**JAVA**

**1)Abstraction vs interface**

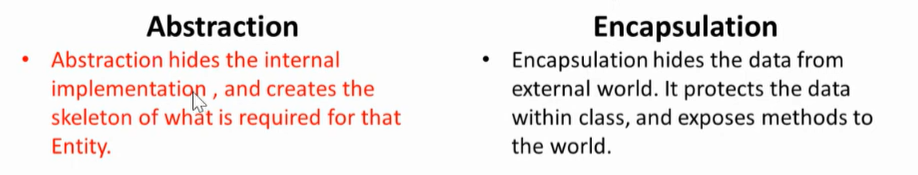
-Abstraction: If you are hiding the internal implementation and showing only the functionality then we call as an abstraction. A class that starts with an abstract keyword and we can't create the object of this class. this class contain abstract as well as concrete methods. this class contain instance and static variables and child class is implementing this class then need to provide implementation of this class abstract method.

-Interface: If we want to achieve the full abstraction, we can go with an interface in which we can't create objects and this class contains all abstract methods by default

all methods present in class are public abstract and variables are static final by default.

-If you have Service requirement specification we should go with abstraction.

**1.1) Abstraction vs Encapsulation**



-when we have to do abstraction object needs to -but in case of encapsulation object doesn’t need Encapsulated to achieve the abstract. Need to be abstracted.

**2) Difference Between**

-HashMap: It's not Thread Safe so performance is high and multiple threads can operate on the map.

-single null key and multiple null values allowed

-HashTable:

-It's Thread Safe so performance is low as the thread needs to wait to get locked. It will lock the entire bucket.

-not even a single null key and null value are not allowed

-SynchronizedMap: -It's Thread Safe so performance is low as the thread needs to wait to get locked. It will lock the entire bucket.

w - null key and multiple null values allowed

-ConcurrentHashMap:

-Its thread safe perfoemance is high as it using segment lock not entire bucket is locked .

-null insertion of key and value are not allowed

**3)OOPS**

**4)String**

**5)Immutable class**

**6)Singleton design pattern and class**

**7)Thread**

**8)exception**

**-exception propagation when we have compile time exception at parent class then only we think about the child class else in runtime exception we don’t need to think**

**9)Collection List Vs set**

**10)hashMap vs hashTable  
10 a)Concurrent HashMap vs HashTable**

**11)wait vs sleep diff:**

wait: It belong to object class which is parent class of all the classes.it will releases the lock of object during synchronization.

Sleep: it belong to Thread class .It will not release the lock of object during synchronization so thread need to wait until particular time to execute.

**12)SOLID design pattern:**

**S:Single Responsibility principle:**

**O:Open close Principle:**

**L:Liscove Substitution Principle:**

**I:Interface Segregation Principle:**

**D:Dependancy Inversion Principle:**

**13)Design pattern**

**I)Singleton,**

**Builder,**

**Factory  
Abstract Factory**

**15)tell me five http header code that you put**

**16)tell me five http status code**

**200,2001,403,404,400,503,500**

**17)why wait notify and Notify all in Object class**

-As every object in java has only one lock or monitor. And as wait notify ad notify all are used for monitor sharing.

-This all method works on Locks. And lock are associated with object not on thread.

18)What is Intern() and join method of String Class.  
**Intern()-**Suppose I am having two string object with literal implementation and one new keyword implementation object

Ex:

String s1=”Shubham”;

String s2=”Shubham”;

String s3=new String(“Shubham”):

SOP(s1==s2);//true as both are pointing to same object in scp

SOP(s1==s3);//false as s1 is at scp and s3 is at heap

So if I want to copy the obj reference of s3 to scp we use intern method

String s4=s3.intern();

SOP(s1==s4);//true;

**JOIN():This Join method to join the string based on the delimiter provided by You**List<String> l = Arrays.*asList*("1", "2", "5", "6");

String op = String.*join*("?", l);

System.***out***.println(op);

1?2?5?6

**18)Fail fast and Fail Safe Iterator**

-In case of Fail fast it used the actual copy and when we do any crud operation while performing iteration it will through concurrent Modification Exception. As it taking and comparing with original class obj

Q)how inheritance and composition is different in java

-Inheritance and composition both are use to create the relationship between the classes.

but they serve different purposes and have different implications.

Inheritance:

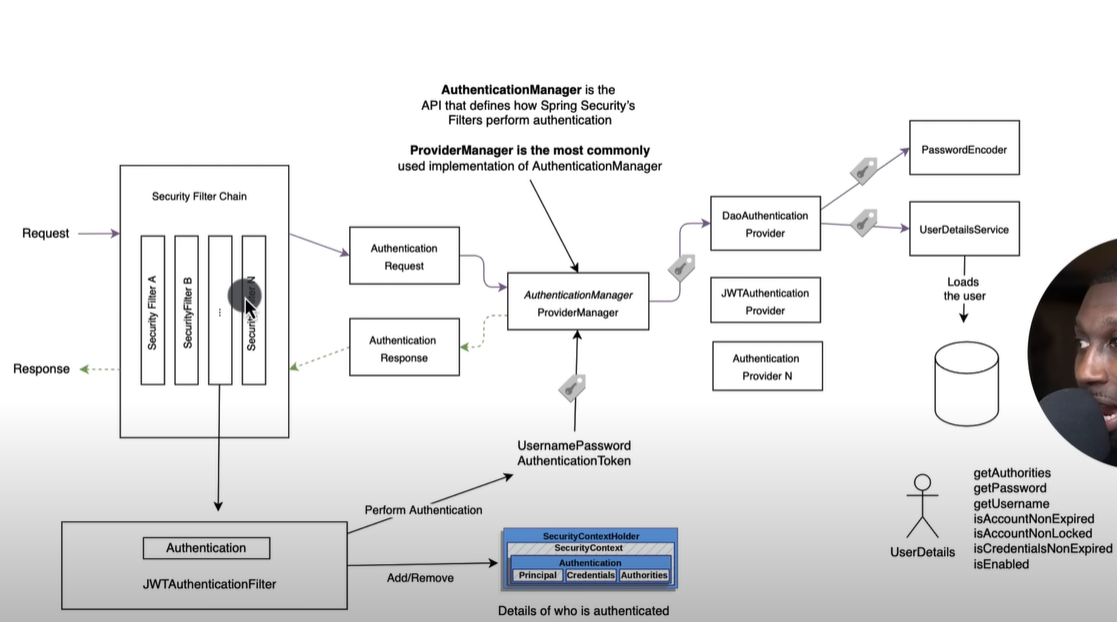
Definition: Inheritance is a mechanism that allows a new class (subclass or derived class) to inherit properties and behaviours from an existing class (superclass or base class).

Relationship: It represents an "is-a" relationship. For example, a Car class might inherit from a more general Vehicle class.

Code Reusability: Inheritance promotes code reusability by allowing the subclass to reuse the code of the superclass.

Access to Superclass Members: The subclass has access to the public and protected members of the superclass. It can override methods of the superclass to provide its own implementation.

Keyword in Java: In Java, the extends keyword is used to establish inheritance.

****

**JAVA 8**

**1)What is an Intermediate Operator in Java 8?**

-Intermediate Operators are the method which are used to process the element of the Stream and it does not return us the value but it will return us

stream or process stream.

EX: filter(),Map(),FlatMap(),Disctinct(),Skip(),Limit ().

**2)What is a Terminal operator in Java 8?**

-This are the method which takes i/p stream and returns us the values after processing. and once a terminal operation is called, the stream cannot be reused for further operations.

E0X: findFirst(),COunt(),ForEach(),Collect().

**3)find the employee from each department whose salary is maximum in each department using java .**

**-I am having list of employee with salary and department entity**

Map<String, Employee> collect2 = list.stream().collect(Collectors.*groupingBy*(Employee::getDepartment, Collectors.*collectingAndThen*(Collectors.*maxBy*(Comparator.*comparing*(Employee::getSalary)), Optional::get)));

collect2.forEach((a, b) -> {

System.***out***.println(a + " " + b.getSalary());

});

O/p:   
MNGR 14100

PHP 29100

HR 38000

IT 28000

**1)find sec highest salary?**

Employee employee = list.stream().sorted(Comparator.*comparing*(Employee::getSalary).reversed()).distinct(.skip(1).findFirst().get();

System.***out***.println(employee);

**2)Frequency of names**

**2.1)Find duplicate from array using set or an using grouping by**

Set<Integer> set = **new** HashSet<>();

List<Integer> collect3 = ll.stream().filter(z -> !set.add(z)).collect(Collectors.*toList*());

System.***out***.println(collect3);

// OR

List<Integer> collect4 = ll.stream().collect(Collectors.*groupingBy*(i -> i, Collectors.*counting*())).entrySet()

.stream().filter(entry -> entry.getValue() > 1).map(entry -> entry.getKey())

.collect(Collectors.*toList*());

System.***out***.println(collect4);

**2.2)Find the frequency of String**  
 // find frequency of String

Map<Character, Long> collect5 = s.chars().mapToObj(x -> (**char**) x).filter(z -> z.*valueOf*(z) != ' ' && z.*valueOf*(z) != ',').collect(Collectors.*groupingBy*(i -> i, Collectors.*counting*()));

System.***out***.println(collect5);

**3)Map vs FlatMap**

**MAP:** *It applies a given function to each element of the stream and returns a new stream with the transformed elements.*

*Only perform the mapping.*

*Produce a stream of value.*

*map() is used only for transformation.*

**FlatMap:** It applies a function to each element of the stream and then flattens the result into a single stream.

Perform mapping as well as flattening.

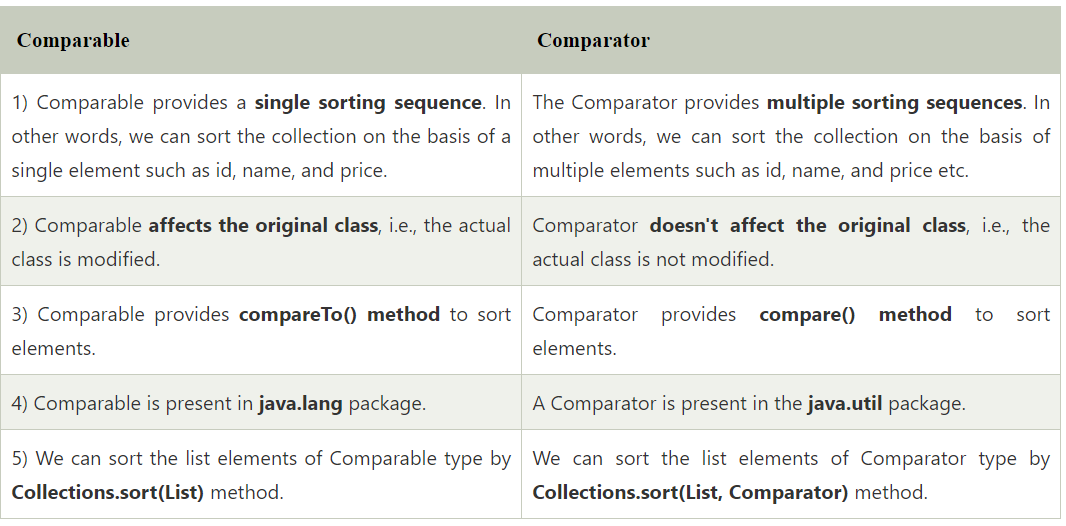
Produce a stream of stream value.

flatMap() is used both for transformation and mapping.

**4)Grouping By**

**5)Stream Api**

**Q. Difference between comparable and comparator**

****