Features of Java 8

- Functional Interfaces and Lambda Expressions
- forEach() method in Iterable interface
- Optional class,
- default and static methods in Interfaces
- Method references
- Java Stream API for Bulk Data Operations on Collections
- Java Date Time API

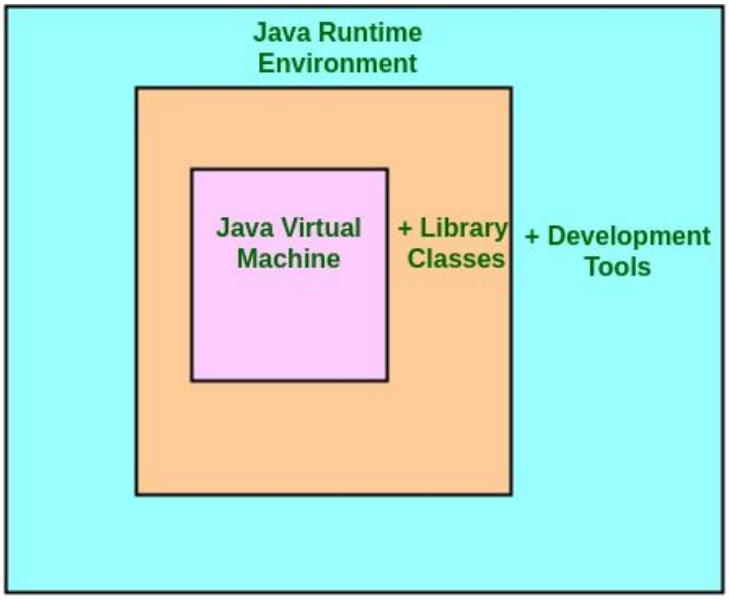
What is concrete class?

A concrete class in Java is a type of subclass, which implements all the abstract method of its super abstract class which it extends to. It also has implementations of all methods of interfaces it implements.

What is the difference between abstract and concrete class?

1. Modifier: An abstract class is declared using abstract modifier. Concrete class should not be declared using abstract keyword, on doing so, it will also become an abstract class.
2. Instantiation: An abstract class cannot be instantiated directly, i.e. object of such class cannot be created directly using new keyword. An abstract class can be instantiated either by a concrete subclass or by defining all the abstract method along with the new statement. A concrete class can be instantiated directly, using a new keyword.

JVM, JRE, JDK?



JDK = JRE + Development Tool
JRE = JVM + Library Classes

What is MVC?

The **Model** contains the underlying data, state and logic that the user wants to see and manipulate.

The **View** presents the model information to the user in the way they expect it and allows to interact with it. A model could have several views that present different parts of the model or present the model in different ways.

Controller interprets the user's interaction with elements in the view, and modifies the model itself.

Advantages of MVC?

1) Controllers make the code better in the following ways:

The view can focus on its main purpose: presenting the user interface, as the controller takes the responsibility of interpreting the input from the user and working with the model based on that input.

The view and the model are not “tightly coupled” when a controller is between them. Features can be added to the model and tested long before they are added to the view.

2)Controllers make the code cleaner and easier to modify.

3)It allows the program to be modular and loosely coupled. Loose coupling is key for large development teams – it allows different teams to work on different parts, so one team can work

on the only the view, and another team can work on only the model. The view and the model can change, without needing knowledge of the other.

What is Immutable Class?

Immutable class 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.

Following are the requirements:

- The class must be declared as **final** (So that child classes can't be created)
- Data members in the class must be declared as final (So that we can't change the value of it after object creation)
- A parameterized constructor
- Getter method for all the variables in it
- No setters(To not have the option to change the value of the instance variable)

Integer, Boolean, Byte, Short these classes don't modify their state, however they create a new instance each time you try to modify them.

```
Integer a =3;2
a += 3;
```

After calling a += 3, a new instance is created holding the value: 6 and the first instance is lost.

Garbage Collection ?

Garbage collection is the process of freeing space in the heap or the nursery for allocation of new objects

Serialization and Deserialization in Java ?

Serialization is a mechanism of converting the state of an object into a byte stream.
Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory
Only the objects of those classes can be serialized which are implementing java.io.Serializable interface.Serializable is a marker interface
A Java object is serializable if its class or any of its superclasses implements either the `java.io.Serializable` interface or its subinterface, `java.io.Externalizable`.

transient variables:- A variable defined with transient keyword is not serialized during serialization process.This variable will be initialized with default value during deserialization. (e.g: for objects it is null, for int it is 0).

static Variables:- A variable defined with static keyword is not serialized during serialization process.This variable will be loaded with current value defined in the class during deserialization.
Only data associated with a specific instance of a class is serialized, therefore static member fields are ignored during serialization.

final fields :- This value is part of an Interface. But as this is constant hence it is saved while serialization.

transient and static & transient and final :- It has no effect, follows the behaviour of static and final

example-link

Marker interface ?

It is an empty interface (no field or methods). Examples of marker interface are **Serializable, Clonnable** and **Remote** interface. All these interfaces are empty interfaces.

REST?

Representational State Transfer (REST) architectural style - It is a set of principles Which follows some standards for implementing API.

The six constraints are:

- ➔ Uniform Interface
- ➔ Stateless
- ➔ Cacheable
- ➔ Client-Server
- ➔ Layered System
- ➔ Code on Demand (optional)

https://www.linkedin.com/learning/designing-restful-apis/rest-apis-the-six-constraints-part-1
https://www.geeksforgeeks.org/rest-api-architectural-constraints/
https://www.dineshonjava.com/what-is-rest-and-rest-architecture-and-rest-constraints/

Which monitor is used when a static method is synchronized?

Static methods do not belong to individual objects. They belong to the class.
And every class loaded by the JVM is simultaneously represented by an object of type Class.
When a thread attempts to enter a synchronized static method, it will ask for the monitor of the Class object that represents the class within which the static method exists.
http://www.programmr.com/blogs/which-monitor-used-when-static-method-synchronized

How Monitor is acquired in case of static method?

For static method in class, the monitor of class objects is acquired.
For static method in interface we cannot use synchronized keyword.
https://stackoverflow.com/questions/18685476/how-is-monitor-acquired-in-case-of-static-function

What is Upper Bound In Java?

Upper-bound is when you specify **(? extends Field)** means argument can be any **Field or subclass of Field**

What is Lower Bound class of a type Wildcard?

Lower-bound is when you specify **(? super Field)** means argument can be any **Field or superclass of Field.**

What is the Upper bound of a type wildcard? Or Upper Bounded Wildcards?

https://docs.oracle.com/javase/tutorial/java/generics/upperBounded.html#:~:text=To%20declare%20an%20upper-bounded,

The Twelve Factors

I. Codebase
    One codebase tracked in revision control, many deploys
II. Dependencies
    Explicitly declare and isolate dependencies
III. Config
    Store config in the environment
IV. Backing services
    Treat backing services as attached resources
V. Build, release, run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

VII. Port binding

Export services via port binding

VIII. Concurrency

Scale out via the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production as similar as possible

XI. Logs

Treat logs as event streams

XII. Admin processes

Run admin/management tasks as one-off processes

There are three types of variables in Java

Local Variables

Instance Variables

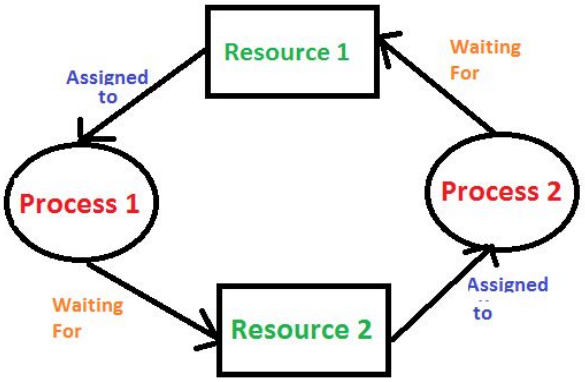
Static Variables

What is type inference variable in Java?

Var is used in JAVA 10. The var keyword allows local variable type inference, which means the type for the local variable will be inferred by the compiler, so you don't need to declare that.

What is Deadlock detection?

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.



Deadlock can arise if the **following four conditions hold simultaneously (Necessary Conditions)**
Mutual Exclusion: One or more than one resource are non-shareable (Only one process can use at a time)
Hold and Wait: A process is holding at least one resource and waiting for resources.
No Preemption: A resource cannot be taken from a process unless the process releases the resource.
Circular Wait: A set of processes are waiting for each other in circular form.

Methods for Handling deadlock

1) Prevention And Avoidance : Use of bankers algorithm :- It allows allocation of predetermined maximum possible amounts of all resources

2) Detection and Recovery : Killing the process, Resource preemption (we preempt some resources from processes and give those resources to other processes.)

3) Ignore the deadlock. If it occurs reboot the system.

What is a nested class? What is serialization?How to instantiate inner class?

https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html

Explain stream api in java?

What is type erasure in java?

Generics were introduced to the Java language to provide tighter type checks at compile time and to support generic programming. To implement generics, the Java compiler applies type erasure to:

- Replace all type parameters in generic types with their bounds or Object if the type parameters are unbounded. The produced bytecode, therefore, contains only ordinary classes, interfaces, and methods.
- Insert type casts if necessary to preserve type safety.
- Generate bridge methods to preserve polymorphism in extended generic types.

Type erasure ensures that no new classes are created for parameterized types; consequently, generics incur no runtime overhead.

What is the difference between a+=1 and a=a+1

a+=1 increments in place. And completes in single evaluation

a=a+1 creates a copy, adds the values and then stores the value.

What happens when we override a static method?

We cannot override static and private methods.

Object Oriented design principle

SOLID stands for:

S - Single-responsibility Principle

O - Open-closed Principle

L - Liskov Substitution Principle

I - Interface Segregation Principle

D - Dependency Inversion Principle

ArrayList, Hashmap methods in java?

Map.putifabsent

Operators in Java Datatypes?

Palindrome for a number

diff b/w set and map

how to add and get elements from list and map

palindrome checking for number and string

wt is the use of static member

What is the final member?

1. concept of polymorphism

What is a Class loader? and how java class is getting loaded in jvm?

The Java ClassLoader is a part of the Java Runtime Environment that dynamically loads Java classes into the Java Virtual Machine. The Java run time system does not need to know about files and file systems because of classloaders.

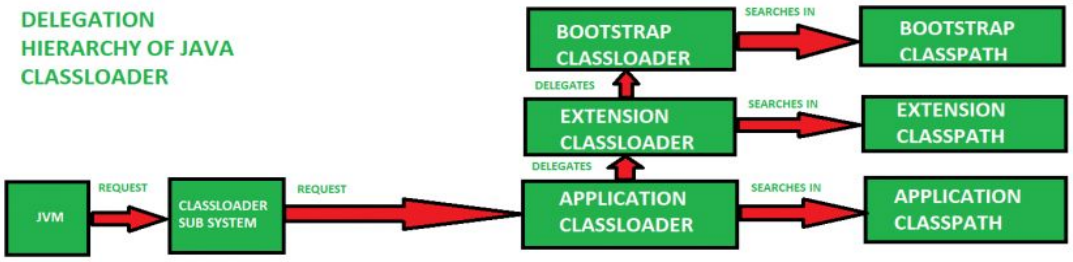
Java classes aren't loaded into memory all at once, but when required by an application. At this point, the Java ClassLoader is called by the JRE and these ClassLoaders load classes into memory dynamically.

A Java Classloader is of three types:

BootStrap ClassLoader: A Bootstrap Classloader is a Machine code which kickstarts the operation when the JVM calls it. It is not a java class. Its job is to load the first pure Java ClassLoader. Bootstrap ClassLoader loads classes from the location rt.jar. Bootstrap ClassLoader doesn't have any parent ClassLoaders. It is also called as the **Primordial** ClassLoader.

Extension ClassLoader: The Extension ClassLoader is a child of Bootstrap ClassLoader and loads the extensions of core java classes from the respective JDK Extension library. It loads files from jre/lib/ext directory or any other directory pointed by the system property java.ext.dirs.

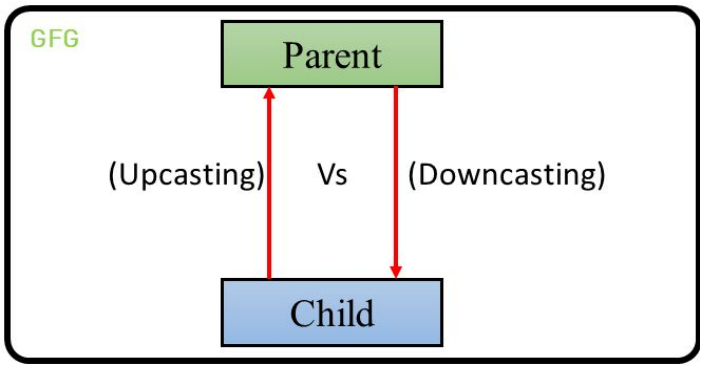
System ClassLoader: An **Application ClassLoader** is also known as a System ClassLoader. It loads the Application type classes found in the environment variable CLASSPATH, -classpath or -cp command line option. The Application ClassLoader is a child class of Extension ClassLoader.



What is upcast and downcast?

Upcasting: Upcasting is the typecasting of a child object to a parent object. Upcasting can be done implicitly. Upcasting gives us the flexibility to access the parent class members but it is not possible to access all the child class members using this feature. Instead of all the members, we can access some specified members of the child class. For instance, we can access the overridden methods.

Downcasting: Similarly, downcasting means the typecasting of a parent object to a child object. Downcasting **cannot be implicitly**.



4. How map is storing the value? What is the mechanism inside?
https://howtodoinjava.com/java/collections/hashmap/how-hashmap-works-in-java/

5. Can map store null as a key? can map store object as a key?

6. What are the difference types in collections?

7. What is Checked exception? What i unchecked exception?

8. What is threading?

9. What is steaming and parallel streaming?

What is join fork?

The fork/join framework is an implementation of the ExecutorService interface that helps you take advantage of multiple processors. It is designed for work that can be broken into smaller pieces recursively. The goal is to use all the available processing power to enhance the performance of your application.

As with any ExecutorService implementation, the fork/join framework distributes tasks to worker threads in a thread pool. The fork/join framework is distinct because it uses a work-stealing algorithm. Worker threads that run out of things to do can steal tasks from other threads that are still busy.

The center of the fork/join framework is the ForkJoinPool class, an extension of the AbstractExecutorService class. ForkJoinPool implements the core work-stealing algorithm and can execute ForkJoinTask processes.

After your ForkJoinTask subclass is ready, create the object that represents all the work to be done and pass it to the invoke() method of a ForkJoinPool instance.

11. Explain filter and its mechanism?

12. Explain Map in stream and its mechanism ?

Can the abstract class have a constructor?wt is the use of it?

Yes. To instantiate other member of class. It can be invoked by its object instance of its subclass.

What is constructor chaining?

The process of invoking a sequence of constructors upon initialization of a class object is called constructor chaining. Constructor chaining is useful when you want to invoke multiple constructors, one after another, by initializing only one instance

Constructor chaining occurs through inheritance. A sub class constructor's task is to call super class's constructor first

What is init block?

Init block executed whenever the constructor of a class is called.

How a method of a parent class is called?

Using super keyword.
Can use this keyword if the same method or variable not available in the child class.
If Available it points to the same method. And if called from the same method, the stack overflow error occur

How a child class method is called?

Using reflection.
Or if it is overridden by child class, parent class can call the member of a child class from its other methods, only if obj is created for child class.

17. What are the disadvantages of micro service architecture?

18. What is the difference between put and patch?

19. When we will use only patch?

20. What is versioning in rest?

1) Characteristics of Hashtable?
2) How to add values to Hashtable?
3) How to declare Hashtable ?
4) What is the characteristics of multi threading?
5) Runnable Interface?
6) Determine BigO for a program (The program was a recursive function like below)

```
fun recur(){
    if(){
        recur()
    }else{
    }
    recur()
}
```

Lambda expression
diff between foreach and map and filter
can we access variable outside lambda
wt is the use of optional

Difference between Final and Effective Final

A variable is final or effectively final when it's initialized once and is never mutated in its owner class; we can't initialize it in loops or inner classes.

Effectively Final:

A keyword before a variable makes it final and no final keyword before a variable make it effectively final provided we never change its value.

Modifying Variables inside lambda

The variable must be final or effectively final, in order to access them inside lambda.
If No : Error will come.

To overcome this, we can come up with 3 potential solutions:
1. Making the variable static.
2. Use Array
3. Atomic Integer

Constraints in java

@Deprecated, @WebServlet, @NotNull

Can we extend or implement enum?
Difference between map and flatmap?
Initialization of arrayList and its size (How can u store 250 records, in which collection)

compare equal with hashing
cookie based or Oauth based
explain rest Api
microservice architecture
spring bean scope
what is bean
what is class and object
tree,hashtable data structure?
how to configure tomcat?
what is continuous monitoring?
how micro service communicate with other?
how join table is configured in hibernate?
what is hsql(hibernate sql)
how to filter data on some condition in hibernate?
how nosql database stores its data
how security is done in project?
try with resource?
how cloning is achieved in java
How map can be sorted using key and value?
When to use string and when to use string buffer?
Can two or more public classes in the same file?
Can we overload main method?
Difference between OS path and Java path?

What is stream ?
What is async process?
how rest api communicates?
how casandra stores it value?
How multi threading can be used?
async process can be used?
sorting string array without sort method?
Define abstract class

How to call another method based on response from db

fail fast & fail safe

Iterators in java are used to iterate over the Collection objects.Fail-Fast iterators immediately throw *ConcurrentModificationException* if there is **structural modification** of the collection. Structural modification means adding, removing or updating any element from collection while a thread is iterating over that collection. Iterator on ArrayList, HashMap classes are some examples of fail-fast Iterator.

Fail-Safe iterators don't throw any exceptions if a collection is structurally modified while iterating over it. This is because, they operate on the clone of the collection, not on the original collection and that's why they are called fail-safe iterators. Iterator on **CopyOnWriteArrayList, ConcurrentHashMap** classes are examples of fail-safe Iterator.

https://anmolsehgal.medium.com/fail-fast-and-fail-safe-iterations-in-java-collections-11ce8ca4180e

What collection to insert data without duplicates and maintain the insertion order?

linkedHashSet

Can we override private method?

No

can we override load static method?

No

What will happen if we override a method and declare it as private in subclass?

No, Compile time error will come

Override rules

The argument list should be exactly the same as that of the overridden method.

The return type should be the same or a subtype of the return type declared in the original overridden method in the superclass.

The access level cannot be more restrictive than the overridden method's access level. For example: If the superclass method is declared public then the overriding method in the sub class cannot be either private or protected.

Instance methods can be overridden only if they are inherited by the subclass.

A method declared final cannot be overridden.

A method declared static cannot be overridden but can be re-declared.

If a method cannot be inherited, then it cannot be overridden.

A subclass within the same package as the instance's superclass can override any superclass method that is not declared private or final.

A subclass in a different package can only override the non-final methods declared public or protected.

An overriding method can throw any uncheck exceptions, regardless of whether the overridden method throws exceptions or not. However, the overriding method should not throw checked exceptions that are new or broader than the ones declared by the overridden method. The overriding method can throw narrower or fewer exceptions than the overridden method.

Constructors cannot be overridden.

What will happen if we declare @functionalInterface and have two abstract method?

Compile time error will come, But if we implement and run the program, it will run successfully.

When will we use lambda expressions?

Can we implement lambda for non functional interface?

Can we catch multiple exceptions in single catch block?
Yes

Why set is sorting values?

Example of functional interface u use?

What is predicate? what it will return?

what is auto configured in spring boot?

how u will setup a micro service from spring boot?

why we use springboot?

advatages of springboot with MVC?

session, cookie, @async, @schedule?

what is compiletime polymorphism and runtime polymorphism?

operator widening?

static, lazy , dynamic binding?

Can we pass request body for DELETE request?

Can we overload operators in java?

Dispatcher servelet?

Do we need to typecast collections.toArray?

mechanism of springboot?

what rest api call will return?

how to call another rest api service from java?

difference between @requestMapping and @getMapping? Whether the URI is same in both?
if @controller and @restController, URI same?

What is single page application?
A single-page application is an app that works inside a browser and does not require page reloading during use. You are using this type of applications every day. These are, for instance: Gmail, Google Maps, Facebook or GitHub.
SPAs are all about serving an outstanding UX by trying to imitate a “natural” environment in the browser — no page reloads, no extra wait time. It is just one web page that you visit which then loads all other content using JavaScript — which they heavily depend on.

Volatile in java

Volatile keyword is used to modify the value of a variable by different threads. It is also used to make classes thread safe. It means that multiple threads can use a method and instance of the classes at the same time without any problem

Comparable Vs Comparator

Comparable	Comparator
1) Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides **multiple sorting sequences**. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.
2) Comparable **affects the original class**, i.e., the actual class is modified.	Comparator **doesn't affect the original class**, i.e., the actual class is not modified.
3) Comparable provides **compareTo() method** to sort elements.	Comparator provides **compare() method** to sort elements.
4) Comparable is present in **java.lang** package.	A Comparator is present in the **java.util** package.
5) We can sort the list elements of Comparable type by **Collections.sort(List)** method.	We can sort the list elements of Comparator type by **Collections.sort(List, Comparator)** method.

https://www.javatpoint.com/difference-between-comparable-and-comparator

Shallow copy and deep copy

1. Creating a copy of object in a different memory location. This is called a Deep copy.

2. Creating a new reference that points to the same memory location. This is also called a Shallow copy.

Shallow Copy	**Deep Copy**
Cloned object and source object are not disjoint completely	Cloned objects and source objects are completely independent of each other.
Changes made in the cloned instance will impact the reference variable of the source object	Changes made in the cloned instance will not impact the reference variable of the source object.
The default version of the clone is the shallow copy	To create deep copy we need to override the clone method of Object class.
Shallow copy is preferred if class variables of the object are only primitive type as fields	A deep copy is preferred if the object's class variables have references to other objects as fields.
It is relatively fast	It is relatively slow.

Alti
====

internal working of hashmap?

what is thread safe map?

concurrent hashmap and synchronized map,which is faster?

ConcurrentHashMap allows concurrent access to data. Whole map is divided into segments.

Read operation ie. get(Object key) is not synchronized even at segment level.

But write operations ie. remove(Object key), get(Object key) acquire lock at segment level. Only part of whole map is locked, other threads still can read values from various segments except locked one.

SynchronizedMap on the other hand, acquire lock at object level. All threads should wait for current thread irrespective of operation(Read/Write).

when we need to insert 100 records which is faster arrayist or linked list?

What collection you will use inorder to store the count of duplicates ?

explain the authorization mechanism?

explain wht is cookie?

how to use session?

what are the methods commonly used in restapi?

difference between REST and SOAP?

difference between put and patch?

how will u call another micro service from one service?

how will u get the data returned by one micro service?

What is adapter pattern? What is the class structure for it?

What is singleton pattern, Define the class structure for it?

The fastest sort algorithm.

What are the ways one microservice communicates with another?

how linkedList implemented in java?

what are all the ways to initialize the collection?

How will u test a service?

What is optional?

JPA repository

reduce function in stream

Throw and throws
what is proxy
dynamic binding
2 primary key in a table, how to do with hibernate
what are class and methods involved in REST call
what are the two classes which can have rest call

how to create custom exceptions
where we will override hashcode and equals

Big o calculation
quickest sorting algorithm
applications of microservice and monolithic
Tree data structure
collection hierarchy

Caching in hibernate
Life Cycle of thread
What is the internal working when we store object as a key in Hashmap
What are the types of functional IF
What is consumer IF?
Comparable Vs Comparator

What will happen if two instanceof a thread started twice?
t1=new threadClss();
t1.start();
t1.start();

Difference between synchronize method and synchronize block?

How to design an immutable class?

Can we have return in finally

Race condition in thread

Sleep in thread

Difference between final, finally and finalize

What is lazy and eager loading in singleton

How to make arrayList thread safe

What are the fetch types available in hibernate

What is cascading in hibernate

What is dialect ?

Where we will overload hashcode and equals method? Wt is the relation between both

How concurrent hashmap is thread safe?

What is Linked Hashmap?

Difference between hashtable and concurrent hashmap

Composite Key in JPA

Diff b/w sleep and wait

Sort collection in reverse order using stream

Load factor of all collections

Working of optional.map