere

Cache is used for storing the data in it. So we are avoiding the round trips to visiting the database every time and storing the data into the cache.

Cache improves the performance of the application.

When to use the cache?

There are several places we can use the cache.

If data is common for throughout the application then go to cache concept.

If we want to share the data among the application then use cache.

Most of the time cache must be singleton only

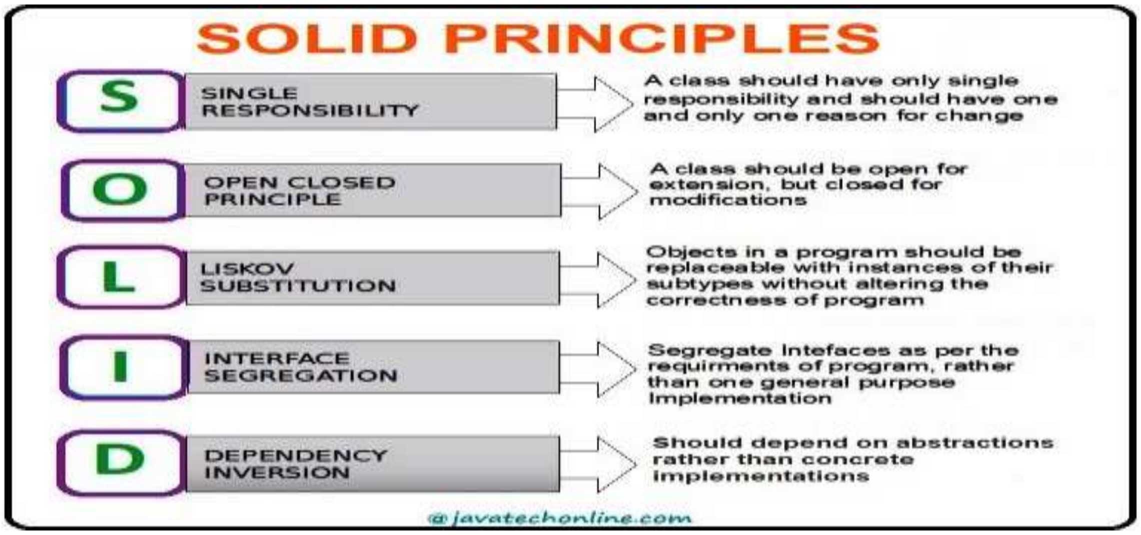
Because data remain same for every request then there is no need to create multiple object of the class.

Using cache we going to avoid duplication logic as part of the application.

Actually in cache data will be stored in the form of key and value.

A cache contains other member methods also which going to shared the state of the object in it.

Question 56: **What is SOLID DESIGN PATTERN**


SOLID PRINCIPLES
S
SINGLE RESPONSIBILITY
A class should have only single responsibility and should have one and only one reason for change
O
OPEN CLOSED PRINCIPLE
A class should be open for extension, but closed for modifications
L
LISKOV SUBSTITUTION
Objects in a program should be replaceable with instances of their subtypes without altering the correctness of program
I
INTERFACE SEGREGATION
Segregate Intefaces as per the requirments of program, rather than one general purpose Implementation
D
DEPENDENCY INVERSION
Should depend on abstractions rather than concrete implementations
@javatechonline.com

Question 57: **What is IOC?**

Answere

IOC is Standard design principle belongs to Object Oriented programming world and it is not provided by the spring people.It collaborating (injecting dependency) an object and managing/manage/release) of the object.IOC is the logical memory in the JVM memory.IOC container memory having two parts the lifecycle (create In memory METAD and Empty (for storing It is used to managing the object (key (id) & value (Object))).

dependency between the objects.

There are two ways to collaborating (injecting dependency) the object.

Dependency pull : Dependency pulling = is the old way of acquiring the objects which is already in place in jee and many languages.

1. Dependency lookup : Our class wants the other class object as dependency, we acquire dependent object by performing a lookup into a registry/container/repository in-short in someplace.

For e.g.. we will take DataSource object from jndi registry of the jee application server by performing lookup which is a form of dependency lookup.

2. Contextual dependency lookup : Contextualized Dependency Lookup. Contextualized Dependency Lookup (CDL) is similar, in some respects, to Dependency Pull, but in CDL, **lookup is performed against the container that is managing the resource, not from some central registry, and it is usually performed at some set point.**

Dependency Injection : To make our classes completely loosely coupled, we have to use dependency injection concept from IOC principle.While performing a dependency injection there are two components are compulsory.

And one component is depends on other components.

One is considered as target and another one is dependent.

Setter Injection = The dependent object is injected into the target class via setter on the target class attribute is called Setter Injection

Construction Injection = The dependent object is injected into the target class by passing dependent as an argument to the target class constructor.

Both are used for injecting dependent object into target class,then what is the use of 2 types of injections, what is the difference between them (or) when we should use what?

Difference between Setter and Constructor Injection

#1

In case of setter injection the dependent object will be injected into Target class after the target class object has been created. But in case of constructor injection the dependent object will be injected into target class during the time of creating the object of target class.

#2

In case of setter injection, injecting the dependent into Target class is optional, we can still create the object of target class and can skip calling setter to inject dependent. But in case of constructor injection without passing dependent as a constructor arg we cannot even create object of Target class, so construction injection makes dependency as mandatory.

Question 58: What is Bean Scope ?

Answere

Whenever we define a class as a bean in spring bean configuration file only one bean definition object will be created in the IOC container because by default the scope of the bean is singleton, if you want multiple object for that bean definition then we can control the instantiation of a bean using scope of a bean.

In spring there are 4 types of scopes are available.

SINGLETON

PROTOTYPE

REQUEST

SESSION

GLOBAL SESSION [port late application] deprecated and removed from spring 3.0.

Singleton Design pattern:

If a class allow to creating only one object of a class and the same object is being used by all the classes within (Class Loader) scope of an application then that class is called as singleton class.

There are several reasons available, why we are creating singleton class.

In some cases an object or a configuration will be common to whole application.

If everyone going to create the object of common thing then it is duplicating among the application, and we are wasting JVM memory.

If there is common requirement then create a singleton class which going to share same object throughout the application

PROTOTYPE

Prototype is another bean scope which is used for creating multiple objects as part of the bean.

IOC container will read the scope and depends on the scope it will create or return the object.

Question 59: What is Spring Boot, what does it provides?

Answere

Spring Boot is an module that has been provided by Spring Framework that addresses the non-functional aspects of building an application using Spring Framework.

it just helps using quickly using Spring Framework modules in developing application

Question 60: How does Spring Boot help us in building the applications quickly?

Answere

Spring Boot has provided 6 features helping us in speeding the developing of application while using Spring Framework.

1.Auto-Configurations

2. Starter dependencies

3. Embedded Servers

4. Actuator endpoints

5. DevTools

6. Spring Boot CLI

1. Auto-Configurations : Spring Boot auto configurations based on the opinionated view takes care of configuring Spring Framework components with default values automatically, we dont need write configuration information in configuring the framework components which eliminates most of the throw-away logic in building an application.

In-Short: Framework will selftune itself in making it usable to the application.

Opininated view in configuring/tuning up the Framework byitself.

Spring Boot looks for certain aspects in deriving which framework components are required to be autoconfigured for your application like

1. looking at a jdbc driver along with Spring Jdbc, it might opinionate that you might building database application will autoconfigure DataSource and JdbcTemplate

2. By looking at a bean definition like HibernateTemplate we configure in our application it might determine or thing we are using ORM and might autoconfigure entityManagerFactory other classes

From the above we can understand Spring Boot AutoConfigurations will automatically configure framework components based on opinionated view.

2. Starter dependencies : While working with Spring Framework we need to add spring module dependencies into the project to build project of that specific technologies.

Spring boot has provided starter dependencies. Starter dependencies are nothing but maven artifacts which are pre-built by Spring Boot developers, upon including them as part of our project will pull all the necessary module dependencies and external third-party libraries along with their compatible verions.

Spring Boot developers has created lot of starter dependencies based on the technologies of the projects we want to build, now its the responsibility of the developer to choose and include right starter as part of the project. So that developer will get jump start experience in building Spring Framework application.

To further help the developers in setting up the project and give an jump start experience Spring Boot has provided Spring boot initializer using which we can quickly create a project of our choice.

3. Embedded Servers : While working on distributed applications, we need to download, install and configure Servlet Containers externally to deploy and run our application, it becomes tedious job in running the application.

Huge amount of automation has to be done to achieve ci/cd integration. for eg.. to release the project for qa to test, we need to setup qa env, install servlet container, configure to delivery the application for qa to test it. But with addition of embedded servlet containers, the servlet containers are shipped as jar dependencies in our project, which doesnt require any additional installation at all.

From git we pulled the code, server is ready.

To run the application we need to start the servlet container from the source code, we have to write the code in configuring the server and deploying the application.

So to release the code to qa, we only need to pull the code and run it, rest of all the things are part of the source code itself.

From the above we can understand delivering the code across the env becomes quite handy and easily we can achieve ci/cd

Spring Boot supports 4 embedded servlet containers currently

1. tomcat (default)

2. jetty

3. undertow

4. netty (reactive application deployment)

Question 61: in Spring Boot how to Remove Embedded Tomcat Server, Enable Jetty Serve ?

Answere Exclude Tomcat – Maven Pom.xml

To exclude tomcat from spring boot, just need to add an additional block to the Spring Boot Starter dependency. In the dependency section, We can add

<exclusions> tags that make sure the given artifact is removed at build time.

```
<dependency>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-web</artifactId>

    <exclusions>

        <exclusion>
```

```
            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-tomcat</artifactId>

        </exclusion>

    </exclusions>

</dependency>
```

You can use that approach to exclude Tomcat from Spring Boot and also for any other exclusions

Question 62: How to Add Jetty Server in Spring Boot

Answere

If you want to use the Jetty server in Spring boot application, first you must need to disable the default tomcat server and then add jetty dependency "

spring-boot-starter-jetty".

```
<dependency>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-jetty</artifactId>

</dependency>
```

After adding jetty in pom.xml then at build time it disables tomcat and maps to the Jetty configurations.

Question 63: How to customize the embedded servlet container while working with Spring Boot?

Answere

In Spring Boot the embedded servlet container is managed through Spring ApplicationContext, So how do we need to customize in configuring and starting an embedded servlet container while working with spring boot?

There are 2 ways are there in customizing the embedded servlet containers

#1 configuration approach

we can configure the server configuration properties in application.yml or properties like below.

```
  server:

    port: 8081

    servlet-context:

      path: /vogo
```

The above properties are read by WebServerFactoryCustomizerAutoConfiguration and configures WebServerFactory bean definition with these values,

#2 programmatic approach

There are 2 ways we can customize the embedded servlet container in programmatic approach

webServerFactory.setPort(8081);

factory.setPort(2020);

4. Actuator endpoints : once the application has been developed and tested we cannot golive, because we need to add certain features/endpoints that helps us in monitoring and managing the application within production environment like

- healthcheck
- metrics
- logs

- statistics etc

unless otherwise monitoring and managing the application becomes quite complex and costlier. From this we can understand we cannot deploy an application straight after completion of its development, still we need to futher develop features that are required for our application to deploy in production env. This will delays the further deployment of our application in production.

We can overcome this problem by using Spring Boot Actuator endpoints. We can build a production grade deployable application straight from the development using actuator endpoints. The spring boot has provided pre-built endpoints as part of actuator, we just need to enable actuator endpoints to use them in monitoring our application.

Spring Boot Actuator is a prepackaged endpoints that are commonly required in monitoring and managing an application in production environment, which are build by the spring boot developers, that can be easily integratable into Spring Boot Application.

To enable or use actuator endpoints within our application, we just only need to add **spring-boot-starter-actuator** dependency in our project which will enable the endpoints by default

```
<dependency>

  <groupId>org.springframework.boot</groupId>

  <artifactId>spring-boot-starter-actuator</artifactId>

</dependency>
```

The actuator module has provided bunch of endpoints which we can use in monitoring our application

1. /info = provides the information about our application to the our partners/endusers like application name, author, version, features, licensing

2. /healthcheck = to check readyness and liveness of our application we use healthcheck

3. /env = to see all the environment variables configured or created on that machine

4. /beans = gives you all the bean definitions of the ioc container in our application

5. /configprops = all the properties we are using and injecting through @ConfigurationProperties can be accessed through /configprops

6. /threaddump = current jvm thread dump information

7. /metrics = memory, cpu utilization etc

8. /loggers = displays the information about loggers and their configuration

9. /logfile = shows the application log

10. /shutdown = we can remote shutdown our application

11. /sessions = displays the http sessions that are there in our application

12. /conditions = the conditional annotations resulted to true based on which auto configurations are executed will be displayed by this endpoint

5. DevTools : DevTools stands for developer tools, which provides handy utilities that helps the developers in reducing the time in debugging the application during development.

If there are changes we made within our source code post deployment of our application within the server, the devtools uses specialized classloader to reload specific class files into jvm memory without the need of repackaging, redeploying and restarting the server, which will saves lot of development time.

In addition in provides live-reloader support where any changes we made in UI pages like HTML/JSP/Thymeleaf will be reflected automatically.

6. Spring Boot CLI: Spring Boot Command-Line Interface a tool that can be installed on our machine to run Spring boot application quickly.

CLI stands for command-line interface through which we can quickly execute the psudeo code of your application.

Maven commands : mvn archetype:generate -DgroupId=boot -DartifactId=bootbasicparent -Dversion=1.0 -DarchetypeArtifactId=maven-archetype-quickstart

Question 64: What is @SpringbootApplication Annotation

Answere

@SpringBootApplication annotation internally imports 3 other annotations

@Configuration

This annotation marks a class as a Configuration class for Java-based configuration. This is particularly important if you favor Java-based configuration over XML configuration.

@ComponentScan

This annotation enables component-scanning so that the web controller classes and other components you create will be automatically discovered and registered as beans in Spring's Application Context. All the@Controller classes you write are discovered by this annotation.

@EnableAutoConfiguration

This annotation enables the magical auto-configuration feature of Spring Boot, which can automatically configure a lot of stuff for you.

For example, if you are writing a Spring MVC application and you have Thymeleaf JAR files on the application classpath, then Spring Boot auto-configuration can automatically configure the Thymeleaf template resolver, view resolver, and other settings automatically.

So, you can say that @SpringBootApplication is a 3-in-1 annotation that combines the functionality of @Configuration, @ComponentScan, and @EnableAutoConfiguration.

Question 65: How to run Spring Boot application, what happens when we trying running the Spring Boot Application?

#1

We need to create an Application class under the Root package of our application.

```
package com.bootbasic;

public class BBApplication {

  public static void main(String[] args) {


  }

}
```

#2 Annotate your Application class with @SpringBootApplication

#3 create the ioc container by using SpringApplication.run

ApplicationContext context = SpringApplication.run(BootBasicApplication.class);

1. It creates an empty environment object

2. it detects the external configuration of your application and loads into the environment object

3. print banner

4. it identifies the type of application by detecting the classpath of your application

4.1 if mvc dependencies found, then it treats WebApplicationType = WEB and instantiates

AnnotationConfigServletWebServerApplicationContext

4.2 else if webflux dependencies are found, then it treats the WebApplicationType = REACTIVE and instantiates

AnnotationConfigReactiveWebServerApplicationContext

4.3 else it treats WebApplicationType = NONE and then instantiates

AnnotationConfigApplicationContext

5. Instantiates Spring Factories and registers with ioc container

6. executes ApplicationContextInitializer

7. prepareContext()

8. refreshContext()

9. executes CommandLineRunners and ApplicationRunners to initialize the application before startup(), once after it returns the object of ioc container to the application.

10. while performing the above steps, at various different stages SpringApplication will publish the events and triggers the listeners

Question 65: What are profiles what is the purpose of it (@Profile)?

We can use profiles for switching from one environment to another environment easily.

```
@Component

class ConnectionManager {

  @Value("${db.driverClassname}")

  private String driverClassname;

  @Value("${db.url}")

  private String url;

  @Value("${db.username}")

  private String username;

  @Value("${db.password}")
  private String password;

  // accessors
}

db-dev.properties
--------------
db.driverClassname=com.mysql.cj.jdbc.Driver
db.url=jdbc:mysql://localhost:3306/db
db.username=root
db.password=root

db-test.properties
------------------
db.driverClassname=com.oracle.jdbc.Driver
db.url=jdbc:oracle:thin:@localhost:1521/xe
db.username=sys
db.password=welcome#123

@Configuration
@PropertySource("classpath:db-dev.properties")
@Profile("dev")
class DevConfig {

}
@Configuration
@PropertySource("classpath:db-test.properties")
```

```
@Profile("test")

class TestConfig {
```

SpringApplication class has been designed to support profiles by default. by default SpringApplication class reads application.properties. To support different profiles SpringApplication has been designed to application-[profile].properties

when we set the profile automatically which is Spring Boot profile support.

Question 65: What is Monolithic Application architecutre?

An application is built out of single source code project where the components across the modules of the application are referenced directly (through the classpath), and the entire application is being deployed and running under one jvm.

Advantages :

1. Easy to develop The IDEs and tools available are designed in support of monolithic application development, so developer can quickly build the application

2. Easy to deploy

The entire application is packaged as a single deployable war/ear, we just need to copy to the target runtime to run it

3. Easy to Scale

We can create quickly the cloned environments to run the application on multiple machines

Disadvantages :

1. overloaded ide

2. overloaded server runtime

3. difficult to achieve re-usability through modularity

4. due to the complexity in understand the sourcecode, the developer will result in poor implementation of the code which will degrades the quality of source code

5. monolithic applications are not easy to be scaled

5.1 only supports one dimensional scaling (horizontal scaling)

5.2 the cost of scaling is very high, because we need to scale always the entire application rather than a single module which requires huge computing capacity

Question 66: What is Microservices Architectre Advantages and Disadvantages

Answere

We decompose the large-scale business systems into multiple smaller services which has the below characteristics

1. loosely coupled

2. independently deployable

3 highly scalable

4 resilient

5 Collaborative

that can be developed by a smaller collaborative team of developers independently

advantages:-

1. Each service is built out of its own source code project

1.1 Independent teams can work in parallel in building and delivering the services which promotes parallel application development

1.2 New developers can quickly understand the application and can become productive

1.3 The developers can understand architecture of the application, so that they produce quality of code and encourages modularity through which we can achieve reusablity and thereby application is easily maintainable

1.4 IDE are not overloaded

1.5 Server Runtimes can quick start due to smaller application size, so developer time will not be wasted during debugging the application

2. Fault Isolation

3. Independently Scalable

4. Can implement CI/CD practices easily

5. Testability of the services are easy as those are

small in size

6. We can adopt quickly the new emerging technologies

for a part of system rather than whole

dis-advantages:-

There are several drawbacks and complexities involved in building applications based on microservices architecture

1. developer has to deal with complexity interms of building distributed services

2. existing IDEs doesnt have support for building distributed solutions based on microservices architecture so developer find it very difficult being productive during development

3. more complexity interms of inter-service communication

4. has to deal with intermediate failures between the services and has to write the exception management logic or retry logic in dealing with such service failures

5. the more the number of service deployments the more consumption of jvm memory and the cost of deployment increases

6. testing the inter-service communication is highly complex

Question 67: What is the support of Spring Framework interms of microservice development?

As part of Spring Cloud + Spring Microservices (latest) has provided below set of tools

1. Eureka Server integration

2. Spring Cloud Config Server/Config Client (Replacing Zookeeper)

3 Spring Circuit breaker (Replacing Hystrix)

4. Spring Api Gateway (Replacing Zuul)

5. Spring Load balancer (Replacing Ribbon)

6. Feign Client Api

Question 68:What is Microservices Architecture

Answere

Discovery Server : Discovery server uses Discovery Registery like Eureka Server or Apache consul

Every service when it is coming to microservces. Get theself registred with Eureka server or consul.I am this service as raw material service spare part service.i am a inventory management services. Every services registered ther service in eureka server.Eureka server keeps the track of the services node information which service is running on which nodes.When the client wants to acces s the microservices Clent has to send request to eureka server asking that what is end point url or where is uri address or ip address on which it was running it want this particular services but he not know on which nodes it run.

So discover server knows which microservices is running on which node.it helps in identifying the microservices on which loction or node it is running . It will helps only in identifyinf the microservices on which node it is running .Eurek server find the information from its discovery registry

Suppose after registering in discovery registry micoservice got crashed and client wants to CCESS THE MICROSERVICES IT GOT 404 error . That's why to avoid this condition Every microservices during the bootstrap after registering the informstion periodically every microservices send health beat Information to the Eureka Server. That he is alive.it is also called   "**Heart Beats".**There after microservices send the heartbeats publishing their live info. If any microservice not registered their liveness to the discovery registered periodically the eureka server deregisterd that microservice fro discovery registery.so its act as a Discovery Engine .

discovery client : The discovery client api is used for communicating with eureka server in identifying the information about where the microservices instances are running. once we get the information we need use RestTemplate (api) for invoking the microservice

Spring Cloud Load balancer: spring cloud loadbalancer has provided an MethodBeforeAdvice internally which contains the logic for discovering the service instances available on the cluster by talking through eureka server and apply round-robbin algorithm in choosing the service construct the url with node details.

and the above advice has to be applied on RestTemplate so that every method call on the RestTemplate class will gothrough Advice and discovery takes place and replaces the URL with which the RestTemplate will invoke the service.

To let the Spring Cloud know we want to apply Loadbalancer advice on RestTemplate we need annotate the RestTemplate bean definition **with @LoadBalanced** so that while creating the object of RestTemplate it applies the advice and creates a proxy and stores the proxied RestTemplate object within the ioc container.

Client side Load Balancer or Ribbon   Suppose there is 5000 client request for raw material services which is on 2 machine as scale up but 5000 requst is on one raw materil services so other machine sit idle and load increase on machine one and it will gone crash . That's why we need client side Load Balancer or Ribbon.Client send the request to Eureks server and ask the information now Eurekaserver get the details of the two raw material microservies end poin or url to the Client side load balancer or ribbon. Now Ribbon on Ron Robbin Algorithm choose the one url and send the request that’s how it manage the traffic on the base of round robin request . It distribute the trafc in Round robin Algorithm

It is mandatory client has to use client side Load balancer or ribbon.

Jenkins : Suppose there is 4 nodes now we configure it in jenekins pipe line. Devops Engineer going to write automation code as installing java starting tomcat. And copying the war fule in particular machine and start the tom cat server . When we wants to scale up the service we create pipe lines in jenekins and gave the machine ip address. Each microservices code pull from the git repository through the Jenkins pipe line and code deployed on respective machine by using shell scripting by ssh on remote machine . from gits it promtes code to jenekins pipeline to j frog repository . now artifact comes to jfrof repository. Now we build deployment pipeline where we take the machine ip address where we wanted to Deployed the application . Mow from jfrog repository the war file file br on remote system via continuous deployment and install tomcat server and jdk 11. Now when tomcat started it registered with eureka server during the boot strap

AWS(Ec2) Instances: I want to scalleup the raw material managemet service in jenekins deployment pipe we will give link or ip address of the raw material services . And we got the computer from AWs cloud plateform as we created Ec2 instances by using **teraform** we may quickly create instances on aws cloud. While we creating Jenekins we create Terraform scripts for creating Ec2 instances . Through Jenekins pipe line it is easy to scale up our Application .

Spring Boot Jars : Spring boot jar replaced tomcate server through this we only neede machine wiwth jdk and jar will execute easily without tomcat so it is fast and easy so it is used for ci/cd frequently

How to Monitor The Microservices ?:

We monitore through Monitoring tools AppDynamics Nagios. AppDynamics and Nagios know how the microservices run is heslthy or not via Accutaor Api

Splunk and Kibana pull the data from AppDynamics and nagios and monitor continuously that which service is running and based on this they go to jenekin pipeline and trigger the application back

(1) For Messaging we have IBM mq/Kafka or Rabbit Mq
(2) For Caching there is Redissh Cache
(3) For Securtity we have Spring-Security
(4) For Transactionality we have Saga Design Pattern
(5) For Logging we have Log4j

Question 69: What do you mean by Open Feign Client?

Answere

open feign : Spring Cloud OpenFeign is a declarative REST Client library that can be used for accessing microservices without writing any client code aspart of our application.

There are many features of feign client which makes it more powerful

1. it is a declarative client library where we are going to annotate the interface methods with spring mvc annotations or openfeign annotations using which we are going to access the microservice

2. the openfeign client is highly customizable through a set of components like

- decoders
- encoders
- loggers
- LoadBalancers
- FeignBuilders

Instead of we writing the HttpInvoker and HttpInvokerAdvice which will automate the process of invoking the HttpEndpoint based on metadata declaration, the OpenFeign api has provided the above components, which we can make use of invoking the REST/HTTP Endpoints.

```
@FeignClient(url="http://localhost:8081/inventorymgmtse rvice/stock/")

interface StoreService {

@GetMapping(value="/{productName}/price", produces

= {"text/plain"})

public double getProductPrice(@PathVariable("productName") String
productName);

}
```

```
@SpringBootApplication @EnableFeignClients

class FeignClientApplication {
```

Question 70: What is circuit breaker?

Answere

Circuit breaker is one of the integration-tier design pattern which is used for protecting the client applications while accessing the remote services.

The circuit breaker is not an spring framework feature or is not only applied for microservices, it an independent design pattern of its own and can be applied anywhere at the client-side when we are accessing remote services in case of

- soap services
- http endpoints
- ejb / rmi invocations

ofcourse it is not limited by language, it is adopted and implemented by many programming languages as it is a design pattern

Spring Cloud Circuit Breaker has provided their own interface with each vendor specific implementation class wrapping the vendor api. Now depends the vendor api we want to use we need to instantiate and configure Spring provided implementation for Vendor api.

To instantiate the Spring Circuit Breaker implementation class, Spring cloud has provided as Factory class

```
interface CircuitBreaker {

T run(Supplier T, Function<Throwable> fallback);

}
```

For this interface Spring Cloud has provided one implementation per one CircuitBreaker vendor library Spring supports.

```
class HystrixCircuitBreaker implements CircuitBreaker {
com.netflix.hystrix.CircuitBreaker cb;

public T run(Supplier<T> s, Function<Throwable> t)

{
```

```
return cb.run(s);

}

}
```

The above class is a wrapper on top of HystrixCircuitBreaker library, which invokes hystrix api classes

```
class Resilience4JCircuitBreaker implements CircuitBreaker {}
```

for the above factory class spring cloud has provided 1 implementation for one vendor

```
class HystrixCircuitBreakerFactory extends CircuitBreakerFactory {

public CircuitBreaker create(String id) {
```

the create method will instantiate the vendor circuitbreaker object and wraps into Spring provided vendor CircuitBreaker implementation and returns to us.

```
}


}
```

```
class Resilience4JCircuitBreakerFactory extends CircuitBreakerFactory
{

public CircuitBreaker create(String id) {
```

the create method instantiates Resilience4J vendor api and wraps into Spring provided vendor implementation class Resilience4JCircuitBreaker object

}

}

Spring Cloud Circuit Breaker

is an wrapper that is provided on top of various different third-party implementations of Circuit breaker libraries. by using Spring Cloud Circuit breaker we can switch from any of the third-party libraries of the Circuit Breaker like

- Hystrix
- Resilience4J
- Spring Retry
- Sentinel

Spring cloud circuit breaker has provided an CircuitBreaker interface with one implementation per one third-party library, where the implementation wraps the Third-party library api inside it.

1. CircuitBreakerConfiguration = we can configure threadshold limits in managing the CircuitBreaker.

The circuitbreaker can be in one of the three states

1. OPEN = requests are no more allowed to the remote service
2. CLOSED = allows the requests to the remote service
3. HALF-OPEN = the number of requests that can be allowed to the remote service before the circuit is fully closed.

2. RateLimitter = RateLimiter configuration helps us in limiting the number of requests to an Remote Service

3. TimeLimitter = how long the client has to wait for a response from the remote service can be configured through TimeLimitter

4. BuilkHeadProvider = Number of concurrent requests allowed to the remote service can be managed through BulkHeadProvider

5. Retry = In case of failure in accessing an RemoteService, we can specify retry automatically.

Question 71: What is Spring Cloud Gateway

There are few common services/requirements we want to enforce across the microservices of our application like

1. **security**
2. **routing**
3. **transformation**
4. **data aggregation**

cloudgateway acts as an single entry point or gateway of receiving microservice requests from the client.

1. Security

So that we can apply security at the gateway level, which indirectly secure all endpoints of the microservices as it acts as an entry point

2. routing

We can configure routes with predicates/conditions based on the request received we can route to a different versions of the microservices

3. transformations

we can modify both request body and response body during the time of request and response by attaching filters to the gateway asking him to apply

4. aggregation

we can write the logic at gateway in calling multiple microservices and aggregate the data and return to the client application

There are lot of api gateway libraries are available in market

1. zuul

2. apigee

in addition to the third-party libraries spring has provided spring-cloud-gateway module to implementation api gateway functionality which works based on reactive streams

Spring Cloud Gateway supports

1. routing based on predicates

2. transformations through filters

3. any how using spring security we can enfore security at api gateway level

but spring cloud gateway doesnt support aggregations.

Question 72: How does the architecture of spring cloud gateway?

Answere

There are 3 major components are there in api gateway

1. **routing =** routing is used for mapping an incoming request to the appropriate microservice

2. **predicates** = conditions to be evaluated based on different parameters of the http request to map the request to the corresponding microservice

3. **filters =** used for applying pre/post process of the request/response while invoking the microservice

There are 2 ways of working with api gateway

1. **through configuration approach**

2. **programmatic api**

Question 73: how to decompose the business system into microservices?

There are 4 ways are there based on which we can identify or break down the system into microservices

1. based on business capabilities

2. based on sub-domain

3. single responsibility principle

4. service per team

Question74: How do we design the database for a microservice based application?

There are 2 patterns in designing the database for microservice application

1. **database per service**

it is recommended to design the database schemas based on service per db pattern so that the microservices will be completely loosely coupled and maintainable

2. shared database

there are reasons why we need to go for single database

/ shared database across the microservices

2.1 we want to implement acid transactions across the microservices

2.2 we want to access the data across the microservices more oftenly

2.3 if we are migrating an legacy application where a shared database is already being used then we can continue use the shared database rather than decomposing

2.4 simple to operate

Question 75: How does microservices communicates with each other?

Let us say we have requirement where a microservice in order to fullfil the functionality it has to gather the data by talking to multiple microservices how can we aggregate the data and communicate with multiple microservices in our application?

There are 2 design patterns we need to apply in intra- communication of the microservices.

1. api composer

implement an api composer service in which we invoke the microservices and performs data aggregation in- memory and returns

advantages:

1. microservices are loosely-coupled

2. resuse the service callouts and data aggregation logic across the services whoever requires the same information

dis-advantages:

1. in-memory data aggregation is not suitable for large volumes of data

2. results in performance problems, if the data has to be access more frequently

#2. command query responsibility segreggation (cqrs)*****

each microservice upon performing an action or an operation will trigger or publish event with the data containing or describing the change.

we register listener for all the events published by the individual services, receives the event performs the processing and stores the data in read-only database.

So the aggregator service will receive the request and fetch the data from read-only

database and returns rather than querying the data across all the services. With this all the dis-advantages we have discussed or resolved.

advantages:-

1. each microservice is loosely coupled

2. no repeated network calls in fetching the same data from the other microservices

3. no in-memory data aggregation so efficient usage of the memory

4. no network congession

5. no consumption of network bandwidth which results in better performance of the microservice

dis-advantage:

1. highly complex to design and implement

Question 76: How to manage transactionality when it comes to microservices?

Answere

To apply transactionality across the microservices we need to implement saga design pattern.

There are 2 ways/types of implementing the saga design pattern

1. **choreography-based saga**
2. **orchestration-based saga**

Saga design pattern

In saga design pattern each microservice performs the operations on their database in local transaction by coordinating and applying an compensating operation incase of a failure with the other.

There are 2 types of saga design pattern

1. choreography-based saga
2. orchestration-based saga

#1. choreography-based saga.

upon receiving an request by a microservice to carry an operation, the microservice performs the business operation and stores the data in its database under local transaction, and inorder to propagate perform an related operation by the other microservice to complete

business operation, it publishes an event in the topic asking the other microservice to complete the business operation.

The other microservice receives the event through listener and performs the operation and based on the outcome of the operation it again publishes an event in reply back topic to the initiator/callee service.

In case of failure the callee microservice executes an compensating transaction to bring the system into consistent state which is called saga design pattern

#2. orchestration-based saga

in case of choreography saga, the initiator service has to be written with lot of logic in coordinating and communicating by exchange messages and apply compensating operations to achieve data consistence and transactionality.

So that the logic in the microservice has been polluted along with business logic even cooridination and transaction managment logic as well, which can be separated out and can be written in another component called orchestrator.

upon receiving the request by the order service, it pass down the request to the orchestrator component to carryout the operation and enforce transactionality through message exchange. So that the original service will have only business logic being implemented and transaction coordination logic is taken care by orchestrator.

For each business operation that spans across the multiple microservices we are going to write one orchestrator component.

Apache Kafka

Apache Kafka is an opensource distributed event platform that works on Streaming api. Apache Kafka is majorly used for building microservices based applications.

Using Spring Boot Kafka Integeration api we can easily build consumers and producers in reading and processing events on Kafka cluster

The spring boot kafka module has provided KafkaTemplate which helps in publishing an message to the specific kafka topic we specified.

Similar we can build Kafka Consumers through Listener api where listerner looks for a message in the kafka topic and when new messages arrives automatically triggers and invokes listener by passing the message for consumption

Interview - Booster

The End