LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Lehrstuhl Bioinformatik • Konstantin Pelz

Erste Java-Programme

(Scopes und Rekursion)

Tutorium Bioinformatik
(WS 18/19)

Konstantin: Konstantin.pelz@campus.lmu.de
Homepage: https://bioinformatik-muenchen.com/studium/propaedeutikum-programmierung-in-der-bioinformatik/

Wiederholung

Die Klasse Hello World als source code:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}
```

Ein Java Programm startet immer mit einer Main-Methode.

Die Main-Methode hat immer den gleichen Kopf:
public static void main(String[] args){ … }

In args sind alle Argumente die in der Console mitgegeben wurden.

java –jar HelloWorld.jar 20 ich bin “eine Eingabe“
args[0] = “20“ args[1] = “ich“ args[2] = “bin“ args[3] = “eine Eingabe“
Alle Eingaben werden als String gespeichert.

System.out.println(“meine Ausgabe“); - Gibt den Text über die Konsole aus
System = Klasse, out = statische Variable, println() = Methode

Wiederholung

Datentypen und Umwandeln von Datentypen

Buchstaben: char (Character)
Wörter: String
Zahlen aus N: int (Integer)
Kommazahlen: float, double

```
String s1 = "5";
String s2 = "10.5";

int f1 = Integer.parseInt(s1);
float f2 = Float.parseFloat(s2);
```

```
String s = "Bioinformatik";
int len = s.length();
char a = s.charAt(0);
char b = s.charAt(len-1);
String c = s.substring(2, 5);
System.out.println(a+c+b);
```

Länge des Strings

Character an einer Stelle im String

Substring von / bis von inklusive bis exklusive

Selbst überlegen ;)

Wiederholung

for - Schleife:

Allgemein:

```
for ( Initialisierung; Boolscher Ausdruck; Iteration){
  Anweisung;
}
```

Code:

```
for ( int i = 0; i < 10; i++){
  System.out.println(i);
}
```

```
0
1
2
3
4
5
6
7
8
9
```

while - Schleife:

Allgemein:

```
while (Boolscher Ausdruck){
  Anweisung;
}
```

Code:

```
int i = 0;

while(i<10){
    System.out.println(i);
    i++;
}
```

```
0
1
2
3
4
5
6
7
8
9
```

Wiederholung

Wie sieht eine Methode allgemein aus?

```
public static Rückgabewert methodenname(Übergabeparameter){
  // Anweisungen
  return Rückgabewert;
}
```

Wo im Code stehen Methoden?

```
1  public class HelloWorld {
2
3    public static Rückgabewert methodenname(Übergabeparameter){
4
5      return Rückgabewert;
6
7    }
8
9    public static void main(String[] args){
10
11     |
12
13   }
14 }
```

Was sind Rückgabewerte?

- int, double, String, char, ... → Alle Datentypen
- void → Dann gibt die Methode nichts zurück

```
public static void nutzlos(){
  return;
}
```

Methoden

Methoden können alle möglichen Typen als Rückgabewert besitzen

```
public class Student {

    public int getMatrikelnr() {
        return matrikelnr;
    }

    public void setMatrikelnr(int matrikelnr) {
        this.matrikelnr=matrikelnr;
    }
```

int als Rückgabewert

void = kein Rückgabewert

Eine Methode mit Rückgabewert muss mit return einen Wert überliefern

Sobald der return Befehl ausgeführt wird stoppt die Methode

Methode die immer zuerst aufgerufen wird: Main Methode (public static void main(String[] args))

Wiederholung

Imperative Programmierung

```
3  public class Rechnen {
4      public static void main(String[] args) {
5          double a = Double.parseDouble(args[0]);
6          String rechen_zeichen = args[1];
7          double b = Double.parseDouble(args[2]);
8          double c = 2.0;
9          if(rechen_zeichen.equals("+")){
10             System.out.println(addition(a,b));
11         } else if (rechen_zeichen.equals("/")){
12             System.out.println(division(a, b));
13         }
14         a = addition(a,b);
15         System.out.println(addition(a,c));
16     }
17
18     public static double addition(double a, double b) {
19         return a + b;
20     }
21
22     public static double division(double a, double b) {
23         return a/b;
24     }
25 }
```

Static:
Von einer static-Methode aus kann man nur wieder eine static-Methode (direkt) aufrufen. Um eine nicht-statische Methode aufzurufen, müsste man ein Objekt der jeweiligen Klasse erzeugen

Ausgabe: bei java –jar Rechnen.jar 1.0 + 4.0

5.0
7.0

Scopes

Variablen haben einen Geltungsbereich („scope“):
Sie gelten nur in dem Block (= zwischen { }), in dem / für den sie deklariert sind.

```
6    public static void printSum(int a, int b) {
7        System.out.println(a+b);
8    }
```

a/b gibt es nur in der Methode

```
16       for (int i=0; i<args.length; i++) {
17           System.out.println(args[i]);
18       }
19
20       int i=0;
21       while (i<args.length) {
22           System.out.println(args[i]);
23           i++;
24       }
```

i gibt es nur in der Schleife

i gibt es auch nach der Schleife!

```
26       for (int i=0; i<10; i++) {
27
28           boolean a = i%2==0; // % ist modulo Operator
29           if (a) {
30               System.out.println(i);
31               a = i%2==0;
32               boolean a = i%2==0;
33           }
34
35           if (i>4) {
36               int test = 3;
37           }
38
39           System.out.println(test);
40       }
```

i gilt auch in verschachteltem Block
a kann neu gesetzt werden
… aber nicht neu deklariert

test ist nur im ‚if‘ Block gültig
…aber nicht mehr danach

Scopes

Variablen haben einen Geltungsbereich („scope“):
Sie gelten nur in dem Block (= zwischen { }), in dem / für den sie deklariert sind.

```
1  package v006;
2
3  public class Taschenrechner {
4      private double a;
5      public double b;
6
7      public Taschenrechner(double a, double b){
8          this.a = a;
9          this.b = b;
10         double c = 2.0;
11     }
12
13     public double addition(){
14         return a + b;
15     }
16
17     public double division(){
18         return a/c;
19     }
20 }
```

a gibt es in der ganzen Klasse
b gibt es in der Klasse und kann durch das Objekt direkt geholt werden, ohne getter.

c gibt es nur in der Methode

c gibt es nur im Konstruktor

ScopeTest

```
3  public class ScopeTest {
4
5
6      private static int a = 2;
7      private int b = 3;
8      private int c;
9
10     public ScopeTest(int a) {
11         if (a<2) {
12             int b = 1;
13             ScopeTest.a = 5;
14         }
15         else {
16             b=2;
17             ScopeTest.a = 7;
18         }
19         c = b;
20     }
21
22
23     public static void main(String[] args) {
24
25         int b = 6;
26
27         ScopeTest d = new ScopeTest(1);
28         System.out.println(d.c);
29         System.out.println(a);
30
31         ScopeTest c = new ScopeTest(b);
32         System.out.println(c.c);
33         System.out.println(a);
34     }
35
36 }
```

Was wird ausgegeben?

3
5

2
7

Rekursion

Iterative Programmierung = Benutzen von Schleifen für Probleme

Rekursive Programmierung = Funktion, die sich selbst aufruft um Problem zu lösen

Methode ruft sich selbst auf, und wird so lange reduziert bis ein Endzustand (Terminalfall) eintritt

Sobald der Terminalfall auftritt werden die Teilschritte aufgelöst und das Ergebnis berechnet

Beispiel: Fibonacci Folge berechnen

$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

n	f_n	n	f_n
1	1	11	89
2	1	12	144
3	2	13	233
4	3	14	377
5	5	15	610
6	8	16	987
7	13	17	1 597
8	21	18	2 584
9	34	19	4 181
10	55	20	6 765

Rekursion

Beispiel: Fibonacci Folge berechnen

$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Iterativer Ansatz

```java
public class Fibonacci {

    public static int iterativFibonacci(int x){
        if(x == 1 || x == 2){
            return 1;
        }
        int fn = 0;
        int fn_minus_1 = 1, fn_minus_2 = 1;
        for(int index = 3; index <= x; index++){
            fn = fn_minus_1 + fn_minus_2;
            fn_minus_2 = fn_minus_1;
            fn_minus_1 = fn;
        }
        return fn;
    }

    public static void main(String[] args) {
        int ergebnis = iterativFibonacci(6);
        System.out.println(ergebnis);
    }
}
```

Rekursiver Ansatz

```java
public class Fibonacci {

    public static int recursiveFibonacci(int x){
        if(x == 1 || x == 2){
            return 1;
        }
        return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
    }

    public static void main(String[] args) {
        int ergebnis = recursiveFibonacci(6);
        System.out.println(ergebnis);
    }
}
```

Iteration

Beispiel: Fibonacci Folge berechnen

$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Iterativer Ansatz

```
3  public class Fibonacci {
4
5      public static int iterativFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          int fn = 0;
10         int fn_minus_1 = 1, fn_minus_2 = 1;
11         for(int index = 3; index <= x; index++){
12             fn = fn_minus_1 + fn_minus_2;
13             fn_minus_2 = fn_minus_1;
14             fn_minus_1 = fn;
15         }
16         return fn;
17     }
18
19     public static void main(String[] args) {
20         int ergebnis = iterativFibonacci(6);
21         System.out.println(ergebnis);
22     }
23 }
```

Programmablauf

fn	fn_minus_1	fn_minus_2	index
0	1	1	
2	2	1	3
3	3	2	4
5	5	3	5
8	8	5	6

Rekursion

Beispiel: Fibonacci Folge berechnen

$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf

Rekursion

Beispiel: Fibonacci Folge berechnen

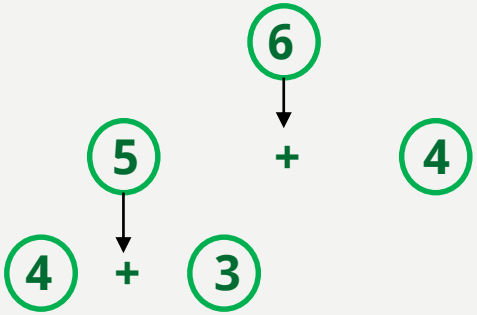
$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

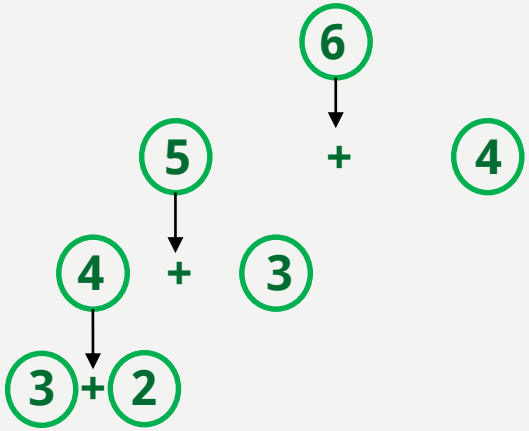
$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

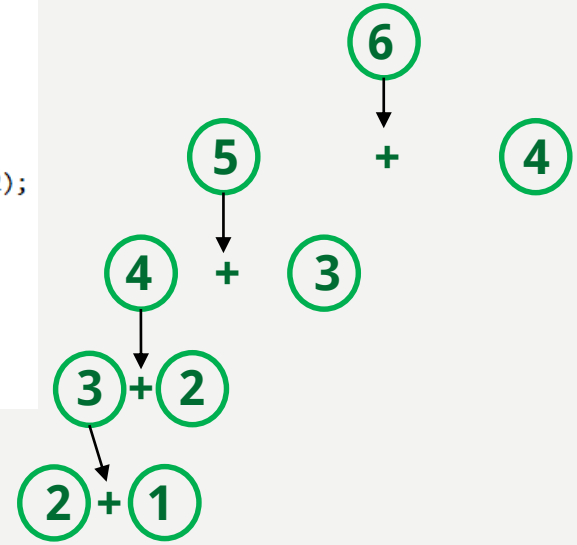
$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

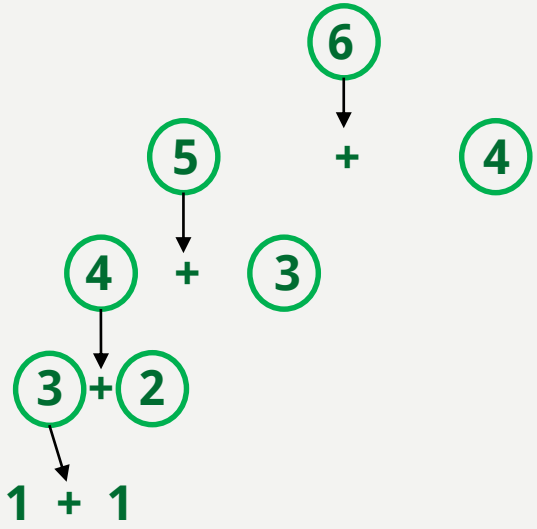
$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

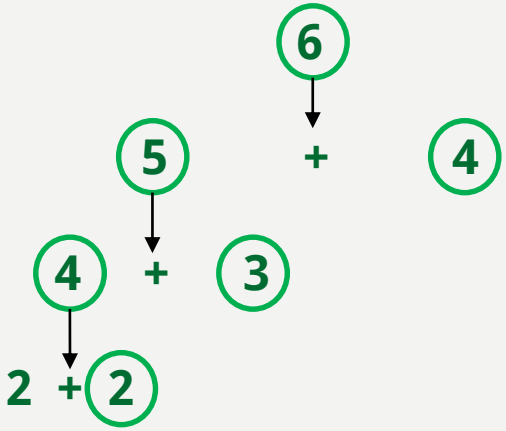
$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

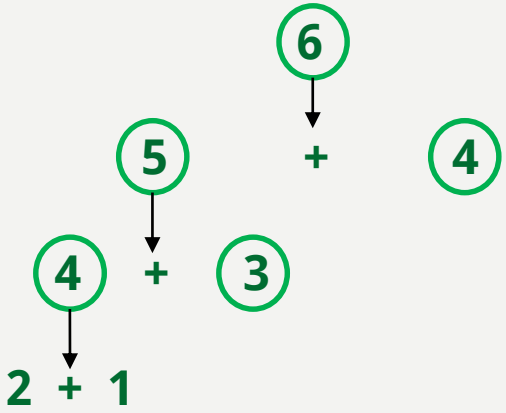
$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

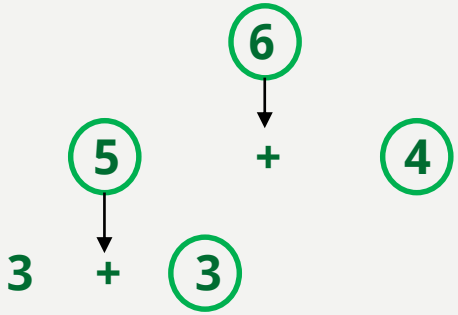
$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

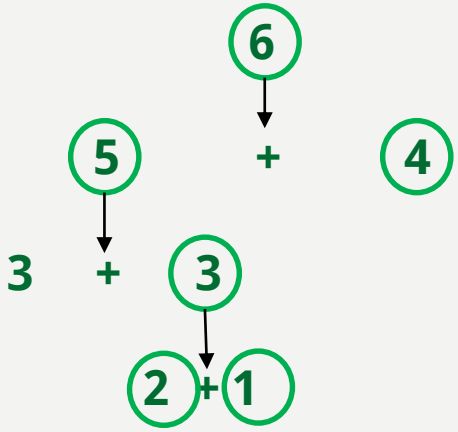
$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

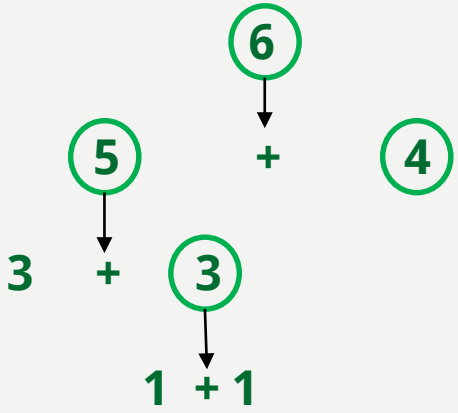
$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

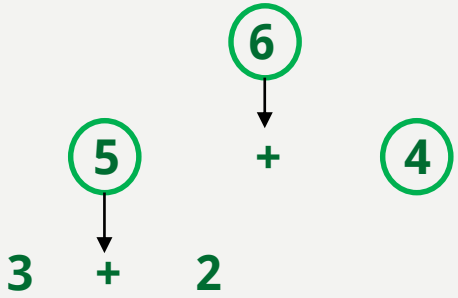
$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf

Rekursion

Beispiel: Fibonacci Folge berechnen

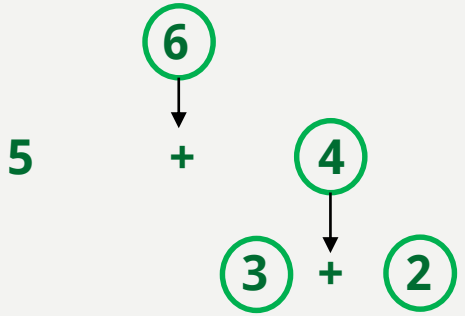
$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

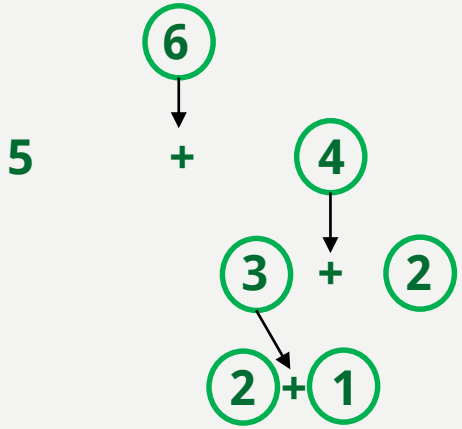
$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

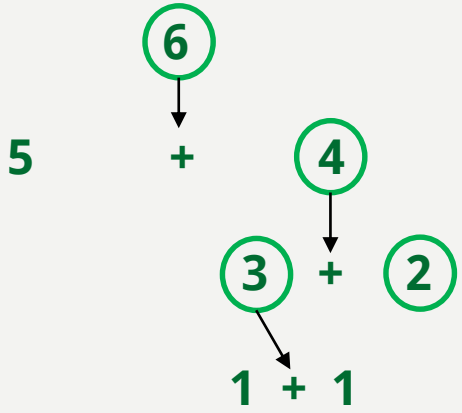
$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

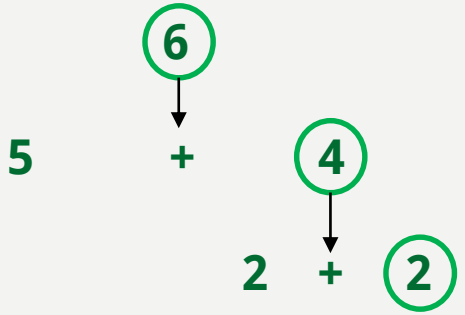
$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

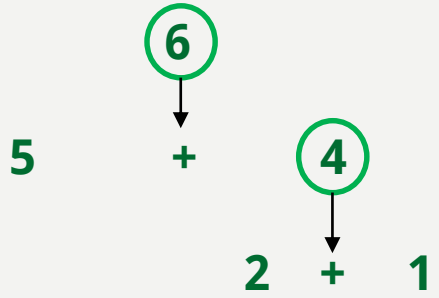
$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf



Rekursion

Beispiel: Fibonacci Folge berechnen

$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf

Rekursion

Beispiel: Fibonacci Folge berechnen

$$f_n = f_{n-1} + f_{n-2} \; für \; n > 2$$

und den Anfangswerten: $f_1 = f_2 = 1$

Rekursiver Ansatz

```
3  public class Fibonacci {
4
5      public static int recursiveFibonacci(int x){
6          if(x == 1 || x == 2){
7              return 1;
8          }
9          return recursiveFibonacci(x-1) + recursiveFibonacci(x-2);
10     }
11
12     public static void main(String[] args) {
13         int ergebnis = recursiveFibonacci(6);
14         System.out.println(ergebnis);
15     }
16 }
```

Programmablauf

8