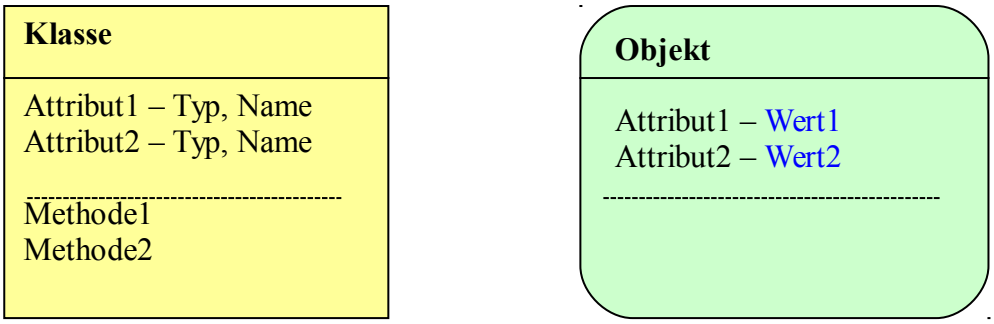
Objektzugriff

Klassen können bekanntlich aus zwei Komponententypen bestehen, nämlich Attributen und Methoden.

Attribute sind in der Klasse mit Datentyp und Namen deklariert – im Objekt sind sie mit jeweils eigenen Werten gefüllt. Man muss daher zwischen Attributen und den Attributwerten unterscheiden. Attribute können aus allen Datentypen bestehen, die in Java erlaubt sind, z.B. Wertdatentypen, Objektdatentypen (Vektoren, Strings), aber auch aus beliebigen anderen selbstdefinierten Objekttypen.



Alle Methoden, die in einer Klasse definiert sind, können prinzipiell von jedem Objekt dieser Klasse benutzt werden (sofern sie **`public`** sind - siehe Kapitel Zugriffskennzeichner). Im Unterschied zu Attributen sind Methoden kein integrierter Bestandteil jedes einzelnen Objektes, sondern sie existieren nur in der Klasse (im Code-Segment) und werden vom Objekt aufgerufen. Innerhalb einer Methode ist immer bekannt, für welches Objekt gerade gearbeitet wird, weil die Objekt-Adresse im `this`-Zeiger innerhalb der Methode bekannt ist.

Daher gilt: Eine Methode kann nur auf die Attribute desjenigen Objekts zugreifen, das sie aufgerufen hat.

Nachdem ein Objekt erzeugt worden ist, will man in der Regel auch etwas mit ihm anfangen, das heißt, man will seine Attributwerte ändern oder seine Methoden aufrufen. Wie macht man das?

Der **Zugriff** auf Attribute und Methoden eines Objekts geschieht durch Nennung

- des Objektnamens (bzw. des Index in einem Vektor), gefolgt von dem
- Punktoperator und
- dem Attributnamen bzw. Methodennamen.

```
einStudent.einkommen = 234; // Attribut

einStudent.gibAus( );     // Methode
```

Zugriffskennzeichnung für Attribute und Methoden

Die Attributwerte eines Objektes sind wichtige Informationen und sollten möglichst vor unbeabsichtigter Veränderung geschützt werden. Dies kann durch Festlegung eines Zugriffskennzeichners bei der Deklaration geschehen. Der Zugriffs-kennzeichner legt fest, ob das Objekt auf ein Attribut direkt zugreifen darf.

Die Zugriffskennzeichner in Java sind formal vergleichbar mit denen von C++, werden aber etwas anders verwendet. In Java werden die Schlüsselworte unmittelbar vor *jedes* Attribut / Methode gesetzt und sind somit Bestandteil des Datentyps. Die Festlegung muss in der Klassendefinition geschehen.

Im Gegensatz zu C++ ist die **Voreinstellung** `public`. Hinweis: Bei der Verwendung von Java-Paketen (siehe Kapitel Packages) müssen öffentliche Methoden ausdrücklich als `public` gekennzeichnet werden.

public: das Objekt kann auf die Komponente lesend und schreibend zugreifen.

private: das Objekt kann auf die Komponente nur über public-Methoden zugreifen.

protected: Ist nur im Zusammenhang mit Vererbung bedeutsam.

static ist hingegen kein Zugriffskennzeichner (siehe nächstes Kapitel). Static kennzeichnet eine private- oder public-Methode oder Attribut als zugehörig zur Klasse (Klassenmethode oder Klassenattribut) und NICHT zu einem bestimmten Objekt. Eine static-Methode kann aufgerufen werden, ohne dass ein Objekt der Klasse vorhanden ist. Ein als static gekennzeichnetes Attribut ist für alle Objekte der Klasse identisch (z.B. Objektzähler).

Es ist in Java üblich, als schützenswert erachtete Attribute einer Klasse als `private` zu deklarieren und dann spezielle **Zugriffsmethoden** (access-methods) zu definieren. Zugriff auf die private-Attribute eines Objektes ist dann ausschließlich über diese Zugriffsmethoden möglich. Dieses Verfahren dient der Datenkapselung und verhindert irregulären Zugriff auf Attributwerte.

Eine Methode zum Ausgeben von Attributwerten wird in Java als Accessor bezeichnet, eine Methode zum Setzen von Attributwerten als Mutator.

Vorsicht: Da jede Java-Anwendung aus einer mindestens einer Klasse besteht, in die `main` als statische Klassen-Methode eingebettet ist, hat der Kennzeichner `private` bei den Attributen dieser Klasse keine Auswirkung. Die main-Methode darf immer auf private-Attribute ihrer Klasse zugreifen (Beispiel 1).

BEISPIEL 1

```
////////////////////////////////
// MAIN ist Bestandteil der Klasse
// Achtung: Zugriff auf private Attribute / Methoden
// ist immer möglich

class Student {

   private int matrikelnr;   // Attribute
   private double einkommen;
   private String vorname, nachname;

   private void gibAus() {       // Accessor
      System.out.println("Student:    " + vorname +
                         " " + nachname);
      System.out.println("Einkommen:  " + einkommen);
      System.out.println("MatrikelNr: " + matrikelnr);
   }

   public static void main(String args[]) {

     Student eineStudentin = new Student(); // Objekt erzeugen

     eineStudentin.vorname    = "Hanna";     // OK !!!!!!!
     eineStudentin.nachname   = "Schneider"; // OK !!!!!!!
     eineStudentin.matrikelnr = 123;         // OK !!!!!!!
     eineStudentin.einkommen  = 4321;        // OK !!!!!!!

     eineStudentin.gibAus();       // Objekt-Methode aufrufen

   }
} // end class STUDENT
```

BEISPIEL 2:

```
////////////////////////////////
// Klasse mit Zugriffs-Methoden


class Student02 {

     private int matrikelnr;   // Attribute
     private double einkommen;
     private String vorname, nachname;

     public void gibAus2() {  // Accessor

        System.out.println("Student: " + vorname +
                           " " + nachname);
        System.out.println("Einkommen:  " + einkommen);
        System.out.println("MatrikelNr: " + matrikelnr);
     }
} // end Student2
```

```
////////////////////////////////////////////////////////
// Klasse mit main:

class Test {

  public static void main(String args[]) {

   Student02 eineFH_Studentin = new Student02();

   eineFH_Studentin.vorname    = "Marta";   // <- geht nicht!!!
   eineFH_Studentin.nachname   = "Schmitt"; // <- geht nicht!!!
   eineFH_Studentin.matrikelnr = 234;       // <- geht nicht!!!
   eineFH_Studentin.einkommen  = 5678;      // <- geht nicht!!!

   eineFH_Studentin.gibAus2();              // das ist OK !!!!!

   }
} // end class Test
```

Der Java-Compiler erzeugt aus den zwei Klassen zwei class-Dateien (`Student02.class` und `Test.class`). Da main Bestandteil der zweiten Klasse ist, sind die Attribute der ersten Klasse **nicht** zugreifbar und die Methode **`gibAus2()`** muss public sein, damit main darauf zugreifen kann.

BEISPIEL 3

```
////////////////////////////
// Zugriffsmethoden

class Student03 {

     ////////////////////////////
     // Attribute

     private int    matrikelnr;
     private double einkommen;
     private String vorname, nachname;

     ////////////////////////////
     // Zugriffs-Methoden

     public void liesEin() {       // Mutator

        System.out.print("Vorname:");
        vorname=In.readWord();
        System.out.print("Nachname:");
        nachname=In.readWord();
        System.out.print("MatrikelNr:");
        matrikelnr=In.readInt();
        System.out.print("Einkommen:");
        einkommen=In.readDouble();
      }

     public void gibAus() {       // Accessor
```

```
            System.out.println("Student:    " + vorname +
                               " " + nachname);
            System.out.println("Einkommen:  " + einkommen);
            System.out.println("MatrikelNr: " + matrikelnr);
        }
} // end class Student03

////////////////////////////////////////////

class Test {

  public static void main(String args[]) {

   Student03 eineStudentin = new Student03();
   eineStudentin.liesEin();
   eineStudentin.gibAus();

  } //end main

} //end class Test
```

Im Schaubild:

Student.java

```
class Student {

        private int Attribut;

        public methode() {
          System.out.print(Attribut);
        }
}
```

Test.java

```
class Test {

 public static void main(...) {
        Student t = new Student();
        t.methode();
 }
}
```