
Java™

Wiederholung

- So sieht ein „leeres“ Java-Programm aus

```
public class Programmname {

    public static void main (String[] args) {

        // Hier stehen die Anweisungen

    }

}
```

Wiederholung

- Anweisungen mit Variablen

  - Deklaration einer Variablen (muss initialisiert werden)
    ```
    Datentyp variablenName;
    ```

  - Deklaration und Initialisierung einer Variablen
    ```
    Datentyp variablenName = Wert;
    ```

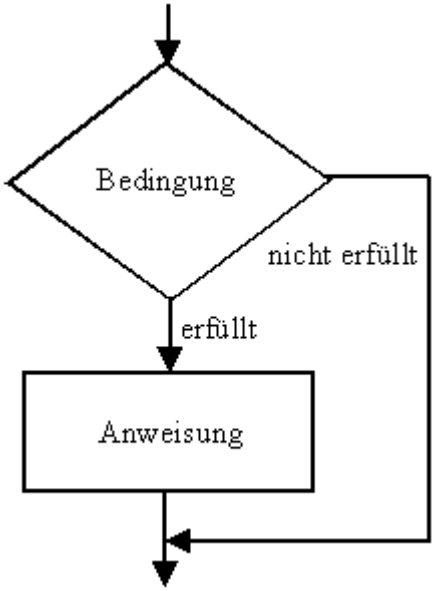
  - Zuweisung eines Wertes zu einer Variablen
    ```
    variablenName = Ausdruck;
    ```

Kontrollstrukturen

- Sequenz
  - Eine Anweisung wird nach der anderen ausgeführt
- Selektion
  - Anweisung wird in Abhängigkeit einer Bedingung ausgeführt
- Iteration
  - Anweisung wird mehrfach wiederholt ausgeführt

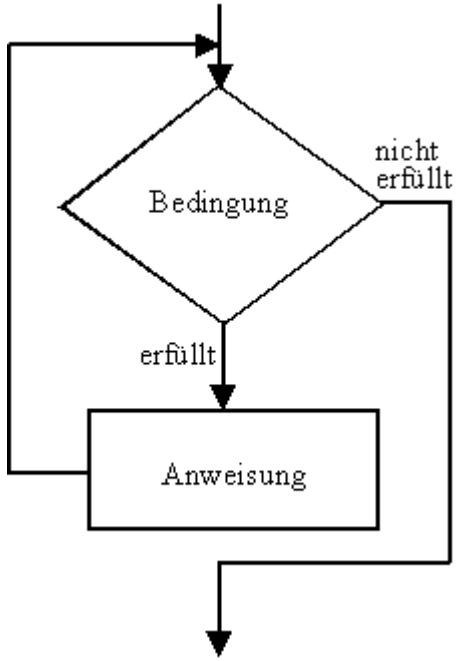
Selektion

```
if (...) {
   ...
}
```



Iteration

```
while (…) {
   …
}
```



Wiederholung

- Anweisungen durch Methodenaufrufe
  - Ausgabe auf der Konsole
    ```
    System.out.println(...);
    ```
  - Benutzereingabe einlesen
    ```
    … = scanner.nextDatentyp();
    ```
  - Mathematische Funktionen
    ```
    Math.sqrt(…), Math.sin(…), Math.random(), usw.
    ```

Weitere Methodenaufrufe

- Ohne Parameter, ohne Rückgabewert
  - `scanner.close();`
  - `System.out.println();`
- Mit Parameter(n), ohne Rückgabewert
  - `Thread.sleep(1000);`
  - `System.out.println(123);`
- Ohne Parameter, mit Rückgabewert
  - `int result = scanner.nextInt();`
  - `double result = Math.random();`
- Mit Parameter(n), mit Rückgabewert
  - `double result = Math.max(2.5,-2.5);`
  - `String result = "Hello World".replace('l','m');`

Eigene Methoden schreiben

- Deklaration einer Methode
  - `public static RückgabeDatentyp name (Parameterliste) {`
    `Anweisungen`
    `}`
  - Rückgabe-Datentyp
    - Beliebiger Datentyp oder `void`, falls keine Rückgabe erfolgt
  - Name
    - Klein Schreiben (wie bei Variablen)
  - Parameterliste
    - Datentyp parameterName
    - Mit Komma getrennt (falls mehrere Parameter verwendet werden)
  - Anweisungen
    - Beliebige Anweisungen (siehe Kontrollstrukturen)
    - Letzte Anweisung muss `return`-Statement sein

Arrays

Arrays

- Array = Feld
  - Mehrere Variablen eines Typs zusammengefasst
- Anzahl der Variablen = Länge
  - Muss bei Deklaration des Arrays festgelegt werden
    - Entweder als fester Wert im Programmtext
    - Oder als Variable, die ggf. zur Laufzeit erst festgelegt wird
  - Ist nicht dynamisch veränderbar

Deklaration und Wertzuweisung

- Elementare Anweisungen mit Variablen

  - Deklaration einer Variablen (muss initialisiert werden)
    ```
    Datentyp variablenName;
    ```

  - Deklaration und Initialisierung einer Variablen
    ```
    Datentyp variablenName = Wert;
    ```

  - Zuweisung eines Wertes zu einer Variablen
    ```
    variablenName = Ausdruck;
    ```

Deklaration und Wertzuweisung

- Elementare Anweisungen bei Arrays

  - Deklaration eines Arrays (muss initialisiert werden)
    ```
    Datentyp[] arrayName;
    ```

  - Deklaration und Initialisierung eines Arrays mit Werten
    ```
    Datentyp[] arrayName = {wert0, wert1, … };
    ```
    Deklaration und Initialisierung eines „leeren“ Arrays der Länge N
    ```
    Datentyp[] arrayName = new Datentyp[N];
    ```

  - Zuweisung eines Wertes zu dem Array-Eintrag an einer Position
    ```
    arrayName[position] = Ausdruck;
    ```

Deklaration und Wertzuweisung

- 1. Beispiel

  - Deklaration eines Arrays (muss initialisiert werden)

```
float[] messwerte;
```

  - Deklaration und Initialisierung eines Arrays mit Werten

```
float[] messwerte = {2.35f, 1.17f, 4.22f};
```

    Deklaration und Initialisierung eines „leeren“ Arrays der Länge 100

```
float[] messwerte = new float[100];
```

  - Zuweisung eines Wertes zu dem Array-Eintrag an erster Position

```
messwerte[0] = 2.35f;
```

Deklaration und Wertzuweisung

- 2. Beispiel

  - Deklaration eines Arrays (muss initialisiert werden)

```
String[] vorlesungen;
```

  - Deklaration und Initialisierung eines Arrays mit Werten

```
String[] vorlesungen = {"Vermessung", "Mathematik"};
```

    Deklaration und Initialisierung eines „leeren“ Arrays der Länge 4

```
String[] vorlesungen = new String[4];
```

  - Zuweisung eines Wertes zu dem Array-Eintrag an zweiter Position

```
vorlesungen[1] = "Informatik";
```

Arrays

- 3. Beispiel

  - Deklaration eines Arrays (muss initialisiert werden)

```
boolean[] bestanden;
```

  - Deklaration und Initialisierung eines Arrays mit Werten

```
boolean[] bestanden = {false, false, false, false};
```

    Deklaration und Initialisierung eines „leeren“ Arrays der Länge 4

```
boolean[] bestanden = new boolean[4];
```

  - Zuweisung eines Wertes zu dem Array-Eintrag an dritter Position

```
bestanden[2] = true;
```

Deklaration und Wertzuweisung

- 4. Beispiel

  - Deklaration einer Variablen (muss initialisiert werden)

```
int[] lotto;
```

  - Deklaration und Initialisierung eines Arrays mit Werten

```
int[] lotto = {9, 32, 35, 38, 42, 43};
```

    Deklaration und Initialisierung eines „leeren“ Arrays der Länge 6

```
int[] lotto = new int[6];
```

  - Zuweisung des Wertes 38 für die 4. Zahl (Index 3)

```
lotto[3] = 38;
```

Arrays

- Speicherung im Rechner

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

Arrays

- Speicherung im Rechner

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

array[2] = 6;
```

0	0	0	0	0	0	0	0

0	0	6	0	0	0	0	0

```
array[4] = 3;

array[6] = 8;

array[2+3] = array[4] * array[2];

array[2+3] -= array[array[2]];

if (array[0] <= array[1])
   array[7] = array[5];
```

Die Länge eines Arrays

- Deklaration und Initialisierung eines Arrays mit N Werten

```
Datentyp[] arrayName = {wert0, wert1, … };
```

- Deklaration und Initialisierung eines „leeren“ Arrays der Länge N

```
Datentyp[] arrayName = new Datentyp[N];
```

- Die Länge N eines Arrays
  - Ist eine finale Variable (kann nicht verändert werden)

```
arrayName.length // N
```

Array Index Out of Bounds Exception

- Erlaubte Zugriffe

```
arrayName[0]
arrayName[1]
arrayName[2]
...
arrayName[N-2]
arrayName[N-1]
```

- Verbotene Zugriffe (Programm stürzt ab)

```
...
arrayName[-2]
arrayName[-1]

arrayName[N]
arrayName[N+1]
arrayName[N+2]
...
```

Zusammenfassung

- Elementare Anweisungen bei Arrays

  - Deklaration eines Arrays (muss initialisiert werden)
    ```
    Datentyp[] arrayName;
    ```

  - Deklaration und Initialisierung eines Arrays mit Werten
    ```
    Datentyp[] arrayName = {wert0, wert1, … };
    ```
    Deklaration und Initialisierung eines „leeren“ Arrays der Länge N
    ```
    Datentyp[] arrayName = new Datentyp[N];
    ```

  - Zuweisung eines Wertes zu dem Array-Eintrag an einer Position
    ```
    arrayName[position] = Ausdruck;
    ```

Übung

- Was passiert mit diesem Array?

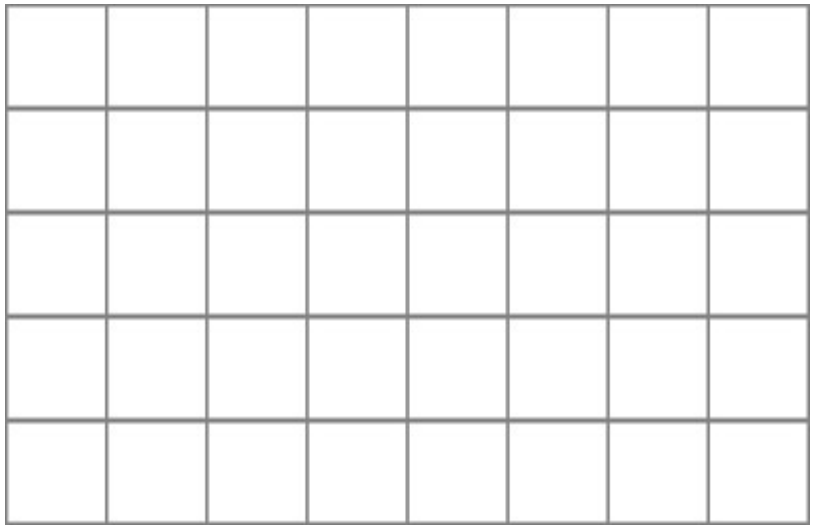
```
int[] array = new int[8];

array[0] = 1;
array[1] = 1;

int i = 2;

while (array[i] < array.length) {

   array[i] = array[array[i]] + array[i] + i - 1;

   i = array[i] - i;

}
```

1	1	0	0	0	0	0	0

Arrays

- Speicherung im Rechner

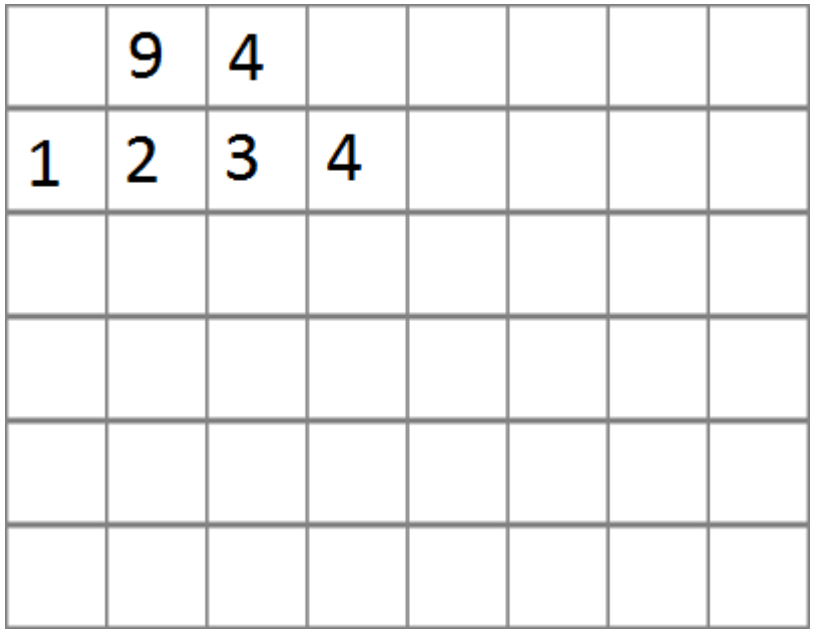
```
int[] array = {1,2,3,4};
```



Arrays

- Speicherung im Rechner

```
int[] array = {1,2,3,4};
```




```
Variable array belegt Speicherplatz 2 und 3

Speicherplatz 2 sagt: das Array beginnt an Speicherplatz 9
Speicherplatz 3 sagt: das Array hat eine Länge von 4
```

Arrays

- Zuweisung von Arrays
  - Flache Kopie

```
int[] a = {1972, 1980, 1996, 2016};

int[] b = a;

b[3] = 2020;

System.out.println(a[3]); // 2020
```

Arrays

- Zuweisung von Arrays
  - Tiefe Kopie

```
int[] a = {1972, 1980, 1996, 2016};

int[] b = a.clone();

b[3] = 2020;

System.out.println(a[3]); // 2016
```

Arrays

- Methodenaufruf
  - Call by Reference

```
public static void methode(int[] array) {
    array[3] = 2020;
}

public static void main(String[] args) {

    int[] a = {1972, 1980, 1996, 2016};

    methode(a);

    System.out.println(a[3]); // 2020

}
```

Arrays

- Methodenaufruf
  - Call by Value

```
public static void methode(int value) {
    value = 2020;
}

public static void main(String[] args) {

    int[] a = {1972, 1980, 1996, 2016};

    methode(a[3]);

    System.out.println(a[3]); // 2016

}
```

Arrays

- Array-Variable

```
int[] a = {1972, 1980, 1996, 2016};

System.out.println(a); // Speicherposition (hexadezimal)
```

Arrays

- Ausgabe eines Arrays
  - Ohne Schleife

```
int[] array = {1954, 1974, 1990, 2014};

System.out.println(array[0]);
System.out.println(array[1]);
System.out.println(array[2]);
System.out.println(array[3]);
```

Arrays

- Ausgabe eines Arrays
  - Mit Schleife (Benutzen Sie diese Variante)

```
int[] array = {1954, 1974, 1990, 2014};

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

Arrays

- Ausgabe eines Arrays
  - For Each Schleife

```
int[] array = {1954, 1974, 1990, 2014};

for (int eintrag : array) {
   System.out.println(eintrag);
}
```

Arrays

- Array mit Zufallszahlen füllen
  - Mit Schleife

```
int anzahl = 10;
int bereich = 100;


int[] array = new int[anzahl];

for (int i=0; i<array.length; i++) {
   array[i] = (int)(Math.random()*bereich);
}
```

Arrays

- Methoden mit Arrays
  - Array ausgeben

```
public static void ausgabe(int[] array) {

   for (int i=0; i<array.length; i++) {
      System.out.println(array[i]);
   }

}
```

Arrays

- Methoden mit Arrays
  - Array mit Zufallszahlen erstellen

```
public static int[] zufzahlen() {

    int[] array = new int[10];
    for (int i=0; i<array.length; i++) {
       array[i] = (int)(Math.random()*100);
    }
    return array;

}
```

Arrays

- Methoden mit Arrays
  - Array mit Zufallszahlen erstellen (mit Parameter)

```
public static int[] zufzahlen(int anzahl){

    int[] array = new int[anzahl];
    for (int i=0; i<array.length; i++) {
       array[i] = (int)(Math.random()*100);
    }
    return array;

}
```

Arrays

- Methoden mit Arrays
  - Hauptprogramm

```
public static void main(String[] args) {

    int[] array = zufzahlen(20);
    ausgabe(array);

}
```

Arrays

- Array sortieren mit der Methode **`Arrays.sort`**
  - Java-Programm

```
public static void main(String[] args) {

    int[] array = zufzahlen(20);
    ausgabe(array);

    Arrays.sort(array);
    ausgabe(array);

}
```

sort

`public static void sort(int[] a)`

Sorts the specified array into ascending numeri

Implementation note: The sorting algorithm is a traditional (one-pivot) Quicksort implementation

Parameters:

a - the array to be sorted

Arrays

- Array sortieren mit eigener Methode
  - Java-Programm

```
public static void main(String[] args) {

    int[] array = zufzahlen(20);
    ausgabe(array);

    mysort(array);
    ausgabe(array);

}
```

Array sortieren mit Merge-Sort

```
public static void mysort(int[] array) {

   if (array.length <= 1)
      return;

   int[] lower = lowerHalf(array);
   int[] upper = upperHalf(array);

   sort(lower);
   sort(upper);

   array = merge(lower,upper);

}
```

Array sortieren mit Merge-Sort

```
public static int[] lowerHalf(int[] array) {

    int[] lower = new int[array.length/2];

    for (int i=0; i < lower.length; ++i) {

        lower[i] = array[i];

    }

    return lower;

}
```

Array sortieren mit Merge-Sort

```
public static int[] upperHalf(int[] array) {

    int[] upper = new int[(array.length+1)/2];

    for (int i=0; i < upper.length; ++i) {

        lower[i] = array[i+array.length/2];

    }

    return upper;

}
```

Arrays

```
public static int[] merge(int[] array1, int[] array2) {

   int[] array = new int[array1.length + array2.length];

   int i1 = 0;
   int i2 = 0;
   for (int i=0; i < array.length; i++) {

      if (i2 == array2.length ||
         (i1 < array.length && array1[i1] < array2[i2])) {
         array[i] = array1[i1];
         i1++;
      } else {
         array[i] = array2[i2];
         i2++;
      }

   }

}
```

Deklaration und Wertzuweisung

- Elementare Anweisungen bei Arrays

  - Deklaration eines Arrays (muss initialisiert werden)
    ```
    Datentyp[] arrayName;
    ```

  - Deklaration und Initialisierung eines Arrays mit Werten
    ```
    Datentyp[] arrayName = {wert0, wert1, … };
    ```
    Deklaration und Initialisierung eines „leeren“ Arrays der Länge N
    ```
    Datentyp[] arrayName = new Datentyp[N];
    ```

  - Zuweisung eines Wertes zu dem Array-Eintrag an einer Position
    ```
    arrayName[position] = Ausdruck;
    ```

Deklaration und Wertzuweisung

- Elementare Anweisungen bei mehrdimensionalen Arrays

  - Deklaration eines Arrays (geht so nicht)

    ```
    Datentyp[][] arrayName;
    ```

  - Deklaration und Initialisierung eines mehrdimensionalen Arrays

    ```
    Datentyp[][] arrayName = {{…}, {…}, … , {…}};
    ```

    Deklaration und Initialisierung einer „leeren Matrix“ der Länge NxM

    ```
    Datentyp[][] arrayName = new Datentyp[N][M];
    ```

    Deklaration und Initialisierung eines mehrdim. Arrays der Länge N

    ```
    Datentyp[][] arrayName = new Datentyp[N][];
    ```

  - Zuweisung eines Wertes zu dem Matrix-Eintrag an einer Position

    ```
    arrayName[zeile][spalte] = Ausdruck;
    ```

Mehrdimensionale Arrays

- Beispiel: Sudoku

```
int[][] sudoku = new int[9][9];
```

```
for (int z = 0; z < 9; ++z) {
   for (int s = 0; s < 9; ++s) {
      sudoku[z][s] = …;
   }
}
```

```
solve(sudoku);
```

Mehrdimensionale Arrays

- Beispiel: Matrix-Vektor-Multiplikation (3D)

```
double[] vector = {0.1, 0.2, 0.3};
double[][] matrix = {{0,1,0},{1,0,0},{0,0,1}};
```

```
double[] ergebnis = new double[3];
for (int z = 0; z < 3; ++z) {
   ergebnis[z] = 0;
   for (int s = 0; s < 3; ++s) {
      ergebnis[z] += matrix[z][s]*vector[s];
   }
}
```

```
ausgabe(ergebnis);
```

Übung

- Wie viele Einträge sind insgesamt in einem 2D-Array?

```
public class ArrayTest {

    public static int length(double[][] array) {









    }

}
```

Übung

- Wie viele Einträge sind insgesamt in einem 2D-Array?

```
public class ArrayTest {

    public static int length(double[][] array) {

        int l = 0;

        for (int i = 0; i < array.length; ++i) {

            length += array[i].length;

        }

        return length;

    }

}
```

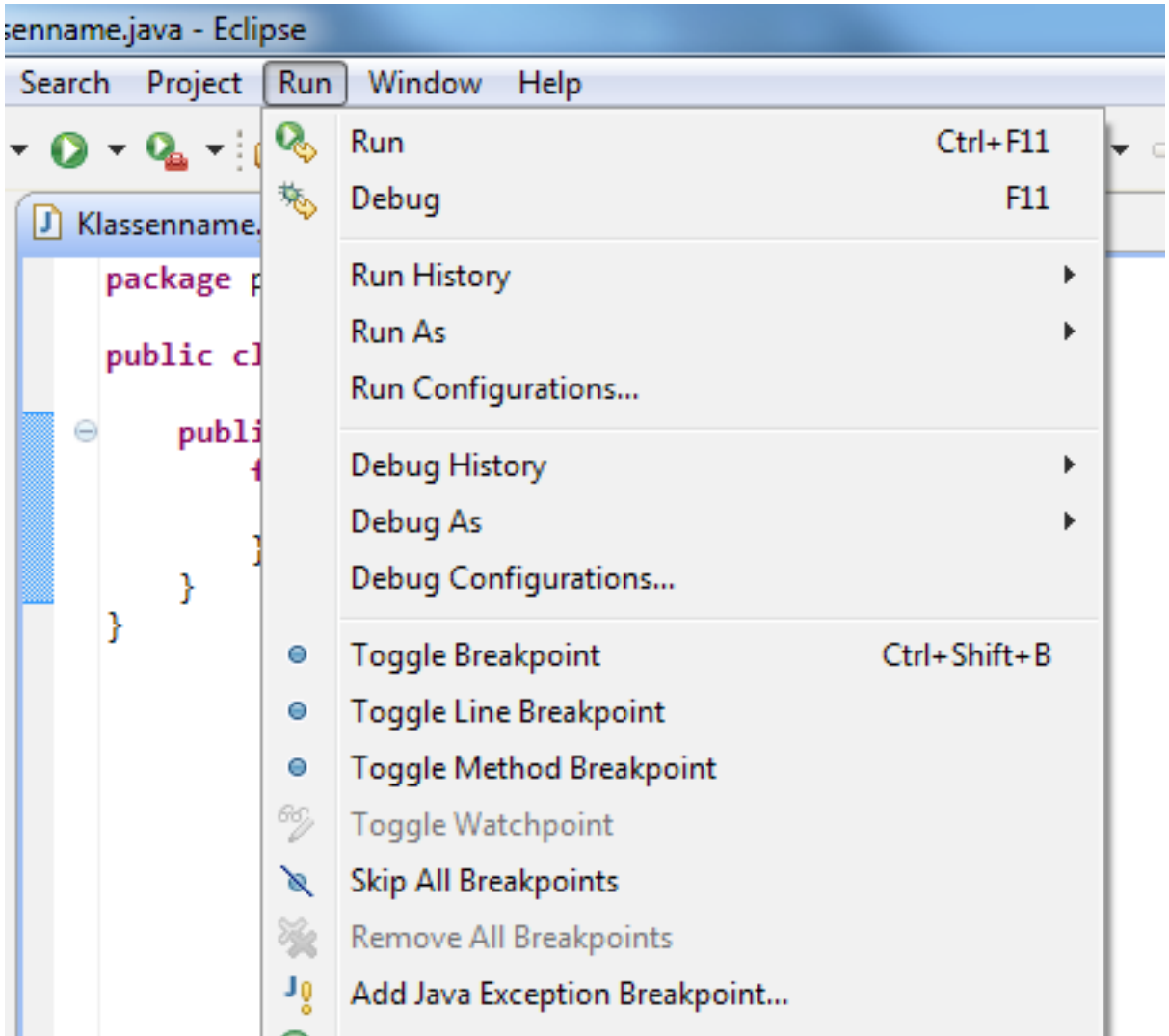
Kommandozeile

Kommandozeilen-Argumente

- Parameter für das Hauptprogramm (`main`-Methode)
- Werden von dem das Hauptprogramm aufrufenden Programm als Parameter übergeben
- Das Programm wird von übergeordneten Programmen wie eine Methode verwendet

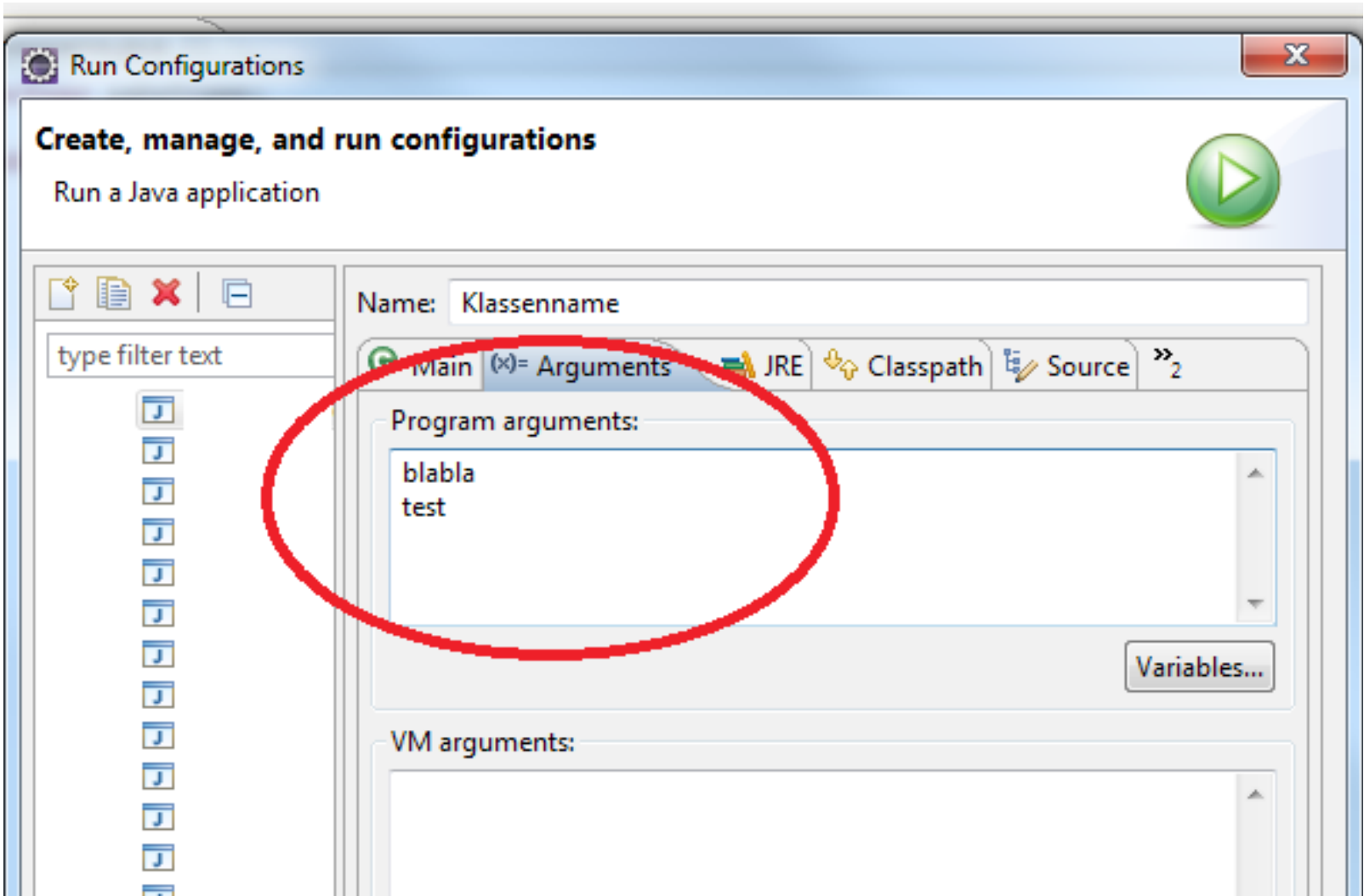
In Eclipse

- Run → Run Configurations...



In Eclipse

- Arguments → Program arguments:



Im Programm

- Beispiel Eingabe: „blabla“ und „test“

```
public class Kommandozeile {

   public static void main (String[] args) {

      System.out.println(args.length); // 2

      System.out.println(args[0]); // blabla

      System.out.println(args[1]); // test

   }

}
```

Hausaufgaben

3.8 Arrays

3.9 Der Einstiegspunkt für das Laufzeitsystem: main()