Java-Speicherlayout (Modell)

Als Java-Programmierer ist es wichtig, den Unterschied zwischen "Referenztypen" und "atomaren Datentypen" genau zu kennen. Auf der anderen Seite ist das Speichermanagement der Java-VM für den Programmierer relativ "unsichtbar".

Wir betrachten folgendes Programm:

```
public class MyClass {
    public int x;
    public int y;

    public static void main() {
        int h = 3;

        MyClass mcObjectA = new MyClass();
        mcObjectA.x = 3;
        mcObjectA.y = 4;

        MyClass mcObjectB = new MyClass();
        mcObjectB.x = 5;
        mcObjectB.y = 6;

        int[] a = new int[3];
        a[0] = 10;
        a[1] = 11;
        a[2] = 12;

        MyClass[] mcArray = new MyClass[2];
        mcArray[0] = mcObjectA;
        mcArray[1] = mcObjectB;

        MyClass dummyA;
        int[] dummyB;
    }
}
```

Im folgenden sind mit den $-Zahlen Speicheradressen gemeint, die in diesem Dokument ausgedacht sind und in Wirklichkeit nicht so einfach ermittelt werden können. Die Situation aus obigem Programm ist zunächst in Tabellenform dargestellt, darunter in Diagrammform. In der Tabelle sieht man z.B. dass in Speicherstelle $101 nicht etwa direkt die Objekteigenschaften x und y gespeichert sind, sondern stattdessen nur eine weitere Adresse ($200). Im Diagramm entspricht dies dem ersten Pfeil links oben.

Adresse	Inhalt	Ansprechbar durch (Typ) Name
$100	3	(int) n
$101	$200	(MyClass) mcObjectA
$102	$250	(MyClass) mcObjectB
$103	$400	(int[]) a
$104	$500	(MyClass[]) mcArray
$105	$null	(MyClass) dummyA
$106	$null	(int[]) dummyB