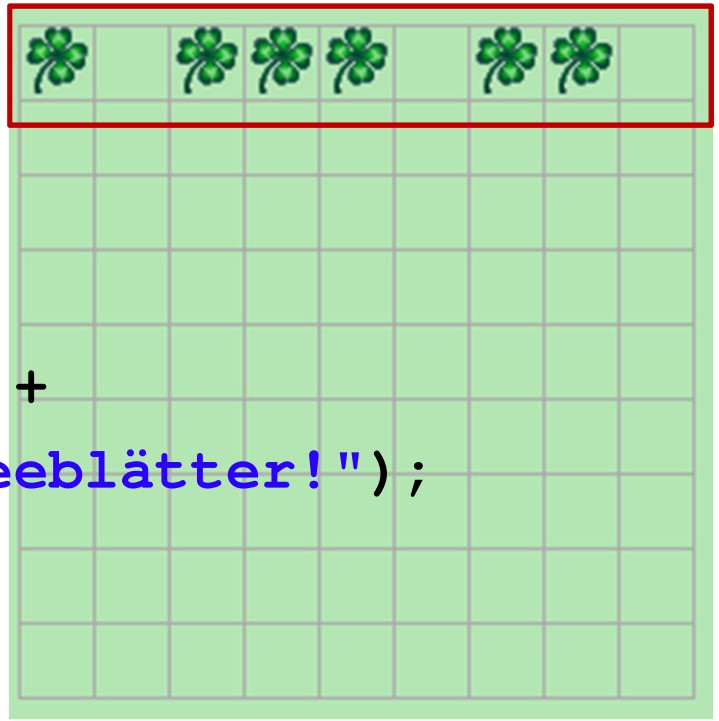
Java programmieren mit JavaKara

Eine Zusammenfassung in Beispielen

Kleeblätter in einer Zeile zählen

```
@Override
public void myMainProgram() {
  int anzahlKleeblaetter = 0;
  for (int x = 0; x < world.getSizeX(); x++) {
    if (world.isLeaf(x, 0)) {
      anzahlKleeblaetter++;
    }
  }
  tools.showMessage("Es hat " +
    anzahlKleeblaetter + " Kleeblätter!");
}
```



Kleeblätter in einer Zeile zählen: Zählschleife

```
@Override
public void myMainProgram() {
  int anzahlKleeblaetter = 0;
  for (int x = 0; x < world.getSizeX(); x++) {
    if (world.isLeaf(x, 0)) {
      anzahlKleeblaetter++;
    }
  }
  tools.showMessage("Es hat " +
    anzahlKleeblaetter + " Kleeblätter!");
}
```

Kleeblätter in einer Zeile zählen: Zählschleife

```
for (int x = 0; x < world.getSizeX(); x++) {
        // Anweisungen
}
```

int x = 0 ist eine Anweisung, die zu Beginn einmal ausgeführt wird

x < world.getSizeX() ist eine Bedingung

Anweisungen werden nur ausgeführt, wenn die Bedingung erfüllt ist

x++ ist eine Anweisung, die nach den Anweisungen ausgeführt wird

Wichtig: Es muss nicht von 0 an gezählt werden; die Bedingung kann beliebig komplex sein; es muss auch nicht um eins erhöht werden. Auch solche Schleifen find möglich (startX, endX sind irgendwelche Variablen):

```
for (int x = startX; (x < endX) && !kara.treeFront(); x = x+2)
```

Kleeblätter in einer Zeile zählen: Bedingung

```
@Override
public void myMainProgram() {
  int anzahlKleeblaetter = 0;
  for (int x = 0; x < world.getSizeX(); x++) {
    if (world.isLeaf(x, 0)) {
      anzahlKleeblaetter++;
    }
  }
  tools.showMessage("Es hat " +
    anzahlKleeblaetter + " Kleeblätter!");
}
```

Kleeblätter in einer Zeile zählen: Bedingung

```
if (world.isLeaf(x, 0)) {
    // Anweisungen
}
```

world.isLeaf(x, 0) ist die Bedingung. Eine Bedingung kann beliebig komplex sein und andere Bedingungen mit &&, ||, ! verknüpfen.

Die Anweisungen werden nur ausgeführt, wenn die Bedingung erfüllt ist.

Eine Bedingung muss immer einen Boole'schen Wert liefern:

- Ein Aufruf einer Methode, die boolean zurückgibt, wie im Beispiel world.isLeaf(x,0).
- Ein Vergleich wie x < world.getSizeX().
- Eine Verknüpfung anderer Bedingungen mit &&, ||, !.

Kleeblätter in einer Zeile zählen: Variablen

```
@Override
public void myMainProgram() {
  int anzahlKleeblaetter = 0;
  for (int x = 0; x < world.getSizeX(); x++) {
    if (world.isLeaf(x, 0)) {
      anzahlKleeblaetter++;
    }
  }
  tools.showMessage("Es hat " +
    anzahlKleeblaetter + " Kleeblätter!");
}
```

Kleeblätter in einer Zeile zählen: Variablen

```
int anzahlKleeblaetter = 0;
anzahlKleeblaetter++;
```

Variablen **speichern Daten.** Sie sind von einem bestimmten Typ, im Beispiel oben Ganzzahl (int).

Variablen müssen **definiert** werden, d.h. ihr Typ muss festgelegt werden: int anzahlKleeblaetter.

Variablen müssen **initialisiert** werden, d.h. es muss ihnen ein erster Wert zugewiesen werden: int anzahlKleeblaetter = 0;

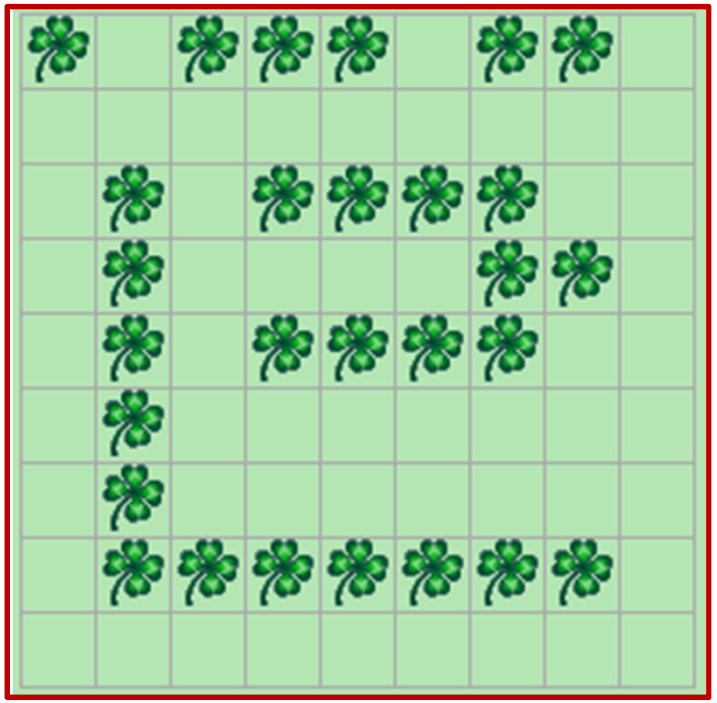
Variablen können beliebig verändert werden:

```
anzahlKleeblaetter++;
anzahlKleeblaetter = anzahlKleeblaetter*10;
...
```

Kleeblätter in Welt zählen

```
@Override
public void myMainProgram() {
  int anzahlKleeblaetter = 0;
  for (int y = 0; y < world.getSizeY(); y++) {
    anzahlKleeblaetter = anzahlKleeblaetter + zaehleKleeblaetter(y);
  }
  tools.showMessage("Es hat " + anzahlKleeblaetter + " Kleeblätter!");
}

int zaehleKleeblaetter(int y) {
  int anzahlKleeblaetter = 0;
  for (int x = 0; x < world.getSizeX(); x++) {
    if (world.isLeaf(x, y)) {
      anzahlKleeblaetter++;
    }
  }
  return anzahlKleeblaetter;
}
```



Kleeblätter in Welt zählen: Methodendefinition

```
int zaehleKleeblaetter(int y) {
  int anzahlKleeblaetter = 0;
  for (int x = 0; x < world.getSizeX(); x++) {
    if (world.isLeaf(x, y)) {
      anzahlKleeblaetter++;
    }
  }
  return anzahlKleeblaetter;
}
```

Kleeblätter in Welt zählen: Methodendefinition, Parameter

Der Parameter y ist innerhalb der Methode zaehleKleeblaetter eine normale Variable.

Die Methode zaehleKleeblaetter erhält eine Kopie von y. Sie könnte y = 0; setzen, ohne dass das Auswirkungen auf die aufrufende Methode hätte. Das gilt für alle Parameter von einfachen Datentypen (int, boolean, ...).

Kleeblätter in Welt zählen: Methodendefinition, Parameter

```
int y = 0;
zaehleKleeblaetter(y);
// y hat immer noch den Wert 0

int zaehleKleeblaetter(int y) {
        y = 7;
        return 42;
}
```

Die Bezeichnung des Parameters spielt dabei überhaupt keine Rolle:

- Im Hauptprogramm wird die Variable y definiert.
- Sie wird als Parameter mit dem Namen y (hier der gleiche Name – kann aber auch ein anderer Name sein!) an die Methoden zaehleKleeblaetter übergeben.
- zaehleKleeblaetter erhält eine Kopie von y. Änderungen an Parameter y haben keine Auswirkungen auf die aufrufende Methode!

Kleeblätter in Welt zählen: Methodenaufruf

```
@Override
public void myMainProgram() {
  int anzahlKleeblaetter = 0;
  for (int y = 0; y < world.getSizeY(); y++) {
    anzahlKleeblaetter =
      anzahlKleeblaetter +
        zaehleKleeblaetter(y);
  }
  tools.showMessage("Es hat " +
    anzahlKleeblaetter + " Kleeblätter!");
}
```

Kleeblätter in Welt zählen, Version 2: Doppelt geschachtelte Schleife

```
@Override
public void myMainProgram() {
  int anzahlKleeblaetter = 0;
  for (int y = 0; y < world.getSizeY(); y++) {
    for (int x = 0; x < world.getSizeX(); x++) {
      if (world.isLeaf(x, y)) {
        anzahlKleeblaetter++;
      }
    }
  }
  tools.showMessage("Es hat " +
    anzahlKleeblaetter + " Kleeblätter!");
}
```

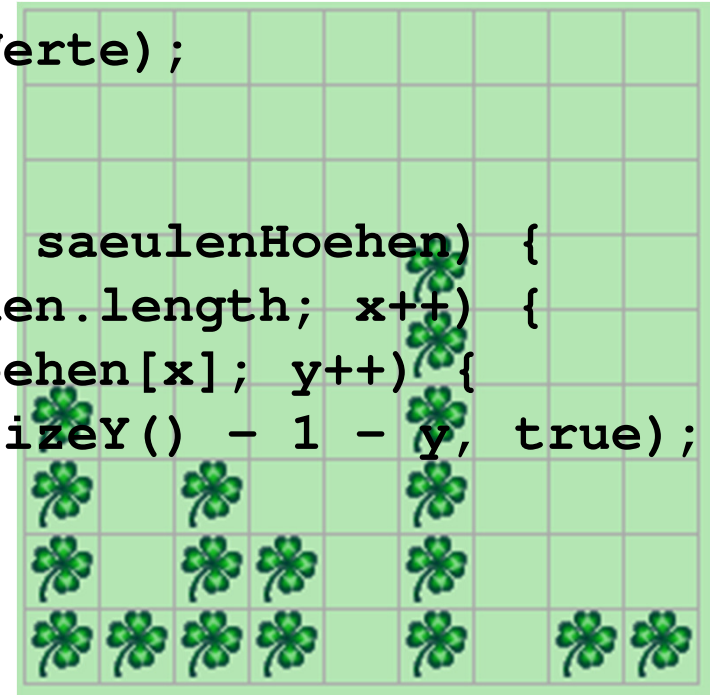
Kleeblätter in Welt zählen, Version 3: Doppelt geschachtelte Schleife

```
@Override
public void myMainProgram() {
  int anzahlKleeblaetter = 0;
  for (int x = 0; x < world.getSizeX(); x++) {
    for (int y = 0; y < world.getSizeY(); y++) {
      if (world.isLeaf(x, y)) {
        anzahlKleeblaetter++;
      }
    }
  }
  tools.showMessage("Es hat " +
    anzahlKleeblaetter + " Kleeblätter!");
}
```

Säulendiagramm zeichnen: Arrays

```
@Override
public void myMainProgram() {
  int[] zufallsWerte = new int[world.getSizeX()];
  for (int x = 0; x < zufallsWerte.length; x++) {
    zufallsWerte[x] = tools.random(world.getSizeY()-1);
  }
  zeichneSaeulenDiagramm(zufallsWerte);
}

void zeichneSaeulenDiagramm(int[] saeulenHoehen) {
  for (int x = 0; x < saeulenHoehen.length; x++) {
    for (int y = 0; y < saeulenHoehen[x]; y++) {
      world.setLeaf(x, world.getSizeY() - 1 - y, true);
    }
  }
}
```



Säulendiagramm zeichnen: Arrays

int[] zufallsWerte = new int[world.getSizeX()];

Variablen speichern Daten. Sie sind von einem bestimmten Typ. zufallsWerte ist eine Variable vom Typ «Array von Ganzzahlen».

Array bedeutet: Eine feste (nicht-veränderliche) Anzahl von Daten vom gleichen Typ, die über einen Index von 0..<Anzahl-1> angesprochen werden.

Variablen müssen definiert werden, d.h. ihr Typ muss festgelegt werden: int[] zufallsWerte.

Variablen müssen initialisiert werden, d.h. es muss ihnen ein erster Wert zugewiesen werden: new int[world.getSizeX()]. Anschliessend haben alle Elemente den Wert 0: zufallsWerte[0] = zufallsWerte[1] = ... = zufallsWerte[world.getSizeX()-1] = 0.

Variablen können beliebig verändert werden:
zufallsWerte[0] = 10; // ändert den Inhalt des ersten Arrayelements

Säulendiagramm zeichnen: Arrays als Parameter

```
void zeichneSaeulenDiagramm(int[] saeulenHoehen) {
  for (int x = 0; x < saeulenHoehen.length; x++) {
    for (int y = 0; y < saeulenHoehen[x]; y++) {
      world.setLeaf(x, world.getSizeY() - 1 - y, true);
    }
  }
}
```

Säulendiagramm zeichnen: Arrays als Parameter

Der Parameter saeulenHoehen ist innerhalb der Methode zeichneSaeulenDiagramm eine normale Variable.

Die Methode zeichneSaeulenDiagramm erhält das Orginal von saeulenHoehen. Würde die Methode zum Beispiel saeulenHoehen[x] = 0; setzen, würde das den Array der aufrufenden Methode ändern. Das gilt für alle Parameter von komplexen Datentypen (Arrays, Objekte, ...).

Säulendiagramm zeichnen: Arrays als Parameter

```
int[] zufallsWerte = new int[world.getSizeX()];
zeichneSaeulenDiagramm(zufallsWerte);
// zufallsWerte[0] hat jetzt Wert 7, nicht mehr 0!

void zeichneSaeulenDiagramm(int[] saeulenHoehen) {
        saeulenHoehen[0] = 7;
}
```

Die Bezeichnung des Arrays (es gibt in diesem Programm nur einen Array!) spielt dabei überhaupt keine Rolle:

- Im Hauptprogramm wird der Array zufallsWerte definiert.
- Er wird als Parameter mit dem Namen saeulenHoehen an die Methoden zeichneSaeulenDiagramm übergeben.
- saeulenHoehnen «verweist» aber auf den gleichen Array, auch wenn es ein anderer Name ist als zufallsWerte!

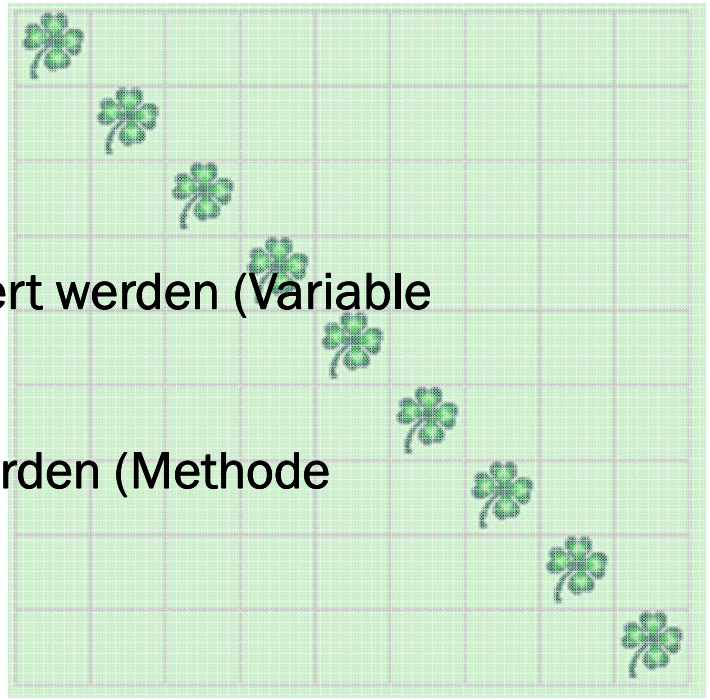
Die Welt von Kara: Zwei-dimensionaler Array (wie Tabelle)

```
@Override
public void myMainProgram() {
  boolean[][] neueFelder =
    new boolean[world.getSizeX()][world.getSizeY()];
  berechneNeueFelder(neueFelder);
  schreibeNeueFelder(neueFelder);
}
```

Das Programm soll eine Welt berechnen.

Die neue Welt soll zunächst zwischengespeichert werden (Variable neueFelder, Methode berechneNeueFelder).

Anschliessend soll die neue Welt dargestellt werden (Methode schreibeNeueFelder).



Die Welt von Kara: Zwei-dimensionaler Array (wie Tabelle)

```
void berechneNeueFelder(boolean[][] neueFelder) {
  for (int y = 0; y < world.getSizeY(); y++) {
    for (int x = 0; x < world.getSizeX(); x++) {
      neueFelder[y][x] =
        berechneNeuenWert(x, y, world.isLeaf(x, y));
    }
  }
}

boolean berechneNeuenWert(int x, int y, boolean l) {
  return x == y;
}
```

Diese Methoden berechnen eine Welt mit Kleeblättern in der Diagonale von links oben nach rechts unten.

Es geht auch drei-dimensional...

```
@Override
public void myMainProgram() {
  boolean[][][] neueFelder =
    new boolean[10] [world.getSizeX()][world.getSizeY()];
  for (int i = 0; i < 10; i++) {
    berechneNeueFelder(neueFelder, i);
  }
  schreibeNeueFelder(neueFelder[9]);
  // neueFelder[9] übergibt den zwei-dimensionalen Array
}
```

Vielleicht soll der zeitliche Verlauf der berechneten Arrays gespeichert werden (in der dritten Dimension)...