Method Reference

Agenda

1. What is Method Reference?
2. Lambda Expression vs Method Reference
3. Types of Method References in Java
4. Method Reference to an Instance Method
5. Method Reference to a Static Method
6. Method Reference to a Constructor
7. Method Reference to an instance method of an arbitrary object

What is Method Reference?

What is Method Reference?

- Method Reference is a new feature added in Java 8
- Method Reference is used to refer a method of Functional Interface
- Functional Interface is an Interface which has only one abstract method
- Method reference is a shorthand notation of a lambda expression to call a method
- The **::** operator is used in method reference to separate the class or object from the method name

Lambda Expression vs Method Reference

Lambda Expression vs Method Reference

```
//Before Java 8, to display all elements from List
Integer arr[] = new Integer[] {1, 2, 3, 4, 5};
List<Integer> list = Arrays.asList(arr);
for(Integer o : list)
    System.out.println(o +" ");


//In Java 8, Using Lambda Expression
list.forEach((Integer o) -> System.out.println(o));


//In Java 8, Using Method Reference
list.forEach(System.out::println);
```

Types of Method References in Java

Types of Method References in Java

Method Reference	Description	Example
Method reference to an Instance Method.	It is used to refer an Instance Method of a Class.	new String()::length equivalent to str.length().
Method reference to a Static Method.	It is used to refer a static method of a Class.	Arrays::sort equivalent to Arrays.sort(arr).
Method reference to a Constructor.	It is used to refer a Constructor of a Class.	ArrayList::new equivalent to new ArrayList()
Method reference to an instance method of an arbitrary object of a particular type	It is used to refer an instance method of an arbitrary object	String::compareToIgnoreCase

Method Reference to an Instance Method

Method Reference to an Instance Method

```
interface MyReference{
     void showMessage();
}

public class MethodReferenceDemo {

     public void myInstanceMethod() {
          System.out.println("Example for Instance Method Reference");
     }

public static void main(String[] args) {
     MethodReferenceDemo obj = new MethodReferenceDemo();
     //Referring Instance Method of a Class
     MyReference reference = obj::myInstanceMethod;
     //Calling the Functional Interface Method
     reference.showMessage();
   }
}
```

Output:

Example for Instance Method Reference

Method Reference to an Instance Method contd..

```
interface MyReference{
     int max(int a, int b);
}

public class MethodReferenceDemo {

     public int findMax(int a, int b) {
          if( a > b)
               return a;
          else
               return b;
}
public static void main(String[] args) {
     MethodReferenceDemo obj = new MethodReferenceDemo();
     //Referring Instance Method of a Class
     MyReference reference = obj::findMax;
     //Calling the Functional Interface Method
     System.out.println(reference.max(10, 20));
     }
}
```

Output:

20

Method Reference to a Static Method

Method Reference to a Static Method

```
interface MyReference{
     void showMessage();
}

public class MethodReferenceDemo {

     public static void myStaticMethod() {
          System.out.println("Example for Static Method Reference");
     }

public static void main(String[] args) {
     //Referring Static Method of a Class
     MyReference reference = MethodReferenceDemo::myStaticMethod;
     //Calling the Functional Interface Method
     reference.showMessage();
   }
}
```

Output:

Example for Static Method Reference

Method Reference to a Static Method contd..

```
interface MyReference{
     int max(int a, int b);
}

public class MethodReferenceDemo {

     public static int findMax(int a, int b) {
          if( a > b)
               return a;
          else
               return b;
}
public static void main(String[] args) {
     //Referring Static Method of a Class
     MyReference reference = MethodReferenceDemo::findMax;
     //Calling the Functional Interface Method
     System.out.println(reference.max(10, 20));
     }
}
```

Output:

20

Method Reference to a Constructor

Method Reference to a Constructor

```
interface MyReference{
     void showMessage();
}

public class MethodReferenceDemo {

     MethodReferenceDemo() {
          System.out.println("Example for Constructor Reference");
     }

     public static void main(String[] args) {
          //Referring Constructor of a Class
          MyReference reference = MethodReferenceDemo::new;
          //Calling the Functional Interface Method
          reference.showMessage();
    }
}
```

Output:

Example for Constructor Reference

Method Reference to a Constructor contd..

```
interface MyReference{
     void max(int a, int b);
}

public class MethodReferenceDemo {
     MethodReferenceDemo(int a, int b) {
          if( a > b)
               System.out.println(a);
          else
               System.out.println(b);
}

public static void main(String[] args) {
     //Referring Constructor of a Class
     MyReference reference = MethodReferenceDemo::new;
     //Calling the Functional Interface Method
     reference.max(10, 20);
   }
}
```

Output:

20

Method Reference to an instance method of an arbitrary object

Method Reference to an instance method of an arbitrary object

```
import java.util.Arrays;
import java.util.Comparator;
interface MyReference {
  void showMessage();
}
public class MethodReferenceDemo {
public static void main(String[] args) {
String[] myArray1 = { "Anitha", "valan", "Yuvashree", "manoj", "Sureka", "sahithi",
"Sirish" };
String[] myArray2 = { "Anitha", "valan", "Yuvashree", "manoj", "Sureka", "sahithi",
"Sirish" };
// using instance method of an arbitrary object of a particular type
Arrays.sort(myArray1, String::compareToIgnoreCase);
```

Method Reference to an instance method of an arbitrary object contd..

```
// Print Data
for (String name : myArray1)
   System.out.println(name);
// without instance method of an arbitrary object of a particular type
System.out.println("***************");
Comparator<String> stringComparator = (first, second) ->
first.compareToIgnoreCase(second);
Arrays.sort(myArray2, stringComparator);
// Print Data
for (String name : myArray2)
   System.out.println(name);
}
}
```

Output:

```
Anitha
manoj
sahithi
Sirish
Sureka
valan
Yuvashree
***************
Anitha
manoj
sahithi
Sirish
Sureka
valan
Yuvashree
```

Thank you