Objekte

Ein Objekt ist eine konkrete Ausprägung seiner Klasse. Was durch die Attribute und Methoden in der Klasse definiert ist, das stellt ein Objekt während des Programmablaufs mit individuellen, konkreten Werten gefüllt dar. Objekte sind eindeutig identifizierbare, individuelle Behälter für Daten (Attributwerte). Jedes Objekt kann die public-Methoden benutzen, die in seiner Klasse definiert sind.

> Im englischsprachigen Bereich ist die Bezeichnung `instance` gebräuchlich; die deutsche Übersetzung „Instanz“ ist missverständlich, wird aber oft verwendet. Eine Klasse (bzw. ein Objekt) zu instantiieren bedeutet, ein konkretes Objekt einer Klasse zu erzeugen.

Objekterzeugung

In Java wird ein Objekt üblicherweise in einem zweistufigen Prozess angelegt:

1. Zunächst wird ein Name vereinbart, unter dem das Objekt angesprochen werden soll. Dieser Name ist aber nicht das Objekt selbst, sondern es handelt sich nur um einen Zeiger (daher auch Referenz genannt). Die Referenz ist zunächst nicht initialisiert und enthält daher keine gültige Adresse, solange das eigentliche Objekt nicht erzeugt worden ist. Es ist deshalb ratsam, Objekt-Referenzen mit einem Nullwert vorzubelegen, damit nicht-initialisierte Objekte vor der Benutzung explizit geprüft werden können (Null-Pointer-Exception).

```
Student eins = null;
Klassenname Objektname;
```

   Die Referenz ist in der Regel von demselben Datentyp wie das Objekt, auf das sie zeigt (nämlich dem Klassentyp). Im Gegensatz zum eigentlichen Objekt wird die Objektreferenz in einem lokalen Gültigkeitsbereich des Programms (auf dem Stack) angelegt.

   Prinzipiell entspricht die Objektreferenz einem Pointer in C und C++. Da es in Java aber keine Bezeichnung für Zeiger (*) gibt, können alle Datentypen als Referenzen benutzt werden, ohne dass sie bei der Deklaration ausdrücklich als solche gekennzeichnet werden müssen.

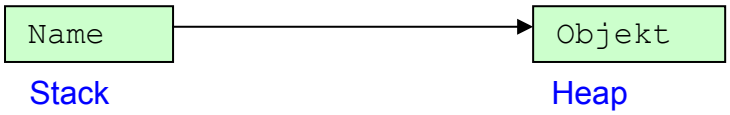
2. Das eigentliche Objekt wird erzeugt mit der Anweisung `new` mit nachfolgendem Konstruktoraufruf, der aus dem Klassennamen gefolgt von runden Klammern besteht. Erst ab diesem Moment ist das konkrete Objekt vorhanden und kann genutzt werden. Der Referenz wird als Rückgabewert die Adresse des neuen Objekts im dynamischen Speicher (heap) übergeben.

```
eins       = new Student();
Objektname = new Klassenkonstruktor();
```

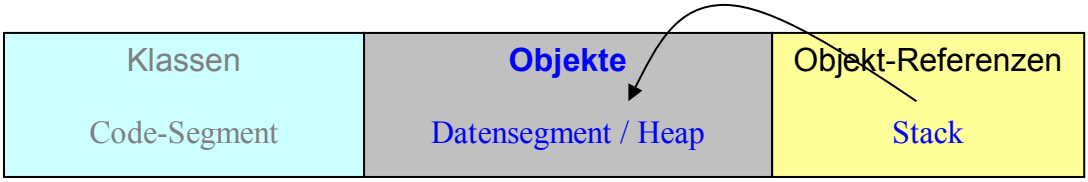
Beim Aufruf können dem Konstruktor Werte übergeben werden, die dazu benutzt werden können, um Objektattribute zu initialisieren (siehe nächstes Kapitel). Falls kein Konstruktor definiert worden ist, wird vom Java-Compiler ein Standardkonstruktor bereitgestellt.

Es ist üblich, die beiden Anweisungen in eine Zeile zusammenzufassen:

```
Student eins = new Student();
```



Jedes Objekt hat eine bestimmte Adresse im Heap des jeweiligen Java-Prozesses. Die Objektreferenz (der Name des Objekts) ist hingegen immer auf dem jeweiligen Stack angesiedelt.



Hinweis: Auch in Java existiert eine interne Speicherverwaltung über Adressen und Zeiger. Da es keine Operatoren für Pointer und Adressen gibt, besteht für den Programmierer keine Möglichkeit zur Speichermanipulation.

```
class Student {

    private String vorname;
    private int matrikelnr;

    Student(String name) { // Konstruktor

       this.vorname=name;
    }
```

```
        void gibAus() {

          System.out.println("Student Name: " + vorname);
        }
}

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

class Test1 {

    public static void main( String[] xyz)  {

        Student eins=null;
        System.out.println("Adresse :" + eins);
        eins = new Student("Hans");
        System.out.println("Adresse :" + eins);
     }
}
```

Die Ausgabe der Objektreferenz zeigt, dass die Speicherverwaltung der Laufzeitumgebung (Java Runtime Environment, Java Virtual Machine) auf eine besondere Weise funktioniert. Objekte haben eine Adresse, die sich als Kombination von Klassenname und einer Art Seriennummer darstellt. Hierbei handelt es sich um eine Art Hashcode:

```
Adresse : null
Adresse : Student@1a16869
```