den aktuellen Wert des statischen Attributs nextNumber im statischen Speicherbereich.

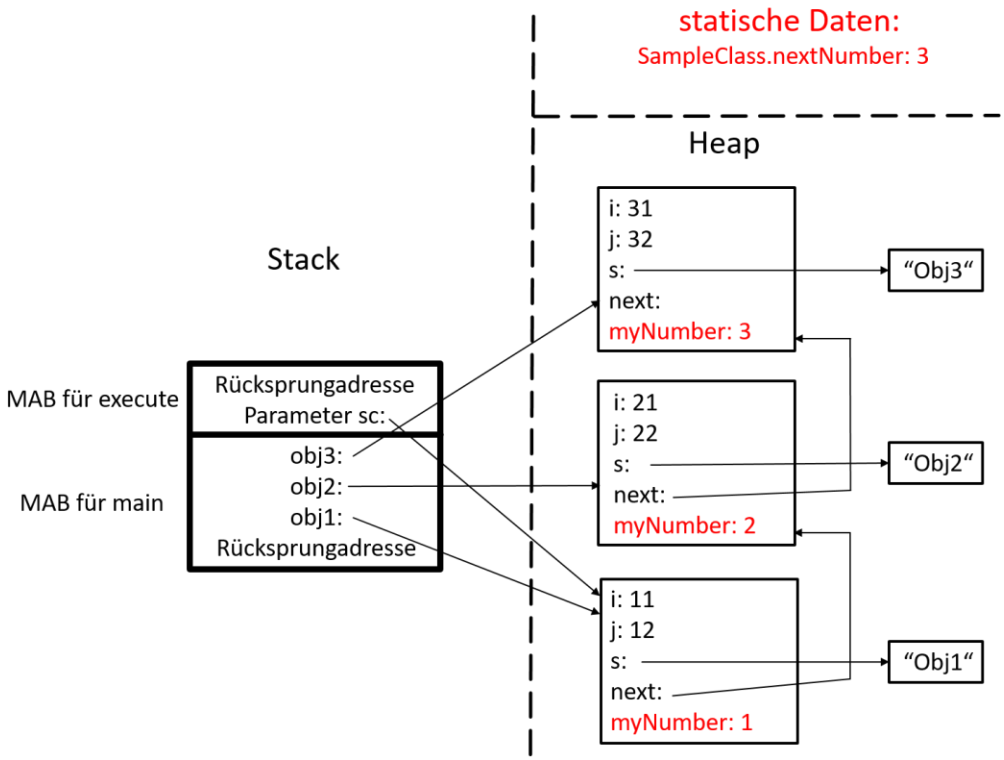

Abbildung 1.14: Situation von Stack und Heap beim Aufruf der Methode execute im obigen Beispielprogramm

Mit der Aufhebung der Beschränkung unserer Betrachtungen auf statische Methoden werden Sie schließlich vollständig verstehen, was beim Ablauf eines Java-Programms passiert.

1.5 Statische und nicht-statische Methoden

Beim Aufruf einer statischen Methode werden genau die Parameter übergeben, die im Programmcode angegeben sind. So war dies in den bisherigen Beispielen auch zu sehen. Eine nicht-statische Methode muss beim Aufruf immer auf ein Objekt angewendet werden. Dies entspricht einem Aufruf einer statischen Methode mit einem zusätzlichen Parameter des Objekts, auf das die Methode angewendet wird. Hierzu ein Code-Beispiel mit der nicht-statischen Methode method:

```
public class X
{
    private String code;

    public void method(String s, int i)
    {
        code = s + i; //bedeutet: this.code = s + i;
    }

    public static void main(String[] args)
    {
        X x = new X();
        x.method("Hallo", 17);
    }
}
```

Dies entspricht folgendem Programmcode, in dem die nicht-statische Methode in eine statische Methode „übersetzt“ wurde:

```
public class X
{
    private String code;

    public static void method(X thisX, String s, int i)
    {
        thisX.code = s + i;
    }

    public static void main(String[] args)
    {
        X x = new X();
        method(x, "Hallo", 17);
    }
}
```

Man sieht also, dass durch das „Übersetzen“ der nicht-statischen Methode eine statische Methode entsteht mit einem zusätzlichen Parameter vom Typ der Klasse, in der sich die Methode befindet (hier X). Der Name des Parameters lautet in Java this. Da this ein Schlüsselwort ist, konnte ich diesen Bezeichner im obigen Beispiel nicht verwenden und habe stattdessen thisX verwendet. Aber bitte beachten Sie, dass dies genau dem Parameter this bei nicht-statischen Methoden entspricht. Entsprechend wird aus dem Aufruf x.method… der Aufruf method(x, …). Das Objekt, auf das die Methode angewendet wird, wird als erster Parameter übergeben. Sie können sich damit den Aufruf nicht-statischer Methoden in einen entsprechenden Aufruf einer statischen Methode übersetzen und sollten damit eine gute Vorstellung haben, worum es bei nicht-statischen Methoden geht.
Wenn wir also den Ablauf des ersten der beiden obigen Programme (das mit nicht-statischer Methode) verfolgen und die Situation kurz vor der Rückkehr aus method betrachten, dann ist der Zustand so, wie er in Abbildung 1.15 zu

sehen ist. Die nicht-statische Methode method besitzt den Parameter this, der die Kennung des Objekts enthält, auf das method angewendet wurde.

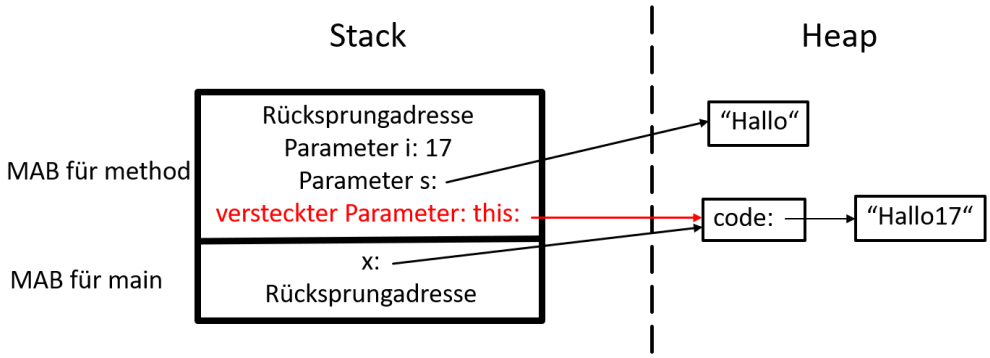

Abbildung 1.15: Situation von Stack und Heap beim Aufruf der nicht-statischen Methode method im obigen Beispielprogramm

Auch die Darstellung aus Abbildung 1.15 kann man sich wieder im Debugger ansehen (s. Abbildung 1.16). Links ist der MAB für method gezeigt unmittelbar nach dem Aufruf der Methode. Neben dem String-Parameter s und dem Parameter i des Typs int ist auch der versteckte Parameter this zu sehen, der eine Objektkennung enthält. Wie immer im Debugger, kann man das Dreieck neben dem Parameternamen ausklappen, um sich die Attribute anzuschauen. Dem Attribut code wurde noch nichts zugewiesen, entsprechend ist sein Wert null. In Abbildung 1.16 rechts ist die Situation unmittelbar nach der Zuweisung und kurz vor der Rückkehr aus method zu sehen. Das Attribut code enthält nun eine Kennung auf ein String-Objekt. Der Debugger zeigt durch die farbliche Hervorhebung an, dass sich der Wert dieses Attributs im letzten Schritt geändert hat.

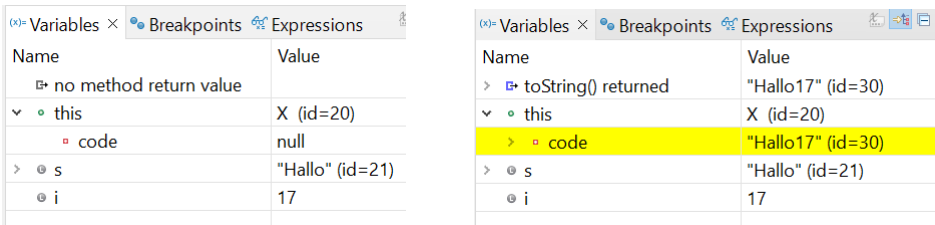

Abbildung 1.16: Variablenansicht im Debugger bei der Ausführung des obigen Programms mit nicht-statischer Methode method, links vor Ausführung der Zuweisung, rechts danach

Man sieht hieran auch nochmals schön, dass die Zuweisung eines Werts an code in der nicht-statischen Methode method eine abkürzende Schreibweise ist und mit code eigentlich this.code gemeint ist.

Zugriff auf die Attribute von Objekten, die im Heap gespeichert werden, erfolgen immer über die Kennung des betreffenden Objekts. Über die Kennung des Objekts gelangt man zunächst zu der Stelle, wo das Objekt im Heap liegt. Die Angabe des Attributnamens erlaubt dann, die genaue Stelle zu finden, wo der Wert des Attributs gespeichert wird. In vorliegenden Fall gelangt man also über this zur Stelle im Heap, wo das Objekt liegt. Durch Angabe des Namens code gelangt man an die richtige Stelle innerhalb des Speicherbereichs für das Objekt.
Wie Sie sicher wissen, wird this auch im folgenden Beispiel automatisch ergänzt, wie im Kommentar angezeigt wird:

```
public class X
{
    public void method1()
    {
        method2(); //bedeutet: this.method2();
    }

    private void method2()
    {
    }
}
```

Das heißt also, dass auch beim Aufruf von method2 der Parameter this vorhanden ist, denn ohne Angabe des Objekts, auf das method2 angewendet wird, wird – wie beim Zugriff auf Attribute (siehe oben) – automatisch this ergänzt (s. Kommentar im obigen Programmcode).
Ihnen sollte nun auch klar sein, warum man in einer statischen Methode nicht auf nicht-statische Attribute dieser Klasse (ohne Angabe eines Objekts) zugreifen kann. Wenn man nämlich einfach ein nicht-statisches Attribut a benutzt, wird bekanntlich automatisch this.a daraus. In einer statischen Methode fehlt aber der Parameter this. Deshalb macht this.a in einer statischen Methode keinen Sinn.
Damit sollten Sie ein ausreichendes Verständnis haben, was bei der Ausführung eines Java-Programms passiert. Im folgenden Abschnitt wollen wir mit diesem Wissen einen Sachverhalt betrachten, den Sie eigentlich schon kennen sollten, den Sie sich aber nun hoffentlich besser erklären können.