Method references

Method references in Java provide a shorthand syntax for defining a lambda expression that executes a specific method. They make the code more readable and concise. There are four kinds of method references:

1. **Reference to a static method**
2. **Reference to an instance method of a particular object**
3. **Reference to an instance method of an arbitrary object of a particular type**
4. **Reference to a constructor**

Let's look at examples for each type of method reference.

1. Reference to a Static Method

This type of method reference refers to a static method of a class.

Example:

```
import java.util.function.Function;

public class Main {
    public static void main(String[] args) {
        // Using a method reference to a static method
        Function<String, Integer> stringToInteger = Integer::parseInt;

        // Applying the function to convert a String to an Integer
        Integer result = stringToInteger.apply("123");

        // Printing the result
        System.out.println(result); // Output: 123
    }
}
```

2. Reference to an Instance Method of a Particular Object

This type of method reference refers to an instance method of a particular object.

Example:

```
import java.util.function.Supplier;

public class Main {
    public static void main(String[] args) {
        // Creating an instance of the class
        Main mainInstance = new Main();
```

```
        // Using a method reference to an instance method of a particular object
        Supplier<String> stringSupplier = mainInstance::instanceMethod;

        // Getting the result from the supplier
        String result = stringSupplier.get();

        // Printing the result
        System.out.println(result); // Output: Hello, world!
    }

    // Instance method
    public String instanceMethod() {
        return "Hello, world!";
    }
}
```

3. Reference to an Instance Method of an Arbitrary Object of a Particular Type

This type of method reference refers to an instance method of an arbitrary object of a particular type. It is often used with functional interfaces like Predicate, Consumer, etc.

Example:

```
import java.util.function.Predicate;

public class Main {
    public static void main(String[] args) {
        // Using a method reference to an instance method of an arbitrary object of a particular type
        Predicate<String> isEmpty = String::isEmpty;

        // Testing the predicate
        boolean result1 = isEmpty.test(""); // true
        boolean result2 = isEmpty.test("hello"); // false

        // Printing the results
        System.out.println(result1); // Output: true
        System.out.println(result2); // Output: false
    }
}
```

4. Reference to a Constructor

This type of method reference refers to a constructor.

Example:

```
import java.util.function.Function;

public class Main {
```

```
    public static void main(String[] args) {
        // Using a method reference to a constructor
        Function<String, StringBuilder> stringBuilderCreator = StringBuilder::new;

        // Creating a new StringBuilder instance
        StringBuilder stringBuilder = stringBuilderCreator.apply("Hello");

        // Printing the result
        System.out.println(stringBuilder.toString()); // Output: Hello
    }
}
```

Summary

Method references are a concise way to use existing methods as lambdas. They improve readability and reduce boilerplate code. Here's a summary of the syntax for each type:

1. **Reference to a static method**: ClassName::staticMethodName
2. **Reference to an instance method of a particular object**: instance::instanceMethodName
3. **Reference to an instance method of an arbitrary object of a particular type**: ClassName::instanceMethodName
4. **Reference to a constructor**: ClassName::new

Using method references can make the code cleaner and more expressive, especially when working with Java's functional interfaces.