## ✅ Java 8 Coding Interview Questions with Answers

### 1. ****What is a lambda expression? Write a basic example.****

**Answer:**

A lambda expression is a **short block of code** which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and can be implemented right in the body of a method.

**Syntax:**

java

CopyEdit

(parameter1, parameter2) -> { //body }

**Example:**

java

CopyEdit

List<String> list = Arrays.asList("A", "B", "C");

list.forEach(s -> System.out.println(s));

### 2. ****Write a Java 8 program to print even numbers from a list.****

**Answer:**

java

CopyEdit

List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6);

numbers.stream()

.filter(n -> n % 2 == 0)

.forEach(System.out::println);

### 3. ****How do you convert a list of Strings to uppercase using streams?****

**Answer:**

java

CopyEdit

List<String> names = Arrays.asList("raj", "amit", "neha");

List<String> upperNames = names.stream()

.map(String::toUpperCase)

.collect(Collectors.toList());

System.out.println(upperNames);

### 4. ****What is a functional interface? Give an example.****

**Answer:**

A functional interface is an interface that contains only **one abstract method**.

**Example:**

java

CopyEdit

@FunctionalInterface

interface MyFunctionalInterface {

void sayHello();

}

You can implement it using a lambda:

java

CopyEdit

MyFunctionalInterface greet = () -> System.out.println("Hello!");

greet.sayHello();

### 5. ****What is the use of**** Optional ****in Java 8?****

**Answer:**  
Optional is a container object used to contain **not-null objects**. It is used to avoid NullPointerException.

**Example:**

java

CopyEdit

Optional<String> name = Optional.of("Raj");

name.ifPresent(System.out::println);

### 6. ****Find the first non-repeated character in a string using Java 8.****

**Answer:**

java

CopyEdit

String input = "programming";

Character result = input.chars()

.mapToObj(c -> (char) c)

.collect(Collectors.groupingBy(c -> c, LinkedHashMap::new, Collectors.counting()))

.entrySet()

.stream()

.filter(entry -> entry.getValue() == 1)

.map(Map.Entry::getKey)

.findFirst()

.orElse(null);

System.out.println(result); // 'p'

### 7. ****How to sort a list of employees by salary using streams?****

**Answer:**

java

CopyEdit

List<Employee> employees = Arrays.asList(

new Employee("John", 3000),

new Employee("Jane", 4000),

new Employee("Alex", 2500)

);

employees.stream()

.sorted(Comparator.comparing(Employee::getSalary))

.forEach(e -> System.out.println(e.getName() + ": " + e.getSalary()));

### 8. ****What is method reference in Java 8? Give examples.****

**Answer:**  
Method reference is a shorthand notation of a lambda expression to call a method.

**Syntax:**

java

CopyEdit

ClassName::methodName

**Example:**

java

CopyEdit

List<String> list = Arrays.asList("A", "B", "C");

list.forEach(System.out::println); // instead of s -> System.out.println(s)

### 9. ****Count number of words in a sentence using streams.****

**Answer:**

java

CopyEdit

String sentence = "Java 8 makes coding easy";

long count = Arrays.stream(sentence.split(" "))

.count();

System.out.println("Word count: " + count);

### 10. ****How to check if a string contains only digits using Java 8?****

**Answer:**

java

CopyEdit

String input = "123456";

boolean isDigitOnly = input.chars().allMatch(Character::isDigit);

System.out.println(isDigitOnly); // true

## 🔄 Want More?

Let me know if you want:

* Real-world scenario-based Java 8 coding tasks
* More advanced Stream API questions
* Java 8 + Spring Boot questions

Would you like 50 or 100 Java 8 coding interview questions in a PDF or list format?

##### You said:

Yes

###### ChatGPT said:

Updated saved memory

Great! I’ll prepare a list of **50–100 Java 8 coding interview questions** with answers and examples, covering:

* ✅ Lambda expressions
* ✅ Stream API
* ✅ Functional interfaces
* ✅ Optional
* ✅ Method references
* ✅ Collectors
* ✅ Date and Time API
* ✅ Real-time coding challenges

I'll provide this as a structured **PDF** and also as a readable list here. You’ll be able to download and use it for interview prep or team practice.

📌 **ETA:** Would you prefer I share it in:

* 📄 PDF format only?
* 📃 List format here and PDF both?

Also, do you want the questions to be:

* For **intermediate** (3–6 years)?
* For **senior level** (8–12 years)?

Let me know so I tailor it properly for you.

##### You said:

explain those with examples : ✅ Lambda expressions ✅ Stream API ✅ Functional interfaces ✅ Optional ✅ Method references ✅ Collectors ✅ Date and Time API ✅ Real-time coding challenges

###### ChatGPT said:

Absolutely! Below is a detailed explanation of **Java 8 key features** with **syntax**, **examples**, and **use cases**, followed by some **real-time coding challenges**.

## ✅ 1. Lambda Expressions

### 🔹 What it is:

Lambda expressions allow you to create **anonymous functions**. They enable **functional programming** and reduce boilerplate code.

### 🔹 Syntax:

java

CopyEdit

(parameters) -> expression

### 🔹 Example:

java

CopyEdit

List<String> names = Arrays.asList("Raj", "Amit", "Neha");

names.forEach(name -> System.out.println(name));

### 🔹 Use Case:

Replacing anonymous inner classes for functional interfaces like Runnable, Comparator, etc.

## ✅ 2. Stream API

### 🔹 What it is:

Stream API processes **collections in a functional style**. You can chain operations like **filter, map, reduce**.

### 🔹 Example:

java

CopyEdit

List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6);

numbers.stream()

.filter(n -> n % 2 == 0)

.map(n -> n \* n)

.forEach(System.out::println); // Prints 4, 16, 36

### 🔹 Use Case:

Processing large collections easily with **less code and better performance** using lazy evaluation.

## ✅ 3. Functional Interfaces

### 🔹 What it is:

An interface with **only one abstract method**. Used as the basis for lambda expressions.

### 🔹 Example:

java

CopyEdit

@FunctionalInterface

interface MyInterface {

void show();

}

public class Test {

public static void main(String[] args) {

MyInterface obj = () -> System.out.println("Hello from Lambda!");

obj.show();

}

}

### 🔹 Built-in Functional Interfaces:

* Predicate<T>
* Function<T, R>
* Consumer<T>
* Supplier<T>

## ✅ 4. Optional

### 🔹 What it is:

Optional is a container that may or may not contain a non-null value. Helps prevent NullPointerException.

### 🔹 Example:

java

CopyEdit

Optional<String> name = Optional.ofNullable(getName());

name.ifPresent(System.out::println); // Only prints if not null

### 🔹 Use Case:

Avoid explicit null checks and write cleaner code when dealing with possibly null values.

## ✅ 5. Method References

### 🔹 What it is:

A shorthand for lambda expressions that **refer to a method** without executing it.

### 🔹 Syntax:

java

CopyEdit

ClassName::methodName

### 🔹 Example:

java

CopyEdit

List<String> list = Arrays.asList("A", "B", "C");

list.forEach(System.out::println); // instead of s -> System.out.println(s)

## ✅ 6. Collectors

### 🔹 What it is:

Part of java.util.stream.Collectors, used to **collect stream output** into a collection or string, map, count, etc.

### 🔹 Example:

java

CopyEdit

List<String> list = Arrays.asList("apple", "banana", "apple");

Map<String, Long> frequencyMap = list.stream()

.collect(Collectors.groupingBy(Function.identity(), Collectors.counting()));

System.out.println(frequencyMap); // {banana=1, apple=2}

### 🔹 Common Collectors:

* toList()
* toSet()
* toMap()
* joining()
* groupingBy()
* partitioningBy()

## ✅ 7. Date and Time API (java.time)

### 🔹 What it is:

New date/time API introduced in Java 8 in package java.time (immutable, thread-safe).

### 🔹 Example:

java

CopyEdit

LocalDate today = LocalDate.now();

LocalDate birthday = LocalDate.of(1990, Month.JULY, 15);

Period age = Period.between(birthday, today);

System.out.println("Age: " + age.getYears());

### 🔹 Common Classes:

* LocalDate, LocalTime, LocalDateTime
* Period, Duration
* DateTimeFormatter

## ✅ 8. Real-Time Coding Challenges

### 🔹 1. ****Find duplicate elements in a list using Java 8****

java

CopyEdit

List<Integer> list = Arrays.asList(1, 2, 3, 2, 4, 3);

Set<Integer> duplicates = list.stream()

.collect(Collectors.groupingBy(Function.identity(), Collectors.counting()))

.entrySet().stream()

.filter(e -> e.getValue() > 1)

.map(Map.Entry::getKey)

.collect(Collectors.toSet());

System.out.println(duplicates); // [2, 3]

### 🔹 2. ****Find the highest salary employee****

java

CopyEdit

Employee emp = employees.stream()

.max(Comparator.comparing(Employee::getSalary))

.orElse(null);

### 🔹 3. ****Group employees by department****

java

CopyEdit

Map<String, List<Employee>> deptMap = employees.stream()

.collect(Collectors.groupingBy(Employee::getDepartment));

### 🔹 4. ****Convert List<String> to comma-separated String****

java

CopyEdit

List<String> items = Arrays.asList("Java", "Spring", "AWS");

String result = items.stream().collect(Collectors.joining(", "));

System.out.println(result); // Java, Spring, AWS

### 🔹 5. ****Find First Non-Repeating Character in a String****

java

CopyEdit

String input = "programming";

Character result = input.chars()

.mapToObj(c -> (char) c)

.collect(Collectors.groupingBy(c -> c, LinkedHashMap::new, Collectors.counting()))

.entrySet().stream()

.filter(entry -> entry.getValue() == 1)

.map(Map.Entry::getKey)

.findFirst().orElse(null);

System.out.println(result); // 'p'

## ✅ 1. Predicate<T>

### 🔹 Description:

Represents a function that accepts a single input argument and returns a **boolean** result.

java

CopyEdit

@FunctionalInterface

public interface Predicate<T> {

boolean test(T t);

}

### 🔹 Example:

java

CopyEdit

Predicate<String> isLongerThan5 = str -> str.length() > 5;

System.out.println(isLongerThan5.test("Java")); // false

System.out.println(isLongerThan5.test("Predicate")); // true

### 🔹 Use Case:

* Filtering lists
* Conditional validations

java

CopyEdit

List<String> names = Arrays.asList("Raj", "Neha", "Prakash", "Tom");

names.stream()

.filter(name -> name.length() > 3)

.forEach(System.out::println);

## ✅ 2. Function<T, R>

### 🔹 Description:

Represents a function that takes an argument of type **T** and returns a result of type **R**.

java

CopyEdit

@FunctionalInterface

public interface Function<T, R> {

R apply(T t);

}

### 🔹 Example:

java

CopyEdit

Function<String, Integer> lengthFunction = s -> s.length();

System.out.println(lengthFunction.apply("Function")); // 8

### 🔹 Use Case:

* Data transformation
* Mapping operations

java

CopyEdit

List<String> names = Arrays.asList("Java", "Spring", "Cloud");

List<Integer> lengths = names.stream()

.map(name -> name.length())

.collect(Collectors.toList());

System.out.println(lengths); // [4, 6, 5]

## ✅ 3. Consumer<T>

### 🔹 Description:

Represents an operation that accepts a single input argument and returns **no result** (i.e., a side-effect action).

java

CopyEdit

@FunctionalInterface

public interface Consumer<T> {

void accept(T t);

}

### 🔹 Example:

java

CopyEdit

Consumer<String> printConsumer = str -> System.out.println("Hello " + str);

printConsumer.accept("Raj"); // Hello Raj

### 🔹 Use Case:

* Logging
* Printing/output operations
* Updating values

java

CopyEdit

List<String> names = Arrays.asList("Java", "8", "Features");

names.forEach(name -> System.out.println("Name: " + name));

## ✅ 4. Supplier<T>

### 🔹 Description:

Represents a supplier of results. It does **not take any input** but returns a result of type **T**.

java

CopyEdit

@FunctionalInterface

public interface Supplier<T> {

T get();

}

### 🔹 Example:

java

CopyEdit

Supplier<String> messageSupplier = () -> "Hello from Supplier!";

System.out.println(messageSupplier.get());

### 🔹 Use Case:

* Generating values
* Lazy loading
* Object creation

java

CopyEdit

Supplier<List<String>> listSupplier = ArrayList::new;

List<String> names = listSupplier.get();

names.add("Java");

System.out.println(names); // [Java]

## 🔚 Summary Table:

| **Interface** | **Method** | **Input** | **Output** | **Use Case** |
| --- | --- | --- | --- | --- |
| Predicate<T> | test() | T | boolean | Filter, condition checks |
| Function<T,R> | apply() | T | R | Transforming data (String → Integer etc.) |
| Consumer<T> | accept() | T | void | Display, logging, side-effect operations |
| Supplier<T> | get() | none | T | Data generation, lazy loading |