# Fragezeichenoperator

Die Zuweisung eines Wertes abhängig von einer Bedingung, kann mit einer if-else-Struktur erreicht werden:

# Fragezeichenoperator

Die Zuweisung eines Wertes abhängig von einer Bedingung, kann mit einer if-else-Struktur erreicht werden:

Bsp. sign-Funktion: if $(x < 0)$

```
        y=-1;
else
        y=+1;
```

# Fragezeichenoperator

Die Zuweisung eines Wertes abhängig von einer Bedingung, kann mit einer if-else-Struktur erreicht werden:

Bsp. sign-Funktion:
```
if (x < 0)
        y=-1;
else
        y=+1;
```

aber kürzer mit den ?-Operator

$$y = x < 0\,?\,-1\,:\,+1;$$

# Fragezeichenoperator

Die Zuweisung eines Wertes abhängig von einer Bedingung, kann mit einer if-else-Struktur erreicht werden:

Bsp. sign-Funktion:

```
if (x < 0)
        y=-1;
else
        y=+1;
```

aber kürzer mit den ?-Operator

```
y = x < 0 ? -1 : +1;
```

Allgemein:

boolescher Ausdruck ? Ausdruck für true : Ausdruck für false ;

# break und continue

werden zur Steuerung von for und while-Schleifen benutzt.

(Die Benutzung von break in der switch-Anweisung ist schon bekannt.)

# break und continue

werden zur Steuerung von for und while-Schleifen benutzt.

Schleifen haben den prinzipiellen Aufbau:

Schleifenanfang

Schleifenende

# break und continue

werden zur Steuerung von for und while-Schleifen benutzt.

Schleifen haben den prinzipiellen Aufbau:

Schleifenanfang

nach break wird die Schleife verlassen

Schleifenende

# break und continue

werden zur Steuerung von for und while-Schleifen benutzt.

Beispiel: Suche des ersten negativen Feldelementes

```
double x[] = new double[8];
int index;
...
for (int i=0; i < x.length; i++){
        if (x[i] < 0) {
            index=i;
            break; // abbruch der schleife
        }
}
```

# break und continue

werden zur Steuerung von for und while-Schleifen benutzt.

Schleifenanfang

nach continue wird ans Schleifenende gesprungen

Schleifenende

# break und continue

werden zur Steuerung von for und while-Schleifen benutzt.

Beispiel: Bestimmung der Anzahl negativer Feldelemente und die Ausgabe der anderen Feldelemente

```
double x[] = new double[8];
int anzahl=0;
...
for (int i=0; i < x.length; i++){
        if (x[i]<0) {
            anzahl++;
            continue; // sprung ans ende
        }
         System.out.println("x(" + i + ")=" + x[i]);
}
```

# break und continue

Zur Steuerung ineinandergeschachtelter Schleifen werden Marken verwendet. Wird sehr selten benötigt!

# break und continue

Zur Steuerung ineinandergeschachtelter Schleifen werden Marken verwendet. Wird sehr selten benötigt!

Eine Marke ist <bezeichner>:

# break und continue

Ziel: Sprung aus äußerer Schleife

```
for (int i=0; i < 4711; i++){
    ...
    for (int j=0; j < 0815; j++){
        ...
        break ; // abbruch der inneren schleife
        ...
    }
}
```

# break und continue

Ziel: Sprung aus äußerer Schleife

```
schleife_1:    for (int i=0; i < 4711; i++){
                   ...
                   for (int j=0; j < 0815; j++){
                       ...
                       break schleife_1;  // abbruch der aeusseren schleife
                       ...
                   }
}
```

# break und continue

Ziel: Sprung aus äußerer Schleife

```
schleife_1:    for (int i=0; i < 4711; i++){
                   ...
                   for (int j=0; j < 0815; j++){
                       ...
                   break schleife_1;  // abbruch der aeusseren schleife
                       ...
                   }
}
```

Analog ist die Anwendung für continue.

# Ausnahmebehandlung

Sei double x[] = new double[3]; ein Feld mit x[0], x[1], x[2].

# Ausnahmebehandlung

Sei double x[] = new double[3]; ein Feld mit x[0], x[1], x[2].

Wird x[3] im Programm test benutzt, so wird es fehlerfrei übersetzt.

# Ausnahmebehandlung

Sei double x[] = new double[3]; ein Feld mit x[0], x[1], x[2].

Wird x[3] im Programm test benutzt, so wird es fehlerfrei übersetzt.

java test liefert aber:

# Ausnahmebehandlung

Sei double x[] = new double[3]; ein Feld mit x[0], x[1], x[2].

Wird x[3] im Programm test benutzt, so wird es fehlerfrei übersetzt.

java test liefert aber:

Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 3

# Ausnahmebehandlung

Sei double x[] = new double[3]; ein Feld mit x[0], x[1], x[2].

Wird x[3] im Programm test benutzt, so wird es fehlerfrei übersetzt.

java test liefert aber:

Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 3

siehe Java 2: java.lang.reflect.
Array.getDouble(Object array, int index) - Method Detail

## getDouble

```
public static double getDouble(Object array,
                               int index)
                        throws IllegalArgumentException,
                               ArrayIndexOutOfBoundsException
```

Returns the value of the indexed component in the specified array object, as a `double`.

**Parameters:**

`array` - the array
`index` - the index

**Returns:**

the value of the indexed component in the specified array

**Throws:**

`NullPointerException` - If the specified object is null
`IllegalArgumentException` - If the specified object is not an array, or if the indexed element cannot
type by an identity or widening conversion
`ArrayIndexOutOfBoundsException` - If the specified `index` argument is negative, or if it is greater tha
the specified array

**See Also:**

`get(java.lang.Object, int)`

Diese Methode "wirft" (throw) mehrere Ausnahmen (Exception).

Diese Methode "wirft" (throw) mehrere Ausnahmen (Exception).

Abfangen einer Ausnahme durch

```
try {
...
}
catch(<welcheException> e){
...
}
```

Beispiel:

```
double x[] = new double[3]; double y;
try {
        System.out.println(" vor y");
        y=x[3];
        System.out.println(" nach y");
}
catch(ArrayIndexOutOfBoundsException e){
       System.out.println("Fehlermitteilung: "+e);
}
System.out.println("nach try-catch");
```

Beispiel:

```
double x[] = new double[3]; double y;
try {
        System.out.println(" vor y");
        y=x[3];
        System.out.println(" nach y");
}
catch(ArrayIndexOutOfBoundsException e){
       System.out.println("Fehlermitteilung: "+e);
}
System.out.println("nach try-catch");
```

Programmausgabe ?

Beispiel:

```
double x[] = new double[3]; double y;
try {
        System.out.println(" vor y");
        y=x[3];
        System.out.println(" nach y");
}
catch(ArrayIndexOutOfBoundsException e){
       System.out.println("Fehlermitteilung: "+e);
}
System.out.println("nach try-catch");
```

Programmausgabe

```
vor y
```

Beispiel:

```
double x[] = new double[3]; double y;
try {
        System.out.println(" vor y");
        y=x[3];
        System.out.println(" nach y");
}
catch(ArrayIndexOutOfBoundsException e){
        System.out.println("Fehlermitteilung: "+e);
}
System.out.println("nach try-catch");
```

Programmausgabe

```
vor y
Fehlermitteilung: java.lang.ArrayIndexOutOfBoundsException: 3
```

Beispiel:

```
double x[] = new double[3]; double y;
try {
        System.out.println(" vor y");
        y=x[3];
        System.out.println(" nach y");
}
catch(ArrayIndexOutOfBoundsException e){
       System.out.println("Fehlermitteilung: "+e);
}
System.out.println("nach try-catch");
```

Programmausgabe

```
vor y
Fehlermitteilung: java.lang.ArrayIndexOutOfBoundsException: 3
nach try-catch
```