

# Die Set-Implementierungen im Detail

## Variante 1: ArrayList

Die einfachste Lösung ist, wenn der ADT Set als interne Datenstruktur eine `ArrayList` verwendet. Beim Einfügen muss immer getestet werden, ob der Wert bereits in der `ArrayList` vorkommt. Dazu muss unter Umständen jedes Element der `ArrayList` untersucht werden.

### Vorteile:

- Einfache Implementation
- Speicherbedarf relativ gering (etwa proportional zur Anzahl der Werte)

### Nachteile:

- Suche nach einem Element bei großen Mengen aufwendig (proportional zur Anzahl der Werte)

## Variante 2: Bitvektor

Man kann eine Menge von Integer-Zahlen auch in eine einzelne Integer-Zahl verpackt darstellen, indem man die entsprechenden Bits in ihrer Binärdarstellung setzt. Man spricht dann von einem **Bitvektor**.

**Beispiel:** Die Menge  $\{0, 3, 4\}$  soll gespeichert werden. Dazu setzt man die Bits an den Stellen 0, 3 und 4 auf 1 und alle anderen auf 0 und bestimmt anschließend die Dezimaldarstellung der Binärzahl. Das Resultat ist die Zahl 25:

31	30	...	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1

 = 25

Da eine Integer-Zahl in Java 32 Bit lang ist, kann man mit einer Integer-Zahl also für die Zahlen von 0 bis 31 darstellen, ob sie in einer Menge enthalten sind oder nicht.

### Ist eine Zahl in der Menge enthalten?

Um herauszufinden, ob eine Zahl enthalten ist, kann man mit dem binären UND-Operator prüfen, ob das entsprechende Bit gesetzt ist:

```
int vier_gesetzt = 25 & 16; // Ergebnis: 16
int eins_gesetzt = 25 & 2;  // Ergebnis: 0
```

	31	30	...	8	7	6	5	4	3	2	1	0
25 =	0	0	...	0	0	0	0	1	1	0	0	1
16 =	0	0	...	0	0	0	0	1	0	0	0	0
2 =	0	0	...	0	0	0	0	0	0	0	1	0
25 & 16 =	0	0	...	0	0	0	0	1	0	0	0	0
25 & 2 =	0	0	...	0	0	0	0	0	0	0	0	0

Der binäre UND-Operator verknüpft zwei `int`-Zahlen (nach Umwandlung in die Binärdarstellung) und setzt im Ergebnis ein Bit auf 1, wenn die entsprechenden Bits in beiden Operanden auf 1 gesetzt sind.

Eine Zahl ist in der Menge enthalten, wenn bei der UND-Verknüpfung ein Wert herauskommt, der nicht 0 ist.

## Zahlen hinzufügen

Um einen Wert zu einem Bitvektor hinzuzufügen, verwendet man die binäre ODER-Verknüpfung.

**Beispiel:** Zur Menge  $\{0, 3, 4\}$  soll die 6 hinzugefügt werden. Man setzt also den Bitvektor auf

```
bitvektor = bitvektor | 64; // neuer Wert 89
```

	31	30	...	8	7	6	5	4	3	2	1	0
25 =	0	0	...	0	0	0	0	1	1	0	0	1
64 =	0	0	...	0	0	1	0	0	0	0	0	0
25   64 =	0	0	...	0	0	1	0	1	1	0	0	1

Der binäre ODER-Operator verknüpft zwei `int`-Zahlen und setzt im Ergebnis ein Bit auf 1, wenn mindestens eines der entsprechenden Bits in den beiden Operanden auf 1 gesetzt ist.

Beim Setzen eines Bits muss nicht geprüft werden, ob es bereits gesetzt ist – der ODER-Operator verändert in diesem Fall den Bitvektor nicht.

## Bitmasken

Zum lesen und setzen von Elementen benötigt man die entsprechenden Zahlen, letztlich in Binärdarstellung. Diese Zahlen, die man zum Auslesen oder Setzen von Bits verwendet werden als **Bitmaske** bezeichnet.

Eine Maske, bei der als niederwertigstes Bit nur das n-te Bit von rechts gesetzt ist, entspricht der Zahl  $2^n$ . Schnell und effizient erhält man diese Zahl, indem man die Zahl 1 um n Stellen nach links „verschiebt“.

From:  
<https://info-bw.de/> -

Permanent link:  
<https://info-bw.de/faecher:informatik:oberstufe:adt:set:implementationen:start?rev=1636912982>

Last update: **14.11.2021 18:03**

