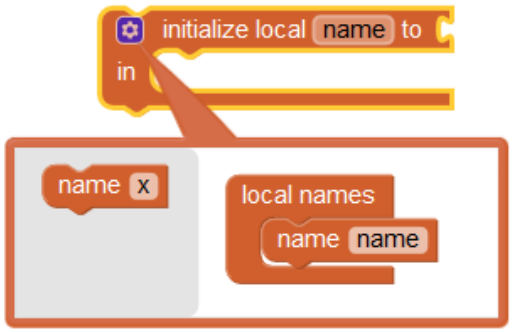
Vergleich Bausteine MIT App Inventor 2 – Programmiersprache Java

MIT App Inventor 2	Java

Variablen

Es wird zwischen globalen und lokalen Variablen entschieden.

Bei globalen Variablen braucht man für jede neue Variable einen eigenen Block, bei lokalen kann man mehrere gleichzeitig nutzen:



Variable können von den verschiedensten Typen sein. Hier drei Initialisierungen:

Für jede Variable gibt es eine get- und eine set-Methode.

Variablen werden entweder global (ganz zu Beginn) oder lokal (in einer Methode oder einer Schleife) deklariert.

Globale Variablen kennt das ganze Programm, lokale nur die Methode (oder Schleife), in der sie deklariert wurden.

Bei der Deklaration der Variable müssen wir deren Typ angeben.

Beispiele:

```
// Deklaration von Variablen

int a = 4; // ganze Zahlen

float b = 6.01; // Kommazahlen

char c = 'A'; // ein Zeichen

String text = "IMP 9"; // eine Zeichenkette

boolean stimmts = false; // ein Wahrheitswert
```

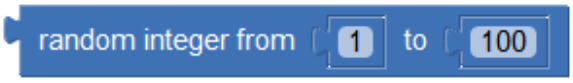
Im Beispiel wurden alle Variablen gleich initialisiert. (Das könnte auch an anderer Stelle passieren.)

```
// Deklaration
int zaehler;
// Initialisierung
zaehler = 0;
// Erhöhen des Wertes der Variable
zaehler = zaehler + 1;
// oder
zaehler += 1;
```

Der Wert der Variable kann genutzt oder verändert werden (s. Abb.).

MIT App Inventor 2	Java

Zufallszahlen




```
float z1 = random(100);
float z2 = random(1,10);
int z3 = round(random(1,6));
```

Mit `random(a)` werden Kommazahlen von 0 bis a erzeugt. Man kann auch den Bereich eingeben. Möchte man ganze Zahlen, muss man runden. (In den Materialien nutzen wir eine Methode `getZufallszahl(int min, int max)`.)

Verzweigungen

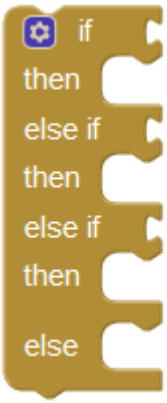
Es gibt einen Block, der beliebig erweitert werden kann:


```
if (bedingung_erfuellt) {
  // hier passiert dann etwas :)
}
```

Zur Erweiterung klickt man auf das Zeichen für Einstellungen und zieht die gewünschten Blockteile nach rechts:


```
if (bedingung_erfuellt) {
  // Anweisungen, wenn Bedingung erfüllt ist
} else {
  // Anweisungen, wenn sie nicht erfüllt ist
}
```

Man könnte hier Verzweigungsblöcke ineinander setzen, kann es aber auch so lösen:

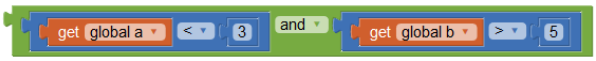



```
if (bedingung_erfuellt) {
  // Anweisungen, wenn Bedingung erfüllt ist
} else if (andere_bedingung_erfuellt) {
  // 1. Bed. ist nicht erfüllt, 2. Bed. ist erfüllt
} else {
  // keine der vorherigen Bedingungen ist erfüllt
}
```

Innerhalb der geschweiften Klammern können weitere Kontrollstrukturen stehen.

Es können auch weitere „else if“ eingefügt werden.

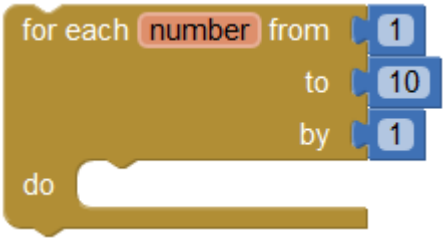
MIT App Inventor 2	Java
	Eine andere Art der Verzweigung: `switch (klasse) { case 6: ausgabe = "Sprachwahl"; break; case 8: ausgabe = "Profilwahl"; break; case 11: ausgabe = "Kurswahl"; break; default: ausgabe = "Es steht keine Wahl an."; }` Lässt man das „break“ weg, werden alle Zweige durchlaufen. Trifft kein Fall zu, werden die Anweisungen unter „default“ ausgeführt.

Bedingungen
Die logischen Verknüpfungen findet man unter Logic, die Vergleichsoperatoren bei Math.
Beispiele:
 
 
	Vergleichsoperatoren: >, <, >=, <=, ==
	<u>Achtung:</u>
	Ein einfaches „=“ bedeutet eine Zuweisung (wie oben bei den Variablen)!
	Möchte man aber überprüfen, ob eine Variable einen Wert hat, braucht man „==“.
	Logische Verknüpfungen:
	&& (und); || (oder); ! (nicht)
	(Bei *und* und *oder* würde auch jeweils ein Zeichen ausreichen. Nimmt man allerdings das Zeichen doppelt, wird die zweite Bedingung nur noch ausgewertet, wenn eine wahre Aussage noch möglich ist.)
	Beispiele:
	`if ( a < 3 && b > 5 ) { // tue etwas }`
	`if ( ! (a == b) ){ // tue etwas }`

MIT App Inventor 2	Java

Schleifen

Zählschleife:




```
for (int i=start; i<=ende; i=i+schritt){
  // tue etwas
}
```

Die Zählvariable `i` läuft im Intervall [`start`; `ende`] mit der Schrittweite `schritt`.

Sie kann in der Schleife verwendet werden.

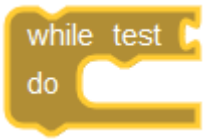
Beispiel:

```
for (int i=0; i<10; i++) {
  ellipse(30, 30, 100-i*10, 100-i*10);
}
```

(i++ entspricht i=i+1)

Hier werden 10 konzentrische Kreise mit Mittelpunkt (30|30) gezeichnet.

Bedingte Schleife:



Solange die Bedingung erfüllt ist, werden die Anweisungen im Schleifenkörper wiederholt.

Wir verwenden hier die while-Schleife, bei der die angegebene Bedingung erfüllt sein muss, um sie auszuführen.

```
while (bedingung_erfuellt) {
  // tue etwas
  // verändere die Voraussetzung für die Bedingung
}
```

Achtung:

Im Schleifenkörper muss darauf geachtet werden, dass die Bedingung irgendwann nicht mehr erfüllt ist und die Schleife damit abbricht.

Beispiel:

```
while (x < width) {
  ellipse(x, 50, 20, 20);
  x=x+10; // der Wert der x-Koordinate wird erhöht
}
```

(Hier wird die x-Koordinate des Mittelpunktes so lange um 10 Pixel erhöht, bis sie größer oder gleich der Fensterbreite ist. Dann wird abgebrochen.)

MIT App Inventor 2	Java

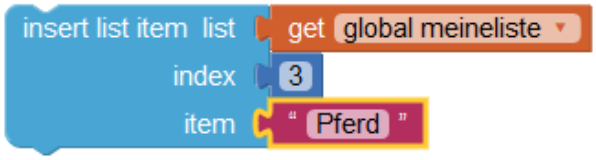
Listen

Erzeugen einer leeren Liste:

Erzeugen einer Liste mit Elementen:

Anhängen eines neuen Elements:

Einfügen eines Elements an einer bestimmten Position:



Element löschen:

Element ersetzen:

Auch in Java gibt es die Möglichkeit, eine Liste zu erzeugen und mit ihr zu arbeiten.

Wir beschränken uns aber auf **Felder** (Arrays).

Deklaration des Feldes:

Typ [ ] Feldname = new Typ[Elementanzahl];

Mit Initialisierung:

Typ [ ] Feldname = {Element 1, Element 2,…};

Beispiele:

```
float[] kommazahlen = new float[10];

int[] zahlenfeld = {1, 2, 4, 8, 16, 32};

char[] zeichenfeld = {'I','M','P','9','!'};
```

Felder haben eine feste Größe, die mit der Eigenschaft `Feldname.length` abgefragt werden kann.

Achtung: Die Zählung beginnt bei 0!

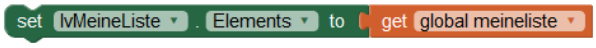
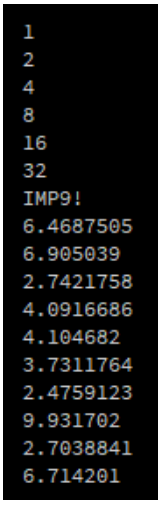
Beispiele zum Durchlaufen von Feldern bzw. dem Füllen eines Feldes mit reellen Zufallszahlen zwischen 1 und 10:

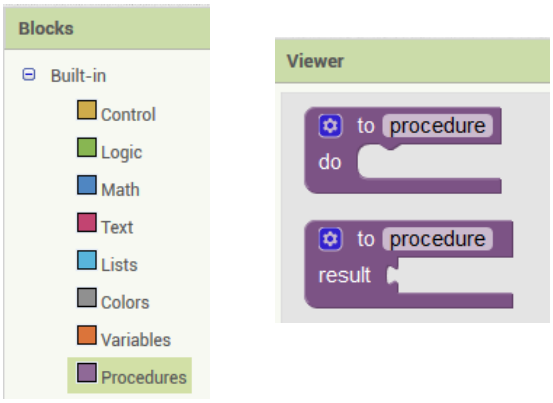
```
for (int i=0; i<zahlenfeld.length; i++) {
  println(zahlenfeld[i]);
}

String z="";
for (int i=0; i<zeichenfeld.length;i++){
  z=z+zeichenfeld[i];
}
println(z);

for (int i=0; i<kommazahlen.length; i++){
  kommazahlen[i]=random(1,10);
  println(kommazahlen[i]);
}
```

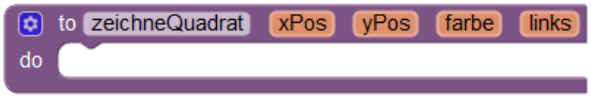
(Ausgabe Abbildung, s.u.)

MIT App Inventor 2	Java
Zugriff auf ein bestimmtes Element:  Zugriff auf die Länge der Liste:  Es gibt noch viele weitere Möglichkeiten, mit Listen zu arbeiten. Siehe Blocks – Lists. Angezeigt werden Listenelemente in einer ListView (User Interface):  	

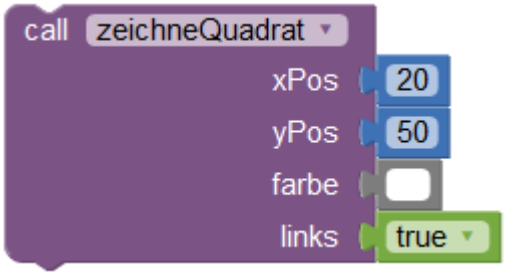
Unterprogramme / Methoden
 	
Erstellen eines neuen Unterprogramms:
 
 	
Der obere Block ist für Methoden ohne Rückgabewert, der untere mit.
	In Java unterscheidet man auch Methoden mit und ohne Rückgabewert. (Siehe auch: Infoblatt zu Methoden.)
	
	Methoden ohne Rückgabewert:
	`void name(<Übergabeparameter>){`
	`  …`
	`};`
	
	Methoden mit Rückgabewert:
	`Typ name(<Übergabeparameter>){`
	`  …`
	`  return rückgabewert;`
	`}`

MIT App Inventor 2

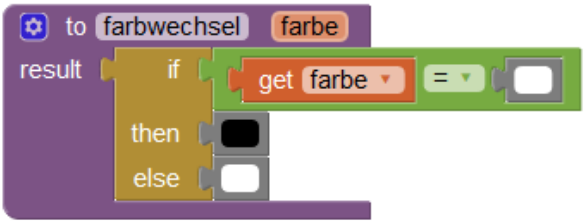
Beispiel des Rahmens einer Methode ohne Rückgabewert mit Übergabeparametern:



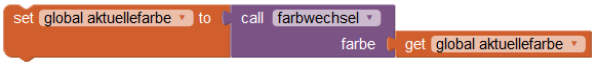
Aufruf der Methode:



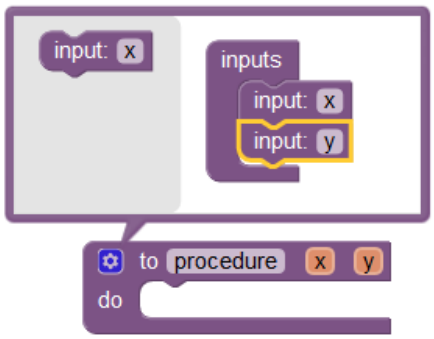
Beispiel für eine Methode mit Rückgabewert:



Aufruf der Methode:



Einfügen der Übergabeparameter:



Java

Beispiele:

```
void zeichneKreis(int x, int y, int r, int farbe){
  fill(farbe);
  ellipse(x,y,r,r);
}

String begruessung(String vorname, String name){
  return "Hallo "+vorname+" "+name+"!";
}

int groesser(int zahl1, int zahl2){
  int erg;
  if (zahl1>zahl2) {
    erg = zahl1;
  } else {
    erg = zahl2;
  }
  return erg;
}
```

Aufruf der Methoden:

```
zeichneKreis(100, 200, 50, #FF0000);

text(begruessung("Hieronymus", "Hase"), 200, 100);

int max;
max = groesser(40, 56); // max erhält hier den Wert 56
```

```
if (klassenstufe == 9) {
   fachwahl="IMP";
}
```