

# Übungsstunde 5

Einführung in die Programmierung

# Nachbesprechung Übung 4

# Aufgabe 1: Sieb des Eratosthenes

```
//Initialisiere das Array sieb und setze alle Werte auf true
boolean[] sieb = new boolean[limit + 1];
for(int i = 2; i < sieb.length; i++){
    sieb[i] = true;
}

//Markiere die Vielfachen der Primzahlen als false
for(int i = 2; i < limit; i++) {
    if(sieb[i]) {
        for(int vielfaches = 2 * i; vielfaches <= limit; vielfaches += i) {
            sieb[vielfaches] = false;
        }
    }
}

//Zähle die Primzahlen
int primzahlen = 0;
for(int i = 2; i < sieb.length; i++){
    if(sieb[i]){
        primzahlen++;
    }
}
```

# Aufgabe 2: Newton-Raphson

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    int z = scanner.nextInt();  
    double root = squareRoot(z);  
    System.out.println("Wurzel von " + z + " = " + root);  
}
```

```
static double squareRoot(int z) {  
    double c = z;  
    double t = c / 2.0;  
    double eps = 0.000000000001;  
  
    while (Math.abs(t * t - c) > eps) {  
        t = (c / t + t) / 2.0;  
    }  
  
    return t;  
}
```

Double-Division  
erzwingen

$\text{Math.abs}(x) = |x|$

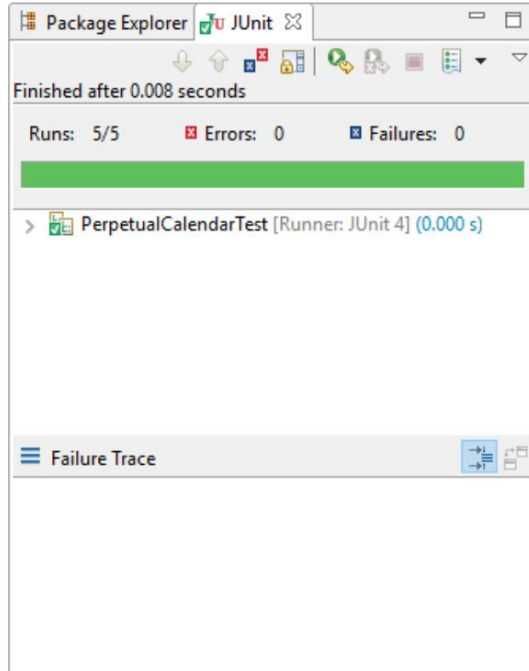
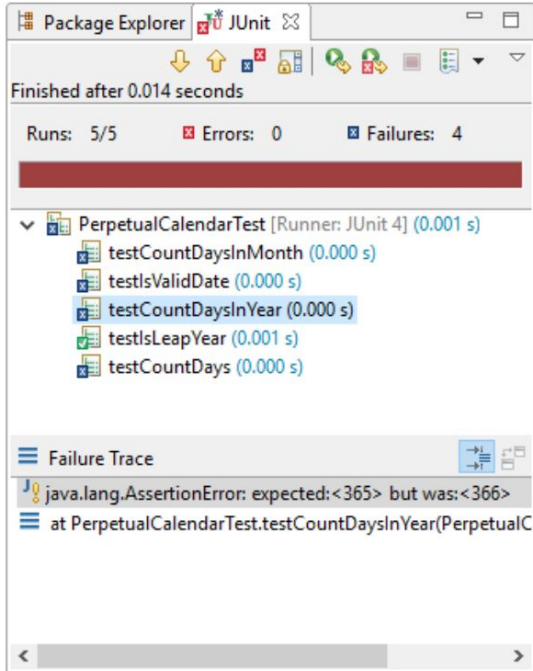
# Aufgabe 3: Loop Invariante

Gegeben sind die Precondition und Postcondition für das folgende Programm

```
public int compute(int a, int b) {  
    // Precondition:  a >= 0  
    int x;  
    int res;  
  
    x = 0;  
    res = b;  
  
    // Loop Invariante: x <= a && res == b - x  
    while (x < a) {  
        res = res - 1;  
        x = x + 1;  
    }  
    // Postcondition:  res == b - a  
    return res;  
}
```

Was ist die Loop Invariante?

# Aufgabe 4: Testen mit JUnit



# Aufgabe 5: Arrays

In dieser Aufgabe implementieren Sie Methoden, welche Arrays verwenden.

1. Implementieren Sie die Methode `ArrayUtil.zeroInsert(int[] x)`, das heisst die Methode `zeroInsert` in der Klasse `ArrayUtil`. Die Methode nimmt einen Array `x` als Argument und gibt einen Array zurück. Der zurückgegebene Array soll die gleichen Werte wie `x` haben, ausser: Wenn eine positive Zahl direkt auf eine negative Zahl folgt oder wenn eine negative Zahl direkt auf eine positive Zahl folgt, dann wird dazwischen eine 0 eingefügt.

Beispiele:

- Wenn `x` gleich `[3, 4, 5]` ist, dann wird `[3, 4, 5]` zurückgegeben.
- Wenn `x` gleich `[3, 0, -5]` ist, dann wird `[3, 0, -5]` zurückgegeben.
- Wenn `x` gleich `[-3, 4, 6, 9, -8]` ist, dann wird `[-3, 0, 4, 6, 9, 0, -8]` zurückgegeben.

Versuchen Sie die Methode rekursiv zu implementieren.

2. Implementieren Sie die Methode `ArrayUtil.tenFollows(int[] x, int index)`. Die Methode gibt einen Boolean zurück. Die Methode soll `true` zurückgeben, wenn im Array `x` ab Index `index` der zehnfache Wert einer Zahl `n` direkt der Zahl `n` folgt. Ansonsten soll die Methode `false` zurückgeben.

Beispiele:

- `tenFollows([1, 2, 20], 0)` gibt `true` zurück.
- `tenFollows([1, 2, 7, 20], 0)` gibt `false` zurück.
- `tenFollows([3, 30], 0)` gibt `true` zurück.
- `tenFollows([3], 0)` gibt `false` zurück.
- `tenFollows([1, 2, 20, 5], 1)` gibt `true` zurück.
- `tenFollows([1, 2, 20, 5], 2)` gibt `false` zurück.

# Aufgabe 6: Bonusaufgabe

Erreichte Punktzahl nach der Korrektur direkt per Git

**Lösungsvorschlag von Studenten**



# File Input

# Lesen aus einer Datei

- **Das können Sie schon!**

```
import java.io.File;  
import java.util.Scanner;  
  
File file = new File("input.txt");  
Scanner scanner = new Scanner(file);  
int zahl = scanner.nextInt();
```

- **Oder kürzer:**

```
Scanner scanner = new Scanner(new File("input.txt"));
```

```
4
5 public class FileIo {
6     public static void main(String[] args)
7         throws FileNotFoundException {
8         File file = new File("example.txt");
9         Scanner scanner = new Scanner(file);
10        int zahl = scanner.nextInt();
11    }
12 }
13
```



- **Jetzt ist der Compiler zufrieden!**
  - Exceptions, welche die **main**-Methode wirft, werden vom System gefangen (und produzieren einen Stacktrace)

# Scanner Methoden

Method	Description
nextInt()	reads an int from the user and returns it
nextDouble()	reads a double from the user
next()	reads a one-word String from the user
nextLine()	reads a one- <i>line</i> String from the user

# Scanner Methoden

Method	Description
<code>hasNext()</code>	returns <code>true</code> if there is a next token
<code>hasNextInt()</code>	returns <code>true</code> if there is a next token and it can be read as an <code>int</code>
<code>hasNextDouble()</code>	returns <code>true</code> if there is a next token and it can be read as a <code>double</code>

- **Diese Scanner Methoden prüfen ob gültiger Input vorliegt**
- **Konsumieren *keinen* Input – liefern nur Information über das nächste Token**
  - **Praktisch um vorauszusehen was für Input kommt um so Laufzeitfehler zu verhindern.**

# Gebrauch der hasNext Methode

- Um nicht über das Ende einer Datei hinauszulesen:

```
Scanner input = new Scanner(new File("example.txt"));  
    if (input.hasNext()) {  
        String token = input.next();    // will not crash!  
        System.out.println("next token is " + token);  
    }
```

# Hybrider Ansatz (bei längeren Files)

Zuerst: Zeile lesen

→ Gibt einen String

Dann diesen String mit einem neuen Scanner durchgehen

# Zeilen-basierte Scanner Aktivität

Method	Description
<code>nextLine()</code>	returns next entire line of input (from cursor to <code>\n</code> )
<code>hasNextLine()</code>	returns true if there are any more lines of input to read (always true for console input)

```
Scanner input = new Scanner(new File("fileName"));
while (input.hasNextLine()) {
    String line = input.nextLine();

    // bearbeiten dieser Zeile
}
```



```
// Counts the words on each line of a file
Scanner input = new Scanner(new File("input.txt"));
while (input.hasNextLine()) {
    String line = input.nextLine();
    Scanner lineScan = new Scanner(line);
    int count = 0;

    // process the contents of this line
    while (lineScan.hasNext()) {
        String word = lineScan.next();
        count++;
    }
    System.out.println("Line has " + count + " words");
}
```

# File Output

# PrintStream-Klasse

- Wie **File** aus **java.io**; erlaubt es Daten auszugeben (z.B. in eine **File**)
- Alle Methoden, die wir von **System.out** kennen, funktionieren auch für **PrintStream**!
  - D.h. **print()**, **println()**

- **Syntax:**

```
File file = new File("example.txt");  
PrintStream output = new PrintStream(file);
```

```
PrintStream output = new PrintStream(new File("example.txt"));
```

# Statt in Konsole, in Datei

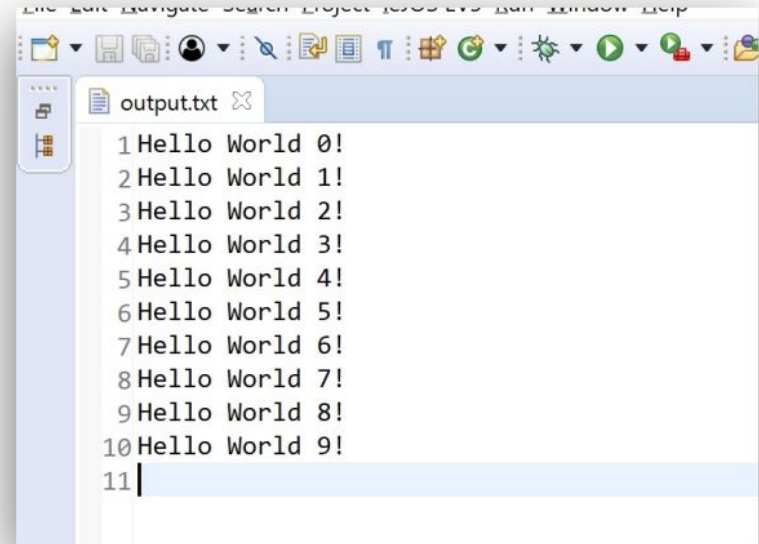
```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintStream;

File file = new File("output.txt");
PrintStream fileOutput = new PrintStream(file);

for (int i = 0; i < 10; i++) {
    fileOutput.print("Hello World ");
    fileOutput.print(i);
    fileOutput.println("!");
}
```

- Wenn Datei nicht existiert, wird sie erstellt
- *Wenn sie existiert, wird sie **überschrieben!***

- Ausgabe ansehen:
  - In Eclipse auf Projekt rechtsklicken → *Refresh*
  - Datei *output.txt* sollte erscheinen (im Projekt, nicht im Java src Folder)
  - Doppelklick



```
1 Hello World 0!  
2 Hello World 1!  
3 Hello World 2!  
4 Hello World 3!  
5 Hello World 4!  
6 Hello World 5!  
7 Hello World 6!  
8 Hello World 7!  
9 Hello World 8!  
10 Hello World 9!  
11 |
```

- **Arbeiten Sie nicht mit der selben Datei als Input (Lesen durch Scanner)  
und Output (Schreiben mit `PrintStream`) zur selben Zeit!**
  - **Sie überschreiben die Input Datei und ersetzen sie durch eine leere Datei (0 Byte).**

GUI

# GUI

- GUI: Graphical User Interface
- Grafische Benutzeroberfläche/-schnittstelle

**Input/Output in einem *Fenster*, d.h. grafisch**





# Einfaches GUI in Java (nicht in JRE)

## Window-Klasse

- Einfache Fenster ohne Steuerelemente oder anpassbares Layout

```
import gui.Window;

public class Empty {
    public static void main(String[] args) {
        Window window = new Window("Empty", 500, 300);
        window.open();
        window.waitUntilClosed();
    }
}
```

```
new Window("Empty", 500, 300);
```

Fenstertitel

Breite

Höhe

```
new Window("Empty", 500, 300);
```

Fenstertitel

Breite

Höhe

```
window.setColor(255, 0, 0);
```

```
new Window("Empty", 500, 300);
```

Fenstertitel

Breite

Höhe

```
window.setColor(255, 0, 0);
```

```
Window window = new Window("Square", 500, 300);
```

Breite

Höhe

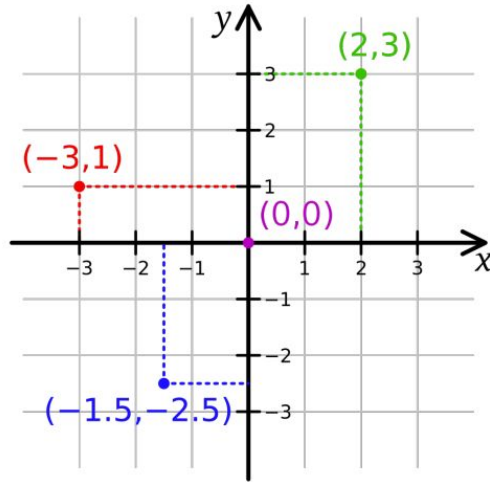
```
window.fillRect(20, 20, 50, 50);
```

x-Koordinate

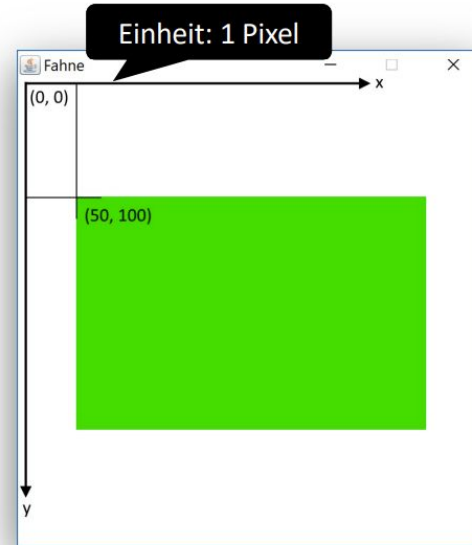
y-Koordinate

```
window.open();
```

# Fenster-Koordinatensystem



■ Mathematik



■ GUIs

# Einfache Zeichen-Methoden

```
fillRect(double x, double y, double width, double height)
```

Zeichnet ein (ausgefülltes) Rechteck der Grösse `width` × `height`, welches die linke obere Ecke beim Punkt `(x, y)` hat

```
fillCircle(double x, double y, double radius)
```

Zeichnet einen (ausgefüllten) Kreis mit Radius `radius`, welches den Mittelpunkt beim Punkt `(x, y)` hat

```
fillOval(double x, double y, double width, double height)
```

...

- **Trick:** `fillRect(x, y, 1, 1)` zeichnet einzelnen Pixel!

# Ändern des Fensterinhalts??

- Bisher: Fenster erstellen, zeichnen, anzeigen.
- Was, wenn wir Fensterinhalt ändern möchten?

```
Window window = new Window("Fenster", 500, 300);
```



```
// Zeichnen
```

```
window.open();
```

```
window.waitUntilClosed();
```

- Geht nicht, Zeichnung muss «veröffentlicht» werden!

# Plotten, animiert!

```
int width = 500, height = 300;
Window window = new Window("Plotter", width, height);
window.open();

for (int i = 0; i < width; i++) {
    double x = 0.05 * i;
    double y = Math.sin(x);
    window.fillRect(i, y * height/4 + height/2, 1, 1);
    window.refresh(10);
}

window.waitUntilClosed();
```



# Plotten, animiert!

```
int width = 500, height = 300;
Window window = new Window("Plotter", width, height);
window.open();

for (int i = 0; i < width; i++) {
    double x = 0.05 * i;
    double y = Math.sin(x);
    window.fillRect(i, y * height/4 + height/2, 1, 1);
    window.refreshAndClear(10);
}

window.waitUntilClosed();
```

# Interaktion

```
int width = 500, height = 300;
Window window = new Window("Moving", width, height);
window.open();

int x = width/2;
while (window.isOpen()) {
    if (window.isKeyPressed("left")) { x--; }
    if (window.isKeyPressed("right")) { x++; }

    window.fillCircle(x, height/2, 10);
    window.refreshAndClear(10);
}
```

# Vorbesprechung Übung 5

# Aufgabe 1: Wörter Raten

Das Programm “WoerterRaten.java” enthält Fragmente eines Rate-Spiels, welches Sie vervollständigen sollen. In dem Spiel wählt der Computer zufällig ein Wort  $w$  aus einer Liste aus und der Mensch muss versuchen, das Wort zu erraten. In jeder Runde kann der Mensch eine Zeichenfolge  $z$  (welche einen oder mehrere Buchstaben enthält) eingeben und der Computer gibt einen Hinweis dazu. Folgende Hinweise sind möglich:

1.  $w$  beginnt mit  $z$
2.  $w$  endet mit  $z$
3.  $w$  enthält  $z$
4.  $w$  enthält nicht  $z$

Tipp? **e**

Das Wort enthält nicht "e"!

Tipp? **a**

Das Wort endet mit "a"!

Tipp? **j**

Das Wort beginnt mit "j"!

Tipp? **v**

Das Wort enthält "v"!

Tipp? **java**

Das Wort ist "java"!

Glückwunsch, du hast nur 5 Versuche benötigt!

# Aus Textdatei lesen

*woerter.txt*

```
8
wort
blasinstrument
computer
schlange
java
programmieren
welt
sugus
```

*Code zum Einlesen der Wörter:*

```
Scanner scanner = new Scanner(new File("woerter.txt"));
String[] woerter = new String[scanner.nextInt()];
for(int i = 0; i < woerter.length; i++) {
    woerter[i] = scanner.next();
}
```

Liest das nächste Wort

Anzahl Wörter = 8

## Absoluter Pfad

- beginnt mit `c:\...` unter Windows oder `/...` unter Linux/macOS

## Relativer Pfad

- ist relativ zum aktuellen Verzeichnis ("working directory") des Programms (bei uns der Projektordner)

# String-Vergleiche

```
String [] buchstaben = {"n", "o"};  
String [] array = {"no"};
```

```
// wort = "no"  
String wort = buchstaben[0] + buchstaben[1];
```

```
System.out.println((wort == array)); ①
```

```
System.out.println((wort == array[0])); ②
```

```
System.out.println((wort.equals(array))); ③
```

```
System.out.println(wort.equals(array[0])); ④
```

```
System.out.println((wort.equals("no"))); ⑤
```

# Aufgabe 2: Datenanalyse

In dieser Aufgabe werden Sie die Körpergrößen einiger Personen analysieren, welche für eine Studie<sup>1</sup> in Kalifornien erhoben wurden. Im Programm “DatenAnalyse.java” sind schon ein paar (leere) Methoden zu Ihrer Hilfe vorgegeben.

- b) Führen Sie als nächstes eine einfache Analyse der Daten durch, indem Sie das Minimum, das Maximum und den Durchschnitt und ausserdem die Anzahl der Körpergrößen ausgeben. Füllen Sie dazu die Methode `einfacheAnalyse()` aus. Beachten Sie folgende Methoden: `Math.min()` und `Math.max()`.

## Beispiel:

```
Anzahl Daten: 26
Minimum: 137 cm
Maximum: 202 cm
Durchschnitt: 173 cm
```

# Histogramm

- c) Die drei Werte, die Sie in b) berechnet haben, sagen nicht viel über die Daten aus. Um die Daten besser zu verstehen, soll Ihr Programm ein **Histogramm** berechnen und ausgeben. Sie können das Histogramm in Text-Form auf der Konsole ausgeben, oder auch mit der Window-Klasse auf ein Fenster zeichnen. Die Textausgabe könnte ungefähr so aussehen:

*groessen.txt*

**26**

168

190

174

170

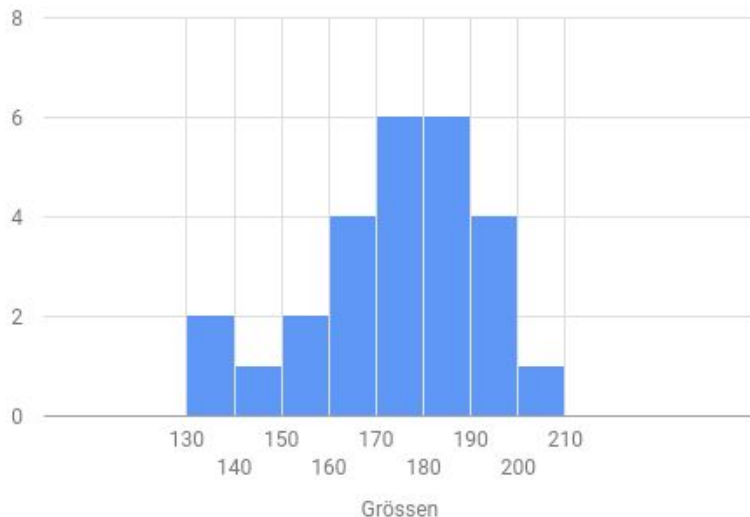
183

145

202

...

Histogram of Grössen



Wie viele Klassen soll das Histogramm enthalten? **10**

[100,115)

[115,130) |

[130,145) ||

[145,160) |||||

[160,175) |||||

[175,190) |||||

[190,205) |||||

[205,220)

[220,235)

[235,250)



# Aufgabe 3: Wellen (Bonus!)

Sei  $M$  eine  $n \times n$  Matrix mit  $n > 0$ . Ein Index  $x = (i, j)$  von  $M$  ist ein Paar von einer Zahl  $i$  für die Zeile und einer Zahl  $j$  für die Spalte mit  $i, j \in \{0, \dots, n-1\}^2$ , welche eine Position in der Matrix  $M$  beschreibt, wobei  $M_x$  der Wert von  $M$  beim Index  $x$  ist. Zum Beispiel, die Matrix in der Figur 1 hat Wert 6 beim Index  $(2, 0)$ . Wir definieren ein Set von Startindexen (START), Endindexen (END), und Nachbarindexen (NEIGHBOR <sub>$x$</sub>  für einen Index  $x$ ) wie folgt:

$$\text{START} = \{(i, j) \mid i = 0 \vee j = 0\} \cap \{0, \dots, n-1\}^2$$

$$\text{END} = \{(i, j) \mid i = n-1 \vee j = n-1\} \cap \{0, \dots, n-1\}^2$$

$$\text{NEIGHBOR}_{(i,j)} = \{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\} \cap \{0, \dots, n-1\}^2$$

# Aufgabe 3: Wellen (Bonus!)

Ein Sequenz von unterschiedlichen Indexen  $X = x_0, \dots, x_s$  mit  $s \geq 0$  beschreibt eine Welle in  $M$ , falls:

1.  $X$  mit einem Startindex anfängt ( $x_0 \in \text{START}$ ) und einem Endindex aufhört ( $x_s \in \text{END}$ ), wobei  $x_0$  gleich  $x_s$  sein kann
2. Aufeinanderfolgende Indexe in  $X$  Nachbarindexe sind:  $\forall i. 0 \leq i < s \Rightarrow x_{i+1} \in \text{NEIGHBOR}_{x_i}$
3. Für jeden Index maximal zwei Nachbarn in  $X$  sind:  $\forall x \in X. |\text{NEIGHBOR}_x \cap X| \leq 2$
4.  $M$  bei jedem Index in  $X$  den gleichen Wert hat:  $\exists v. \forall x \in X. M_x = v$
5.  $M$  bei jedem Index, welcher nicht in  $X$  ist, aber ein Nachbarindex eines Indexes aus  $X$  ist, einen kleineren Wert hat:  $\forall x \in X, y \in (\text{NEIGHBOR}_x \setminus X). M_y < M_x$

6	6	7	3	4	2
6	2	7	1	4	3
6	6	7	2	4	4
4	5	7	7	2	3
9	9	1	7	7	4
7	9	4	5	7	4

Die Matrix  $M$  enthält  $k$  Wellen, falls es  $k$  disjunkte Sequenzen gibt (das heisst, dass die  $k$  Sequenzen kein gemeinsames Element enthalten), welche Wellen in  $M$  beschreiben. Zum Beispiel, die Matrix in Figur 1 enthält drei Wellen, welche mit blau markiert sind. Eine der Wellen wird durch die Sequenz  $(0, 4), (1, 4), (2, 4), (2, 5)$  beschrieben, bei welchen die Matrix überall den Wert 4 hat.

## Aufgabe 3: Wellen (Bonus!)

6	6	7	3	4	2
6	2	7	1	4	3
6	6	7	2	4	4
4	5	7	7	2	3
9	9	1	7	7	4
7	9	4	5	7	4

Implementieren Sie die Methode `Waves.waves(int [][] matrix)`, welche eine quadratische Matrix als Argument nimmt und die maximale Anzahl Wellen in `matrix` zurückgibt. Die Datei `“WavesTest.java”` enthält bereits einige Tests. Wir empfehlen die Tests anzusehen, da die Tests einige Beispiele zeigen. Testen Sie Ihr Programm ausgiebig—am besten mit JUnit—und pushen Sie die Lösung vor dem Abgabetermin. **Tipp:** Überprüfen Sie für jeden Index in `START`, ob er der Anfang einer Welle ist, indem Sie über die mögliche Welle iterieren.

## Aufgabe 4: Schweizerfahne (GUI)

- Instanziere die Window-Klasse und benutze die Methoden setColor() und fillRect()
- RGB-Farbwerte sind zwischen 0 und 255
- Die (x,y)-Koordinaten beginnen in der linken oberen Ecke mit (0,0)

Das Verhältnis von Breite und Länge der Kreuzarme beträgt also 6:7, von Breite und Gesamtlänge des Kreuzbalkens 6:20.

Das Grössenverhältnis des Kreuzes zum Quadrat ist nicht definiert.

*(wikipedia)*

```
Scanner s = new Scanner(System.in);  
int size = s.nextInt(); // input the flag size
```

```
Window window = new Window("Fahne", size, size);
```

```
window.setColor(255, 0, 0); // red  
window.fillRect(0, 0, size, size);
```

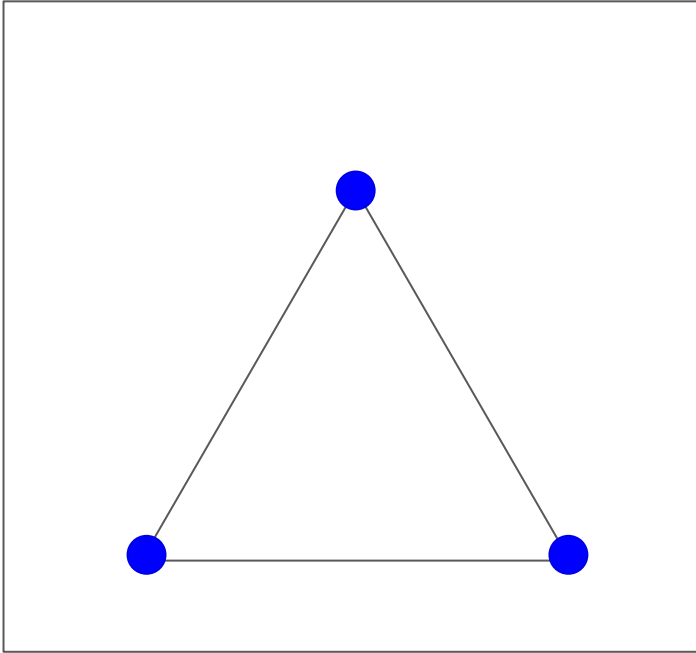
```
window.setColor(255, 255, 255); // white  
window.fillRect(0.2 * size, 0.4 * size, 0.6 * size, 0.2 * size);  
window.fillRect(0.4 * size, 0.2 * size, 0.2 * size, 0.6 * size);
```

```
window.open();  
window.waitForClosed();
```



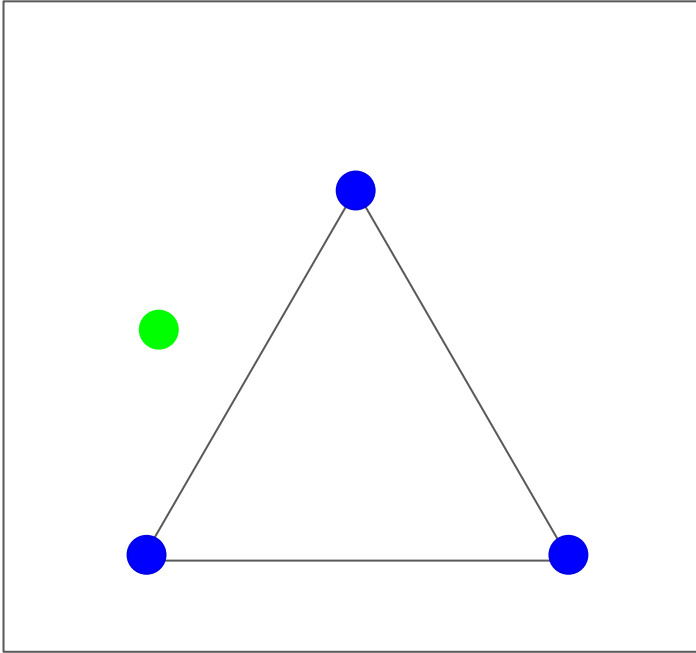
wiederverwendbar

# Aufgabe 5: Chaos Game



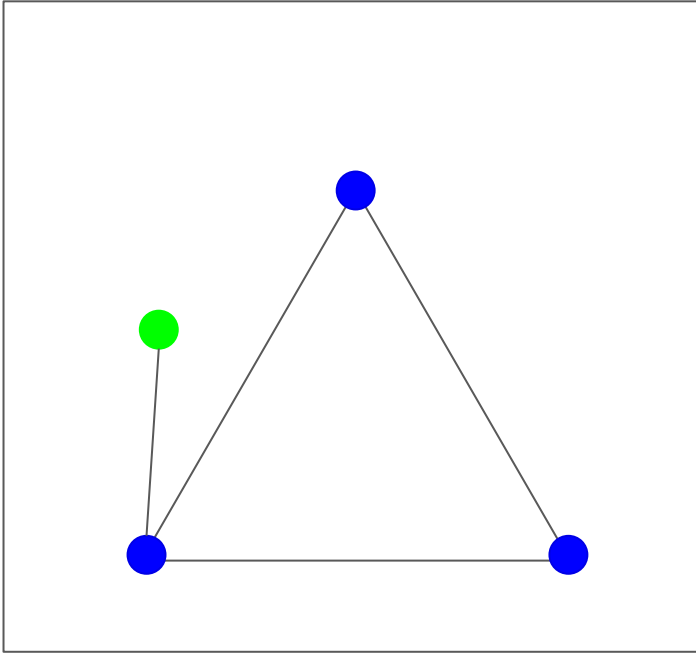
- Definiere die drei Eckpunkte eines Dreiecks.

# Aufgabe 5: Chaos Game



- Definiere die drei Eckpunkte eines Dreiecks.
- Wähle einen zufälligen Punkt  $p$  im Fenster.

# Aufgabe 5: Chaos Game



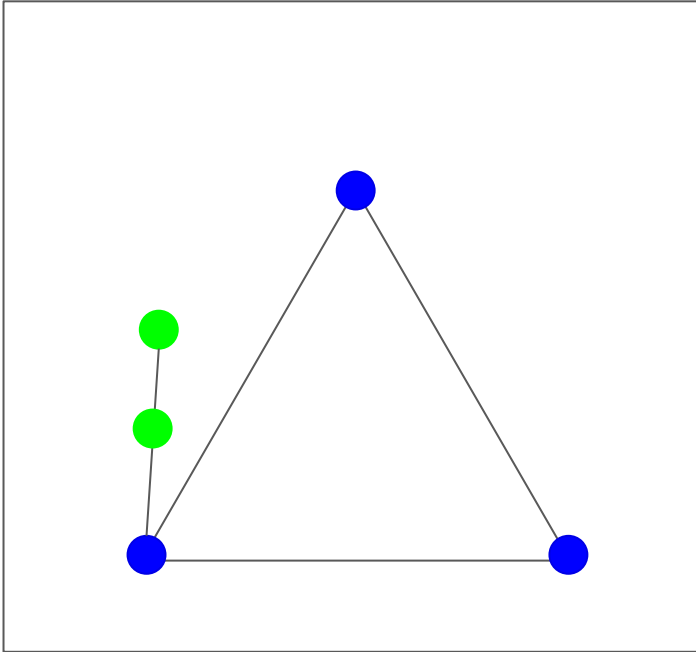
- Definiere die drei Eckpunkte eines Dreiecks.
- Wähle einen zufälligen Punkt  $p$  im Fenster.

Wiederhole:

- Wähle zufällig einen Eckpunkt des Dreiecks aus.



# Aufgabe 5: Chaos Game

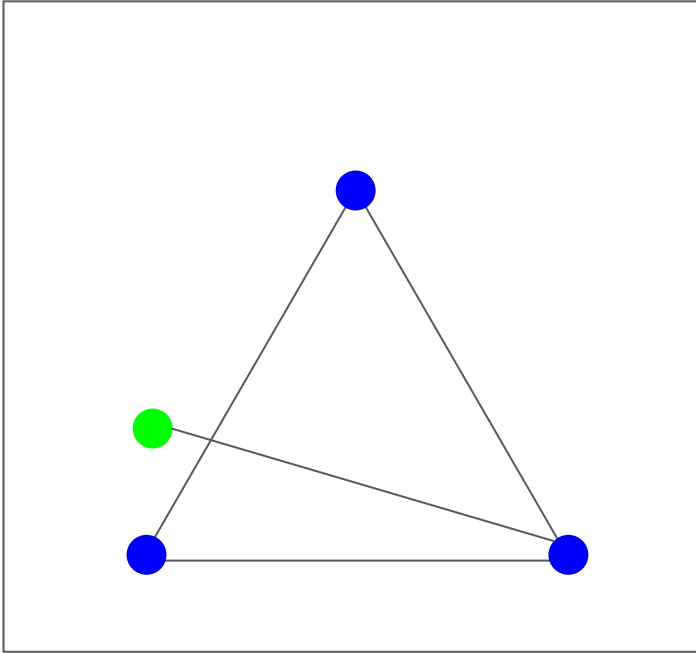


- Definiere die drei Eckpunkte eines Dreiecks.
- Wähle einen zufälligen Punkt  $p$  im Fenster.

Wiederhole:

- Wähle zufällig einen Eckpunkt des Dreiecks aus.
- Finde die Mitte zwischen diesen beiden Punkten. Dies ist jetzt  $p$ .
- Zeichne  $p$  ein.

# Aufgabe 5: Chaos Game

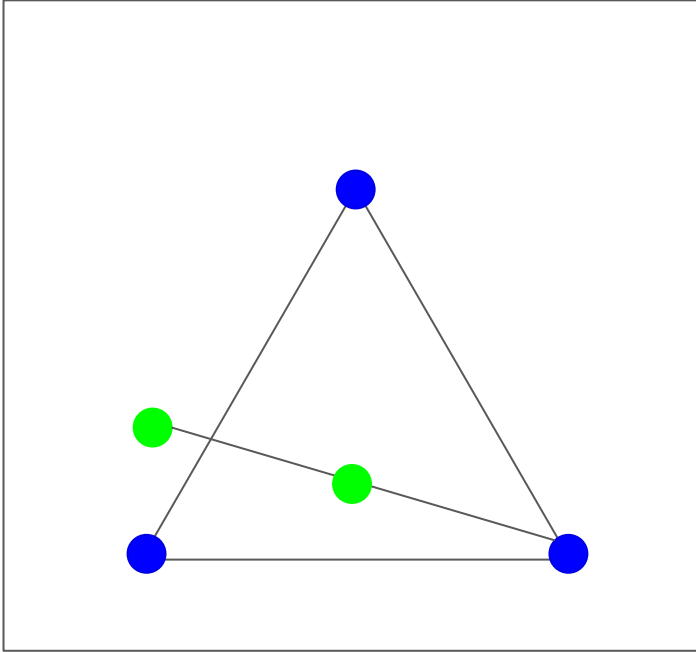


- Definiere die drei Eckpunkte eines Dreiecks.
- Wähle einen zufälligen Punkt  $p$  im Fenster.

Wiederhole:

- Wähle zufällig einen Eckpunkt des Dreiecks aus.
- Finde die Mitte zwischen diesen beiden Punkten. Dies ist jetzt  $p$ .
- Zeichne  $p$  ein.

# Aufgabe 5: Chaos Game

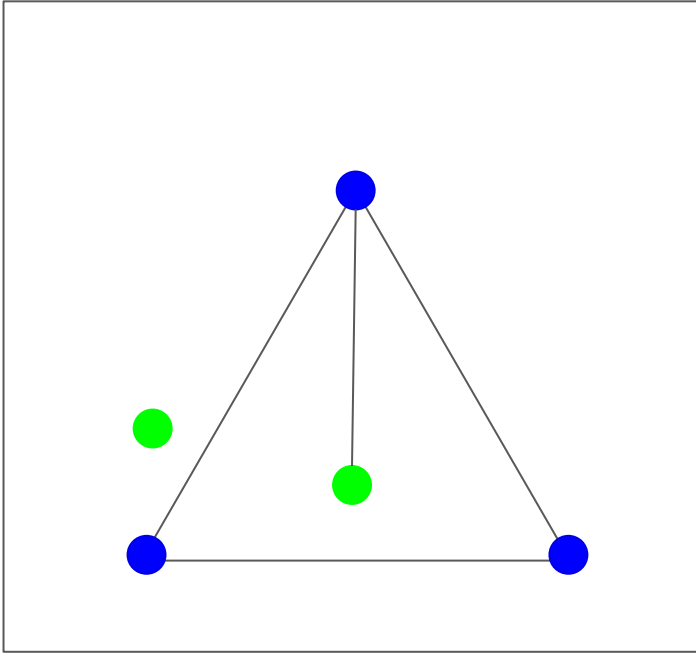


- Definiere die drei Eckpunkte eines Dreiecks.
- Wähle einen zufälligen Punkt  $p$  im Fenster.

Wiederhole:

- Wähle zufällig einen Eckpunkt des Dreiecks aus.
- Finde die Mitte zwischen diesen beiden Punkten. Dies ist jetzt  $p$ .
- Zeichne  $p$  ein.

# Aufgabe 5: Chaos Game

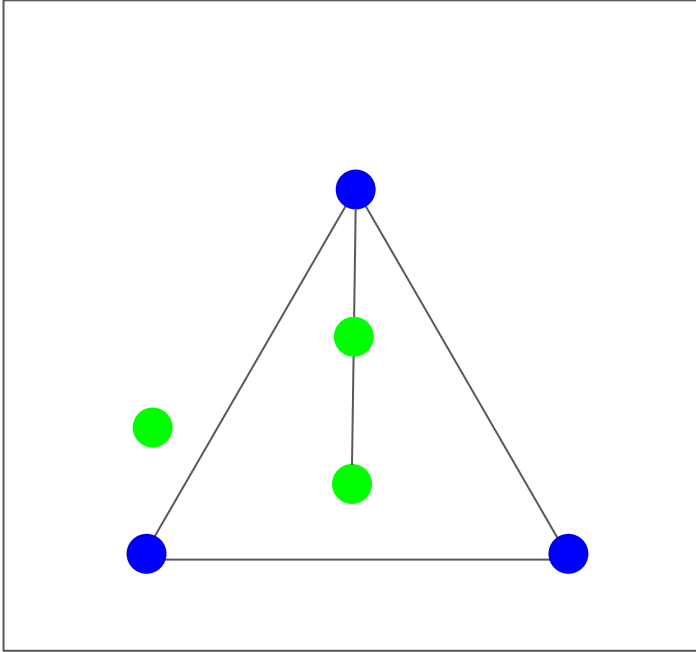


- Definiere die drei Eckpunkte eines Dreiecks.
- Wähle einen zufälligen Punkt  $p$  im Fenster.

Wiederhole:

- Wähle zufällig einen Eckpunkt des Dreiecks aus.
- Finde die Mitte zwischen diesen beiden Punkten. Dies ist jetzt  $p$ .
- Zeichne  $p$  ein.

# Aufgabe 5: Chaos Game

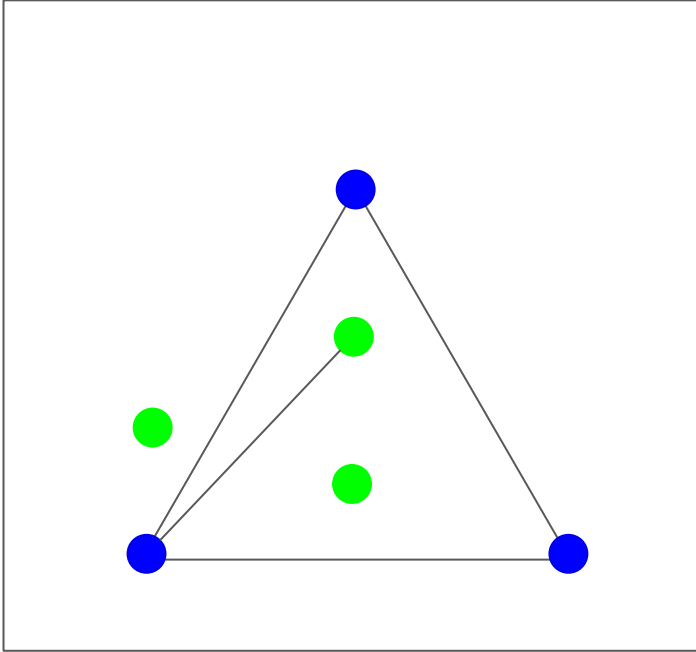


- Definiere die drei Eckpunkte eines Dreiecks.
- Wähle einen zufälligen Punkt  $p$  im Fenster.

Wiederhole:

- Wähle zufällig einen Eckpunkt des Dreiecks aus.
- Finde die Mitte zwischen diesen beiden Punkten. Dies ist jetzt  $p$ .
- Zeichne  $p$  ein.

# Aufgabe 5: Chaos Game



- Definiere die drei Eckpunkte eines Dreiecks.
- Wähle einen zufälligen Punkt  $p$  im Fenster.

Wiederhole:

- Wähle zufällig einen Eckpunkt des Dreiecks aus.
- Finde die Mitte zwischen diesen beiden Punkten. Dies ist jetzt  $p$ .
- Zeichne  $p$  ein.

# Aufgabe 6: EBNF

In dieser Aufgabe erstellen Sie erneut verschiedene EBNF-Beschreibungen. Speichern Sie diese wie gewohnt in der Text-Datei "EBNF.txt", welche sich in Ihrem "u04"-Ordner, bzw. im "U04 <N-ETHZ-Account>"-Projekt befindet. Sie können die Datei direkt in Eclipse bearbeiten.

1. Erstellen Sie eine Beschreibung `<pyramid>`, welche alle Zahlen zulässt, in denen die Ziffern zuerst strikt auf- und dann strikt absteigen. Beispiele sind: 14, 121, 1341.  
Sie dürfen annehmen, dass das leere Wort auch zugelassen wird. (Als Challenge können Sie probieren, das leere Wort auszuschliessen.)
2. Erstellen Sie eine Beschreibung `<digitsum>`, welche alle natürlichen Zahlen als Symbole zulässt, deren Quersumme eine gerade Zahl ist.
3. Erstellen Sie eine Beschreibung `<xyz>`, welche alle Wörter zulässt, die aus X, Y und Z bestehen und bei welchen jedes X mindestens ein Y im Teilwort links und rechts von sich hat. Beispiele sind: Z, YXY, YXXY, ZYXXY.
4. Erstellen Sie eine Beschreibung `<term>`, welche alle wohlgeformten arithmetischen Terme, bestehend aus ganzen Zahlen, Variablen (x, y, z), Addition und Klammern zulässt. Geklammerte Terme müssen mindestens eine Addition enthalten. Beispiele sind: 1 + 4, (1 + 4), 1 + (3 + 4), (1 + 1) + x + 5.

# Einschub: Rekursionsübung

Input: String, mit paarweise verschiedenen Elementen

Output: alle Permutationen des Strings auf der Konsole

Anforderungen: rekursiv, Reihenfolge egal

Beispiel: `printPermutations("abc")` →  
abc  
acb  
bac  
bca  
cab  
cba



Lösungsvorschlag?

# Aufgabe 7: Teilfolgen

Schreiben Sie ein Programm `Teilfolgen`, welches mithilfe von Rekursion alle Teilfolgen der Länge  $N$  eines Strings  $S$  ausgibt. Lesen Sie  $S$  und  $N$  (eine nicht-negative ganze Zahl) von der Konsole ein.

Eine Teilfolge ist definiert als eine Folge von Zeichen (`String`), die durch das Weglassen von Zeichen aus einer ursprünglichen Folge von Zeichen entsteht. Beispielsweise sind alle Teilfolgen der Länge  $N=2$  des Strings  $S="apple"$ :

`"ap", "al", "ae", "pp", "pl", "pe", "le"`

# Beispiele

- $S = abcd, j = 3$

abc

abd

bcd

acd

- $S = abcd, j = 5$

- $S = abcd, j = 0$

*leerer String ("")*

- $S = accb, j = 2$

ac

ac

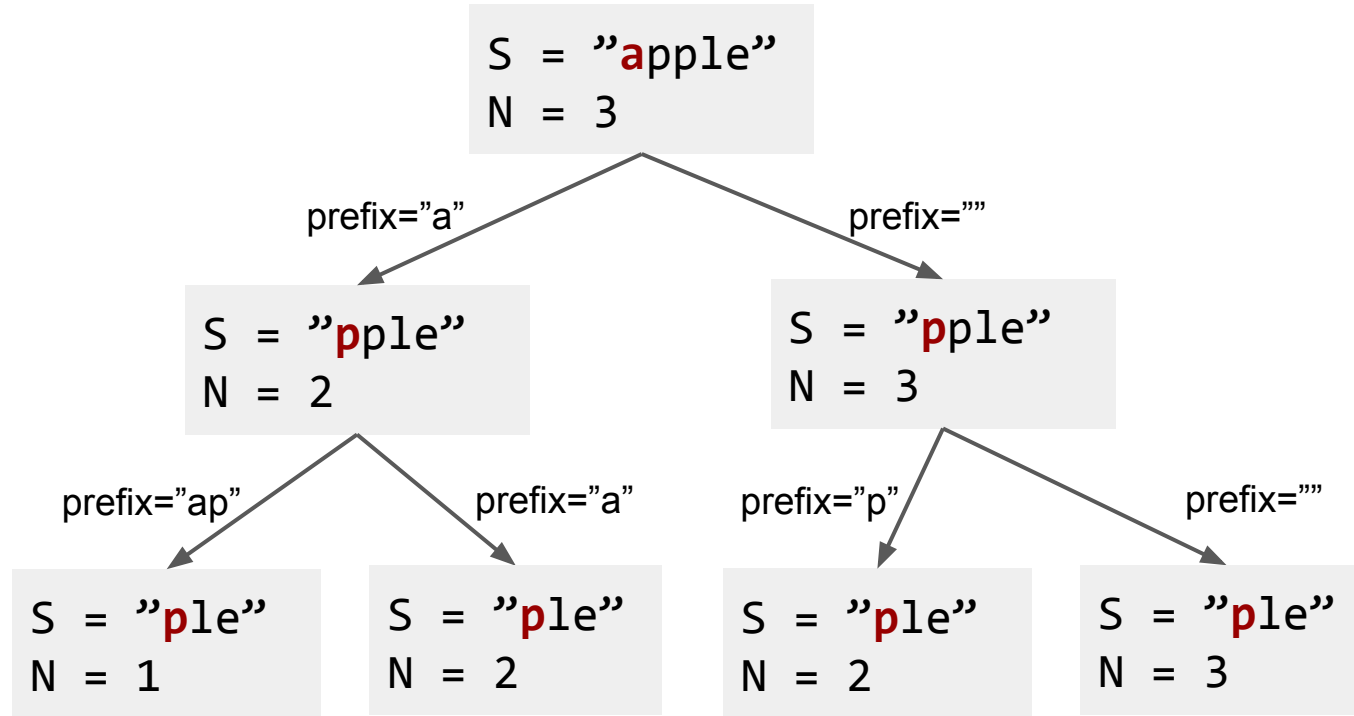
ab

cc

cb

cb

# Rekursion: Teilprobleme identifizieren



# Rekursionsende

Wann haben wir eine Teilfolge gefunden?

→ Wenn 'prefix' die richtige Länge hat (oder  $N=0$  ist).

Wie könnte eine rekursive Lösung aussehen?

# Teilfolgen: Pseudocode

```
subSequences(prefix, S, N) {  
    if (N == 0) {  
        // 'prefix' ist eine Teilfolge!  
        print prefix;  
    } else if (S is not empty) {  
        rest = S ohne erstes Zeichen;  
        // Reduziertes Teilproblem im Rest von S lösen:  
        subSequences(prefix + S[0],  
                      rest, N - 1);  
        // Gleiches Problem im Rest von S lösen:  
        subSequences(prefix, rest, N);  
    }  
}
```

# Zusatzübungen

# Postfix-Increment Operator

Was sind die Werte der Variablen (alles `int`)?

```
x = 0;  
y = 0;  
z = 1;  
  
y = x++ + z - x;
```

```
x = 0;  
y = 0;  
z = 1;  
  
y = x++ + z * x;
```

```
x = 0;  
y = 0;  
z = 1;  
  
y = z * x + x++;
```



# Arrays

Was gibt das folgende Code-Segment aus?

```
int [] a = new int [10];

for (int i = 0; i < 10; i++) {
    a[i] = 9 - i;
}

for (int i = 0; i < 10; i++) {
    a[i] = a[ a[i] ] ;
}

for (int i = 0; i < 10; i++) {
    System.out.print(a[i]);
}
```

Wie könnte man dieses Code-Segment flexibler formulieren?

Kahoot