**Java 8**

**What are the significant advantages of Java 8?**

* Compact, readable, and reusable code.
* Less boilerplate code.
* Parallel operations and execution.
* Can be ported across operating systems.
* High stability.
* Stable environment.
* Adequate support

### What is MetaSpace? How does it differ from PermGen?

**PremGen:** MetaData information of classes was stored in PremGen (Permanent-Generation) memory type before Java 8. PremGen is fixed in size and cannot be dynamically resized. It was a contiguous Java Heap Memory.

**MetaSpace:** Java 8 stores the MetaData of classes in native memory called 'MetaSpace'. It is not a contiguous Heap Memory and hence can be grown dynamically which helps to overcome the size constraints. This improves the garbage collection, auto-tuning, and de-allocation of metadata

### What are functional or SAM interfaces?

Functional Interfaces are an interface with only one abstract method. Due to which it is also known as the Single Abstract Method (SAM) interface. It is known as a functional interface because it wraps a function as an interface or in other words a function is represented by a single abstract method of the interface.

Functional interfaces can have any number of default, static, and overridden methods. For declaring Functional Interfaces @FunctionalInterface annotation is optional to use. If this annotation is used for interfaces with more than one abstract method, it will generate a compiler error.

### Can a functional interface extend/inherit another interface?

A functional interface cannot extend another interface with abstract methods as it will void the rule of one abstract method per functional interface.

It can extend other interfaces which do not have any abstract method and only have the default, static, another class is overridden, and normal methods.

### What is the default method, and why is it required?

A method in the interface that has a predefined body is known as the default method. It uses the keyword default. default methods were introduced in Java 8 to have 'Backward Compatibility in case JDK modifies any interfaces. In case a new abstract method is added to the interface, all classes implementing the interface will break and will have to implement the new method. With default methods, there will not be any impact on the interface implementing classes. default methods can be overridden if needed in the implementation. Also, it does not qualify as synchronized or final.

### 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.

### What are some standard Java pre-defined functional interfaces?

Some of the famous pre-defined functional interfaces from previous Java versions are Runnable, Callable, Comparator, and Comparable. While Java 8 introduces functional interfaces like Supplier, Consumer, Predicate, etc. Please refer to the java.util.function doc for other 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

### What are the various categories of pre-defined function interfaces?

**Function:** To transform arguments in returnable value.

**Predicate:** To perform a test and return a Boolean value.

**Consumer:** Accept arguments but do not return any values.

**Supplier:** Do not accept any arguments but return a value.

**Operator:** Perform a reduction type operation that accepts the same input types.

### What is the lambda expression in Java and How does a lambda expression relate to a functional interface?

Lambda expression is a type of function without a name. It may or may not have results and parameters. It is known as an anonymous function as it does not have type information by itself. It is executed on-demand. It is beneficial in iterating, filtering, and extracting data from a collection.

As lambda expressions are similar to anonymous functions, they can only be applied to the single abstract method of Functional Interface. It will infer the return type, type, and several arguments from the signature of the abstract method of functional interface.

### What is the basic structure/syntax of a lambda expression?

Lambda expression can be divided into three distinct parts as below:

1. List of Arguments/Params:

(String name)

A list of params is passed in () round brackets. It can have zero or more params. Declaring the type of parameter is optional and can be inferred for the context.

2. Arrow Token:

->   
Arrow token is known as the lambda arrow operator. It is used to separate the parameters from the body, or it points the list of arguments to the body. 3. Expression/Body:

A body can have expressions or statements. {} curly braces are only required when there is more than one line. In one statement, the return type is the same as the return type of the statement. In other cases, the return type is either inferred by the return keyword or void if nothing is returned.

### In Java 8, what is Method Reference?

Method reference is a compact way of referring to a method of functional interface. It is used to refer to a method without invoking it. :: (double colon) is used for describing the method reference. The syntax is **class::methodName**

### What is an Optional class?

Optional is a container type which may or may not contain value i.e. zero(null) or one(not-null) value. It is part of java.util package. There are pre-defined methods like isPresent(), which returns true if the value is present or else false and the method get(), which will return the value if it is present.

### What are Java 8 streams?

A stream is an abstraction to express data processing queries in a declarative way.

A Stream, which represents a sequence of data objects & series of operations on that data is a data pipeline that is not related to Java I/O Streams does not hold any data permanently.  
The key interface is **java.util.stream.Stream<T>**. It accepts Functional Interfaces so that lambdas can be passed. Streams support a fluent interface or chaining.

### What is the stateful intermediate operation? Give some examples of stateful intermediate operations.

To complete some of the intermediate operations, some state is to be maintained, and such intermediate operations are called stateful intermediate operations. Parallel execution of these types of operations is complex.

For Eg: sorted() , distinct() , limit() , skip() etc.

Sending data elements to further steps in the pipeline stops till all the data is sorted for sorted() and stream data elements are stored in temporary data structures.

**What is the most common type of Terminal operations?**

* collect() - Collects single result from all elements of the stream sequence.
* reduce() - Produces a single result from all elements of the stream sequence
  + count() - Returns the number of elements on the stream.
  + min() - Returns the min element from the stream.
  + max() - Returns the max element from the stream.
* Search/Query operations
  + anyMatch() , noneMatch() , allMatch() , ... - Short-circuiting operations.
  + Takes a Predicate as input for the match condition.
  + Stream processing will be stopped, as and when the result can be determined.
* Iterative operations
  + forEach() - Useful to do something with each of the Stream elements. It accepts a consumer.
  + forEachOrdered() - It is helpful to maintain order in parallel streams.