Klassen

Ein wichtiges Grundprinzip der objektorientierten Programmierung ist die Abstrahierung des darzustellenden Realitätsausschnitts in Typengruppen, den sogenannten **Klassen**. Eine Klasse ist so etwas wie ein Bauplan, der alle Bestandteile (Komponenten) eines bestimmten Typs definiert. Man kann sich eine Klasse als einen abstrakten „Begriff“ vorstellen, der dann durch Objekte realisiert werden kann – oder anders ausgedrückt: eine Klasse schreibt Eigenschaften und Verhaltensweisen vor, die von den Objekten mit individuellen Werten gefüllt werden.

Daraus geht hervor,

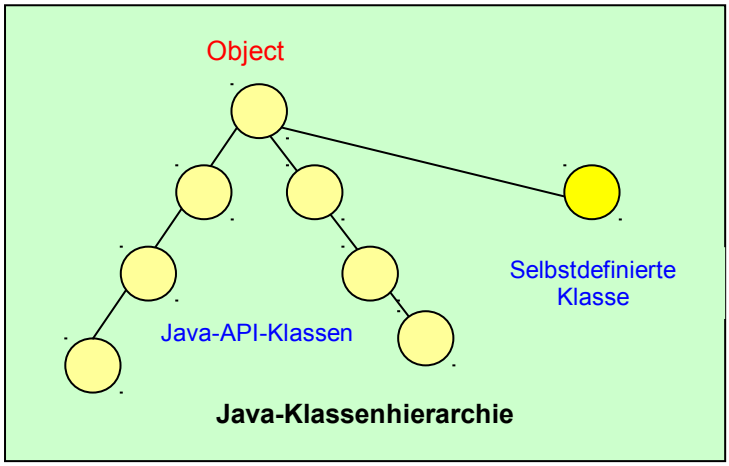
1. dass eine Klasse deklariert sein muß, bevor sie zum Erzeugen von Objekten benutzt werden kann,
2. dass die Existenz von Objekten von der Existenz der Klasse abhängig ist,
3. dass eine Klasse auch existieren kann, ohne dass es Objekte gibt.

```
class MyFirstClass {

     int a;                  // ATTRIBUTE
     double b;

     void set(int a1, b1) { // METHODE
          a = a1;
          b = b1;
     }
}
```

Java-Klassen können in „vertikaler“ und „horizontaler“ Abhängigkeit zueinander stehen (siehe Kapitel: Assoziation und Vererbung). Klassenbeziehungen und Klassenvererbung sind wichtige Modellierungsmittel der objektorientierten Programmierung. Die Beziehungen, die innerhalb einer Klasse definiert sind, werden durch die Objekte umgesetzt.

Außer den selbstdefinierten Klassen gibt es in Java eine große Anzahl von vordefinierten Systemklassen (API-Klassen), die bestimmte Funktionalitäten anbieten, die häufig gebraucht werden. Die API-Klassen ordnen sich in die große Java-Klassenhierarchie ein, die baumartig von einer einzigen Wurzel ausgeht.

Alle (auch selbstgeschriebene) Klassen sind automatisch von dieser Wurzelklasse abgeleitet und erben von ihr Attribute und Methoden, die grundlegenden Eigenschaften und Verhaltensweise von allen Objekten regeln. Es gibt also keine „freischwebenden“ Klassen wie beispielsweise in C++, sondern jede Java-Klasse ist grundsätzlich an einen bestimmten Ort im Java-Universum gebunden.



Weitere Bedeutung von Klassen

Klassen werden nicht nur zur Objekterzeugung gebraucht. Klassen sind auch in rein formaler Hinsicht die Grundeinheiten der Java-Programmierung, denn alles muß in Form von Klassen implementiert werden. Es gibt keine ausführbaren Sprachbestandteile, die außerhalb einer Klasse stehen dürfen (ausgenommen Anweisungen wie `package` oder `import`). Zum Verständnis kann der Hinweis darauf dienen, dass der durch den Kompiliervorgang erzeugte Bytecode immer in eine Klasse eingehüllt sein muß, um überhaupt ausführbar bzw. übertragbar sein zu können.

Auch in einer strikt objektorientierten Sprache wie Java werden Klassen (Hüllklassen) benötigt, die *nicht* zur Objekterzeugung dienen, sondern allgemeine Eigenschaften oder Funktionalitäten zur Verfügung stellen, die nicht an die Existenz von Objekten gebunden sind (entsprechend den prozeduralen C-Funktionen). Nicht-objektabhängige Methoden werden mit dem Schlüsselwort `static` gekennzeichnet - **`main()`** ist bspw. eine solche Methode.

In einer Klasse müssen alle Methoden mit Rumpf implementiert werden – (für C++-Programmierer: es gibt nur „inline-Methoden“). Die Reihenfolge, in der die Methoden implementiert werden, ist gleichgültig. Es wird keine Vorausdeklaration (Prototyp) benötigt, da bei der Kompilierung generell der gesamte Quellcode überprüft wird. Auch die Position von main innerhalb einer Klasse ist gleichgültig.

Speichernutzung: Klassendeklarationen sind immer Bestandteil des Code-Segments innerhalb der Java Virtual Machine - Objekte sind hingegen Bestandteil des Daten-Segments (Heap). Daraus geht hervor, dass Objekte keine Methoden „besitzen“ können, sondern nur Daten. Damit Methoden ausgeführt werden können, übergeben Objekte ihre eigene Adresse an die Methode und die Adresse wird dort im `this`-Zeiger aufgefangen – dies geschieht „unsichtbar“.