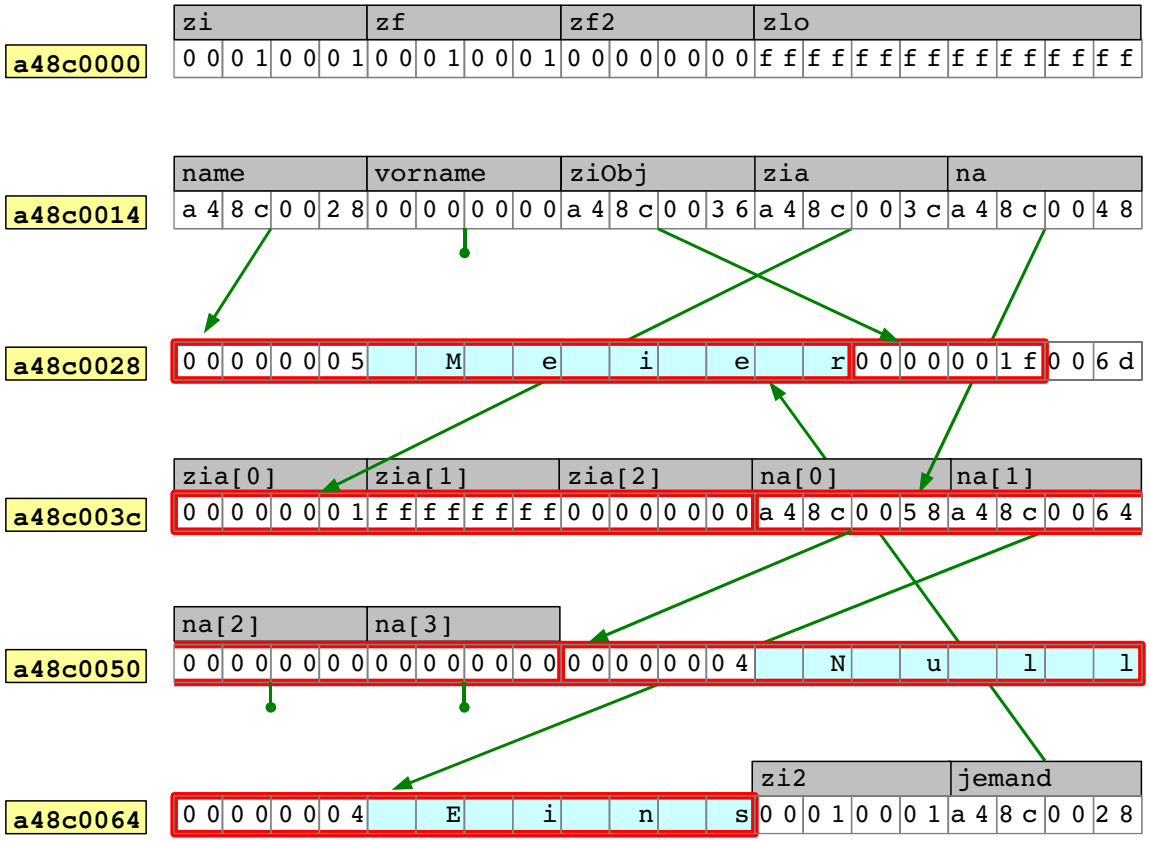
Wie in Java verschiedene Datentypen abgespeichert werden

```
// Einige primitive Typen
int zi=65537;
float zf=9.1837E-41;
float zf2;
long zlo=-1;
// Objekte
String name="Meier";
String vorname;
Integer ziObj=new Integer(31);
// Array aus primitiven Typen
int[] zia=new int[3];
zia[0]=1;
zia[1]=-1;
// Array aus Objekten
String[] na=new String[4];
na[0]="Null";
na[1]="Eins";
na[2]=null;
// noch zwei Zuordnungen
int zi2=zi;
String jemand=name;
```



Adresse

Name

Objekt

in hexadezimal

in Character

Jedes Kästchen ist ein Byte. Alle Bytes sind durchnummeriert, die Nummer nennt man Speicheradresse! Das Bild zeigt einen Speicherauszug ab der Adresse `a48c0000`. (Nur die erste Adresse in jeder Zeile wird angezeigt.)

In jedem Byte ist immer irgendetwas abgespeichert. Wird in Java eine Variable erzeugt und nichts zugewiesen, so wird trotzdem ein Wert dort abgelegt. Bei Zahlen ist es die Zahl 0, bei Objekten ist es der Verweis nirgendwo hin, genannt `null`. (Natürlich muss auch `null` als eine bestimmte Bitkombination abgespeichert werden, man verwendet die Kombination aus lauter Nullbits.) Um Buchstaben besser erkennen zu kennen wurde z.B. der Inhalt `0045` als farbig hinterlegtes `E` dargestellt (erster Buchstabe von „Eins“.

Die Darstellung lügt ein bisschen: Zu Objekten wird jeweils eine gewisse Menge Verwaltungsinformation mit abgespeichert. Dies wurde bei Strings durch die Angabe der Länge angedeutet. In Wirklichkeit ist es aber noch mehr. Bei den Arrays kommt in Wirklichkeit ebenfalls noch Verwaltungsinformation dazu. Aber das ist nicht so wichtig für das Verständnis.

Die Namensangaben über den Speicherzellen dienen der Orientierung; sie sind selbst nicht im Speicher. Woher der Prozessor z.B. weiß, welche 4 Bytes die Variable `zi1` beinhalten, ist eine andere Frage, die das Bild nicht beantworten will.