Arrays und Vektoren

Arrays – Felder – Listen

Nicht alles ist dasselbe, nicht alles ist unterschiedlich!

Beachte:

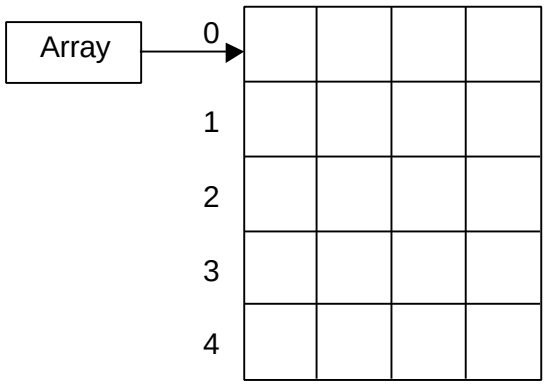
- Dies ist keine allgemein anerkannte Klassifizierung. Die Ausdrücke werden oft vermischt oder in einem anderen Sinn verwendet!
- Bei objektorientierter Programmierung wird unter „Feldern“ auch etwas ganz anderes verstanden: „Variablen“ einer Klasse
- Eine „Liste“ ist auch eine Datenstruktur, die eine bestimmte Definition hat (mit mehreren Varianten; Siehe dazu Vorlesung und Übung Algorithmen und Datenstrukturen 2).

Arrays (manchmal auch Felder genannt):

Geordnete Sammlung von Elementen genau identischen Typs und Länge, die über ganze Zahlen indiziert werden.

Beispiel (Modula):

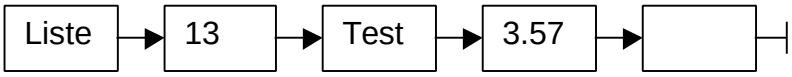
```
Array: ARRAY 5 OF ARRAY 4 OF INTEGER;
```



Listen:

Linear geordnete Sammlung von Elementen. Beispiel (Lisp):

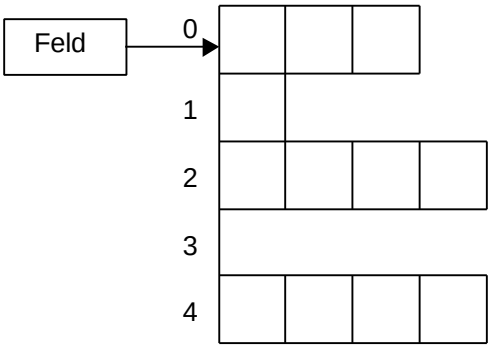
```
(setq Liste ´(”13” ”Test” ”3.57” ” ”))
```



Felder:

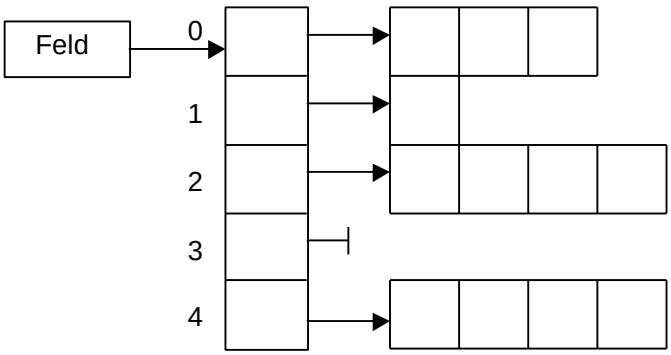
In Java haben alle Elemente von Feldern (=Arrays) genau denselben Typ, auch wenn dies nicht immer auffällt. Beispiel (Java):

```
int [5][] Feld; Feld[0]=new int[3]; Feld[1]=new
int[1]; Feld[2]=new int[4]; Feld[4]=new int[4];
```



Diese Zeichnung ist jedoch FALSCH!

In Wirklichkeit besteht das Array „Feld“aus 5 **Referenzen**, die jeweils auf int[] verweisen und daher den gleichen Typ und die gleiche Länge (die Länge einer Referenz) haben. Die **RICHTIGE** Darstellung ist daher:



Beispiel 1: Leere und nicht-existente Arrays

```
import java.io.PrintWriter;

public class Bsp1
{

static public void main(String args[])
{
    System.out.println("Programmstart\n\n");
    PrintWriter out=new PrintWriter(System.out,true);
    int[][] a={{1,2},{}};
        // Leeres Array b[1], aber es existiert
    int[][] b={{1,2},null};
        // Sub-Array b[1] existiert gar nicht !!! Kann
        // beim Kompilieren NICHT erkannt werden !!!
    int i,j;
    out.println("Array a ausgeben:");
    for(i=0;i<a.length;i++) // Arrays ausgeben
        for(j=0;j<a[i].length;j++)
            out.println(a[i][j]);
```

Array a ausgeben:

1

2

```
    out.println("Das Element a[1] wird zu einem Array der
Laenge 3 geaendert.");
    // Array einfuegen (altes wird automatisch zerstoert)
    a[1]=new int[3];
    a[1][0]=3;
    a[1][1]=4;
    a[1][2]=5;
    out.println("Array a (Version 2) ausgeben:");
    for(i=0;i<a.length;i++)
        for(j=0;j<a[i].length;j++)
            out.println(a[i][j]);
```

Das Element a[1] wird zu einem Array der Laenge 3 geaendert.

Array a (Version 2) ausgeben:

1

2

3

4

5

```
    // Array b ausgeben (Vorsicht!)
    out.println("\nArray b ausgeben:");
    for(i=0;i<b.length;i++)
        for(j=0;j<b[i].length;j++)
            out.println(b[i][j]);
    System.out.println("\n\nProgramende");
    // Wird nie erreicht, NullPointerException.
}
}
```

Array b ausgeben:

1

2

java.lang.NullPointerException

at Bsp1.main(Bsp1.java:40)

at symantec.tools.debug.MainThread.run(Agent.java:56)

Application terminated

Beispiel 2: Arrays vergleichen

```
public class Bsp2
{

static public boolean IsEqual(int[][][] a,int[][][] b)
{
// Prüfung der Parameter fehlt hier: Dimensionen
// müssen gleich sein und Elemente duerfen nicht
// "null" sein; Besser: While-Schleifen verwenden
    for(int i=0;i<a.length;i++)
        for(int j=0;j<a[i].length;j++)
            for(int k=0;k<a[i][j].length;k++)
                if(a[i][j][k]!=b[i][j][k])
                    return false;
    return true;
}

static public void main(String args[])
{
    System.out.println("Programmstart\n\n");
    int [][][] a={{{0,1,2,3},{4,5,6,7},{8,9,10,11}},
        {{12,13,14,15},{16,17,18,19},{20,21,22,23}}};
    // Welche Initialisierung per Index ist noetig, um
ein Array mit den gleichen Werten zu erhalten?
    int [][][] b=new int[2][3][4];
    b[0][0][0]=0; b[0][0][1]=1; b[0][0][2]=2;
    b[0][0][3]=3; b[0][1][0]=4; b[0][1][1]=5;
    b[0][1][2]=6; b[0][1][3]=7; b[0][2][0]=8;
    b[0][2][1]=9; b[0][2][2]=10; b[0][2][3]=11;
```

```
    b[1][0][0]=12; b[1][0][1]=13; b[1][0][2]=14;
    b[1][0][3]=15; b[1][1][0]=16; b[1][1][1]=17;
    b[1][1][2]=18; b[1][1][3]=19; b[1][2][0]=20;
    b[1][2][1]=21; b[1][2][2]=22; b[1][2][3]=23;
    int i,j,k;
    System.out.println("Array a ausgeben:");
    for(i=0;i<a.length;i++)
        for(j=0;j<a[i].length;j++)
            for(k=0;k<a[i][j].length;k++)
                System.out.println("a["+i+"]["+j+"]["+k+
                                   "]="+a[i][j][k]);
    System.out.println("\nArray b ausgeben:");
    for(i=0;i<b.length;i++)
        for(j=0;j<b[i].length;j++)
            for(k=0;k<b[i][j].length;k++)
                System.out.println("b["+i+"]["+j+"]["+k+
                                   "]= "+b[i][j][k]);
```

Array a ausgeben:

a[0][0][0]= 0

a[0][0][1]= 1

……

a[1][2][2]= 22

a[1][2][3]= 23

Array b ausgeben:

b[0][0][0]= 0

b[0][0][1]= 1

……

b[1][2][2]= 22

b[1][2][3]= 23

```
    // Vergleichen der beiden Arrays
    System.out.println("\nVergleich mit a.equals(b)");
    if(a.equals(b))
        System.out.println("a.equals(b)=TRUE");
    else
        System.out.println("a.equals(b)=FALSE");
/* Geht nicht, da a[0][0][0] ein int ist und kein Objekt
der Klasse Integer, daher koennen keine Methoden
aufgerufen werden!
    if(a[0][0][0].equals(b[0][0][0]))
        System.out.println("a.equals(b)=TRUE");
    else
        System.out.println("a.equals(b)=FALSE");
*/
```

Vergleich mittels a.equals(b)

a.equals(b)=FALSE *// Warum das, enthalten doch dieselben Werte (Siehe oben)????*

```
    // Händisch vergleichen
    System.out.println("\nVergleich mittels Prozedur");
    if(IsEqual(a,b))
        System.out.println("IsEqual(a,b)=TRUE");
    else
        System.out.println("IsEqual(a,b)=FALSE");
```

Vergleich mittels Prozedur

IsEqual(a,b)=TRUE

```
    // Das Vergleichen etwas genauer prüfen
    System.out.println("\na[0][0][0]=30 zuweisen");
    a[0][0][0]=30;
    System.out.println("a[0][0][0]="+a[0][0][0]+
                       ",b[0][0][0]="+b[0][0][0]);
    System.out.println("\nVergleich mittels equals");
    if(a.equals(b))
        System.out.println("a.equals(b)=TRUE");
    else
        System.out.println("a.equals(b)=FALSE");
```

a[0][0][0]=30 zuweisen

a[0][0][0]=30, b[0][0][0]=0

Vergleich mittels equals

a.equals(b)=FALSE *// Wie vorher*

```
    System.out.println("\nVergleich mittels Prozedur");
    if(IsEqual(a,b))
        System.out.println("IsEqual(a,b)=TRUE");
    else
        System.out.println("IsEqual(a,b)=FALSE");
```

Vergleich mittels Prozedur

IsEqual(a,b)=FALSE

```
    // Wir weisen b a zu und vergleichen erneut
    System.out.println("\nArray a wird b zugewiesen");
    b=a;
    System.out.println("Vergleich mittels equals");
    if(a.equals(b))
        System.out.println("a.equals(b)=TRUE");
    else
        System.out.println("a.equals(b)=FALSE");
```

Array a wird b zugewiesen

Vergleich mittels equals

a.equals(b)=TRUE

```
    System.out.println("\na[0][0][0] 30 zuweisen");
    a[0][0][0]=30;
    // Was ist jetzt im Array b enthalten????
    System.out.println("a[0][0][0]="+a[0][0][0]+
                       ", b[0][0][0]="+b[0][0][0]);
    System.out.println("Vergleich mittels equals");
    if(a.equals(b))
        System.out.println("a.equals(b)=TRUE");
    else
         System.out.println("a.equals(b)=FALSE");
```

Variable a[0][0][0] wird 30 zugewiesen

a[0][0][0]=30, b[0][0][0]=30

// Wieso ist nun auch b veraendert???

Vergleich mittels equals

a.equals(b)=TRUE *// Wir haben doch nur a veraendert????*

```
    System.out.println("\nVergleich mittels Prozedur");
    if(IsEqual(a,b))
        System.out.println("IsEqual(a,b)=TRUE");
    else
        System.out.println("IsEqual(a,b)=FALSE");
    // Zuweisung wird spaeter naeher untersucht!
    System.out.println("\n\nProgrammende");
}

}
```

Vergleich mittels Prozedur

IsEqual(a,b)=TRUE

Beispiel 3: Arrays kopieren und clonen

```
public class Bsp3
{

static public void PrintArray(int [][][] array)
// Hilfsprozedur zum Ausgeben. Hier keine Fehlerprüfung
// enthalten(laenge=3, nicht null, ...)
{
    int i,j,k;
    for(i=0;i<2;i++)
        for(j=0;j<2;j++)
            for(k=0;k<2;k++)
                System.out.println("["+i+"]["+j+"]["+k+
                    "]: "+array[i][j][k]);
}

static public void main(String args[])
{
    System.out.println("Programmstart\n\n");
    int count=0;
    int [][][] threeDimArray=new int[2][2][2];
    int [][][] anotherThreeDimArray;
    System.out.println("Array angelegt");
    int i,j,k;
```

```
    // Initialisieren mit eindeutigen Werten
    for(i=0;i<2;i++)
        for(j=0;j<2;j++)
            for(k=0;k<2;k++)
                threeDimArray[i][j][k]=count++;
    // Ausgeben
    PrintArray(threeDimArray);
    System.out.println("Länge: "+threeDimArray.length);
```

Array angelegt

[0][0][0]: 0

[0][0][1]: 1

[0][1][0]: 2

[0][1][1]: 3

[1][0][0]: 4

[1][0][1]: 5

[1][1][0]: 6

[1][1][1]: 7

Länge des Feldes: 2

```
    System.out.println("Kopieren in zweites Array:");
    // Kopieren per einfacher Zuweisung (Siehe Bsp. 2!)
    anotherThreeDimArray=threeDimArray;
    System.out.println("Neues Array veraendern");
    anotherThreeDimArray[1][1][1]=30;
    System.out.println("Original ausgeben:");
    PrintArray(threeDimArray);
    System.out.println("Wieso ist hier Original
veraendert worden?????????");
    System.out.println("Wiederherstellen");
    threeDimArray[1][1][1]=7;
```

Kopieren in zweites Array: *//anotherThreeDimArray=threeDimArray;*

Neues Array veraendern

Original ausgeben:

[0][0][0]: 0

[0][0][1]: 1

[0][1][0]: 2

[0][1][1]: 3

[1][0][0]: 4

[1][0][1]: 5

[1][1][0]: 6

[1][1][1]: 30

Wieso ist hier Original veraendert worden?????????

Wiederherstellen

```
    System.out.println("Neu kopieren, diesmal anders!");
    anotherThreeDimArray=new int[2][][];
    for(i=0;i<2;i++)
        anotherThreeDimArray[i]=threeDimArray[i];
    anotherThreeDimArray[1][1][1]=30;
    System.out.println("Neues Array veraendern");
    anotherThreeDimArray[1][1][1]=30;
    System.out.println("Original ausgeben:");
    PrintArray(threeDimArray);
    System.out.println("Wieso ist hier Original
veraendert worden????????");
    System.out.println("Wiederherstellen");
    threeDimArray[1][1][1]=7;
```

Neu kopieren, diesmal anders!

// for(i=0;i<2;i++)

// anotherThreeDimArray[i]=threeDimArray[i];

Neues Array veraendern

Original ausgeben:

[0][0][0]: 0

[0][0][1]: 1

[0][1][0]: 2

[0][1][1]: 3

[1][0][0]: 4

[1][0][1]: 5

[1][1][0]: 6

[1][1][1]: 30

Wieso ist hier Original veraendert worden????????

Wiederherstellen

```
    System.out.println("Neu kopieren, wieder anders!");
    anotherThreeDimArray=new int[2][2][];
    for(i=0;i<2;i++)
        for(j=0;j<2;j++)
          anotherThreeDimArray[i][j]=threeDimArray[i][j];
    System.out.println("Neues Array veraendern");
    anotherThreeDimArray[1][1][1]=30;
    System.out.println("Original ausgeben:");
    PrintArray(threeDimArray);
    System.out.println("Noch immer falsch?????????");
    System.out.println("Wiederherstellen");
    threeDimArray[1][1][1]=7;
```

Neu kopieren, wieder anders!

// for(i=0;i<2;i++)

// for(j=0;j<2;j++)

// anotherThreeDimArray[i][j]=threeDimArray[i][j];

Neues Array veraendern

Original ausgeben:

[0][0][0]: 0

[0][0][1]: 1

[0][1][0]: 2

[0][1][1]: 3

[1][0][0]: 4

[1][0][1]: 5

[1][1][0]: 6

[1][1][1]: 30

Noch immer falsch?????????

Wiederherstellen

```
    System.out.println("Neu kopieren, 4. Mal anders!");
    anotherThreeDimArray=new int[2][2][2];
    for(i=0;i<2;i++)
        for(j=0;j<2;j++)
            for(k=0;k<2;k++)
                anotherThreeDimArray[i][j][k]=
                      threeDimArray[i][j][k];
    System.out.println("Neues Array veraendern");
    anotherThreeDimArray[1][1][1]=30;
    System.out.println("Original ausgeben:");
    PrintArray(threeDimArray);
System.out.println("Endlich! Original bleibt erhalten.");
    System.out.println("Kopie ausgeben:");
    PrintArray(anotherThreeDimArray);
```

Neu kopieren, ein viertes Mal anders!

Neues Array veraendern

Original ausgeben:

[0][0][0]: 0

[0][0][1]: 1

……..

[1][1][0]: 6

[1][1][1]: 7

Endlich! Original bleibt erhalten.

Kopie ausgeben:

[0][0][0]: 0

[0][0][1]: 1

……..

[1][1][0]: 6

[1][1][1]: 30

```
System.out.println("\nNochmal, diesmal mit \"clone\"!");
    for(i=0;i<2;i++)
        for(j=0;j<2;j++)
            for(k=0;k<2;k++)
                threeDimArray[i][j][k]=count++;
    PrintArray(threeDimArray);
    System.out.println("Kopieren in zweites Array:");
   anotherThreeDimArray=(int[][][])threeDimArray.clone();
    System.out.println("Neues Array veraendern");
    anotherThreeDimArray[1][1][1]=30;
    System.out.println("Original ausgeben:");
    PrintArray(threeDimArray);
    System.out.println("Wieso ist Orig. veraendert????");
    System.out.println("Wiederherstellen");
    threeDimArray[1][1][1]=15;
```

Dasselbe nochmal, diesmal aber mit "clone"!

[0][0][0]: 8

……..

[1][1][0]: 14

[1][1][1]: 15

Kopieren in zweites Array:

// anotherThreeDimArray=(int[][][])threeDimArray.clone();

Neues Array veraendern

Original ausgeben:

[0][0][0]: 8

……..

[1][1][0]: 14

[1][1][1]: 30

Wieso ist hier Original veraendert worden?????????

Wiederherstellen

```
    System.out.println("Neu kopieren, diesmal anders!");
    anotherThreeDimArray=new int[2][][];
    for(i=0;i<2;i++)
        anotherThreeDimArray[i]=
                    (int[][])threeDimArray[i].clone();
    anotherThreeDimArray[1][1][1]=30;
    System.out.println("Neues Array veraendern");
    anotherThreeDimArray[1][1][1]=30;
    System.out.println("Original ausgeben:");
    PrintArray(threeDimArray);
    System.out.println("Wieso ist hier Original
veraendert worden?????????");
    System.out.println("Wiederherstellen");
    threeDimArray[1][1][1]=15;
```

Neu kopieren, diesmal anders!

// for(i=0;i<2;i++)

// anotherThreeDimArray[i]=(int[][])threeDimArray[i].clone()

Neues Array veraendern

Original ausgeben:

[0][0][0]: 8

[0][0][1]: 9

[0][1][0]: 10

[0][1][1]: 11

[1][0][0]: 12

[1][0][1]: 13

[1][1][0]: 14

[1][1][1]: 30

Wieso ist hier Original veraendert worden?????????

Wiederherstellen

```
    System.out.println("Neu kopieren, wieder anders!");
    anotherThreeDimArray=new int[2][2][];
    for(i=0;i<2;i++)
        for(j=0;j<2;j++)
            anotherThreeDimArray[i][j]=
                    (int[])threeDimArray[i][j].clone();
    System.out.println("Neues Array veraendern");
    anotherThreeDimArray[1][1][1]=30;
    System.out.println("Original ausgeben:");
    PrintArray(threeDimArray);
    System.out.println("Endlich! Original bleibt
erhalten.");
    System.out.println("Kopie ausgeben:");
    PrintArray(anotherThreeDimArray);
    System.out.println("\n\nProgrammende");
}
}
```

Neu kopieren, wieder anders!

// for(i=0;i<2;i++)

// for(j=0;j<2;j++)

// anotherThreeDimArray[i][j]=(int[])threeDimArray[i][j].clone();

Neues Array veraendern

Original ausgeben:

[0][0][0]: 8

[0][0][1]: 9

[0][1][0]: 10

[0][1][1]: 11

[1][0][0]: 12

[1][0][1]: 13

[1][1][0]: 14

[1][1][1]: 15

Endlich! Original bleibt erhalten.

Kopie ausgeben:

[0][0][0]: 8

[0][0][1]: 9

[0][1][0]: 10

[0][1][1]: 11

[1][0][0]: 12

[1][0][1]: 13

[1][1][0]: 14

[1][1][1]: 30

Programmende

Application terminated

BEACHTE:

- Unterschied zwischen

  "threeDimArray.length" und "threeDimArray.clone()":

  length ist eine Variable, clone eine Methode (Funktion, nicht Prozedur)!

- Folgende zwei Codestücke sind äquivalent:

1. anotherThreeDimArray=new int[2][2][];

   for(i...)

   for(j...)

   anotherThreeDimArray[i][j]=(int[])threeDimArray[i][j]**.clone()**;

2. anotherThreeDimArray=new int[2][2][**2**];

   for(i...)

   for(j...)

   for(k...)

   anotherThreeDimArray[i][j]**[k]**=threeDimArray[i][j]**[k]**;

Unterschied Arrays Û Vektoren

ACHTUNG: Dies ist keine allgemeine Definition von "Vektor", sondern beschreibt lediglich die in java.util.Vector definierte Klasse!

1. Arrays haben eine feste Anzahl von Elementen: Bei **new** festgelegt
   ⇒ Vektoren können beliebig viele Elemente enthalten (bei Bedarf werden sie automatisch dynamisch vergrößert)
2. Arrays können int, boolean, char, … und Referenztypen enthalten
   ⇒ Vektoren können **nur** Referenztypen enthalten
3. Bei Arrays haben alle Elemente den selben Typ
   ⇒ Vektoren können beliebige Typen gleichzeitig enthalten (Siehe dazu objektorientierte Programmierung)

Beispiel 4: Die Klasse Vector

```
import java.util.Vector;

public class Bsp4
{

static public void main(String args[])
{
    System.out.println("Programmstart\n\n");
    Vector vec=new Vector();
    // 10 Elemente einfuegen
    for(int i=0;i<10;i++)
        vec.addElement(new Integer(i+1));
    System.out.println("Capacity="+vec.capacity());
    if(vec.contains(new Integer(2)))
        System.out.println("Enthaelt Integer(2)");
    else
        System.out.println("Enthaelt NICHT Integer(2)");
    System.out.println("1. El.: "+vec.firstElement());
    System.out.println("Letzt. El.: "+vec.lastElement());
    System.out.println("El. an St. 3:"+vec.elementAt(3));
```

Capacity=10

Enthaelt Integer(2)

Erstes Element: 1

Letztes Element: 10

Element an Stelle 3: 4

```
    System.out.println("Index des Wertes 4, beginnend
           bei Index 4: "+vec.indexOf(new Integer(4),4));
    System.out.println("Index des Wertes 4: "+
                         vec.indexOf(new Integer(4)));
    vec.insertElementAt(new Integer(8),0);
    System.out.println("Index des Wertes 4 nach einfügen
von 8 an Index 0: "+vec.indexOf(new Integer(4)));
    System.out.println("Letzter Index des Wertes 8: "+
                       vec.lastIndexOf(new Integer(8)));
    System.out.println("Letzter Index des Wertes 8,
beginnend bei Index 4: "+
                       vec.lastIndexOf(new Integer(8),4));
```

Index des Wertes 4, beginnend bei Index 4: -1
Index des Wertes 4: 3
Index des Wertes 4 nach einfügen von 8 an Index 0: 4
Letzter Index des Wertes 8: 8
Letzter Index des Wertes 8, beginnend bei Index 4: 0

```
    System.out.println("Groesse des Vekt.: "+vec.size());
    vec.removeElement(new Integer(4));
    System.out.println("Index des Wertes 4 nach
removeElement: "+vec.indexOf(new Integer(4)));
    System.out.println("Groesse des Vekt.: "+vec.size());
    System.out.println("toString: "+vec.toString());
```

Groesse des Vektors (size): 11
Index des Wertes 4 nach removeElement: -1
Groesse des Vektors (size): 10
toString: [8, 1, 2, 3, 5, 6, 7, 8, 9, 10]

```
    vec.removeElementAt(3);
    System.out.println("rem.El.At(3): "+vec.toString());
    vec.setElementAt(new Integer(15),3);
    System.out.println("nach setElementAt(
                new Integer(15),3): "+vec.toString());
    vec.setSize(10);
/* println("nach setSize(10): "+vec.toString());
    Fehler, da an Index 9 der Wert null eingefuegt wird,
    der nicht in einen String umgewandelt werden kann! */
    vec.setSize(5);
    System.out.println("setSize(5): "+vec.toString());
```

nach removeElementAt(3): [8, 1, 2, 5, 6, 7, 8, 9, 10]

nach setElementAt(new Integer(15),3): [8, 1, 2, 15, 6, 7, 8, 9, 10]

nach setSize(5): [8, 1, 2, 15, 6]

```
    if(vec.isEmpty())
        System.out.println("Vektor ist leer");
    else System.out.println("Vektor ist nicht leer");
    System.out.println("removeAllElements()");
    vec.removeAllElements();
    if(vec.isEmpty())
        System.out.println("Vektor ist leer");
    else
        System.out.println("Vektor ist nicht leer");
}
}
```

Vektor ist nicht leer

removeAllElements()

Vektor ist leer

Enumerationen

Enumerationen (=Iteratoren) sind ein wichtiges und komfortables Mittel um Datenstrukturen zu durchwandern. Dennoch müssen sie mit Vorsicht verwendet werden (insbesondere bei Multithreading)!

Was passiert, wenn während des Durchlaufs

- neue Elemente eingefügt werden?
- Elemente gelöscht werden?
- die gesamte Datenstruktur gelöscht wird?

Iteratoren, die diese Operationen „unbeschadet“ überstehen, werden allgemein als „robuste Iteratoren“ bezeichnet (und sind sehr selten!).

Achtung: Lesen Sie bei jedem Iterator immer genau nach, welche Reaktion auf obige Aktionen erfolgt!

Ist nichts erwähnt, **müssen** Sie davon ausgehen, daß der Iterator **nicht** robust ist.

Beispiel 5: Enumerationen

```
import java.util.Vector;
import java.util.Enumeration;

public class Bsp5
{

static public void main(String args[])
{
    System.out.println("Programmstart\n\n");
    Vector vec=new Vector();
    vec.addElement(new Integer(1));
    vec.addElement(new Integer(2));
    vec.addElement(new Integer(3));
    vec.addElement(new Integer(4));
    vec.addElement(new Integer(5));
    Enumeration enum=vec.elements();
    if(enum.hasMoreElements() )
        System.out.println("Noch Elemente vorhanden");
    else
        System.out.println("Keine Elemente mehr
vorhanden");
```

Noch Elemente vorhanden

```
    // Wir stehen vor 1
    System.out.println("Naechstes Element sollte 1 sein:"
                        +(Integer)enum.nextElement());
    System.out.println("Naechstes Element sollte 2 sein:"
                        +(Integer)enum.nextElement());
    vec.removeElementAt(0); // 1 loeschen
    System.out.println("Element an Stelle 0 geloescht,
                naechstes Element sollte nun 3 sein: "
                +(Integer)enum.nextElement());
```

Naechstes Element: sollte 1 sein: 1

Naechstes Element: sollte 2 sein: 2

// Element „3“ fehlt, da es durch removeElementAt(0) an

// der Marke für das nächste Element „vorbeigerutscht“ ist.

Element an Stelle 0 geloescht, naechstes Element sollte nun 3 sein: 4

```
    vec.insertElementAt(new Integer(1),0);
    System.out.println("Element 1 an Stelle 0 eingefuegt,
                naechstes Element nach 4 sollte 5 sein:"
                +(Integer)enum.nextElement());
    System.out.println("Vollstaendiger Durchlauf:");
    for (enum=vec.elements();enum.hasMoreElements();)
        System.out.print(enum.nextElement()+", ");
    System.out.println();
```

Element 1 an Stelle 0 eingefuegt, Element nach 4 sollte 5 sein: 4

// Doppelt, da Element „4“ durch Einfügen um eine Stelle nach hinten verschoben wurde!

Vollstaendiger Durchlauf:

1, 2, 3, 4, 5,

```
    enum=vec.elements();
     enum.nextElement();
     enum.nextElement();
     Integer test=(Integer)enum.nextElement();
     vec=null;
    System.out.println("Vector geloescht (vec=null),
         existiert Element noch (sollte 3 sein): "+test);
    if(enum.hasMoreElements() )
        System.out.println("Noch Elemente vorhanden");
    else
        System.out.println("Keine Elemente mehr
vorhanden");
    System.out.println("\n\nProgrammende");
}
}
```

Vector geloescht (vec=null), existiert Element noch (sollte 3 sein): 3

Noch Elemente vorhanden

// Vektor existiert noch immer, da in der Enumeration eine Referenz auf

// ihn enthalten ist (daher nicht vom GC gelöscht)!