

# Гадание на кофейной гуще, или передача по значению в Java

к.ф.м.н. Андрей Озеров

(Примечание редактора колонки Андрея Терешко, далее — прим. редактора: Не стоит гадать на кофейной гуще, надо больше пить кофе — этот прелестный напиток. ;-) )

Однажды, придя в очередной раз на техническое интервью для устройства на работу в одну IT — компанию, с целью проверки моих скромных познаний основ языка Java, меня спросили: “Как происходит передача объектов в методы в качестве параметров?”. Данный вопрос не застал меня врасплох, и я с гордостью ответил: “Передача примитивов в методы в качестве параметров происходит по значению, а объектов – по ссылке”. Данный ответ вполне удовлетворил моего будущего коллегу, и после непродолжительной беседы я был принят в компанию в качестве разработчика программного обеспечения. (прим. редактора: если бы спрашивающий более досконально разбирался в программировании на Java, то автора статьи не приняли бы на работу и ... пропал бы программист для Java. ;-) ).

Однако спустя некоторое время я все больше и больше стал задумываться над проблемой передачи и возвращении объектов в/из методов. На мои размышления огромное влияние оказала книга Брюса Эккеля, где он подробно описывает поведение объекта при передаче ссылки на него в качестве параметра методу. Да, я не оговорился, действительно передача объекта в метод происходит путем передачи ссылки на этот объект. (прим. редактора: точнее - передача методу в качестве параметра переменной некоторого типа. Тип может быть "примитивным" — это `boolean`, `char`, `int` и т.д. и "объектным" — это то, что не относится к примитивному типу. В случае объектного типа методу в качестве параметра действительно передается ссылка на объект — не передавать же весь объект в метод? Но, что же это за ссылка передается?). Тогда возьмем для примера простую программку и посмотрим, изменится ли объект, если в методе переопределить эту ссылку. (прим. редактора:

## Гадание на кофейной гуще, или передача по значению в Java

более точно, данный пример проверяет, окажет ли какое либо влияние в вызывающем методе изменение, совершенное в вызванном методе и заключающееся в изменении значения ссылки на параметр объектного типа, переданный методу).

```
public class HelloWorld {
    static void changeIt(String value) {
        value = new
String("Hello!");
    }
    public static void main(String[] argv)
{
        String test = new String("Hello
World!");
        changeIt(test);
        System.out.println("After changing : " +
test);
    }
}
```

Листинг 1.

Результат выполнения программы из Листинга 1 немного удивил меня. Объект *test* в методе *main* не изменил своего состояния после вызова статического метода класса — *changeIt*. (прим. редактора: если бы это произошло, то это было бы печально.) Следовательно, в данном случае здесь происходит какое-то дополнительное действие, которое я упустил.

Самое время обратиться к первоисточникам, вот что по этому поводу сказано в спецификации самого языка: “*Method parameters* (§8.4.1) name argument values passed to a method. For every parameter declared in a method declaration, a new parameter variable is created each time that method is invoked (§15.12). The new variable is initialized with the corresponding argument value from the method invocation. The method parameter effectively ceases to exist when the execution of the body of the method is complete.” Иными словами для каждого параметра метода создается своя локальная копия. Следовательно, в Java методах все аргументы передаются по значению. В случае, когда аргумент – примитивный тип, передача по значению означает то, что метод не может изменить оригинальное значение. Когда же аргумент – ссылка на объект, создается локальная копия ссылки, которая указывает на тот же самый объект, при

## Гадание на кофейной гуще, или передача по значению в Java

изменении которой оригинальное состояние объекта остается неизменным. (прим. редактора: верно - ВСЕ ПАРАМЕТРЫ В JAVA ПЕРЕДАЮТСЯ ПО ЗНАЧЕНИЮ. Если параметр — ссылка на объект, то ЗНАЧЕНИЕМ является ЗНАЧЕНИЕ самой ссылки, а не значение разнообразных полей в объекте, коих может быть великое множество, как по количеству, так и по разнообразию типов).

Теперь все становится на свои места. В случае программы из Листинга 1. мы передали ссылку на объект типа `String` в метод, где создавалась временная копия этой ссылки. (прим. редактора: точнее, сначала была создана копия ссылки (временная или нет — это отдельный разговор. Возможно, эта копия "переживет" и свой оригинал, ведь вызванный метод объекта может сохранить эту копию в поле своего или чужого объекта), а затем эта копия ссылки и была передана вызванному методу. То есть, в памяти уже содержатся две ссылки, содержащие одно и то же значение! И если бы в вызванном методе был вызван еще другой метод с передачей ему параметра в виде ссылки на тот же объект, то в памяти находилось бы уже три ссылки, содержащих одно и то же значение! И т.д.) Затем мы переопределили ссылку внутри метода, теперь она указывает на другой объект, но по возвращении из метода мы продолжаем работать с оригинальной ссылкой `test`, которая осталась неизменной.

Теперь давайте рассмотрим другой пример.

```
public class HelloWorld {
    private int iValue;
    static void changeIt(HelloWorld value) {
        value.iValue = 10;
    }
    public static void main(String[] argv)
    {
        HelloWorld hw = new HelloWorld();
        hw.iValue = 20;
        changeIt(hw);
        System.out.println("After changing : " +
hw.iValue);
    }
}
```

Листинг 2.

В данном случае мы создаем новый объект `hw` типа `HelloWorld`, присваиваем целому

## *Гадание на кофейной гуще, или передача по значению в Java*

полю значение `20`, и затем передаем ссылку на этот объект в статический метод класса, где пытаемся изменить состояние объекта. В результате после возвращения в метод `main` мы видим, что состояние оригинального объекта было изменено. Здесь мы передаем ссылку на оригинальный объект в метод, где через сам объект пытаемся изменить его состояние. В этом случае первоначальное состояние объекта, несомненно, будет изменено. (прим. редактора: точнее, передаем по значению(!) ссылку на объект, и, далее, используя эту переданную ссылку можно в вызванном методе изменять значение полей объекта, на который указывает переданная ссылка).

Подводя итог всему вышесказанному, хочется лишь отметить, что однозначного мнения нет, что имели в виду авторы и разработчики языка Java, когда определяли возможности передачи объектов в качестве параметров методов. (прим. редактора: авторы языка Java ОДНОЗНАЧНО имели в виду то, что ВСЕ ПАРАМЕТРЫ В JAVA ПЕРЕДАЮТСЯ ПО ЗНАЧЕНИЮ.) Однако на самом деле, вряд ли стоит уделять этому столь значительное внимание. Важно лишь помнить, что если Вы передаете ссылку на Ваш объект в метод, Вы должны быть готовы, что внутри этого метода состояние объекта может измениться. (прим. редактора: под "состоянием объекта" тут надо понимать — "значения полей объекта", которые действительно могут измениться. Но есть "непробиваемые" объекты — это объекты типа `String` и объекты типов классов-оболочек: `Boolean`, `Character`, `Integer` и т.п. Значение полей в объектах этих типов вообще нельзя изменить, даже владея ссылкой на объекты этого типа. И таких "непробиваемых" объектов в Java полно и, конечно, сам программист может создавать свои типы "непробиваемых" объектов, например, устанавливая модификатор доступа `private` для поля объекта и не предоставляя метод для его модификации.)

(Заключительное примечание редактора: Отсутствие "однозначного мнения", особенно у начинающих программировать на Java или, что более характерно, перешедших к программированию на Java с других языков вызвано, по моему мнению, тем, что в других языках программирования ссылка может содержать значение(!), указывающее на другую(!) ссылку. В Java ОБЪЕКТНАЯ ССЫЛКА ВСЕГДА СОДЕРЖИТ ЗНАЧЕНИЕ, УКАЗЫВАЮЩЕЕ НА ОБЪЕКТ или значение `null`. В Java ссылка не может содержать значение, указывающее на другую ссылку!

Но это не означает, что в Java нельзя строить связанные списки или древовидные структуры. В Java объектная ссылка может содержать значение, указывающее на

## Гадание на кофейной гуще, или передача по значению в Java

объект, поле которого содержит объектную ссылку, содержащую значение, указывающее на другой объект... и т.д.

Если же программисту необходимо(!) сделать так, чтобы вызванный метод изменил значение ссылки, то надо "обернуть" эту ссылку специальным объектом (или использовать массив) и передать эту "обертку" в качестве параметра методу, который и изменит значение "завернутой в обертку" ссылки.

Можно запрограммировать и так, чтобы для изменения значения ссылки использовать возвращаемое значение метода:

```
String value = new String("Hello!");  
value = modifyValue(value);
```

Если программисту требуется, чтобы вызванный метод изменил переданные ему значения переменных "примитивных" типов, то эти переменные можно также "обернуть" специальным объектом (или использовать массив) и передать эту "обертку" в качестве параметра.)

(P.S. редактора колонки:

Представьте себе, что Вы, уважаемый читатель, совершаете путешествие. Во время своего изумительного путешествия Вы встречаете многих людей. Некоторым из встречных Вы КОПИРУЕТЕ часть своей дорожной карты, обращаясь к ним с просьбой показать Ваш дальнейший путь. Вам в ответ те, кого Вы спрашиваете, возвращают частичные КОПИИ своих дорожных карт, руководствуясь которыми Вы и продолжаете свое изумительное путешествие. Что было бы, если бы Вы вынуждены были для продолжения своего путешествия отдавать оригинал(!) своей дорожной карты в руки встречного? Вашу дорожную карту могли бы разорвать! И Ваше изумительное путешествие могло бы тогда закончиться весьма печально.

Конечно, если какой-нибудь встреченный Вами плохой человек захочет сжечь город, узнав от Вас местоположение(!) этого незащищенного города, по предоставленной ему частичной копии Вашей дорожной карты, то он это вполне может сделать. Но, если на частичной копии Вашей дорожной карты, предоставленной этому плохому человеку, нанесены только защищенные(!) крепости, то у Вас есть очень большая уверенность, что к моменту прихода в эту крепость Вы не найдете там одни развалины, а Вас и согреют и накормят, как Вы и рассчитывали.

И, руководствуясь заветами товарища Штирлица (запоминается последняя фраза), еще раз повторю — ВСЕ ПАРАМЕТРЫ В JAVA ПЕРЕДАЮТСЯ ПО ЗНАЧЕНИЮ.)

## **1. Литература**

- James Gosling, Bill Jo, Guy Steel, Gilad Bracha The Java™ Language Specification, Second Edition
- [http://java.sun.com/docs/books/jls/second\\_edition/html/typesValues.doc.html#28536](http://java.sun.com/docs/books/jls/second_edition/html/typesValues.doc.html#28536)
- Брюс Эккель, “Философия Java”, серия “Библиотека программиста”, издательство “Питер”, Санкт-Петербург, 2001 г.