Entsprechend den verschiedenen Datentypen gibt es Grundoperatoren zur Verarbeitung dieser Daten. In Java wird zwischen **arithmetischen**, **logischen** und **Vergleichsoperatoren** unterschieden, die nur auf die entsprechenden Datentypen angewandt werden können.

1. Logische Operatoren (Boolesche Verknüpfungen)

	Sym.	Syntax	Beschreibung	
①	&&	Erg = Var_1 && Var_2	UND - Verknüpfung	Logische Operatoren verknüpfen boolesche Ausdrücke (hier Var_1 und Var_2). Das Ergebnis ist ein boolescher Ausdruck. **ERG = {true | false}**
②	||	Erg = Var_1 || Var_2	ODER - Verknüpfung	
③	!	Erg = ! Var_1	NEGATION	

2. Vergleichsoperatoren

	Sym.	Syntax	Beschreibung	
①	==	Erg = Var_1 == Var_2	Gleichheit (*)	Vergleichsoperatoren werden auf numerische (* auch für boolesche) Werte angewendet. Das Ergebnis ist ein boolescher Ausdruck. **ERG = {true | false}**
②	!=	Erg = Var_1 != Var_2	Ungleichheit (*)	
③	>=	Erg = Var_1 >= Var_2	Grösser (Gleich)	
④	<=	Erg = Var_1 <= Var_2	Kleiner (Gleich)	

3. Arithmetische-Operatoren

	Sym.	Syntax	Beschreibung	
①	+	Erg = Var_1 + Var_2	Addition	Arithmetische Operatoren werden auf numerische Werte angewendet. Hierbei ist der Datentyp von Erg abhängig von den Datentypen (siehe unten) der Operanden Var1 und Var2.
②	-	Erg = Var_1 - Var_2	Subtraktion	
③	*	Erg = Var_1 * Var_2	Multiplikation	
④	/	Erg = Var_1 / Var_2	Ganz. Division	
⑤	%	Erg = Var_1 % Var_2	Divisionssrest	

4. Bit Operatoren (bitweise Verknüpfungen)

	Sym.	Syntax	Beschreibung	
①	&	Erg = Var_1 & Var_2	bitweise AND Verk.	Bit-Operatoren Operatoren verknüpfen bitweise numerische Ausdrücke (hier Var_1 und Var_2). Das Ergebnis ist wieder ein numerischer Ausdruck.
②	|	Erg = Var_1 | Var_2	bitweise OR Verk.	
③	^	Erg = Var_1 ^ Var_2	bitweise XOR Verk.	
④	~	Erg = ~ Var_1	Bitweise NEGATION	

Durch den Wertebereich der numerischen Datentypen ergibt sich folgende „Reihenfolge“ bezüglich der Größe und Genauigkeit. Daraus resultiert (auch) der Datentyp arithmetischer Operationen.

byte > **short** > **int** > **long** > **float** > **double**

Bei Operationen mit gleichen Typen ist die Rückgabe gleich dem Typ aber mind. `int`

```
1.    byte + byte      = int
2.   short + short    = int
3.     int + int       = int
4.    long + long     = long
5.   float + float    = float
6.  double + double  = double
```

Bei Operationen mit ungleichen Typen ist die Rückgabe (mind.) `int` oder der größere Typ

```
1.  byte + short     = int;
2.  byte + long       = long;
3.   int + float     = float;
4.  long + double  = double;
```

Hinweis : Diese Regelen gelten nicht uneingeschränkt bei verkürzter Wertzuweisung !