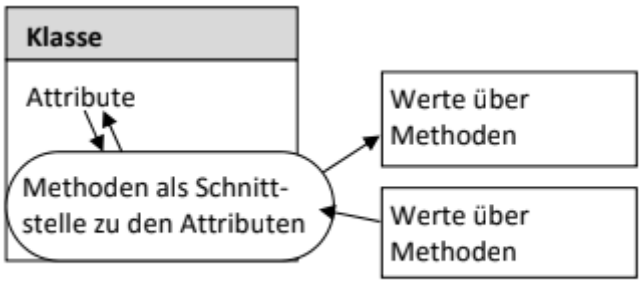
Kapselung

1 Zugriff auf Attribute kontrollieren

Eine Klasse kombiniert Daten (=Attribute / Instanzvariablen) und die Funktionalität (=Methoden) in einer Struktur. Sind diese Elemente sinnvoll vereint, spricht man grundlegend von Kapselung. Die Idee der Datenkapselung ist es, Attribute gegen aussen zu schützen, damit nicht jeder direkten Zugriff auf diese Daten hat und sie nur in kontrollierter Form verändert werden können. Nur die Methoden der Klasse selbst haben Zugriff auf die Attribute. Dies wird auch Information Hiding genannt.

Konzeptidee:



Kontrollierter Zugriff auf gekapselte Attribute

2 Zugriffsmodifizierer

Zugriffsmodifizierer sind Schlüsselwörter, mit denen die Zugriffsrechte für die Elemente einer Klasse (Attribute und Methoden) festgelegt werden. Bei der Definition werden Sie unmittelbar **vor** dem Datentyp definiert:

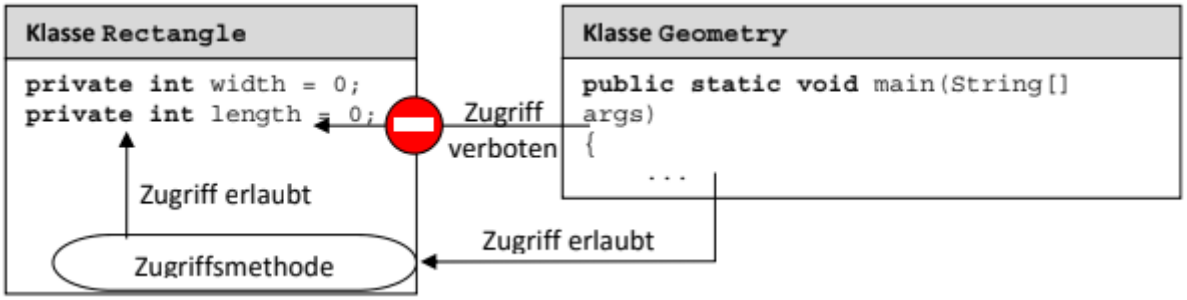
```
[modifier] type identifier;
```

Beispiel:

```
private double breite;
```

2.1 Attribute mit dem Zugriffsmodifizierer private kapseln

Um Attribute in einer Klasse zu kapseln (also gegen ausserhalb der Klasse zu verstecken), müssen diese `private` deklariert werden. Die so gekennzeichneten Attribute sind nur innerhalb der Klasse selbst sichtbar. Der Zugriff ist so weit eingeschränkt, dass nur Methoden der Klasse selbst darauf zugreifen können.



3 Zugriffsmethoden

In obigem Beispiel wurde die Klasse Rectangle definiert. Ein Rechteck besteht aus einer Breite (width) und einer Länge (length). Da die Attribute private deklariert wurden, müssen Methoden zur Verfügung gestellt werden, um diese Werte auszulesen oder zu verändern.

3.1 Getter Methoden

Um die Werte der Attribute auslesen zu können, werden sogenannte getter-Methoden definiert:

```
public class Rectangle {
      private int width = 0;
      private int length = 0;
              ①
      public int getWidth() {
            return width;
      }
      public int getLength() {
            return length;
      }
}
```

Die Methoden sind `public` definiert (somit von überall her sichtbar) und definieren einen Rückgabewert ①. Dieser Datentyp muss mit dem Datentyp der Variable, die mit `return` zurückgeliefert wird, übereinstimmen. Die Anweisung `return width;` liefert den Inhalt der Variable `width` als Rückgabe dem Aufrufer der Methode zurück.

- Der Name der Methode wird gewöhnlich aus dem Wort get („get" = erhalten) und dem Namen des Attributs gebildet, der dann mit einem Großbuchstaben beginnt.
- Eine Ausnahme bilden Attribute des Datentyps `boolean`. Den Namen für Getter-Methoden auf diese Attribute stellen Sie das Wort „is" voran. Die Methode heißt beispielsweise `isFilled`.

Von «aussen», also einer anderen Klasse, können nun diese Methoden auf einem Objekt der Klasse Rectangle aufgerufen werden, um die Werte auszulesen.

3.2 Setter Methoden

Die Attribute haben in diesem Beispiel zu Beginn den Wert 0:

```
public class Rectangle {
	private int width = 0;
	private int length = 0;
```

Sollen nun Werte für diese Attribute gesetzt werden können, müssen sogenannte setter-Methoden definiert werden:

```
public class Rectangle {
	private int width = 0;
	private int length = 0;
	//...
                              ①
	public void setWidth(int width) {
		this.width = width;
	}

	public void setLength(int length) {
		this.length = length;
	}
}
```

Setter Methoden definieren jeweils einen Parameter ①. Damit kann beim Aufruf der Methode ein Wert vom Typ int übergeben werden. In der Methode wird der Wert dieses Parameters in die Instanzvariable width gespeichert. Heissen Parameter und Instanzvariable gleich, muss vor die Instanzvariable das Schlüsselwort `this.` geschrieben werden. Somit ist klar, dass die Instanzvariable gemeint ist.

3.3 Konsequent mit getter- und setter-Methoden arbeiten

Sie sollten immer die entsprechenden Setter- und Getter-Methoden verwenden, auch wenn Sie aus anderen Methoden der Klasse auf die Attribute zugreifen möchten. Dies hat folgende Vorteile:

- Sofern spezielle Gültigkeitsregeln für ein Attribut gelten, müssen Sie diese nur einmal in der Setter-Methode berücksichtigen.
- Eine eventuell erforderliche Bearbeitung eines Wertes vor dem Speichern oder nach dem Auslesen eines Attributs muss nur einmal in der jeweiligen Getter- bzw. Setter-Methode programmiert werden.
- Änderungen beim Schreiben (Speichern) und Auslesen von Attributen müssen ebenfalls nur in den Zugriffsmethoden vorgenommen werden. Zugriffsmethoden senken somit den Wartungsaufwand.