Java 8 Key Features with Examples

# Lambda Expressions

Lambda expressions allow you to write anonymous methods/functions.

Example:

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

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

# Stream API

Stream API allows functional-style operations on collections.

Example:

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);

# Functional Interfaces

An interface with a single abstract method, used with lambdas.

Example:

@FunctionalInterface

interface MyInterface { void show(); }

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

obj.show();

# Optional

Container for values that might be null. Helps avoid NullPointerExceptions.

Example:

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

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

# Method References

Short-hand syntax for lambdas calling existing methods.

Example:

list.forEach(System.out::println);

# Collectors

Used to collect stream results into lists, maps, strings, etc.

Example:

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

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

Date and Time API

New date/time classes are immutable and thread-safe.

Example:

LocalDate today = LocalDate.now();

LocalDate dob = LocalDate.of(1990, 7, 15);

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

# Real-Time Coding Challenges

- Find duplicates:

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());

- Highest salary employee:

Employee emp = employees.stream()

.max(Comparator.comparing(Employee::getSalary)).orElse(null);

- Group by department:

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

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

- List to comma-separated string:

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

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

- First non-repeating character:

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);