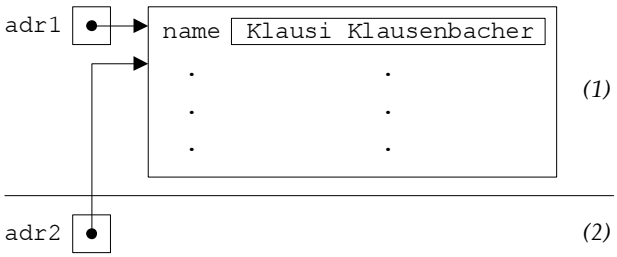

Abbildung 5.15: Klassen als Referenzdatentyp

Wir wollen unser Adressprogramm mit möglichst wenig Aufwand so erweitern, dass es statt einer *zwei* Adressen verwaltet. Hierzu schaffen wir zwei Objekte, die wir durch die Variablennamen `adr0` und `adr1` referenzieren:

```
Adresse adr0=new Adresse();
Adresse adr1=new Adresse();
```

Nun war es in unserem vorigen Programm (siehe Abschnitt 5.2.4) so, dass sämtliche Operationen (Ein- und Auslesen von Adressen) auf der Variablen `adr` ausgeführt wurden. Wie können wir das Programm so anpassen, dass es sowohl mit `adr0` als auch mit `adr1` arbeitet?
An dieser Stelle kommen uns eben die Referenzen zugute: Um etwa Daten aus dem Objekt `adr0` auszulesen, setzen wir einfach die Referenzen neu:

```
Adresse adr=adr0;
```

Auf diese Art verweisen `adr` und `adr0` auf dasselbe Objekt. Wenn wir also zum Beispiel durch den Befehl

```
adr.postleitzahl=IOTools.readInteger("PLZ        : ");
```

den Postleitzahleintrag von `adr` neu setzen, setzen wir damit automatisch auch den Eintrag von `adr0`. Wollen wir uns stattdessen um die Daten aus `adr1` kümmern, so müssen wir lediglich die Referenz neu setzen:

```
int n=IOTools.readInteger("Neue Adressennummer "
                          +"(zwischen 0 und 1):");
adr=(n==0)?adr0:adr1;
```

Diese Zeilen können wir etwa in den **`switch`**-Block unseres alten Programms einbauen und unser Menü somit um einen weiteren Auswahlpunkt („aktuelle Adresse wechseln“) erweitern. Die anderen Programmteile können wir direkt übernehmen, da sie sich allesamt mit der (von uns angepassten) Referenz `adr` befassen. Unser erweitertes Listing unterscheidet sich somit kaum von den auf Seite 135 dargestellten Zeilen:

```
1  import Prog1Tools.IOTools;
2
3  public class AdressBuch_v2 {
4
```