Der ADT Set

Ein weiterer abstrakter Datentyp ist **Set** oder **Menge**, der eine Menge im mathematischen Sinne repräsentiert.

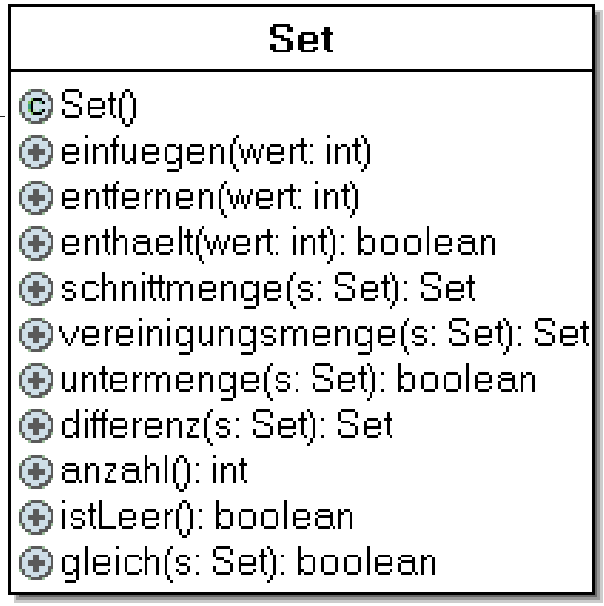
Eigenschaften von Mengen:

- Eine Menge kann **beliebig viele Elemente** enthalten.
- Jedes Element kann **höchstens einmal** vorhanden sein.
- Es kommt **nicht auf die Reihenfolge** der Elemente an.

Mehrere Implementationsvarianten mit Vor- und Nachteilen.

Wir betrachten Mengen von vorzeichenlosen int-Werten.

Mengenoperationen I



Set bietet die folgenden Operationen an:

- Konstruktor **`Set()`** – erzeugt eine leere Menge
- **`einfuegen(wert: int)`** – fügt den Wert `wert` in die Menge ein, falls er noch nicht vorhanden ist
- **`entfernen(wert: int)`** – entfernt den Wert `wert` aus der Menge, falls er vorhanden ist; andernfalls wird die Menge nicht verändert.
- **`enthaelt(wert: int): boolean`** – gibt `true` zurück, wenn `wert` in der Menge enthalten ist, sonst `false`.
- **`anzahl(): int`** – gibt die Anzahl der Elemente in der Menge zurück
- **`istLeer(): boolean`** – gibt `true` zurück, wenn die Menge leer ist, sonst `false`.

Mengenoperationen II

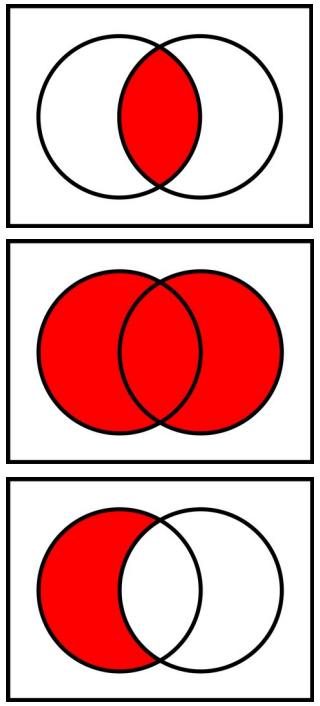
Set

- Set()
- einfuegen(wert: int)
- entfernen(wert: int)
- enthaelt(wert: int): boolean
- schnittmenge(s: Set): Set
- vereinigungsmenge(s: Set): Set
- untermenge(s: Set): boolean
- differenz(s: Set): Set
- anzahl(): int
- istLeer(): boolean
- gleich(s: Set): boolean

- `schnittmenge(s: Set): Set` – gibt eine Menge zurück, die genau die Elemente enthält, die in dieser Menge **und** in `s` enthalten sind.

  `vereinigungsmenge(s: Set): Set` – gibt eine Menge zurück, die genau die Elemente enthält, die in dieser Menge **oder** in `s` oder in beiden enthalten sind.

- `differenz(s: Set): Set` – gibt eine Menge zurück, die genau die Elemente enthält, die in dieser Menge, **aber nicht** in `s` enthalten sind.

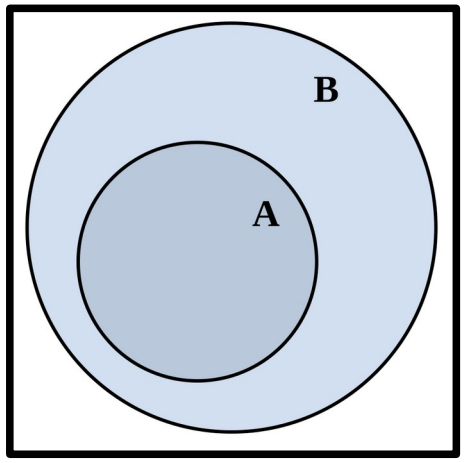


Mengenoperationen III

Set

- Set()
- einfuegen(wert: int)
- entfernen(wert: int)
- enthaelt(wert: int): boolean
- schnittmenge(s: Set): Set
- vereinigungsmenge(s: Set): Set
- untermenge(s: Set): boolean
- differenz(s: Set): Set
- anzahl(): int
- istLeer(): boolean
- gleich(s: Set): boolean

- `untermenge(s: Set): boolean` – gibt true zurück, wenn jedes Element dieser Menge in `s` enthalten ist
- `gleich(s: Set): boolean` – gibt true zurück, wenn diese Menge und s die gleichen Elemente enthält; sonst `false`.



Sets - subset A of B (GNU Free Documentation License)
https://commons.wikimedia.org/wiki/File:Set_subsetAofB.svg

Implemenationsvariante 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)