Tagesprogramm

Referenzen

String

Array

Referenztypen versus elementare Typen

elementare Typen:

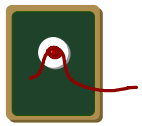
boolean, byte, short, char, int, long, float, double

```
int i = 1;
```

Referenztypen:

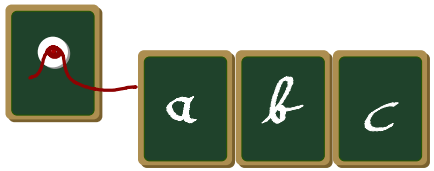
String, Scanner, System, ...

String[], int[], long[], ...



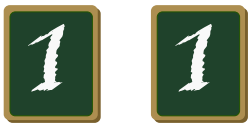
```
int[] ints = { 1, 2, 3 };
```


```
String abc = "abc";
```



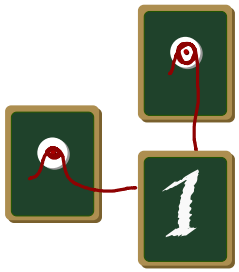
Auswirkung von Zuweisungen

```
int a = 1;
int b = a;
```

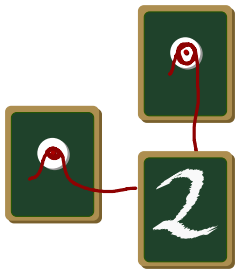



```
a = 2;
System.out.println(a + "," + b); → 2,1
```


```
int[] as = { 1 };
int[] bs = as;
```



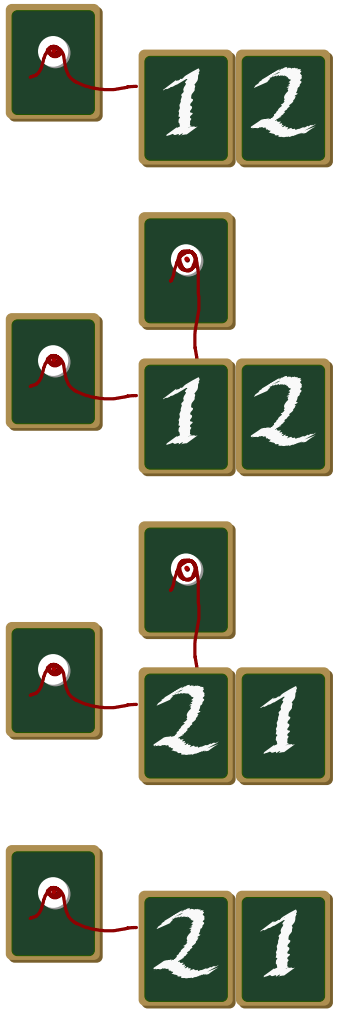

```
as[0] = 2;
System.out.println(as[0] + "," + bs[0]); → 2,2
```



Änderung eines Arrays als Parameter

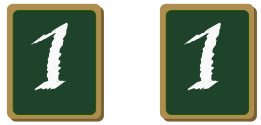
```
private static void swap(int[] pair) {
    int help = pair[0];
    pair[0] = pair[1];
    pair[1] = help;
}
```

```
    ...
    int[] is = { 1, 2 };
    swap(is);
    System.out.println(is[0] + "," + is[1]);
```



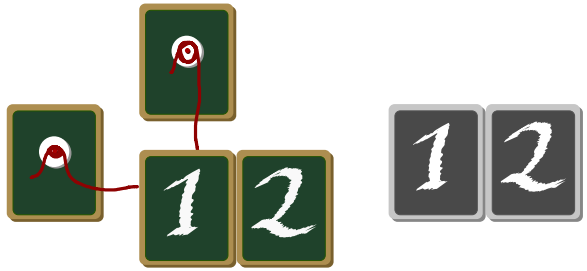
Unterscheidung referenzierter Objekte

```
int i = 1;
int j = 1;
System.out.println(i == j); // true
```



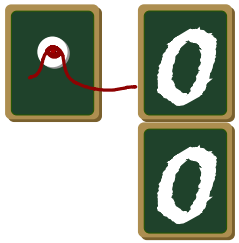

```
int[] is = { 1, 2 };
int[] js = { 1, 2 };
System.out.println(is == js); // false
```


```
is = js;
System.out.println(is == js); // true
```

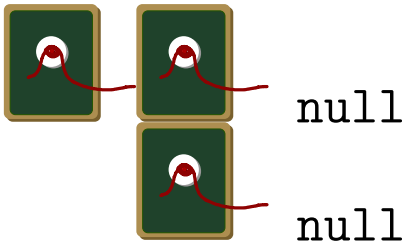


Null

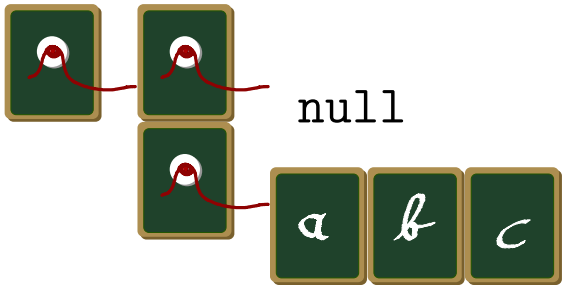
```
int[] ints = new int[2];
System.out.println(ints[1]); → 0
```



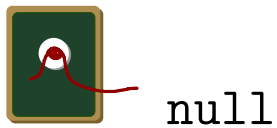

```
String[] strs = new String[2];
System.out.println(strs[1]); → null
```




```
strs[1] = "abc";
System.out.println(strs[1]); → abc
```




```
String str = null;
```



Aufgabe: Null in lokaler Variable

Warum müssen wir einer lokalen Variable eines Referenztyps explizit `null` zuweisen, obwohl Arrayeinträge automatisch mit `null` initialisiert werden?

A Weil alle lokalen Variablen explizit initialisiert werden müssen.

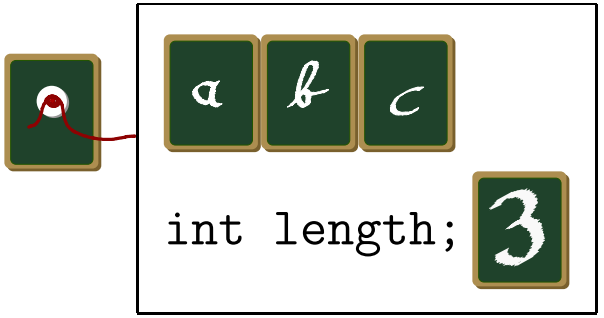
B Weil `null` als Anfangswert lokaler Variablen oft nicht passt.

C Weil der Umgang mit `null` fehleranfällig ist.

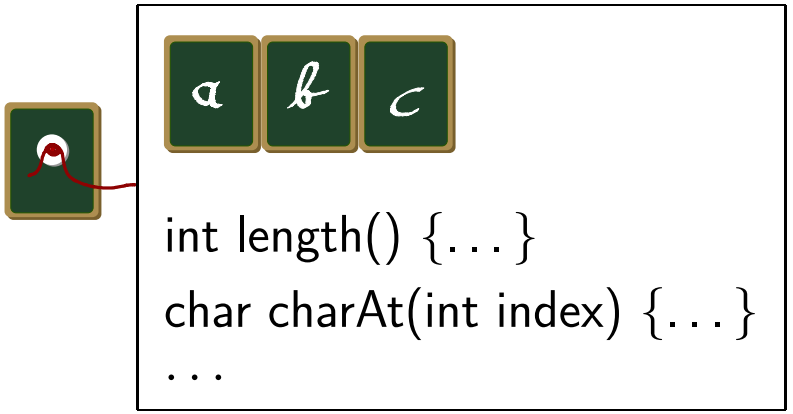
D Weil der Compiler damit überfordert wäre.

Typ bestimmt Verwendbarkeit

```
char[] cs = { 'a', 'b', 'c' };
```




```
String str = "abc";
```

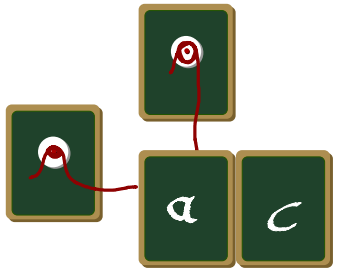


Umgang mit Strings

Zeichen in Strings nicht änderbar, daher Unterscheidung gleicher Strings nicht nötig


```
String s1 = "ac";
String s2 = "ac";
```

oder




```
String s3 = s1 + s2;
```


```
System.out.println(s1 == s2);       // ??? - nicht verwenden!
System.out.println(s1.equals(s2)); // true
System.out.println(s1 == s3);       // false
System.out.println(s1.equals(s3)); // false
```

Häufig verwendete String-Methoden

`s1.length()` → `int`

`s1.equals(s2)` → `boolean`

`s1.charAt(1)` → `char`

`s1.toCharArray()` → `char[]`

`s1.toUpperCase()` → `String`

`s1.toLowerCase()` → `String`

Aufgabe: Vergleiche

Welche der folgenden Vergleiche sind richtig bzw. empfehlenswert?

A
```
int i, j;       ... if (i.equals(j)) { ...}
```

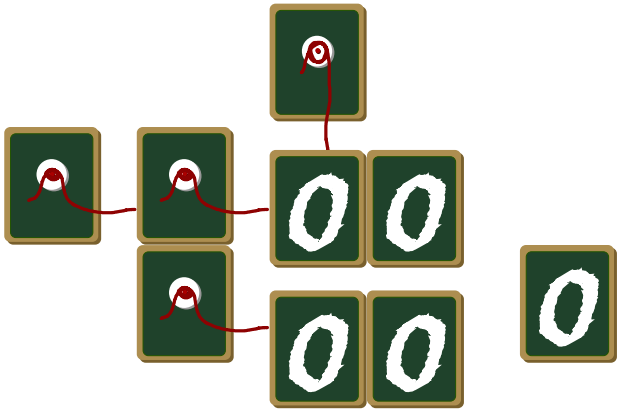
B
```
String s, t;    ... if (s.equals(t)) { ...}
```

C
```
double d, e;    ... if (d == e) { ...}
```

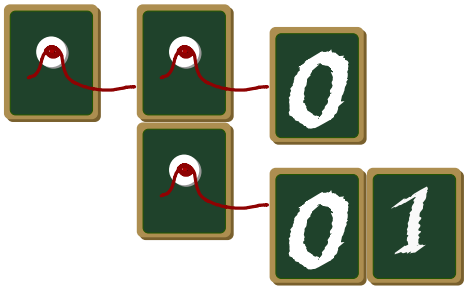
D
```
boolean a, b; ... if ((a == b) && (a ^ b)) { ...}
```

Mehrdimensionale Arrays

```
int[][] iss = new int[2][2];
int[] is = iss[0];
int i = iss[0][1];
```




```
int[][] jss = new int[2][];
jss[0] = new int[1];
jss[1] = new int[2];
jss[1][1] = 1;
```



Pascalsches Dreieck als Array

```
private static long[][] triangle(int lines) {
    long[][] triangle = new long[lines][];
    for (int n = 0; n < lines; n++) {
        triangle[n] = new long[n + 1];
        triangle[n][0] = 1;
        triangle[n][n] = 1;
        for (int k = 1; k < n; k++) {
            triangle[n][k] = triangle[n - 1][k - 1]
                           + triangle[n - 1][k];
        }
    }
    return triangle;
}
```

ForEach-Schleife

```
long[][] triangle = triangle(...);

for (int n = 0; n < triangle.length; n++) {
    for (int k = 0; k < triangle[n].length; k++) {
        System.out.print(triangle[n][k] + " ");
    }
    System.out.println();
}


for (long[] line : triangle) {
    for (long elem : line) {
        System.out.print(elem + " ");
    }
    System.out.println();
}
```

Aufgabe: Konstruktion eines Arrays

Suchen Sie in Gruppen zu 2 bis 3 Personen eine Antwort:

Wie kann man das Array, das an folgende Methode übergeben wird, so konstruieren, dass es nach Ausführung das Pascalsche Dreieck enthält?

```
private static void triangle(long[][] triangle) {
    for (int n = 0; n < triangle.length; n++) {
        triangle[n][0] = 1;
        triangle[n][n] = 1;
        for (int k = 1; k < n; k++) {
            triangle[n][k] = triangle[n - 1][k - 1]
                           + triangle[n - 1][k];
        }
    }
}
```