# Konstruktor

- spezielle Methode zum Initialisieren bei Erzeugen eines Objekts mit `new`
- trägt Namen der Klasse
- hat keinen Rückgabetyp
- keiner angegeben: Compiler erzeugt Standard-Konstruktor (analog: Attribute werden mit Standard-Werten erzeugt)
- kann überladen werden

Beispiel:
Zweiter Konstruktor
für Klasse `Rational`

```
public Rational(int i) {
   Zaehler = i;
   Nenner = 1;
 }
```

# Statische Attribute

- `static`-Attribute einer Klasse:
  „Klassenvariable“

  ★ existieren **immer** genau einmal, unabhängig von Anzahl Instanzen

  ★ können von allen Instanzen gemeinsam verwendet werden (z.B.: Instanzzähler)

  ★ Zugriff von außen:
  *`Klassenname.Variablenname`*

Beispiel: `Math.PI`

# Statische Methoden

- `static`-Methoden einer Klasse: „Klassenmethoden“
- können unabhängig von der Existenz einer Instanz verwendet werden

- Zugriff von außen:
  *`Klassenname`*`.`*`Methodenname`*`(`*`[Parameter]`*`)`

  Beispiel: `Math.sin(`*`x`*`)`

- `public static void main(...) { ... }`

  macht aus jeder beliebigen Klasse ein Java-Programm

# Arrays
# (Vektoren, Matrizen)

# Arrays

- geordnete Sammlung von Elementen des gleichen Datentyps
- Deklaration:

```
Typ[] Arrayname;
```

- Erzeugung:

```
new Typ[Ausdruck];
```

- *Ausdruck*: ganzzahlig (!) – legt Größe fest
- Zugriff auf *n*tes Element (beginnend bei 0):

```
Ausdruck[n];
```

# Arrays

Beispiel:

```
int[] x;
x = new int[10];
int[] X = new int[10];    // dasselbe kuerzer
for(int i=0; i<X.length; i++)
    X[i] = i;

int[] y = {17, 0, 3, 2, -4};
System.out.println("y[3] = " + y[3]);
```

- können mit Komma-separierter Liste von Elementen in „{ ... }“ initialisiert werden
- Objekte – Attribut *Arrayname*`.length`: Anzahl der Elemente

# Arrays

Beispiel:

```
// Datei: Beispiel7.java

public class Beispiel7 {
        static double norm(double[] v)
         {  // Norm eines Vektors
          double summe=0;
          for(int i=0; i<v.length; i++)
            summe += v[i]*v[i];
          return Math.sqrt(summe);
         }

       /* Hauptprogramm */
       public static void main(String[] args) {
         System.out.println(norm(new double[] {2, -1, 1, Math.sqrt(3)}) );
    }
}
```

- Beachte : Aufruf von `norm()` mit „anonymem Array“ `new double[] { ... }`
- in `norm()`: Zugriff auf Elemente `v[i]`, `i=0, ..., v.length-1`

# Kommandozeilen-Parameter

```
// Datei: Beispiel8.java

public class Beispiel8 {
     public static void main(String[] Argumente) {
        for(int i=0; i<Argumente.length; i++)
          System.out.println((i+1) + ". Argument: "
                  + Argumente[i]);
    }
}
```

- Kommandozeilen-Parameter werden der Methode `main()` als Arrays vom Typ `String` übergeben
- Beispiel:

```
java Beispiel8 A Zwei Argument3 4
```

4 Argumente

# Mehrdimensionale Arrays

Beispiel:

```
int[][] M = new int[2][3];
```

- erzeugt 2x3-Matrix `M`
- ist Array von Referenzen auf Arrays – lange Variante:

```
int[][] M;
M = new int[2][];
for(int i=0; i<M.length; i++)
   M[i] = new int[3];
```

- Initialisierung: Schachtelung geschweifter Klammern:

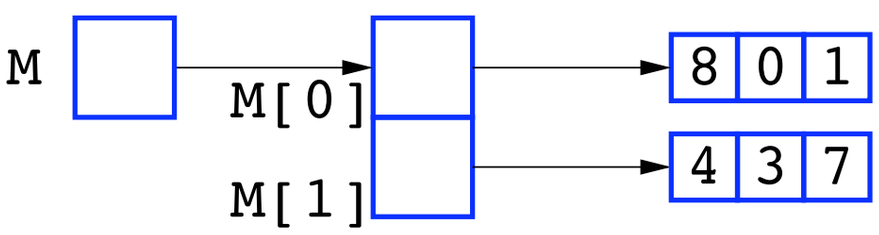
```
int[][] M = {{8, 0, 1}, {4, 3, 7}};
```

# Mehrdimensionale Arrays

Beispiel:

```
int[][] M = new int[2][3];
```

- erzeugt 2x3-Matrix `M`
- ist Array von Referenzen auf Arrays – lange Variante:




```
int[][] M;
M = new int[2][];
for(int i=0; i<M.length; i++)
   M[i] = new int[3];
```

- Initialisierung: Schachtelung geschweifter Klammern:

```
M.length = 2
M[0].length = 3
```

```
int[][] M = {{8, 0, 1}, {4, 3, 7}};
```

# Gauß-Elimination

Ein lineares Gleichungssystem $A\,\vec{x} = \vec{b}$

hat folgende Eigenschaften:

1. Das Vertauschen zweier beliebiger Zeilen von $A$ und der entsprechenden „Zeilen“ von $\vec{b}$ hat keinen Einfluss auf die Lösung (Umsortieren der Gleichungen).
2. Lösung $\vec{x}$ bleibt unter Linearkombination der Zeilen von $A$ unverändert, wenn die „Zeilen“ von $\vec{b}$ entsprechend transformiert werden.

# Gauß-Elimination

$A$: $n \times n$ Matrix

Für alle Spalten $s = 1, ..., n$

1. (*Pivotisierung*) Suche das Matrixelement mit dem größten $|A_{rs}|$. Vertausche dann die Zeilen $r$ und $s$ von $A$ und $\vec{b}$.

2. (*Elimination*) Für alle Zeilen $r$=1, ..., $n$ mit $r \neq s$ und $A_{rs} \neq 0$ ersetze

$$A_{rt} \leftarrow A_{rt} - \frac{A_{rs}}{A_{ss}} A_{st} \qquad \text{für } t = s, \ldots, n ,$$

$$b_r \leftarrow b_r - \frac{A_{rs}}{A_{ss}} b_s .$$

$\Rightarrow$ **Lösung**: $x_r = \dfrac{b_r}{A_{rr}}$ .

# Klasse `Arrays`

```
// Datei: Beispiel9.java
import java.util.*;  // Importiere Klasse Arrays

public class Beispiel9 {
        public static void main(String[] args) {
          String[] s = {"hase", "Igel", "Reh", "Ratte", "Fuchs",
                        "Ente", "Katze", "Hund", "hausKatze" };
          Arrays.sort(s);     // Sortiere s
          System.out.println("Sortiert: ");
          for(int i=0; i<s.length; i++)   // Ausgabe
           {
             System.out.print(s[i]);
             if(i < s.length-1)
               System.out.print(" , ");
             else
               System.out.println();
           }
    }
}
```

- implementiert nützliche Methoden, insbesondere `sort()` (sortieren)
  - `static void sort(`*Typ*`[] a)` sortiert die Elemente des Arrays `a` aufsteigend
  - `static void sort(`*Typ*`[] a, int from, int to)` sortiert im Bereich `from` bis `to-1`