Unter einer **Klasse** (auch *Objekttyp* genannt) versteht man in der objektorientierten Programmierung ein abstraktes Modell bzw. einen *Bauplan* für eine Reihe von ähnlichen Objekten. Die Klasse dient als Bauplan für die Abbildung von realen Objekten in Softwareobjekte und beschreibt Attribute (Eigenschaften) und Methoden (Verhaltensweisen) der Objekte (in C++ spricht man häufig vom Klassen-Member also Daten-Member und Funktions-Member). Verallgemeinernd könnte man auch sagen, dass eine Klasse dem Datentyp eines Objekts entspricht.

Quelle: https://de.wikipedia.org/wiki/Klasse_(Objektorientierung)

Hinweise:

- ③ und ④ deklarieren die Daten-Member der Klasse.
- ⑩ und ⑪ sind die Funktions-Member der Klasse.
- Bei ⑩ handelt es sich um eine Inline-Implementierung.
- Bei ⑪ zusammen mit ⑫-⑮ ist Definition und Deklaration voneinander getrennt.
- Mit ② und ⑤ werden die zulässigen Zugriffsmöglichkeiten der nachfolgenden Abschnitte festgelegt.
- ⑥ bis ⑨ sind spezielle Funktionen zum Anlegen, Kopieren oder Löschen von Objekten dieser Klasse.
- Eine Klassendatei muss nicht zwingend alle beschriebenen Elemente enthalten.

| | exemplarischer (typischer) Quellcode einer Klassendatei |
|---|---|
| ① | `class klassenName {` |
| ② | `  private:` |
| ③ | `  Datentyp attribut1;` |
| ④ | `  Datentyp attribut2 = wert;` |
| | `  :` |
| ⑤ | `  public:` |
| ⑥ | `  klassenName();` |
| ⑦ | `  klassenName(Datentyp);` |
| ⑧ | `  klassenName(klassenName & A);` |
| ⑨ | `  ~klassenName();` |
| | `  :` |
| ⑩ | `  void methode1(){ // Anweisungen }` |
| ⑪ | `  Datentyp methode2(Datentyp);` |
| | `  :` |
| | `}` |
| ⑫ | `klassenName::methode2(Datentyp variable){` |
| ⑬ | `  // Anweisungen` |
| ⑭ | `  return Datentyp;` |
| ⑮ | `}` |
| | `  :` |
| | klassenName.cpp |

Die Erzeugung von Objekten einer Klasse wird als Instanziierung bezeichnet. Dieses kann auf verschiedene Weisen (ähnlich wie beim Anlegen von Variablen) geschehen:

| | | |
|---|---|---|
| ❶ | **Automatisch:** Objekte einer Klasse werden automatisch instanziiert, wenn eine Variable dieser Klasse angelegt wird. | `klassenName objekt1;`<br>`klassenName objket2(wert);` |
| ❷ | **Dynamisch:** für das Objekt muss Speicher allokiert werden. Dieses geschieht mit dem **new** – Operator, der einen Pointer zurückliefert. Der mit **new** allokierter Speicher muss mit **delete** wieder freigegeben werden. | `klassenName * objekt3;`<br>`objekt3 = new klassenName;`<br><br>`objekt3 = new klassenName(wert);` |

Der Zugriff auf die Methoden erfolgt: im Fall ❶ über den Punktoperator (Strukturzugriff)
`objekt1.methode1();` bzw. mit Rückgabe `variable = objekt1.methode2(wert);`

im Fall ❷ über den Pfeiloperator (Indirektzugriff)
`objekt3 -> methode1();` analog `variable = objekt3->methode2(wert);`

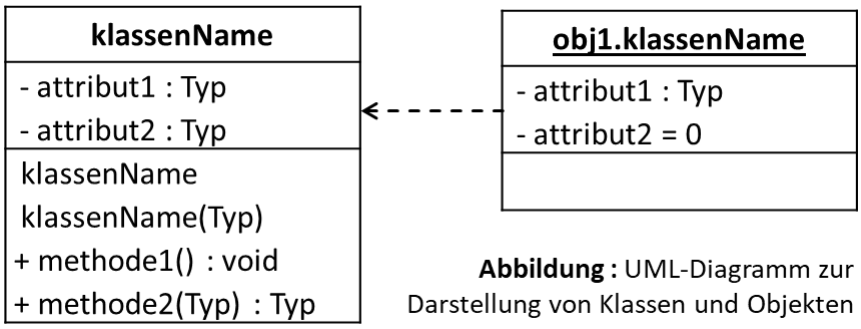


**Abbildung :** UML-Diagramm zur Darstellung von Klassen und Objekten