Abstrakte Klassen

Abstrakte Klassen sind durch das Schlüsselwort $abstract$ ausgewiesen.

Es gibt keine Instanzen einer abstrakten Klasse.

Sie dürfen abstrakte Methoden enthalten, auch mit $abstract$ gekennzeichnet.

Abstrakte Methoden haben keinen Rumpf.

Es werden also nur für alle Unterklassen die Bezeichungen festgelegt.

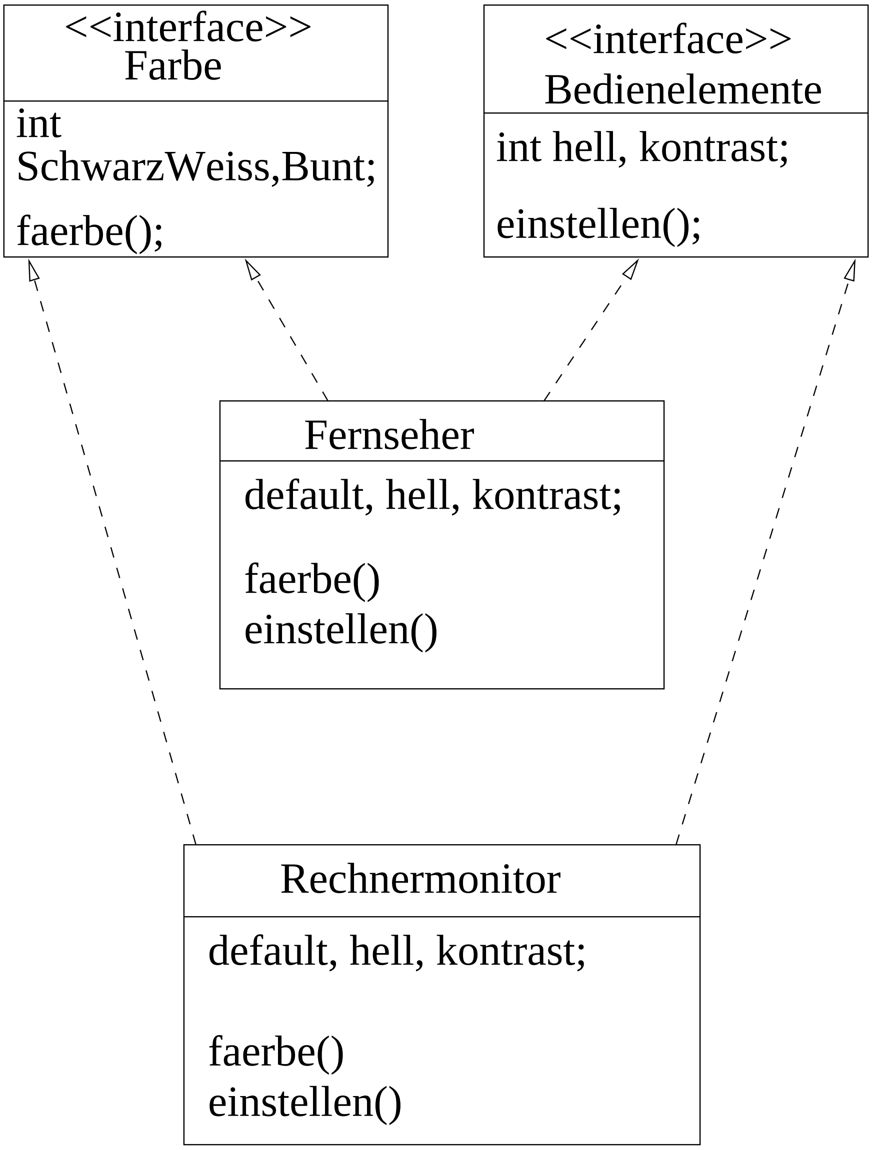
Abstrakte Klassen dienen eigentlich der Kommunikation unter ProgrammiererInnen.

Schnittstellen

Eine Klasse, die mit dem Schlüsselwort *interface* anstelle von *class* ausgezeichnet ist.

- Eine Schnittstelle ist eine Klasse, die ausschließlich abstrakte Methoden hat.
- Eine Schnittstelle kann keine Objekte haben.
- Eine Schnittstelle kann von anderen Schnittstellen abstrakte Methoden erben. Das Schlüsselwort ist wie bei Klassen *extends*: Vererbung im Sinne von **A ist ein B**.
- Eine Klasse kann Unterklasse von mehreren Schnittstellen sein. Dies wird durch das Schlüsselwort *implements* angegeben: **A implementiert B**. Dann implementiert sie die abstrakten Methoden all dieser Schnittstellen.

Beispiel


<<interface>>
Farbe
int
SchwarzWeiss,Bunt;
faerbe();
<<interface>>
Bedienelemente
int hell, kontrast;
einstellen();
Fernseher
default, hell, kontrast;
faerbe()
einstellen()
Rechnermonitor
default, hell, kontrast;
faerbe()
einstellen()

... in JAVA

```
interface Farbe {
    int SchwarzWeiss=0, Bunt=1;
    public void faerbe ();
}

interface Bedienelemente {
    int Hell=3, Kontrast=3;
    public void einstellen ();

}
```

Verwendung der Schnittstelle

```
class Fernseher implements Farbe, Bedienelemente {
    int Default,Hell,Kontrast;
    public void faerbe () {
        Default=Bunt;
    }

    public void einstellen () {
        Hell=1;
        Kontrast=1;

    }
    public void drucke () {
        System.out.println ("TV-Farbe: "+Default);
        System.out.println ("TV-Helligkeit: "+Hell);
        System.out.println ("TV-Kontrast: "+Kontrast);
    }
}

class Rechnermonitor implements Farbe, Bedienelemente {
    int Default,Hell,Kontrast;

    public void faerbe () {
        Default=SchwarzWeiss;
    }
    public void einstellen () {
        Hell=2;
```

```
        Kontrast=2;
    }
    public void drucke () {
        System.out.println ("Monitor-Farbe: "+Default);
        System.out.println ("Monitor-Helligkeit: "+Hell);
        System.out.println ("Monitor-Kontrast: "+Kontrast);
    }
}

class InterfaceTest {

    public static void main (String[] argv) {
        Fernseher tv=new Fernseher ();
        tv.faerbe ();
        tv.einstellen ();
        tv.drucke ();
        Rechnermonitor monitor=new Rechnermonitor ();
        monitor.faerbe ();
        monitor.einstellen ();
        monitor.drucke ();
    }
}
```

Ergebnis

`java InterfaceTest`

TV-Farbe: 1

TV-Helligkeit: 1

TV-Kontrast: 1

Monitor-Farbe: 0

Monitor-Helligkeit: 2

Monitor-Kontrast: 2

Felder

Mehrere Daten desselben Datentyps können zu einem Feld (engl.: array) zusammengefaßt werden. Die Elemente des Feldes (components) sind der Reihe nach nummeriert, beginnend bei 0. Ein Feld wird deklariert durch den Datentyp seiner Elemente und eckige Klammern.

```
int[] feldInt;
char[] feldChar;
boolean[] feldBoolean;
```

Die Länge eines Feldes wird bei der Konstruktion eines neuen Feldes durch eine Zahl in den eckigen Klammern angegeben. Die Länge eines Feldes ändert sich nie.

```
feldInt = new int[8];
```

Zugriff auf ein Element des Feldes

Ein Feldelement wird durch die Laufvariable (Zeiger) bezeichnet. Es referenziert auf einen Wert. Dieser Wert kann einer Variablen zugewiesen werden. Dem Feldelement kann auch ein Wert zugewiesen werden.

```
value = feldDouble[i];
feldInt[2] = 6;
feldBoolean[0] = regnet | !regnet;
```

```
import AlgoTools.IO;
public class Feld {
    public static void main (String argv[]) {
        double[] a;                // eindimensionales double−Feld
        int i, j;                                   // Laufvariablen
        a = new double[8];                 // Platz fuer 8 Double
        for (i=0; i < a.length; i++)   // durchlaufe das Feld
        IO.println (a[i]);             // drucke jedes Feldelement
        double[][] mat = new double[4][3];        // 4x3 Matrix
        for (i=0; i<mat.length; i++) {            // jede Zeile
            for (j=0; j < mat[i].length; j++)   // die Spalten
            IO.print (mat[i][j], 8, 2); // drucke Matrixelement
            IO.println ();                  // gehe auf neue Zeile
        }
        int[][] diagonal = { { 1 } , { 2,3 } , { 4,5,6
} } ;                               // untere linke Dreiecksmatrix
    }
}
```

Ergebnis

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.00 0.00 0.00

0.00 0.00 0.00

0.00 0.00 0.00

0.00 0.00 0.00

```
class FeldTestO {
    public static void main (String[] argv) {
        int i;
        Object[] oFeld = new Object[5];
        oFeld[1] = "schreibtisch";
        for (i=0; 5 > i; i++)                //so sieht man die
        System.out.println (""+oFeld[i]);    //Elemente
    }
}
```

null

schreibtisch

null

null

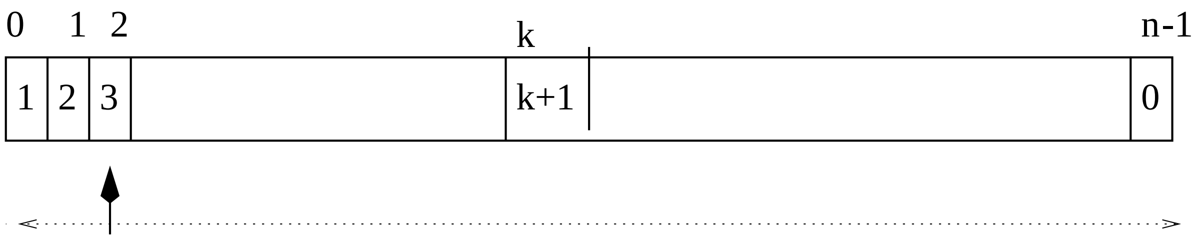
null

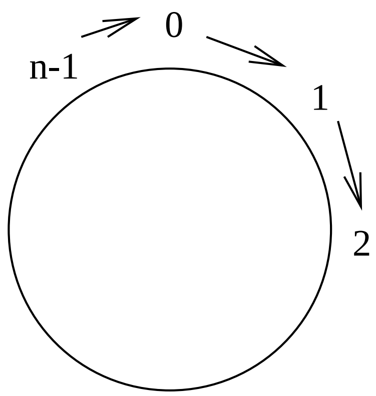
Ring als Feld

Beispiel

n Kinder stehen im Kreis, das k-te Kind scheidet aus

solange, bis nur noch 1 Kind übrig ist.





index = 3

0,	1,	2,	3,	0,	1,	2,	3
1,	2,	3,	1,	2,	3,	1,	2
1,	2,	1,	2,	1,	2,	1,	2

```
import AlgoTools.IO;
public class ArrayAbzaehlreim {
    public static void main (String argv[]) {
        int[] next;                                   // Feld mit Indizes
        int i, index, n, k;                           // Laufvariablen
        n = IO.readInt ("Wie viele Kinder ? ");
        k = IO.readInt ("Wie viele Silben ? ");
        next = new int[n];                 // Platz fuer Index-Feld
        for (i = 0; i < n; i++)            // initiale Aufstellung
        next[i] = (i+1) % n;
        index = n-1;
        while (next[index] != index) {          //abschlagen, bis
                            // jemand sein eigener Nachfolger ist
            for (i=1; i< k; i++)                // gehe k-1 mal
            index = next[index]; // zum jeweiligen Nachfolger
            IO.print ("Ausgeschieden: ");
            IO.println (next[index],5);
            next[index] = next[next[index]];         // Index neu
        }
        IO.println ("Es bleibt uebrig: "+ index);
    }
}
```

Ergebnis

Wie viele Kinder ? 10

Wie viele Silben ? 8

Ausgeschieden: 7

Ausgeschieden: 5

Ausgeschieden: 4

Ausgeschieden: 6

Ausgeschieden: 9

Ausgeschieden: 2

Ausgeschieden: 1

Ausgeschieden: 8

Ausgeschieden: 3

Es bleibt uebrig: 0