Method Reference

@FunctionalInterface

interface StringProcessor {

void process(String str);

}

public class LambdaMethodReferenceExample {

public static void main(String[] args) {

// Using a lambda expression

StringProcessor processor = str -> System.out.println(str.toUpperCase());

// Calling the process method

processor.process("hello");

}

}

or

public class LambdaMethodReferenceExample {

// A static method that converts a string to uppercase and prints it

public static void printUpperCase(String str) {

System.out.println(str.toUpperCase());

}

public static void main(String[] args) {

// Using a method reference

StringProcessor processor = LambdaMethodReferenceExample::printUpperCase;

// Calling the process method

processor.process("hello");

}

}

constructor reference

class Person {

private String name;

private int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public String toString() {

return "Person{name='" + name + "', age=" + age + "}";

}

}

@FunctionalInterface

interface PersonFactory {

Person create(String name, int age);

}

public class ConstructorReferenceExample {

public static void main(String[] args) {

// Using a constructor reference

PersonFactory personFactory = Person::new;

// Creating a new Person object using the factory

Person person = personFactory.create("Alice", 30);

// Printing the created Person object

System.out.println(person);

}

}

Line 1: PersonFactory personFactory = Person::new;

In this line, we are defining a PersonFactory instance using a constructor reference:

PersonFactory: PersonFactory is a functional interface that we've defined. It declares a single method create which takes two parameters (String name and int age) and returns a Person object.

Person::new: This is the constructor reference syntax. Here's what each part means:

Person: This is the name of the class whose constructor we are referring to.

::: This double colon (::) operator in Java is used for method references and constructor references.

new: This signifies that we want to reference the constructor of the Person class.

So, Person::new effectively means "a reference to the constructor of the Person class".

personFactory: This is the variable name that will hold the reference to our PersonFactory. Since PersonFactory is a functional interface, it can be assigned a reference to any method or constructor that matches its method signature (create(String name, int age) in this case).

Line 2: Person person = personFactory.create("Alice", 30);

In this line, we are using the personFactory instance to create a new Person object:

personFactory: This is the variable we declared earlier, which holds a reference to the constructor of the Person class (through the PersonFactory functional interface).

create("Alice", 30): This calls the create method on the personFactory instance. Remember, personFactory holds a reference to Person::new, so calling personFactory.create("Alice", 30) essentially calls the constructor new Person("Alice", 30).

Person person = ...: This line declares a variable person of type Person. The result of personFactory.create("Alice", 30) is assigned to person, which means person now holds a reference to a newly created Person object with the name "Alice" and age 30.