# Tagesprogramm

Werte und Typen

Methoden und Parameter

Ergebnisse und Seiteneffekte

# Literale unterschiedlicher Typen

| | | |
|---|---|---|
| `int` | ganze Zahl | `0, 1, -1, 2, -2, ...` |
| `double` | Fließkommazahl | `1.23, 12345.0, -1.23e12, 1.23e-12, ...` |
| `String` | Zeichenkette | `"Das ist ein Text", "3", "", ...` |
| `char` | Buchstabe | `'a', 'B', '7', ...` |
| `boolean` | Wahrheitswert | `true, false` |

# Typdeklaration für Variablen

```
int i;
```

`i` ist Variable vom Typ `int`,
jeder Wert in `i` ist ganze Zahl

```
String s = "first";
```

`s` ist Variable vom Typ `String`,
Wert in `s` ist anfangs `"first"`,
durch Zuweisung in andere Zeichenkette änderbar

```
double d = 3.0;
```

Typen von Variable und Anfangswert stimmen überein

```
boolean b = (i <= 3);
```

Typen von Variable und Anfangswert stimmen überein,
<= auf ganzen Zahlen liefert Wahrheitswert

# Typkonsistenz

```
boolean b; b = true;
boolean b; b = 'b';

if (2 <= 3) {...}
if (2 + 3) {...}

boolean b = (4 <= 3);   while (b) {...}
int i = 3;   while (i) {...}

String s = (b ? "true" : "false");
String s = ((3 + 4) ? "true" : "false");
String s = (b ? b : i);
```

# Aufgabe: Typen konsistent?

In welchen der folgenden Variablendeklarationen mit Initialisierungen werden Typen konsistent verwendet?

A
```
int string = 27;
```

B
```
String wahrheitswert = true;
```

C
```
char a = "a";
```

D
```
boolean wahr = false;
```

# Methode als Abstraktionswerkzeug

```
public class MehrereDivisionen {

    /* Methode mit           Parameter                    */
    public static void main(String[] args) {
        divide(5, 2);         /* Methodenaufrufe mit ...  */
        divide(9, 5);         /* ... je zwei Argumenten   */
        divide(1234, 35);
        divide(19, 17);
    }

    /* Methode mit              Parameter,    Parameter   */
    private static void divide(int dividend, int divisor) {
        System.out.println(dividend / divisor);
        System.out.println(dividend % divisor);
    }
}
```

# Methodenaufruf mit Parameterübergabe

```
divide(5, 2);
```

```
int dividend 5      int divisor 2

System.out.println(dividend / divisor 2 );
System.out.println(dividend % divisor 1 );
```

```
divide(9, 5);
```

```
int dividend 9      int divisor 5

System.out.println(dividend / divisor 1 );
System.out.println(dividend % divisor 4 );
```

# Aufgabe: Namen von formalen Parametern

```
public class Parameternamen {
    public static void main(String[] args) {
        int x = 1;
        int y = 2;
        xyMitMinus(y, x);
    }
    private static void xyMitMinus(int x, int y) {
        System.out.println(x - y);
    }
}
```

Welche Zahl gibt dieses Programm aus?

A -1

B 0

C 1

# Methodenergebnisse

```
public class MehrereDivisionenUndMethoden {

    public static void main(String[] args) {
        prnt(divRest(5, 2));          /* verwende Ergebnis */
        int rest = divRest(9, 5);
        prnt(rest);
    }

    /* Methode ohne Ergebnis (void)                        */
    private static void prnt(int value) {
        System.out.println(value);    /* Seiteneffekt      */
    }

    /* Methode mit Ergebnis vom Typ int                    */
    private static int divRest(int dividend, int divisor) {
        prnt(dividend / divisor);     /* Seiteneffekt      */
        return (dividend % divisor); /* Rückgabe Ergebnis */
    }
}
```

# Rückgabe von Ergebnissen

```
divRest(5, 2);
```


```
int dividend 5     int divisor 2

prnt(dividend / divisor 2 );
return(dividend % divisor 1 );
```

```
divRest(9, 5);
```


```
int dividend 9     int divisor 5

prnt(dividend / divisor 1 );
return(dividend % divisor 4 );
```

# Aufgabe: Ergebnisse, Seiteneffekte verstehen

Finden Sie in Gruppen zu 2 bis 3 Personen heraus, welche Seiteneffekte dieses Programm hat und welche Ergebnisse zurückgegeben werden:

```
public class SeiteneffekteUndErgebnisse {

    public static void main(String[] args) {
        int m = 1027;    /* m und n: Anfangswerte änderbar */
        int n = 395;     /* m > 0 und n > 0                */

        while (m != n) {
            m = minimiere(m, n);
            n = minimiere(n, m);
        }
        System.out.println(m);
    }

    private static int minimiere(int x, int y) {
        return ((y < x) ? (x - y) : x);
    }
}
```