What are the new Features in Java 8 ?

There are various new features in Java 8, but the following are the most important:

Lambda Expressions –a new feature of the language that allows us to **consider actions as objects**

Method References – allow us to define Lambda Expressions by **explicitly referring to methods by their names**

Optional – Optionality is expressed via a special wrapper class.

Functional Interface – A Lambda Expression can be used to **implement an interface with a maximum of one abstract method.**

Default methods – allow us to provide **entire implementations in interfaces in addition to abstract methods**

Nashorn, JavaScript Engine – JavaScript code execution and evaluation engine based on ava.

Stream API –a particular **iterator class that lets us to efficiently process collections of items**

Date API –a Date API inspired by **JodaTime that is enhanced and immutable**

What are the two methods in Collection interface to generate a Stream?

Collection interface has two methods to generate a Stream.

stream() – Stream API is used to process collections of objects. A stream is sequence of objects that supports various methods which can be pipelined to produce the desired result. Stream takes input from the **collections, Arrays or I/O channels. Streams don't change the original data structure, they only provide the result as per the pipelined methods.**

A sequential stream is executed in a single thread running on one CPU core. The elements in the stream are processed sequentially in a single pass by the stream operations that are executed in the same thread.

Aggregation operations such as filter, reduce, match, find, etc are supported by stream.

Visitation of the elements present in the stream can only be done once during the lifetime of a stream. A new stream must be created to revisit the same elements present in the source.

Why we required stream
1. for functional programming
2. using lambda expressions we can reduce the code and we can do method chaining.
3. bulk operations.

parallelStream() – Returns a parallel Stream considering collection as its source.

A parallel stream allows us to use multi-core processing by executing the stream operation parallel across multiple CPU cores. **It is suggested to use a parallel stream when the order of execution of individual items does not affect the final result.**

Lambda Expression :-->

The Expression through which we can represent an Anonymous function.
Anonymous means Nameless/Unknown.
Function or method don't have any name or prefix called Anonymous function.
Using Lambda Expression we can convert all abstract method to Anonymous function.
Lambda Expression only applicable for functional interface.

Anonymous function : A method who don't have any name or modifier.

```
Syntax :  Parameter     Expression     Body
             ()            ->          { System.out.println("Lambda Syntax"); }
```

Lambda expressions are nothing but the implementation of the functional interface. Lambda expression is to reduce the amount of code.
A lambda expression is a short block of code which takes in parameters and returns a value.

There are Three type method references that are as follows:

Reference to a static method.	ContainingClass::**staticMethodName**
Reference to an instance method.	containingObject::**instanceMethodName**
Reference to a constructor.	ClassName::**new**

Example :--

1) Reference to a Static Method

```
interface Sayable{
   void say();
}

public class MethodReference {
  public static void saySomething(){
    System.out.println("Hello, this is static method.");
  }
  public static void main(String[] args) {
    // Referring static method
    Sayable sayable = MethodReference::saySomething;
    // Calling interface method
    sayable.say();
  }
}
```

2. Reference to an instance method.

```
interface Sayable{
void say();
}
public class InstanceMethodReference {
public void saySomething(){
    System.out.println("Hello, this is non-static method.");
}
public static void main(String[] args) {
    // Creating object
    InstanceMethodReference methodReference = new InstanceMethodReference();

    // Referring non-static method using reference
      Sayable sayable = methodReference::saySomething;

    // Calling interface method
       sayable.say();


       // Referring non-static method using anonymous object
       // You can use anonymous object also
        Sayable sayable2 = new InstanceMethodReference()::saySomething;

        // Calling interface method
        sayable2.say();
}
}
```

3. Reference to a constructor.

```
    interface Messageable{
        Message getMessage(String msg);
    }
    class Message{
        Message(String msg){
            System.out.print(msg);
        }
    }
    public class ConstructorReference {
        public static void main(String[] args) {
            Messageable hello = Message::new;
            hello.getMessage("Hello");
        }
    }
```

What is method reference vs constructor references in Java 8?

A method reference is similar to lambda expression used to refer a method without invoking it while constructor **reference used to refer to the constructor without instantiating the named class.**

What is @FunctionalInterface.?

A functional interface is nothing but an interface that has only one abstract method. To declare an interface as a functional interface, Java provides an annotation @FunctionalInterface.

The Interface who contains o**nly one abstract method but can have multiple default and static method is called functional interface.**
It can contain any number of Object class method

Example of functional Interface in JAVA:-->

```
Runnable    -----------------> run()
Callable    -----------------> call()
Comparable  -----------------> compareTo()
Comparator  -----------------> compare()
```

Runnable , Callable, Comparable and Comparator , these 4 are functional interface. These functional interface contains only 1 method.

Runnable	functional interface contains only 1 method	run()
Callable	functional interface contains only 1 method	call()
Comparable	functional interface contains only 1 method	compareTo()
Comparator	functional interface contains only 1 method	compare()

What is difference between normal interface and functional interface?
Java interface can have any number of abstract methods however functional interface must have only one abstract method.

Why do we use functional interface?
A functional interface is an interface that contains only a single abstract method **(a method that doesn't have a body).** we can use functional interfaces in a lambda expression and method references.

Functional Interfaces Can Be Implemented by a Lambda Expression.

The Java Function interface (java.util.function.Function) interface.
The Function interface represents **a function (method) that takes a single parameter and returns a single value.**

Consumer, Predicate and supplier are predefined functional interface in JAVA8.

If you want to play with stream API, then we must need to understand these 3 API.

What is Consumer Functional Interface :-->
A Consumer is a functional interface that accepts **a single input and returns no output.** This is a functional interface whose functional **method is accept(Object)**

Where is Consumer interface used?
It can be used as the assignment target for a lambda expression or method reference. **The Consumer Interface accepts a single argument and does not return any result.**

Consumer Functional Interface have 1 abstract method void accept (T t) which not returns any value. and have default Consumer<T> andThen(Consumer<? super T>,after) method. The Java Consumer interface is a functional interface that represents an function that consumes a value without returning any value.

What is Predicate Functional Interface :-->
This Predicate Functional Interface used for conditional check.

Predicate<T> is a generic functional interface that represents a single argument function **that returns a boolean value (true or false).** This interface available in java.util. function package and **contains a test(T t) method that evaluates the predicate of a given argument.**

The Java Predicate interface, java.util.function.Predicate, represents a **simple function that takes a single value as parameter, and returns true or false.**

what is supplier functional interface :-->
Supplier can be used in all contexts where **there is no input required but an output is expected.**

Java 8 Supplier is a functional interface **whose functional method is get().**
The Supplier interface represents an operation that takes no argument and returns a result.

supplier interface is represents an operation that takes no argument and returns a result, whose functional method is get() .

What is Java8 :- java 8 useful to write compact code with the help of lambda expression and functional programming.

The main features in java 8 , stream, lambda expression, functional interface , date api , default methods in the functional interface,
static methods in functional interface, optional, method reference, constructor reference, nashorom javascript engine.

The main advantage of JAVA 8 to create compact code, to create More readable & resuable code, to do parallel operations.

What is stream concept in Java 8
collection is nothing but group of elements as single entity and It is the source for the Stream. Stream API is used to process collections of objects with functional style of coding using lambda expression.

A stream is a sequence of objects that supports various methods which can be pipelined to produce the required result.

The features of java stream are

1. A stream is not a data structure instead it takes input from **collectios, Array or (I/O) channels.**

2. **Streams don't change the original data structure,** they only provide the **result as per the pipelined methods.**

When we want to process collections then go for stream , stream it is special iterator class that allows collection processing in functional manner.

What are the Stream methods ?
1. filter
2. forEach
3. map
4. flatMap
5. reduce
6. groupingBy
7. count
8. collect

What is difference between Stream and parallel Stream ?
Both are used to process of group of objects, **but stream execute it in a single core of your machine with sequential flow.** and **parallel Stream usages multiple core of your machines.**

What is CompletableFuture ?
CompletableFuture is used for **asynchronous programming in java.** Asynchronous programming is a means of writing non-blocking code by running a task on separate thread than the main application thread and notifying the main thread about its progress, completion or failure.

what is the use of functional interface in java 8?

Functional interfaces is used to as reference to lambda expression

Functional interfaces can contain only one abstract method. However, they can include any quantity of default and static methods.
we can assign lambda expression to its functional interface
in java 8 below are functional interfaces:-

Runnable -> This interface only contains the run() method.
Comparable -> This interface only contains the compareTo() method.
ActionListener -> This interface only contains the actionPerformed() method.
Callable -> This interface only contains the call() method.

Why functional interface have only 1 abstract method?
The functional interface also known as Single Abstract Method Interface was introduced to facilitate Lambda functions. Since a lambda function can only provide the implementation for 1 method it is mandatory for the functional interface to have only one abstract method.

what is the difference between hashtable and hashmap?

HashMap is non-syncronized and is not thread safe while HashTable is thread safe and is synchronized.

HashMap allows one null key and values can be null whereas HashTable doesn't allow null key or value.

HashMap is faster than HashTable. HashMap iterator is fail-safe where HashTable iterator is not fail-safe.

We can use Collections.synchronizedMap() to make a thread-safe HashMap.

HashMap is not synchronized, therefore it's faster and uses less memory than Hashtable. Generally, **unsynchronized objects are faster than synchronized** ones in a single threaded application.

what is synchronizedmap() and hashtable

In **synchronized HashMap and HashTable ,** all operations are synchronized. That means, whatever the operation you want to perform on the map, **whether it is read or update, you have to acquire object lock.**

But in **ConcurrentHashMap** , only **update operations are synchronized. Read operations are not synchronized.**

Why Lambda expression use functional interface only ?

As you know functional programming is supported in JAVA 8 with the help of Lambda expression. The Lambda expression supported the Functional interface.

These interfaces also are known as the Single Abstract method(SAM). The Lambda expressions support only those interfaces which have only one abstract method. It provides a body to the abstract method and an assigned as the reference of the interface.

Whenever you are using the lambda expression for the Function interface, the compiler strictly ensures the interface has only one abstract method. If the interface contains more than one abstract method the program shows an error.

Because the lambda expression provides a body to only a single abstract method. When you write lambda expression the compiler assumes the statements of lambda expression will be the body of the abstract method.

Why functional interface have only 1 abstract method?

The functional interface also known as Single Abstract Method (SAM) Interface was introduced to facilitate Lambda functions.

Since a lambda function can only provide the implementation for 1 method,if We add more than 1 abstract methods, then calling method won't know, which method to call out of those 2 abstract methods.

Why Strings are Immutable in Java?

An immutable object is an object which state is guaranteed to stay identical over its entire lifetime.

the immutable string or objects that cannot be modified once it is created. But we can only change the reference to the object. We restrict to change the object itself. The String is immutable in Java because of the security, synchronization and concurrency, caching, and class loading. **The reason of making string final is to destroy the immutability and do not allow others to extend it.**

difference between permgen and metaspace in java

PermGen always has a fixed maximum size. Metaspace by default auto increases its size depending on the underlying OS.

PermGen (Permanent Generation) is a special heap space separated from the main memory heap.

The JVM keeps track of loaded class metadata in the PermGen. Additionally, the JVM stores all the static content in this memory section.
This includes all the static methods, primitive variables, and references to the static objects.

Metaspace is a new memory space – starting from the Java 8 version; it has replaced the older PermGen memory space.

Differences between synchronized collection and concurrent collection in java

synchronize means only one thread is allowed to operate on an object at a time and Synchronized Collection has low performance than Concurrent Collection because at a time only one thread is allowed to operate on an object so it increases the waiting time of the threads.
Synchronized Collection acquires the lock on the entire Collection object which provides thread-safety

concurrent means only multiple threads are allowed to operate on an object at a time Concurrent Collection has high performance than Synchronized Collection because at a time multiple threads are allowed to operate on an object so it decreases the waiting time of the threads and More than one threads can perform read-write operation concurrently still it provides Thread Safety.

HashMap :- HashMap is not thread safe means it is not Synchronized so there is no locking mechanism so n number thread will can operate HashMap object.
HashMap returns fail fast itreator and it can throw the concurnent modification exception. HashMap allows one null key and multiple null values. we can use the hashmap in single threded environment.

HashTable :- HashTable is thread safe means it is Synchronized so there is locking mechanism so only one thread can operate HashMap object so other thread will goes in queue until the first thread complete its task.
HashTable **returns fail safe itreator** so it won't throw the concurnent modification exception. HashTable not allows null key and null value. It is a legacy class,so it is not recomended to use.

SynchronizedHashMap :- SynchronizedHashMap is thread safe means it is Synchronized so there is locking mechanism so only one thread can operate SynchronizedHashMap object so other thread will goes in queue until the first thread complete its task. SynchronizedHashMap returns **fail fast itreator** that means it can throw the concurnent modification exception. S**ynchronizedHashMap allows one key as null and multiple null values** . we can use the **SynchronizedHashMap in Multi threded environment.**

ConcurrentHashMap :-- ConcurrentHashMap is not Synchronized the whole map basically only write operations are Synchronized here. so you can perform n number of threads on n number of segement , basically **ConcurrentHashMap will create 16 segment by default so 16 threads can perfomed the write operation on 16 segment and there is no lock for read operation.** ConcurrentHashMap not allows null key and null values. ConcurrentHashMap returns **fail safe itreator** so it won't throw the concurnent modification exception. **we can use the ConcurrentHashMap in Multi threded environment.**

Finally we can say ConcurrentHashMap is faster and can use Multi threded environment.

What Is the Difference Between Intermediate and Terminal Operations in Java 8 Stream API ?

Stream operations are combined into pipelines to process streams And all operations are either intermediate or terminal

intermediate Operations :-- intermediate Operations return itself for further operations on a stream and these operations are lazy and intermediate Operations can only process data when there will be Terminal Operations. Means without Terminal Operations , we can’t use **intermediate Operations.**

Intermediate Operations are :-- flatMap() map(), filter(), distinct(), sorted(), limit(), skip()

Terminal Operations :-- terminal operation terminate the stream pipepline and initiate the stream processing.

Terminal Operations are :-- forEach(), toArray(), reduce(), collect(),sum() , min(), max(), count(), anyMatch(), allMatch(), noneMatch(), findFirst(), findAny()

What are the differences between Heap and Stack Memory in Java?

Stack is generally used to store the order of method execution and local variables. In contrast, Heap memory is used to store the objects. After storing, they use dynamic memory allocation and deallocation.

What is Object Cloning?

An ability to recreate an object entirely similar to an existing object is known as Object Cloning in Java. Java provides a clone() method to clone a current object offering the same functionality as the original object.

What are Brief Access Specifiers and Types of Access Specifiers?

Access Specifiers are predefined keywords used to help JVM understand the scope of a variable, method, and class. We have four access specifiers.

- **Public** Access Specifier
- **Private** Access Specifier
- **Protected** Access Specifier
- **Default** Access Specifier

Can a constructor return a value?

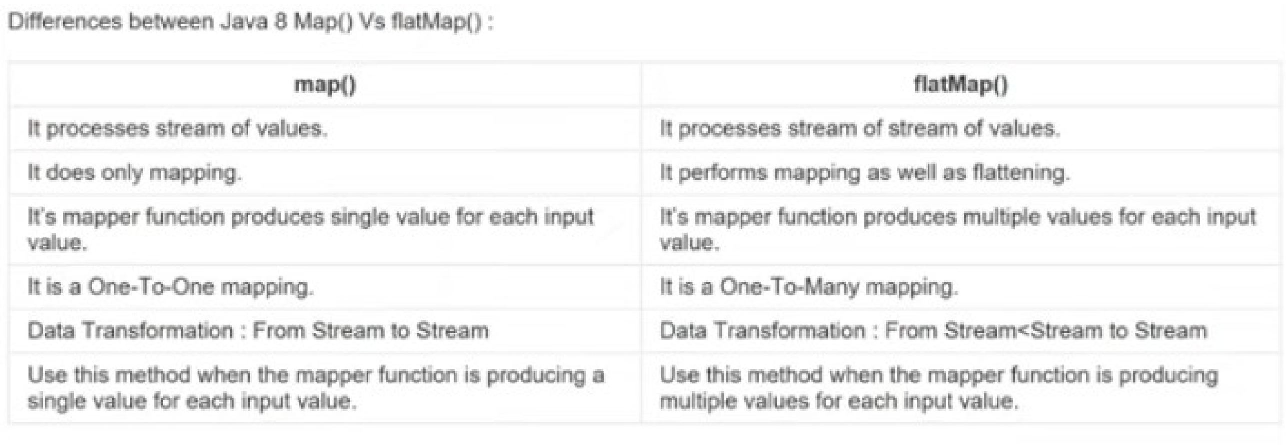
Yes, A constructor can return a value. It replaces the class's current instance implicitly; you cannot make a constructor return a value explicitly.

What is map and flatmap in java 8 ?

- flatmap is used for both transformation and flattening,
- flatMap() function is a combination of map and flattening operations.
- flatMap provides one or more values for each input value.
- it is also referred to as One-To-Many mapping.

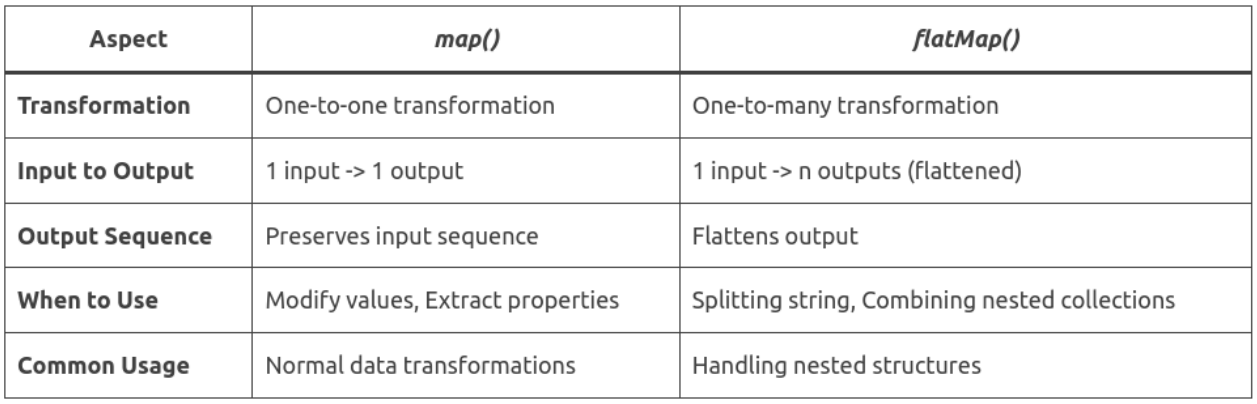
- The map() function is used to transform a stream from one form to another.
- The map produces one output value for each input value
- it is also referred to as One-To-One mapping.

- **We can use map() to execute a function on every element of a stream.**
- **flatMap() converts a stream of collections into a single “flat” stream.**
- **The map() method is used to transform each element of a stream into another object.**
-
- **The flatMap() method is used to transform each element of a stream into zero or more elements of a new stream.**



Differences between Java 8 Map() Vs flatMap() :

map()	flatMap()
It processes stream of values.	It processes stream of stream of values.
It does only mapping.	It performs mapping as well as flattening.
It's mapper function produces single value for each input value.	It's mapper function produces multiple values for each input value.
It is a One-To-One mapping.	It is a One-To-Many mapping.
Data Transformation : From Stream to Stream	Data Transformation : From Stream<Stream to Stream
Use this method when the mapper function is producing a single value for each input value.	Use this method when the mapper function is producing multiple values for each input value.



Aspect	*map()*	*flatMap()*
Transformation	One-to-one transformation	One-to-many transformation
Input to Output	1 input -> 1 output	1 input -> n outputs (flattened)
Output Sequence	Preserves input sequence	Flattens output
When to Use	Modify values, Extract properties	Splitting string, Combining nested collections
Common Usage	Normal data transformations	Handling nested structures

IMPORTANT POINTS OF MAP & FLATMAP

1. Map and FlatMap both reaturn stream.
2. Map are used for transform purpose.
3. FlatMap are used for transform and Flatten purpose.

Note :-
1. transform means change the actual value.
2. Flatten means convert the nested stream into to single stream.

Example of Map

Suppose you have the following list of integers and you want to multiply each factor by 2 digits that means transform , because, here your are changing the actual value.

Example :--

```
List<Integer> numberList = List.of(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

System.out.println(numberList.stream().map(no -> no * 2)
.collect(Collectors.toList()));
```

Explanation :- You have a list of integers and you multiply each element by 2 digits and return it as a single (stream).

Note :- so you have used map because here are single list.

Example of FlatMap

if you have a nested list of integers and you want to multiply each element by 2 digits (that means transform) and combine the nested list as a single list and return as a single (stream).

Example :--

```
        List<List<Integer>> numberList = List.of(List.of(1, 2, 3),
                List.of(4, 5, 6),
                List.of(7, 8, 9, 10)) ;

        System.out.println(numberList.stream().flatMap(no -> no.stream())
                .collect(Collectors.toList())
               .stream().map(no -> no * 2)
               .collect(Collectors.toList()));
```

Explanation :- You have a nested list of integers, so you create a single (stream) using FlatMap then you used map and multiply each element by 2 digits and finally return it as a single (stream).

Note :- so you have used flatap because here are nested List.

What is the mono and flux?

Mono & Flux are Publisher.

Mono: Returns 0 or 1 element. The Mono API allows producing only one value.
So mono is nothing but single entity.

When we're expecting single result from database, or external service , then we should use Mono.

Flux: Returns 0…N elements. The Flux can be endless, it can produce multiple values. When we're expecting multiple results from   database, or external service   then we should use Flux.
So Flux is list of elements.

What is the zipWith?

The zipWith() method combines the result from this mono and another mono object.

To use Mono and Flux, make sure that you add Project Reactor Core dependency:

```
    <dependency>
      <groupId>io.projectreactor</groupId>
      <artifactId>reactor-core</artifactId>
      <version>3.5.1</version>
    </dependency>
```

What is WebClient

WebClient is a reactive HTTP client that supports non-blocking and asynchronous operations for making HTTP requests.

In addition to reactive operations, it also supports synchronous and blocking requests.

What are objects and classes in Java?
Objects are basic building blocks in Java that contains state and behavior. Classes are templates that define objects and their behavior.

What are constructors in Java?
Constructors are special methods in Java that are used to initialize objects. The constructor is invoked when an object of a class is created. It has the same name as the class and does not have a return type.

What is method overloading and overriding in Java?
Method overloading is defining methods with the same name but different parameters. Overriding is providing a specific implementation of a method already defined in the parent class.

What is abstraction in Java?
Abstraction refers to hiding the implementation details and exposing only the functionality to users. Abstract classes and interfaces are used to achieve abstraction in Java.

What is final keyword in Java?
The final keyword is used to apply restrictions on classes, methods and variables. Final class cannot be inherited, final method cannot be overridden and final variable value cannot be changed.

What is static in Java?
Static is a keyword in Java used to denote a class member belongs to a type rather than to an instance. Static members can be used without creating an object of class.

What is encapsulation in Java?
Encapsulation is the mechanism of wrapping data (variables) and code acting on data (methods) together as a single unit. In Java, encapsulation is achieved by making fields private and providing public setter and getter methods.

What is inheritance in Java?
Inheritance represents parent-child relationship between classes in Java. It allows a derived class to inherit commonly used state and behavior from its parent class.

What is polymorphism in Java?
Polymorphism means ability to take different forms. In Java, polymorphism allows assigning a variable and method call to take different forms or classes. Method overloading and overriding uses polymorphism.

What is JIT compiler in Java?
JIT (Just-In-Time) compiler in Java converts bytecode into native machine code at runtime when required by the program. It improves performance by compiling bytecode lazily.

What is multithreading in Java?

Multithreading allows concurrent execution of multiple parts of a Java program. The main ways to create threads in Java are extending Thread class and implementing Runnable interface.

Explain different ways to create a thread in Java.
There are two ways to create a thread in Java – 1. Extending Thread class 2. Implementing Runnable Interface. Runnable is preferred because Java does not support multiple inheritance.

What are access modifiers in Java?
Java provides access control through public, protected, private and default modifiers. Public grants access from anywhere. Private restricts access to the class itself. Default and protected have specialized uses.

What is Collections Framework in Java?
Java Collections Framework provides ready-made architecture to store and manipulate group of objects. It contains interfaces like List, Set, Queue and classes like ArrayList, LinkedHashSet, PriorityQueue etc.

What is singleton class in Java and how can we make a class singleton?
Singleton class means that only one instance of the class can be created. Singleton pattern involves a single private constructor, a static variable and a static public method that returns the instance.

What is Java Serialization API?
Java Serialization API provides a standard mechanism to serialize objects to stored or transmitted across streams. Only serializable objects can be serialized.

What is autoboxing and unboxing in Java?
Autoboxing is automatic conversion of primitive types to object wrapper classes. Unboxing is the reverse process of converting wrapper objects to primitives.

What is final, finally and finalize in Java?
final is a keyword – final class can't be inherited, final method can't be overridden, final variable value can't change. finally is a block – used with try/catch to put code that executes always. finalize is a method – called by Garbage collector before object is collected.

What is try-with-resources in Java?
try-with-resources is a way to automatically close resources after usage without needing an explicit finally block. Any class that implements AutoCloseable interface can be used in try-with-resources.

What is difference between Heap and Stack memory?
Heap memory is used by all parts of the application whereas stack memory is used only by one thread of execution. Objects are created in Heap, Stack is used for local primitive variables and references to objects in Heap.

What is Java String Pool?
Java String Pool refers to collection of Strings stored in heap memory. String literals and constants are stored in the String pool for reuse to optimize memory usage.

How is a string immutable in Java?
In Java, string objects are immutable meaning their state cannot be changed once created. Whenever changes are made to a string, a new instance is created. This optimizes performance by reusing strings from pool.

What is ThreadPoolExecutor in Java?
ThreadPoolExecutor is a thread pool implementation added from Java 5 that provides more configurable thread pools to execute tasks. It allows configuring pool size, rejection policies, thread factories etc.

What is memory leak in Java?
Memory leak occurs when objects are no longer used by the application but Garbage Collector fails to recognize them as unused. This results in out of memory errors if too many objects are unreferenced.

What is shallow copy and deep copy in Java?
Shallow copy is copying an object’s field references into another instance. Deep copy is making separate copy of all the objects in the original object graph.

What are transient and volatile keywords in Java?
transient – skip field during serialization, volatile – field will not be cached and always read from main memory.

Explain Generics in Java?
Generics allow defining type-safe classes, interfaces and methods which work with different types while avoiding duplicity. Generics work only with reference types in Java.

What is Comparable and Comparator interface in Java?
Comparable is used to provide natural ordering of objects of a class. Comparator provides custom ordering and added flexibility of sorting objects.

Explain different ways to iterate over a collection in Java?
Iterating collections can be done through Iterator, for-each loop, forEach(), forEachRemaining() and ListIterator. Iterator allows removing elements during iteration while ListIterator can iterate in reverse.