

# Woche 04

Kontrollstrukturen, Kontrollflussdiagramme

# Kontrollstrukturen

- Selektion (bedingte Auswahl)

- if-Anweisung
- if-else-Anweisung
- else if
- switch

- Iteration (wiederholte Ausführung)

- while-Schleife
- do-while-Schleife
- for-Schleife

# if-Anweisung

## Allgemeine Form:

```
1 if(Bedingung A) {  
2     // Code, der läuft, wenn A true ergibt  
3 }
```

## Beispiel:

```
1 public static void printIfZero(int number){  
2     if(number == 0) {  
3         System.out.println("Zero!");  
4     }  
5 }
```

# if-else Anweisung

## Allgemeine Form:

```
1 if(Bedingung A) {  
2     // Code, der läuft, wenn A true ergibt  
3 } else {  
4     // Code, der läuft, wenn A false ergibt  
5 }
```

## Beispiel:

```
1 public static void printIfZero(int number){  
2     if(number == 0) {  
3         System.out.println("Zero!");  
4     } else {  
5         System.out.println("Not zero!");  
6     }  
7 }
```

# else if

in Beziehung darauf

```
1  if(number == 0) {  
2      System.out.println("Zero!");  
3  } else {  
4      if(number < 0) {  
5          System.out.println("Negative!");  
6      } else {  
7          System.out.println("Positive");  
8      }  
9  }
```

kann kompakter dargestellt werden als:

→

```
1  if(num == 0) {  
2      System.out.println("Zero!");  
3  } else if(num < 0) {  
4      System.out.println("Negative!");  
5  } else {  
6      System.out.println("Positive!");  
7  }
```

# while-Schleife

## Allgemeine Form:

```
1 while(Bedingung A){  
2     // Code, der wiederholt läuft, solange A true ergibt  
3 }
```

## Beispiel:

```
1 public static void countDown(int start){  
2     while(start > 0){  
3         System.out.println(start);  
4         start--;  
5     },  
6     System.out.println("GO!");  
7 }
```

Output countDown(3):

No. lap	S	start > 0 ?	op
1	3	true	3.
2	2	true	2.
3	1	true	1.
4	0	false.	GO!

# while-Schleife

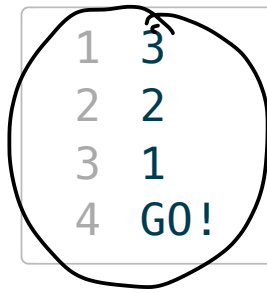
## Allgemeine Form:

```
1 while(Bedingung A){  
2     // Code, der wiederholt läuft, solange A true ergibt  
3 }
```

## Beispiel:

```
1 public static void countDown(int start){  
2     while(start > 0){  
3         System.out.println(start);  
4         start--;  
5     }  
6     System.out.println("GO!");  
7 }
```

Output countDown(3):



```
1 3  
2 2  
3 1  
4 GO!
```

# do-while-Schleife

## Allgemeine Form:

```
1 do {  
2     // Code, der wiederholt läuft, solange A true ergibt  
3 }while(Bedingung A);
```

## Beispiel:

```
1 public static void passwordCheck(){  
2     String password;  
3     do {  
4         password = getInput();  
5     } while (!isPasswordCorrect(password));  
6 }
```

Hier wird der Code in den geschweiften Klammern garantiert einmal ausgeführt.



# for-Schleife

## Allgemeine Form:

A: (Deklaration und) Initialisierung einer (oder mehrerer) Zählervariablen

C: Veränderung (z.B. Hochzählen) der Zählervariablen

```
1 for(Anweisung A ; Bedingung B ; Anweisung C){  
2     // Code, der wiederholt läuft, solange B true ergibt  
3 }
```

A

```
for( ; B ; C){  
    //Code . . .  
}
```

## Beispiel:

```
1 public static void countDown(int start){  
2     for(int i=start; i>0; i--){  
3         System.out.println(i);  
4     }  
5     System.out.println("GO!");  
6 }
```

Output countDown(3):

Schleife	i	i>0?	output
1	3	true	3
2	2	true	2
3	1	true	1
4	0	false	GO!

# for-Schleife

## Allgemeine Form:

A: (Deklaration und) Initialisierung einer (oder mehrerer) Zählervariablen

C: Veränderung (z.B. Hochzählen) der Zählervariablen

```
1 for(Anweisung A ; Bedingung B ; Anweisung C){  
2     // Code, der wiederholt läuft, solange B true ergibt  
3 }
```

## Beispiel:

```
1 public static void countDown(int start){  
2     for(int i=start; i>0; i--){  
3         System.out.println(i);  
4     }  
5     System.out.println("GO!");  
6 }
```

Output countDown(3):

```
1 3  
2 2  
3 1  
4 GO!
```

## Unterbrechen. Schleif :

- keyword: break
- mit if - else / If Anweisungen. in While-, for-Schleife verwendet.

- Beispiels: max. Primzahl innerhalb n..

```
int n = 20;  
while ( n > 0 ) {  
    if ( isPrima(n) ) {  
        break;  
    }  
    n--;  
}  
System.out.println (n);
```

- Weitere Anwendung von "break" : später.

# W04P01 - Kontrollstrukturen

Bearbeite nun die Aufgabe [W04P01 - Kontrollstrukturen](#)

# Geht das einfacher?

```
1  /**
2   * @param month Number of the month (1-12)
3   */
4  public static int daysInMonth(int month) {
5      int days;
6      if (month == 4 || month == 6
7          || month == 9 || month == 11) {
8          days = 30;
9      } else if (month == 2) {
10         days = 28;
11     } else {
12         days = 31;
13     }
14     return days;
15 }
```

# Geht das einfacher?

```
1  /**
2  * @param month Number of the month (1-12)
3  */
4  public static int daysInMonth(int month) {
5      int days;
6      if (month == 4 || month == 6
7          || month == 9 || month == 11) {
8          days = 30;
9      } else if (month == 2) {
10         days = 28;
11     } else {
12         days = 31;
13     }
14     return days;
15 }
```

Ja, mit  
switch!

- Variable oder Ausdruck  
auf mehrere mögliche Werte  
=> viele Bedingungen auf  
die gleiche Variable

# Switch

```
1  /**
2  * @param month Number of the month (1-12)
3  */
4  public static int daysInMonth(int month) {
5      int days;
6      switch (month) {
7          case 4, 6, 9, 11:
8              days = 30;
9          case 2:
10             days = 28;
11         default:
12             days = 31;
13     }
14     return days;
15 }
```

## Output

System.out.println(daysInMonth(2));

# Switch

```
1  /**
2  * @param month Number of the month (1-12)
3  */
4  public static int daysInMonth(int month) {
5      int days;
6      switch (month) {
7          case 4, 6, 9, 11:
8              days = 30;
9          case 2:
10             days = 28;
11         default:
12             days = 31;
13     }
14     return days;
15 }
```

## Output

System.out.println(daysInMonth(2));

1 31

Warum?



# Switch

```
1 /**
2  * @param month Number of the month (1-12)
3  */
4  public static int daysInMonth(int month) {
5      int days;
6      switch (month) {
7          case 4, 6, 9, 11:
8              days = 30;
9              break;
10         case 2:
11             days = 28;
12             break;
13         default:
14             days = 31;
15             break;
16     }
17     return days;
18 }
```

## Output

System.out.println(daysInMonth(2));

1 28

Breaks sind wichtig!

Ohne break → Fall-Through

# Switch

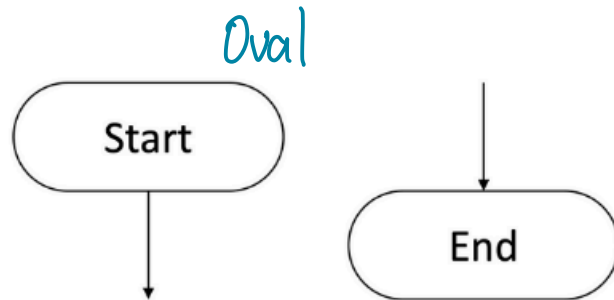
```
1  /**
2  * @param month Number of the month (1-12)
3  */
4  public static int daysInMonth(int month) {
5      int days = switch (month) {
6          case 4, 6, 9, 11 -> 30;
7          case 2 -> 28;
8          default -> 31;
9      };
10     return days;
11 }
```

Und noch kompakter

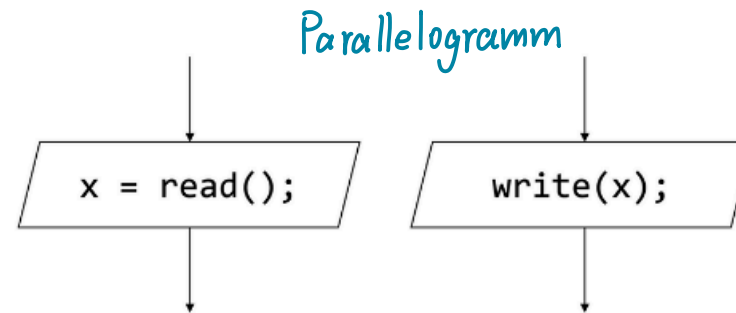
# W04P02 - Taschenrechner

Bearbeite nun die Aufgabe [W04P02 - Taschenrechner](#)

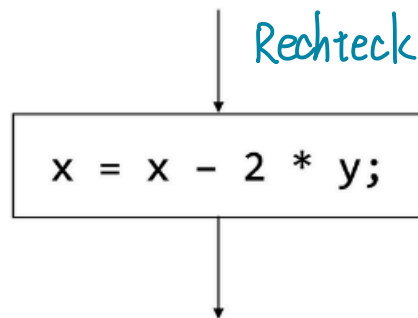
# Kontrollflussdiagramme Eidl Klausur. ☆



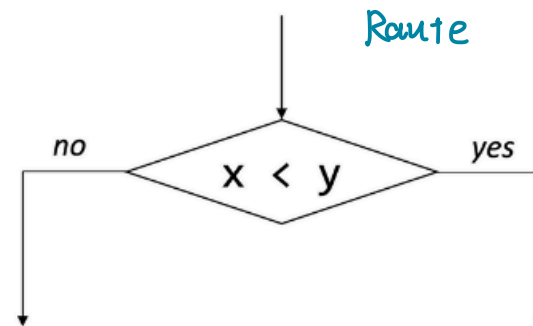
Anfang bzw. Ende des Programms



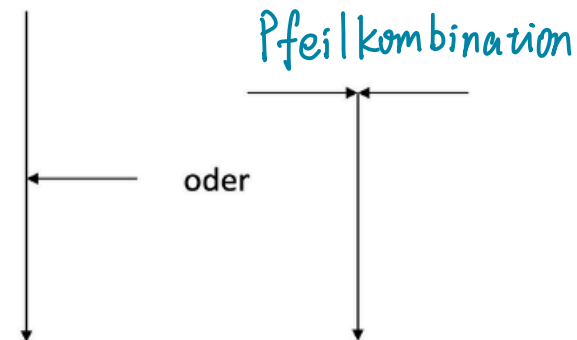
Ein- und Ausgabe



Zuweisung



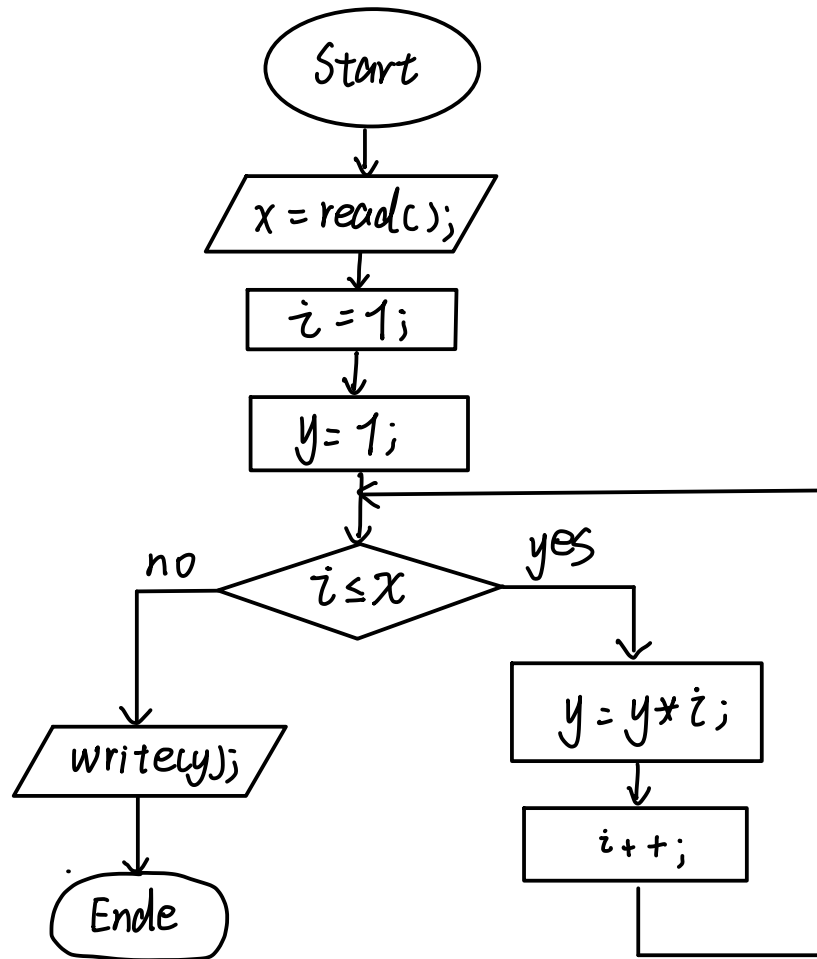
Verzweigung



Zusammenlauf

# W04P03 - Kontrollflussdiagramm

Bearbeite nun die Aufgabe [W04P03 - Kontrollflussdiagramm](#)



# W04P03 - Kontrollflussdiagramm

Bearbeite nun die Aufgabe [W04P03 - Kontrollflussdiagramm](#)

# Slide und Code von Aufgaben

**GitHub** : Code aller P-Aufgaben

- <https://github.com/SikiaLi/PGDP.git>