@techwithvishalraj

Java

Method References

Follow

Share

@techwithvishalraj

Method References

- Java provides a new feature called method reference in Java 8.
- Method reference is used to refer method of functional interface.
- Some times, It is compact and easy form of lambda expression.
- Each time when you are using lambda expression to just referring a method, you can replace your lambda expression with method reference.

- Types of Method References:

  1. Reference to a static method.
  2. Reference to an instance method.
  3. Reference to a constructor.

@techwithvishalraj

1. Reference to a static method.

You can refer to static method defined in the class. Following is the syntax and example which describe the process of referring static method in Java.

ContainingClass :: staticMethodName

```
@FunctionalInterface
interface Welcome{
   public void hello();
}

class Greet{
   public static void greeting(){
      System.out.println("Good Morning :) ");
   }
}

public class Test1 {
   public static void main(String[] args) {
      //return type(Welcome) must be Functional Interface
      Welcome welcome =Greet::greeting;
      welcome.hello();
   }
}
```

output: Good Morning :)

@techwithvishalraj

1. Reference to a static method.

You can also overload static methods by referring methods.

```
import java.util.function.BiFunction;
class Arithmetic{
    public static int add(int a, int b){
        return a+b;
    }

    public static double add(double a, double b){
        return a+b;
    }
}
public class Test2 {
    public static void main(String[] args) {
        BiFunction<Integer, Integer, Integer> addition = Arithmetic::add;
        BiFunction<Double, Double, Double> addition1 = Arithmetic::add;
        System.out.println(addition.apply(2, 5));
        System.out.println(addition1.apply(2.3, 5.4));
    }
}
```

output: 7
7.7

@techwithvishalraj

2. Reference to an Instance Method

instanceMethodRefClassObject :: instanceMethodRefName
new InstanceMethodRefClassName() :: instanceMethodRefName

```
@FunctionalInterface
interface Welcome{
   public void hello();
}
class Greet{
   public void greeting(){
      System.out.println("Good Morning :) ");
   }
}
public class Test3 {
   public static void main(String[] args) {
      //Referring non-static method using reference
      Greet greet = new Greet();
      Welcome welcome = greet::greeting;
      welcome.hello();
     //Referring non-static method using anonymous object
      Welcome welcome1 = new Greet()::greeting;
      welcome1.hello();
   }
}
```

output: Good Morning :)
Good Morning :)

@techwithvishalraj

3. Reference to a Constructor

ReferenceClassName :: new

```
@FunctionalInterface
interface Messageable{
    public void setMessage(String message);
}

class Message{
    Message(String message) {
        System.out.println(message);
    }
}

public class Test4 {
    public static void main(String[] args) {
        Messageable messageable = Message::new;
        messageable.setMessage("Hello");
    }
}
```

output: Hello

@techwithvishalraj

Thank you!

vishal-bramhankar

techwithvishalraj

Vishall0317