Java 8 introduced **Method References**, a shorthand notation of **Lambda expressions** to refer directly to **existing methods or constructors**.

They make code more readable and concise when you're just **calling a method** without modifying or adding logic.

**🔷 What Are Method References?**

Instead of writing a lambda like:

str -> System.out.println(str)

You can use a method reference:

System.out::println

**✅ Syntax**

ClassName::methodName

objectRef::methodName

ClassName::new (for constructors)

**🧠 Types of Method References in Java 8**

| **Type** | **Example** | **Used When** |
| --- | --- | --- |
| **Static method reference** | ClassName::staticMethod | Lambda calls a static method |
| **Instance method on specific obj** | object::instanceMethod | Lambda calls an instance method on known object |
| **Instance method on class type** | ClassName::instanceMethod | Lambda calls method on lambda parameter |
| **Constructor reference** | ClassName::new | Lambda returns a new object |

**🔹 1. Static Method Reference**

**Example:**

class MathUtil {

public static int square(int n) {

return n \* n;

}

}

Function<Integer, Integer> func = MathUtil::square;

System.out.println(func.apply(5)); // 25

**🔹 2. Instance Method of a Particular Object**

class Printer {

void print(String s) {

System.out.println(s);

}

}

Printer printer = new Printer();

Consumer<String> c = printer::print;

c.accept("Hello"); // Hello

**🔹 3. Instance Method of an Arbitrary Object of a Particular Type**

List<String> names = Arrays.asList("john", "alice", "bob");

// Instead of: names.sort((s1, s2) -> s1.compareToIgnoreCase(s2));

names.sort(String::compareToIgnoreCase);

🔹 Here, Java knows compareToIgnoreCase will be called like s1.compareToIgnoreCase(s2).

**🔹 4. Constructor Reference**

class Person {

String name;

Person(String name) {

this.name = name;

}

}

Function<String, Person> personCreator = Person::new;

Person p = personCreator.apply("Avinash");

**🔁 Method References vs Lambdas**

| **Lambda** | **Method Reference** |
| --- | --- |
| x -> x.toLowerCase() | String::toLowerCase |
| (a, b) -> Integer.compare(a, b) | Integer::compare |
| () -> new ArrayList<>() | ArrayList::new |
| (str) -> System.out.println(str) | System.out::println |

**🧪 Practical Example**

import java.util.\*;

import java.util.function.\*;

public class MethodRefExample {

public static void main(String[] args) {

List<String> names = Arrays.asList("john", "alice", "bob");

// Method reference to static method

names.forEach(MethodRefExample::printName);

// Method reference to instance method of object

MethodRefExample obj = new MethodRefExample();

names.forEach(obj::greet);

// Method reference to constructor

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

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

}

public static void printName(String name) {

System.out.println("Name: " + name);

}

public void greet(String name) {

System.out.println("Hello, " + name);

}

}

**📌 Summary**

| **Type** | **Syntax Example** | **Equivalent Lambda** |
| --- | --- | --- |
| Static method | Class::staticMethod | (x) -> Class.staticMethod(x) |
| Instance method on object | obj::instanceMethod | (x) -> obj.instanceMethod(x) |
| Instance method on parameter | Class::instanceMethod | (x, y) -> x.instanceMethod(y) |
| Constructor | Class::new | () -> new Class() |

**❓ Common Interview Questions**

1. Can method references replace all lambdas?  
   → ❌ No, only when you're calling an existing method directly.
2. Is method reference just syntax sugar?  
   → ✅ Yes, it's a more concise form of lambda.
3. Can method references be chained?  
   → ❌ No, you combine them with other functional constructs like map().