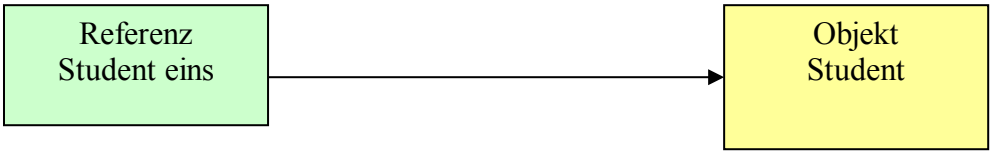
Referenz-Operationen

Es wurde bereits darauf hingewiesen, dass beim Umgang mit Objekten stets der wesentliche Unterschied zwischen der *Referenz* (Objektname) und dem eigentlichen *Objekt* zu beachten ist - die Referenz stellt lediglich einen Zeiger auf das Objekt dar und beinhaltet *nicht* die eigentlichen Objektdaten, sondern nur deren Speicheradresse.

Daher haben alle Referenz-Operationen (Operationen, die auf eine Referenz ausgeführt werden), keine direkten Auswirkungen auf das Objekt – sie betreffen zunächst nur den Inhalt der Referenzvariablen, also die ADRESSE des Objekts. Aus diesem Grund auch sind die von den Wert-Datentypen bekannten elementaren Operationen (arithmetische, logische, relationale..) für Referenzen nicht direkt verwendbar.

Letztlich haben Referenz-Operationen natürlich auch Auswirkungen auf die Objekte – aber anders, als man es erwarten würde.




```
Student eins = new Student ( );
```

Die unerwartete Auswirkung von Referenz-Operationen ist besonders in zwei Fällen zu beachten.

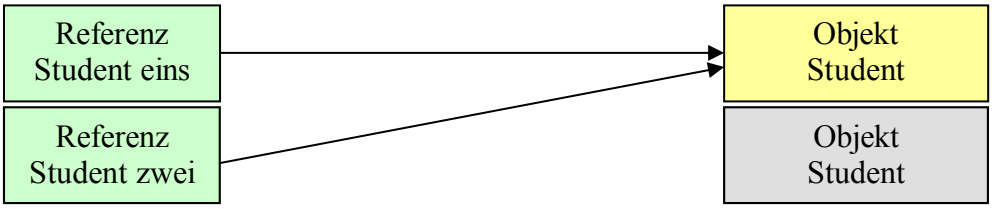
- Zuweisung von Referenzen mit dem Zuweisungs-Operator (=),
- Vergleich von Referenzen mit dem Vergleichsoperator (==).

Zuweisungsoperator (=)

Bei der Zuweisung von Referenzen mit dem **Zuweisungsoperator =** wird nur der Inhalt des Zeigers (nämlich die Adresse des Objektes) kopiert. Dabei werden keine Objektwerte kopiert, sondern der Inhalt der rechtsstehenden Referenz wird in die linksstehende Referenz kopiert.
Folge: Beide Referenzen zeigen hernach auf ein- und dasselbe Objekt und der ursprüngliche Inhalt der bei der Zuweisung links stehenden Referenz geht verloren. Damit verfällt das zweite Objekt der Garbage-Collection und wird zerstört. Man nennt diese Art der Referenz-Zuweisung auch "flache Kopie", weil hierbei keine Objektdaten kopiert werden.

```
Student eins = new Student ( );
Student zwei = new Student ( );

zwei = eins;  // Referenzen zeigen auf dasselbe Objekt
```



Vergleichsoperator (==)

Beim Vergleich von Referenzen mit dem Vergleichsoperator **==** wird NICHT der Wert der Objekte verglichen, sondern die **Adressen** der Objekte. Es ergibt sich nur dann der Wert `true`, wenn die Adressen identisch sind, es sich also tatsächlich um ein und dasselbe Objekt handelt.

```
Student eins = new Student ( );
Student zwei = new Student ( );

System.out.println( eins == zwei );        // false
System.out.println( eins.equals(zwei));    // false

eins = zwei; // Jetzt Zuweisung !

System.out.println( eins == zwei );        // true
System.out.println( eins.equals(zwei));    // true
```

Allgemeine Objekt-Methoden (von der Basisklasse `Object` abgeleitet)

Alle Objekte (auch Objektdatentypen) sind in Java von einer gemeinsamen Grundklasse namens `Object` abgeleitet. In dieser Klasse sind viele Methoden vorgegeben, die in jeder Klasse benutzt und überschrieben werden können. Die eben genannte Methode `equals` gehört hierzu.

Eine häufig benutzte Operation ist die Verkettung von Strings mit dem Operator **+**. Diese ruft implizit die Methode **`toString()`** der übergeordneten Basisklasse **`Object`** auf. Diese wandelt das angesprochene Objekt in einen String. Deshalb kann mit + eine Anzahl von Strings verkettet ausgegeben werden.

Die Methode `clone` erzeugt ein neues Objekt der Klasse, das exakt denselben Inhalt besitzt.

Die Methode `hashCode` berechnet einen Schlüssel, der dazu verwedet werden kann, um das Objekt in einer Hash-Tabelle zu speichern, bzw. wiederaufzufinden.

Weitere Informationen zu Object-Methoden sind der API-Dokumentation zu entnehmen.

http://java.sun.com/j2se/1.4.1/docs/api/