1. What are the main features introduced in Java 8?

Answer:
Java 8 introduced lambda expressions, functional interfaces, Stream API, default and static methods in interfaces, Optional class, method references, and the new Date and Time API.

---

2. What is a Lambda Expression?

Answer:
A lambda expression is a concise way to represent an anonymous function. It is mainly used to implement functional interfaces and helps reduce boilerplate code.

---

3. What is a Functional Interface?

Answer:
A functional interface is an interface with exactly one abstract method. It can have multiple default or static methods. Examples are Runnable, Comparator, and Callable.

---

4. Why do we need Functional Interfaces?

Answer:
Functional interfaces are required to support lambda expressions. They provide a clear contract for implementing behavior in a functional programming style.

---

5. What is @FunctionalInterface annotation?

Answer:
It is an optional annotation that ensures the interface contains only one abstract method and provides compile-time validation.

---

6. What is Stream API?

Answer:
Stream API is used to process collections in a functional and declarative way. It supports operations like filtering, mapping, and reducing data without modifying the original collection.

---

7. Difference between Collection and Stream?

Answer:
A Collection stores data, whereas a Stream processes data. Streams are lazy, single-use, and do not store elements.

---

8. What are Intermediate and Terminal Operations?

Answer:
Intermediate operations like filter() and map() return a stream and are lazy. Terminal operations like forEach(), collect(), and reduce() trigger execution.

---

9. Difference between map() and flatMap()?

Answer:
map() transforms each element one-to-one, while flatMap() flattens nested structures into a single stream.

---

10. Difference between filter() and map()?

Answer:
filter() is used for condition-based selection, while map() is used for transforming data.

---

11. What is Optional?

Answer:
Optional is a container object that may or may not contain a value. It is used to avoid NullPointerException and makes null handling explicit.

---

12. Why Optional is better than null?

Answer:
Optional forces developers to handle the absence of values explicitly and reduces runtime NullPointerException issues.

---

13. What is Method Reference?

Answer:
Method reference is a shorthand syntax of lambda expression used to refer to an existing method.

---

14. Types of Method References?

Answer:
There are four types:

1. Reference to static method
2. Reference to instance method of a particular object
3. Reference to instance method of an arbitrary object
4. Reference to constructor

---

15. What are Default Methods in Interface?

Answer:
Default methods allow interfaces to have method implementations without breaking existing implementations.

---

16. Why were Default Methods introduced?

Answer:
They were introduced to support backward compatibility, especially to enhance existing interfaces like Collection without breaking implementations.

---

17. Can we override default methods?

Answer:
Yes, implementing classes can override default methods.

---

18. What if two interfaces have same default method?

Answer:
The implementing class must override the method to resolve the ambiguity.

---

19. What is forEach() method?

Answer:
forEach() is a terminal operation used to iterate over elements in a stream or collection.

---

20. What is Predicate Functional Interface?

Answer:
Predicate represents a condition and returns a boolean value. It is commonly used in filtering operations.

---

21. What is Consumer Functional Interface?

Answer:
Consumer accepts a value and performs an operation but does not return any result.

---

22. What is Supplier Functional Interface?

Answer:
Supplier supplies a value without taking any input and is commonly used for lazy value generation.

---

23. What is Function Functional Interface?

Answer:
Function takes an input and returns an output. It is often used in map operations.

---

24. What is reduce() operation?

Answer:
reduce() combines stream elements into a single result, such as summing or multiplying values.

---

25. What is Collectors class?

Answer:
Collectors is a utility class that provides predefined methods to convert stream results into collections like List, Set, or Map.

---

26. What is parallelStream()?

Answer:
parallelStream() processes elements in parallel using multiple threads and can improve performance for large datasets.

---

27. Difference between stream() and parallelStream()?

Answer:
stream() processes data sequentially, while parallelStream() processes data concurrently using multiple threads.

---

28. Is Stream API thread-safe?

Answer:
Streams themselves are not thread-safe, but parallel streams internally manage threads using the ForkJoinPool.

---

29. What is the new Date and Time API in Java 8?

Answer:
Java 8 introduced the java.time package which provides immutable, thread-safe, and well-designed date and time classes.

---

30. Why was the old Date class replaced?

Answer:
The old Date class was mutable, not thread-safe, and had poor API design, so Java 8 introduced a better alternative.