Java 8 Lambda Expressions & Functional Interfaces – Interview Guide

This document is a complete, interview-oriented reference for Java 8 Lambda Expressions and Functional Interfaces. It is designed for developers with 3–8+ years of experience and covers concepts, examples, problems, and commonly asked interview questions.

1. Lambda Expressions – Definition

A lambda expression is a concise way to represent an anonymous function that implements the single abstract method of a functional interface.

Syntax: (parameters) -> expression OR (parameters) -> { statements }

Example: Runnable r = () -> System.out.println("Hello Lambda");

2. Why Lambda Expressions Were Introduced

Lambda expressions reduce boilerplate code, improve readability, and enable functional-style programming in Java.

3. Functional Interfaces

A functional interface contains exactly one abstract method and may contain default or static methods.

Example: @FunctionalInterface interface Calculator { int calculate(int a, int b); }

4. Predefined Functional Interfaces

Predicate<T>: Takes input and returns boolean. Used for filtering.

Example: Predicate isEven = n -> n % 2 == 0;

Function<T, R>: Transforms input to output.

Example: Function lengthFn = s -> s.length();

Consumer<T>: Consumes input without returning result.

Example: Consumer printer = s -> System.out.println(s);

Supplier<T>: Supplies values without input.

Example: Supplier dateSupplier = () -> new Date();

UnaryOperator<T> and **BinaryOperator<T>**: Specialized functional interfaces.

5. Lambdas with Streams

Lambdas are widely used in Stream API operations like map, filter, forEach, and reduce.

Example: names.stream().map(s -> s.length()).forEach(System.out::println);

6. Lambda-Based Interview Problems

• Sorting using lambda comparator

• Filtering using Predicate

• Transformation using Function

• Side-effects using Consumer

• Custom functional interfaces

• reduce(), allMatch(), flatMap() usage

7. Tricky Lambda Concepts

• Variables must be effectively final

• Lambdas do not create a new scope

• Checked exceptions handling

• Lambdas are objects but not classes

8. Common Interview Questions

What is a lambda expression?

Can lambda exist without functional interface?

Difference between lambda and anonymous class?

Why variables must be effectively final?

When should lambdas be avoided?

9. How to Explain in Interview

Lambda expressions provide a concise way to implement functional interfaces, enable functional programming, and work seamlessly with Java Streams to process data declaratively.