Quelle: https://o7planning.org/de/10213/vergleichen-und-sortieren-in-java J. Rau 08.2019

Primitive Datentypen vs. Reference Typen

1- einfacher Datentyp und Referenz-Typ in Java

Einfache Typen

Type	Bit/Bytes	Range
boolean	1 bit	true or false
char	16 bit/ 2 bytes	0 to 65535
byte	8 bit/ 1 byte	-128 to 127
short	16 bit/ 2 bytes	-32768 to 32767
int	32 bits/ 4 bytes	-2^{31} to -2^{31}
long	64 bits/ 8 bytes	-2^{63} to -2^{63}
float	32 bits/ 4 bytes	$-3.4028235 \times 10^{38}$ to 3.4028235×10^{38}
double	64 bits/ 8 bytes	-1.8×10^{308} to 1.8×10^{308}

Alle anderen Typen sind Objekt-Typen, sie sind die Reference Typen

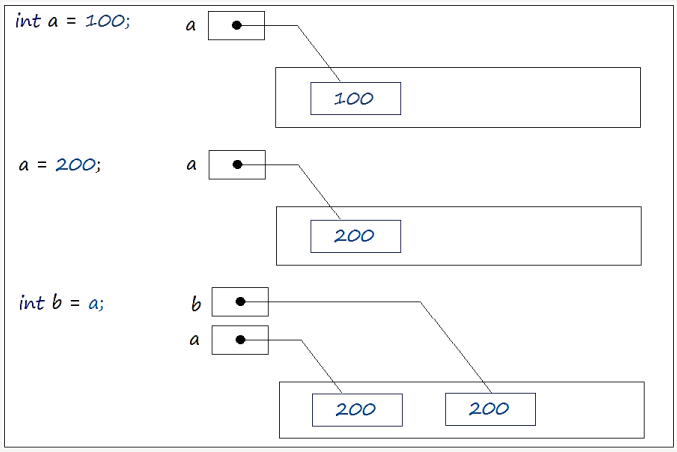
2- Speicherzuweisungen in Java

```
// Variable 'a' erstellen und die Wert 100 zuweisen.
int a = 100;

// neuen Wert für 'a' zuweisen
a = 200;

// Die Variable 'b' erstellen, b = a zuweisen.
int b = a;
```

Dann sind die Speicherzuweisungen von Java:



Referenz-Typ in der Speicherung

Der **new-Operato**r (z.B. v = new Vogel ()), erstellt im Speicher ein neues Objekt. Es wird eine Variable angemeldet und ihr Wert durch den **new-Operato**r erstellt**,** zum Beispiel:
`Vogel v;`
im Konstruktor: `a = new Object();`

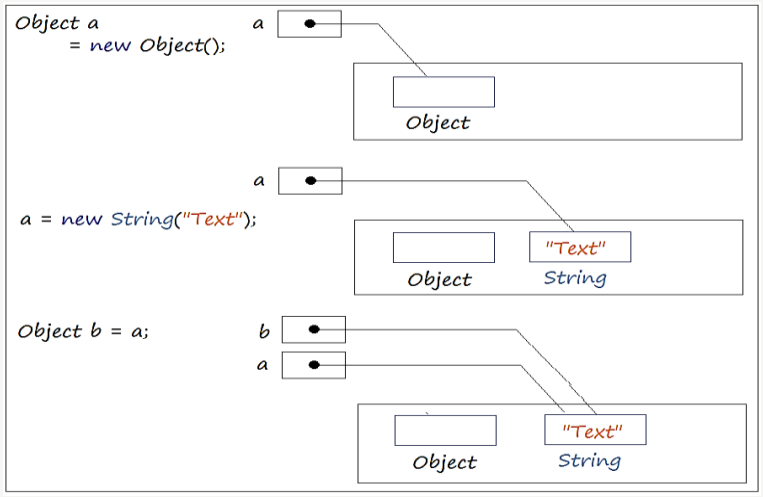
Java erstellt ein neues Objekt auf dem Heap und eine Reference **v**, die die Speicherposition des neu erstellten Vogel-Objekts auf dem Heap zeigt.

Wenn Sie eine Variable **b** mit **Objekt b = a initialisieren;** wird kein neues Objekt im Speicher erstellt. Java legt eine **Reference b an**, die auf die **gleiche Position wie a** zeigt.

```
// Das Objekt festlegen und initialisieren.
     Object a = new Object();

// neuen Wert für das Objekt 'a' zuweisen.
     a = new String("Text");

// Das Objekt 'b' festlegen und 'a' zuweisen
     Object b = a;
```



3- Daten und Objekte in Java auf Gleichheit prüfen

Im **Java** gibt es 2 Vergleichsmöglichkeiten: **== Operator** oder die Methode **equals (..)** benutzen.

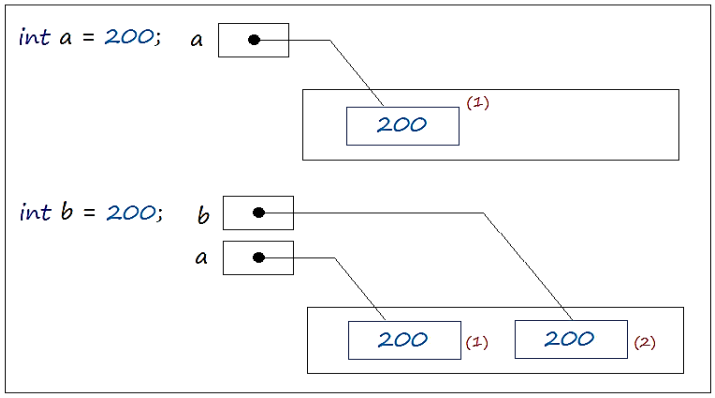
Der **== Operator** funktioniert für das Vergleichen von **primitiven Typen** und **Objekttypen, er vergleicht den Verweis der Speicherreferenz**.

equals (..) ist eine Methode zum **inhaltlichen** Vergleich für **Objekttypen.**

3.1- == Operator, um primitive Datentypen zu vergleichen

```
// Variable a erstellen, Wert 200 zuweisen.
// Ein Speicher (1) auf dem Stack wird für den Wert 200 erstellt.
     int a = 200;
// Eine Variable b erstellen, Wert 200 zuweisen
// Ein Speicher (2) auf dem Stack wird für den Wert 200 erstellt.
     int b = 200;
// Obwohl a und b auf 2 unterschiedlichen Speicher zeigen, gibt a == b
// true zurück, denn es werde die Referenzen verglichen.
```

Quelle: https://o7planning.org/de/10213/vergleichen-und-sortieren-in-java J. Rau 08.2019



3.2- == Operator, um Referenztypen zu vergleichen

```
// Achtung: Die folgenden Objekt-Initialisierungen sind nicht gleich:
     String str1 = "String 1";
     String str2 = new String ("String 1");

// new-Operator erstellt Speicherraum (1), der "This is text" enthält.
// und 's1' ist eine Referenz zu dem Speicherraum (1).
     String s1 = new String ("This is text");

// new-Operator erstellt Speicherraum (2), der "This is text" enthält.
// und 's1' ist eine Referenz zu dem Speicherraum (2).
     String s2 = new String ("This is text");

// Operator== um s1 und s2 zu vergleichen, Ergebnis: false.
// Der Grund ist, dass der Operator== mit dem Referenzstyp
// die Positionen in der Speicherung vergleicht.
     s1 == s2; // false
_____________________________________________________
// Es gibt keinen new-Operator.
// Java erstellt eine Referenz von obj und zeigt zum Speicherraum,
// zu dem s1 zeigt.
     Object obj = s1;
     obj == s1; // true
```

3.3- Methode equals (..) um die Referenz-Typen zu vergleichen

```
     String s1 = new String ("This is text");
     String s2 = new String ("This is text");

// Vergleich zwischen s1 und s2 durch die Methode equals(..)
// Gleicher Inhalt des Speicherbereichs
     boolean e1 = s1.equals(s2); //true

     s2 = new String (" This is new s2 text");

// Ungleicher Inhalt des Speicherbereichs
     boolean e2 = s1.equals(s2); // false
```