**Java 8 Interview Questions:**

Q: What are the main features introduced in Java 8?

A: The main features introduced in Java 8 include lambda expressions, functional interfaces, streams, default methods, and the new date and time API.

Q: What is a lambda expression in Java 8?

A: A lambda expression is a concise way to represent an anonymous function. It allows you to pass behavior as an argument to a method or to define functionality inline.

Q: How do you define a lambda expression in Java 8?

A: A lambda expression is defined using the following syntax:

(parameters) -> { body }

Q: What is a functional interface in Java 8?

A: A functional interface is an interface that contains exactly one abstract method. It is used to provide target types for lambda expressions and method references.

Q: What is a stream in Java 8?

A: A stream is a sequence of elements that can be processed in parallel or sequentially. It allows you to perform operations such as filtering, mapping, and reducing on collections.

Q: How do you create a stream in Java 8?

A: You can create a stream from a collection using the stream() method or from an array using the Arrays.stream() method.

Q: What are the differences between a stream and a collection in Java 8?

A: A collection is a data structure that holds elements, while a stream is a sequence of elements that can be processed. Collections are eagerly evaluated, whereas streams are lazily evaluated.

Q: What are the intermediate and terminal operations in a stream?

A: Intermediate operations are operations that transform a stream into another stream, such as filter() and map(). Terminal operations are operations that produce a result or a side-effect, such as forEach() and collect().

Q: How do you iterate over a collection using streams in Java 8?

A: You can use the forEach() method or other terminal operations like collect() or toArray() to iterate over a collection using streams.

Q: What is a method reference in Java 8?

A: A method reference is a shorthand syntax for lambda expressions that refer to methods or constructors by their names.

Q: What is the Optional class in Java 8?

A: The Optional class is a container object that may or may not contain a non-null value. It is used to avoid null pointer exceptions and to provide a more expressive way of handling null values.

Q: What is a default method in Java 8?

A: A default method is a method defined in an interface with a default implementation. It allows adding new methods to existing interfaces without breaking compatibility with implementing classes.

Q: How do you handle exceptions in lambda expressions?

A: Lambda expressions can throw exceptions, but they need to be declared in the functional interface's abstract method using the throws clause.

Q: How do you sort a list of objects using a lambda expression in Java 8?

A: You can use the sort() method from the List interface and provide a lambda expression or a method reference as a comparator to define the sorting order.

Q: What is the purpose of the Predicate functional interface in Java 8?

A: The Predicate functional interface represents a boolean-valued function of one argument. It is often used for filtering elements in streams or collections.

Q: How do you group elements in a stream based on a property using Java 8?

A: You can use the groupingBy() collector from the Collectors class, providing a lambda expression or a method reference that extracts the property to group by.

Q: What is the purpose of the flatMap() operation in Java 8 streams?

A: The flatMap() operation is used to flatten nested collections or maps into a single stream.

Q: How do you find the maximum or minimum element in a stream using Java 8?

A: You can use the max() or min() method from the Stream interface, providing a comparator to define the ordering.

Q: What is the purpose of the Collectors class in Java 8?

A: The Collectors class provides various useful collectors for collecting elements from a stream into a collection or performing reduction operations.

Q: How do you convert a list of objects into a map based on a property using Java 8?

A: You can use the toMap() collector from the Collectors class, providing lambda expressions or method references to extract the key and value from each object.

Q: How do you perform parallel processing in Java 8 using streams?

A: You can use the parallelStream() method on a collection or a stream to process elements in parallel, leveraging multiple threads.

Q: What is the purpose of the Supplier functional interface in Java 8?

A: The Supplier functional interface represents a supplier of results. It is often used to lazily initialize objects or generate values.

Q: How do you use the peek() operation in Java 8 streams?

A: The peek() operation allows you to perform a side-effect for each element in a stream without modifying the stream itself.

Q: What is the purpose of the Comparator interface in Java 8?

A: The Comparator interface provides a way to define custom ordering for objects. It is commonly used for sorting and searching.

Q: How do you convert a stream of objects to an array in Java 8?

A: You can use the toArray() method from the Stream interface to convert a stream of objects into an array.

Q: What is the purpose of the Function functional interface in Java 8?

A: The Function functional interface represents a function that takes one argument and produces a result. It is often used for mapping or transforming elements in streams.

Q: How do you filter null values from a stream using Java 8?

A: You can use the filter() operation and a lambda expression that checks for null values to filter them out from a stream.

Q: What is the purpose of the BiFunction functional interface in Java 8?

A: The BiFunction functional interface represents a function that takes two arguments and produces a result. It is often used for operations that involve two inputs.

Q: How do you find the sum or average of elements in a stream using Java 8?

A: You can use the sum() or average() methods from the DoubleStream or IntStream interfaces to calculate the sum or average of elements in a stream.

Q: What is the purpose of the Predicate functional interface in Java 8?

A: The Predicate functional interface represents a boolean-valued function of one argument. It is often used for filtering elements in streams or collections.

Q: How do you use the reduce() operation in Java 8 streams?

A: The reduce() operation allows you to perform a reduction on the elements of a stream using a binary operator.

Q: What is the purpose of the BiConsumer functional interface in Java 8?

A: The BiConsumer functional interface represents an operation that takes two arguments and returns no result. It is often used for consuming or performing operations on pairs of elements.

Q: How do you remove duplicate elements from a stream using Java 8?

A: You can use the distinct() operation to remove duplicate elements from a stream based on their equality.

Q: What is the purpose of the LongSupplier functional interface in Java 8?

A: The LongSupplier functional interface represents a supplier of long values. It is often used for generating or providing long values.

Q: How do you concatenate strings in a stream using Java 8?

A: You can use the Collectors.joining() method to concatenate strings in a stream into a single string.

Q: What is the purpose of the Consumer functional interface in Java 8?

A: The Consumer functional interface represents an operation that takes one argument and returns no result. It is often used for consuming or performing operations on elements.

Q: How do you iterate over a map using Java 8 streams?

A: You can use the entrySet().stream() method on a map to iterate over its key-value pairs using streams.

Q: What is the purpose of the Supplier functional interface in Java 8?

A: The Supplier functional interface represents a supplier of results. It is often used to lazily initialize objects or generate values.

Q: How do you convert a stream of strings to uppercase in Java 8?

A: You can use the map() operation and a lambda expression that applies the toUpperCase() method to each string in the stream.

Q: What is the purpose of the UnaryOperator functional interface in Java 8?

A: The UnaryOperator functional interface represents an operation that takes one argument of a specific type and returns a result of the same type. It is often used for operations that transform elements.

Q: How do you find the first element in a stream using Java 8?

A: You can use the findFirst() method from the Stream interface to find the first element in a stream.

Q: What is the purpose of the IntPredicate functional interface in Java 8?

A: The IntPredicate functional interface represents a predicate that takes an int value as an argument. It is often used for filtering or testing conditions on int values.

Q: How do you convert a stream of objects to a list in Java 8?

A: You can use the collect() method with the Collectors.toList() collector to convert a stream of objects into a list.

Q: What is the purpose of the BinaryOperator functional interface in Java 8?

A: The BinaryOperator functional interface represents an operation that takes two arguments of the same type and returns a result of the same type. It is often used for reduction operations.

Q: How do you iterate over a range of numbers using Java 8 streams?

A: You can use the IntStream.range() or LongStream.range() methods to create a stream of numbers in a specific range.

Q: What is the purpose of the DoubleFunction functional interface in Java 8?

A: The DoubleFunction functional interface represents a function that takes a double value as an argument and produces a result of a specific type. It is often used for mapping or transforming double values.

Q: How do you convert a stream of objects to a set in Java 8?

A: You can use the collect() method with the Collectors.toSet() collector to convert a stream of objects into a set.

Q: What is the purpose of the ToIntFunction functional interface in Java 8?

A: The ToIntFunction functional interface represents a function that takes an object as an argument and produces an int value. It is often used for extracting integer properties from objects.

Q: How do you find the distinct elements in a stream using Java 8?

A: You can use the distinct() operation to find the distinct elements in a stream based on their equality.

Q: What is the purpose of the DoubleSupplier functional interface in Java 8?

A: The DoubleSupplier functional interface represents a supplier of double values. It is often used for generating or providing double values.