3-Minute Class

⏳

Master Lambda Expressions & Method References!

Swipe

@Tasawar Abbas

What are Lambda Expressions?

Lambda Expressions in Java allow you to define anonymous methods, making your code more concise and readable. They enable you to pass methods as arguments, treat them as values, and bring functional programming to Java.

Introduced in Java 8, Lambdas simplify how we work with interfaces having a single abstract method, also known as **Functional Interfaces**.

@Tasawar Abbas

Swipe

Syntax of Lambda Expressions

Basic Syntax

LambdaExpression.java

```
1  (parameters) → expression
```

codesnap.dev

Example 1 (Single Statement):

LambdaExpression.java

```
1  (int a, int b) → a + b;
```

codesnap.dev

@Tasawar Abbas

Swipe

Example 2 (Multiple Statements):

LambdaExpression.java

```java
(int x, int y) → {
    int result = x * y;
    return result;
};
```

Note: Lambdas can omit parameter types if they can be inferred, and if there's only one parameter, you can omit parentheses.

LambdaExpression.java

```java
a → a * 2;
```

@Tasawar Abbas

Swipe

Benefits of Using Lambda Expressions

- **Concise Code** 📝

Lambdas reduce boilerplate code. No need to write anonymous classes, making your code much shorter and cleaner.

- **Enhanced Readability** 👓

Code becomes easier to read, especially when dealing with complex operations such as filtering and mapping collections.

- **Promotes Functional Programming** 🧠

Allows a more functional approach to Java programming, encouraging immutability and avoiding side effects.

- **Efficient Use in Streams API**

Works seamlessly with Java's Stream API, making collection processing and transformations more expressive.

What are Method References?

- **Method References** are a shorthand for lambda expressions that refer to methods or constructors directly.
- They use the :: operator and can be used to refer to **static methods, instance methods,** and **constructors**.

Method references are more concise when your lambda expression is simply calling a method.

Types of Method References

1. Static Method Reference: Refers to a static method of a class.

MethodReference.java

```
1  ClassName::staticMethod
2  //Example:
3  Math::max
4
```

2. Instance Method Reference of a Particular Object: Refers to an instance method of an existing object.

MethodReference.java

```
1  object::instanceMethod
2  //Example:
3  myPrinter::print
4
```

@Tasawar Abbas

Swipe

3. Instance Method Reference of an Arbitrary Object of a Particular Type: Refers to an instance method of an object type.

MethodReference.java

```
1  ClassName::instanceMethod
2  //Example:
3  String::toLowerCase
4
```

4. Constructor Reference: Refers to a class constructor.

MethodReference.java

```
1  ClassName::new
2  //Example:
3  ArrayList::new
4
```

@Tasawar Abbas

Swipe

Example of Lambda Expression vs Method Reference

MethodReferenceVsLambda.java

```java
// 1) Lambda Expression:
list.forEach(item → System.out.println(item));

// 2) Equivalent Method Reference:
list.forEach(System.out::println);
```

Explanation:

- The lambda expression takes an item from the list and prints it. The method reference System.out::println does exactly the same thing but more concisely, directly referring to the println method.

@Tasawar Abbas

Swipe

When to Use Lambda Expressions?

- **When passing behavior as a method argument: e.g.** Passing logic to process elements of a collection or handle events.

```
Runnable task = () → System.out.println("Task executed");
```

- **For functional-style operations on collections: e.g.** Filtering, mapping, or reducing a list with the Stream API.

```
List<Integer> evenNumbers = numbers.stream()
    .filter(n → n % 2 == 0)
    .collect(Collectors.toList());
```

- **For concise, inline code blocks: e.g.** Event listeners in GUI applications.

```
button.setOnAction(event → handleButtonClick());
```

@Tasawar Abbas

Swipe

When to Use Method References?

When a lambda expression is just calling an existing method

Method references are a more concise and readable alternative when the lambda expression is merely calling a method.

```
employees.forEach(Employee::printName);
```

For improved code readability

Method references simplify lambdas when they refer to existing methods, making the code easier to understand and maintain.

In constructor references for object creation

Method references can refer to constructors, making object creation more compact:

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

@Tasawar Abbas

Swipe

Conclusion

Lambda expressions and M**ethod references** are **powerful tools** that make Java code more expressive, concise, and functional. Mastering them can lead to more efficient, cleaner, and modern Java code. 🚀

@Tasawar Abbas

Swipe

🔓Unlock More 👨‍💻 Programming Power!

👍 Drop a **like** and share your thoughts or questions about **Lambda Expressions & Method References** in the **comments** below!

TASAWAR ABBAS

✨Inspiring Minds with Code Today 🤖