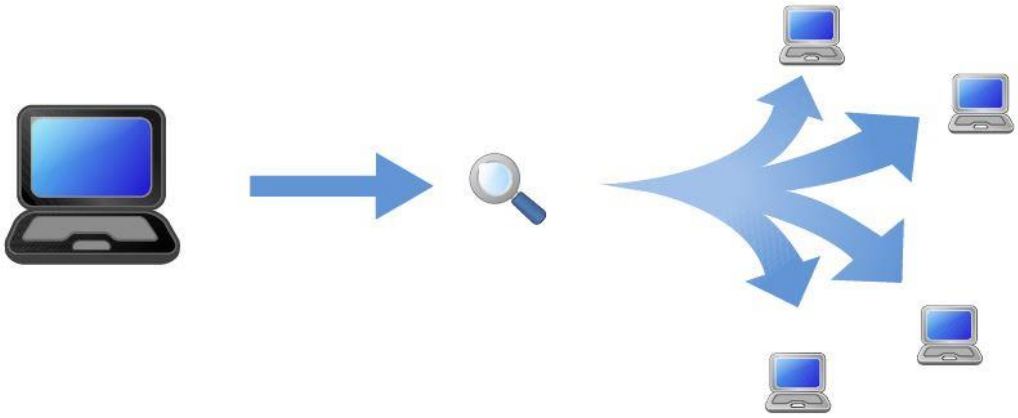
Ping Scanner



Projektarbeit von:	Lukas Randegger, Murat Tokmakçi, David Keller und Jonas Bollinger
Bildungsinstitution:	TEKO Zürich
Dozent:	Stephan Kessler
Abgabedatum:	17.03.2020

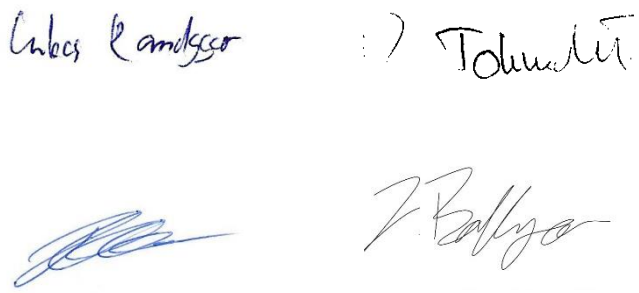
1 Inhaltsverzeichnis

2 Selbstständigkeitserklärung

Ich (wir) bestätige(n) hiermit, dass ich (wir) die vorstehende Projektarbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und sowohl wörtliche, als auch sinngemäss verwendete Textteile, Grafiken oder Bilder kenntlich gemacht habe(n). Diese Arbeit ist in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt worden.

Ort, Datum Unterschrift/en

Zürich, 16. März 2020



3 Initialisierung und Planung

3.1 Auftrag des Dozenten

Die an die Studierenden gestellte Aufgabenstellung lautete wie folgt:

Ihr Auftrag ist es einen Ping-Scanner zu programmieren. Dabei soll Ihr Programm in einem Netzwerkbereich nach aktiven Geräten suchen. Der Benutzer gibt dazu über ein GUI den Bereich an und erhält in einer GUI-Liste alle aktiven Geräte.

*In der GUI-Oberflache gibt der Benutzer eine IP-Adresse (IPV4) und einen Suchbereich an. Dabei kann der Benutzer zwischen den Suchbereichen **24**, **16** und **8** aussuchen. Diese Zahlen stehen dabei für die verkürzte Netzmaskenform. Wenn der Benutzer die IP-Adresse 192.168.1.7 angibt, wird bei entsprechenden Angaben mit folgenden Bereichen gesucht:*

- ***24** → 192.168.1.1 – 192.168.1.254*
- ***16** → 192.168.1.1 – 192.168.254.254*
- ***8** → 192.1.1.1 – 192.254.254.254*

Der Benutzer soll darauf über einen Knopf die Suche starten. Alle Hosts die darauf antworten sollen mit einer IP-Adresse und dem Hostnamen angezeigt werden. Der Benutzer soll zusätzlich eine Option am Anfang haben, dass der Hostname nicht ermittelt wird. Ist diese Option ausgeschaltet, wird kein Hostname angegeben bzw. leer gelassen.

3.2 Zielformulierung

Als Ziel wurde in der Gruppe ein funktionierendes Programm mit einem verständlichen bzw. selbsterklärenden GUI definiert.

3.3 Projektablaufplanung

Zur Durchführung wurden nebst 2 Meetings direkt in der Schule, 4 weitere Web-Meetings abgemacht und durchgeführt. Die Web-Meetings waren durch Ihre nicht ortsgebundene Natur effizient und praktikabel. Dazu wurden die Tools Discord[1] und Hangouts[2] verwendet.

[1] Discord: Ein kostenloser, sicherer und umfangreicher Sprach- und Textchat. https://discordapp.com/
[2] Hangouts: Google Hangouts ist ein Videokonferenz- und Instant-Messaging-Dienst. https://hangouts.google.com/?hl=de

4 Realisierung

4.1 Programmstruktur

4.1.1 IP Adressen generieren

Die Host-IP-Adressen im Subnetz werden in der Klasse Scanlauf über die Methode addNibble(String base_adress, int number_of_nibbles_to_add) generiert. Der erste Parameter ist der Netzwerkbereich der IP-Adresse. Dieser ist für alle IP-Adressen gleich. Der zweite Parameter bestimmt die Anzahl Nibbles die für die Hosts offen sind. Bsp. Eingabe IP-Adresse: 192.168.1.18, Subnetz: 16. Die Parameter wären dann, 192.168 und 2.
Die Methode generiert die IP-Adressen mit einem rekursiven Ansatz. In jedem Durchlauf wird numberOfNibblesToAdd um 1 reduziert. Ist numberOfNibblesToAdd 0, dann endet die Methode und gibt die erstelle IP-Adresse zurück.

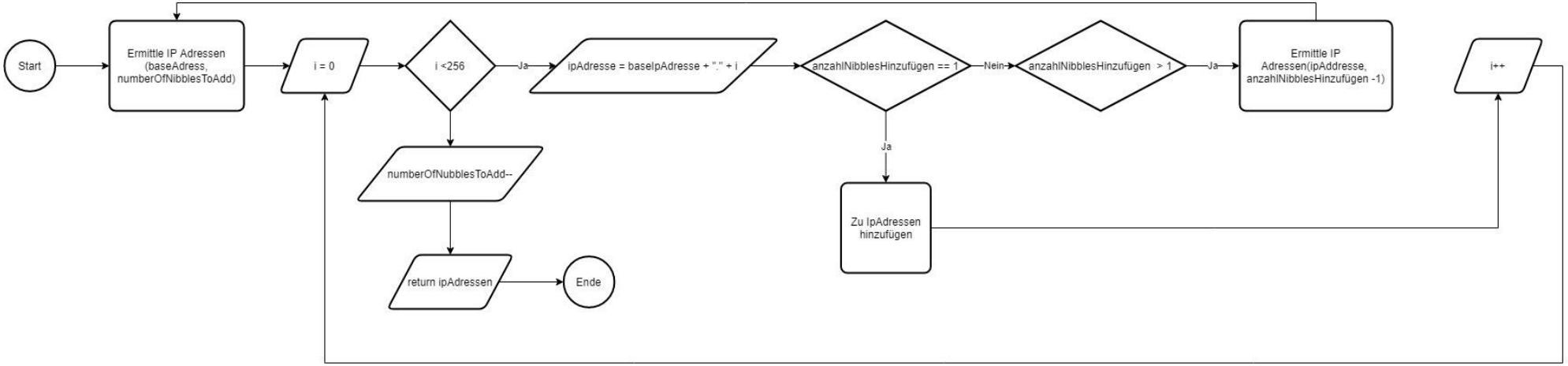


Figure 1: Ablaufschema IP generieren

4.1.2 Auflösen der Host Adressen

In der Klasse IpScanner über die Methode loopAllIps(ArrayList<String> ips) werden die IP-Adressen aufgelöst. Über eine For-Schleife passiert für jede IP-Adresse folgendes: Die Methode InetAddress.getByName() wandelt die IP-Adresse die aktuell ein String ist, zu einer InetAddress um. Diese Klasse erlaubt uns mit der Methode isReachable(int timeout) festzustellen, ob der Host mit dieser IP erreichbar ist. Falls der Host innerhalb des Timeouts antwortet, ist der Rückgabewert True. Falls im UI die Option «Hostnamen anzeigen» aktiviert ist, wird über die Methode getHostname() der Name des Hosts aufgelöst. Ist die Option nicht aktiviert, bleibt der Host-Name leer. Die erreichbaren Hosts werden über die Klasse Host abgebildet. Diese hat 2 Properties: ip_adresse und host_name.

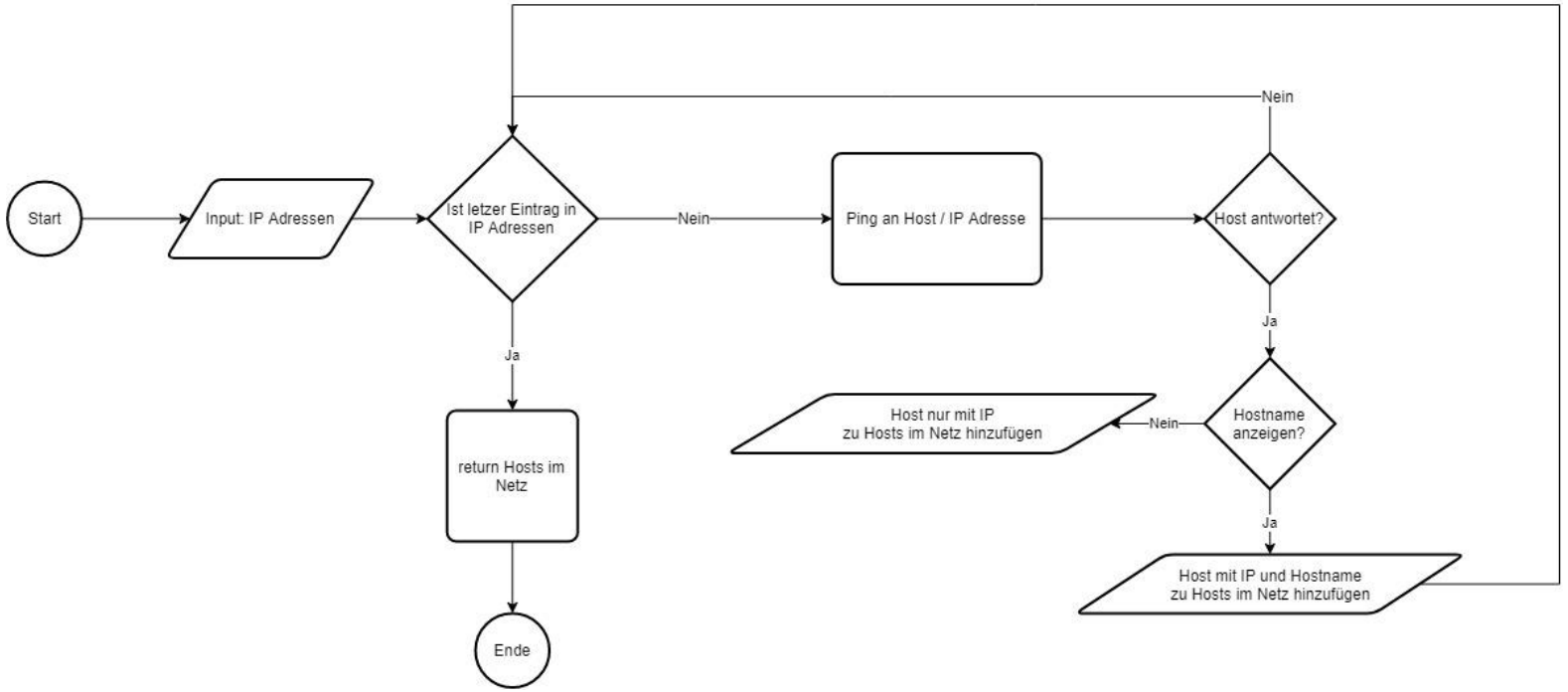


Figure 2: Ablaufschema ping und Namensauflösung IP

4.2 GUI

Für die Eingabe in den Ping-Scanner wurde ein GUI erstellt. Dieses vereinfacht die Ein- und Ausgabe und macht diese anschaulicher.

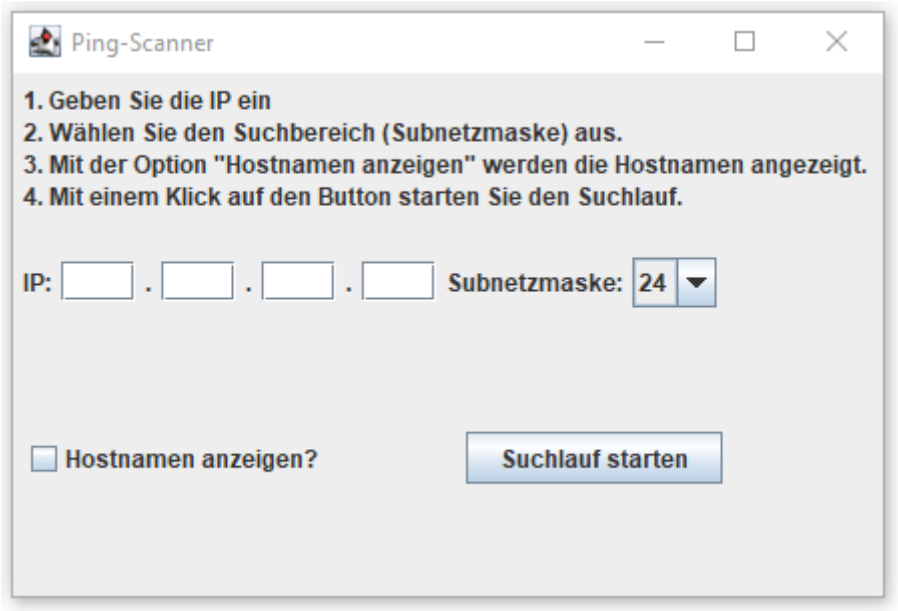


Figure 3: Ping-Scanner

4.2.1 Aufbau des GUI

Beim Layout wurde darauf geachtet, dass es einen logischen Aufbau hat.

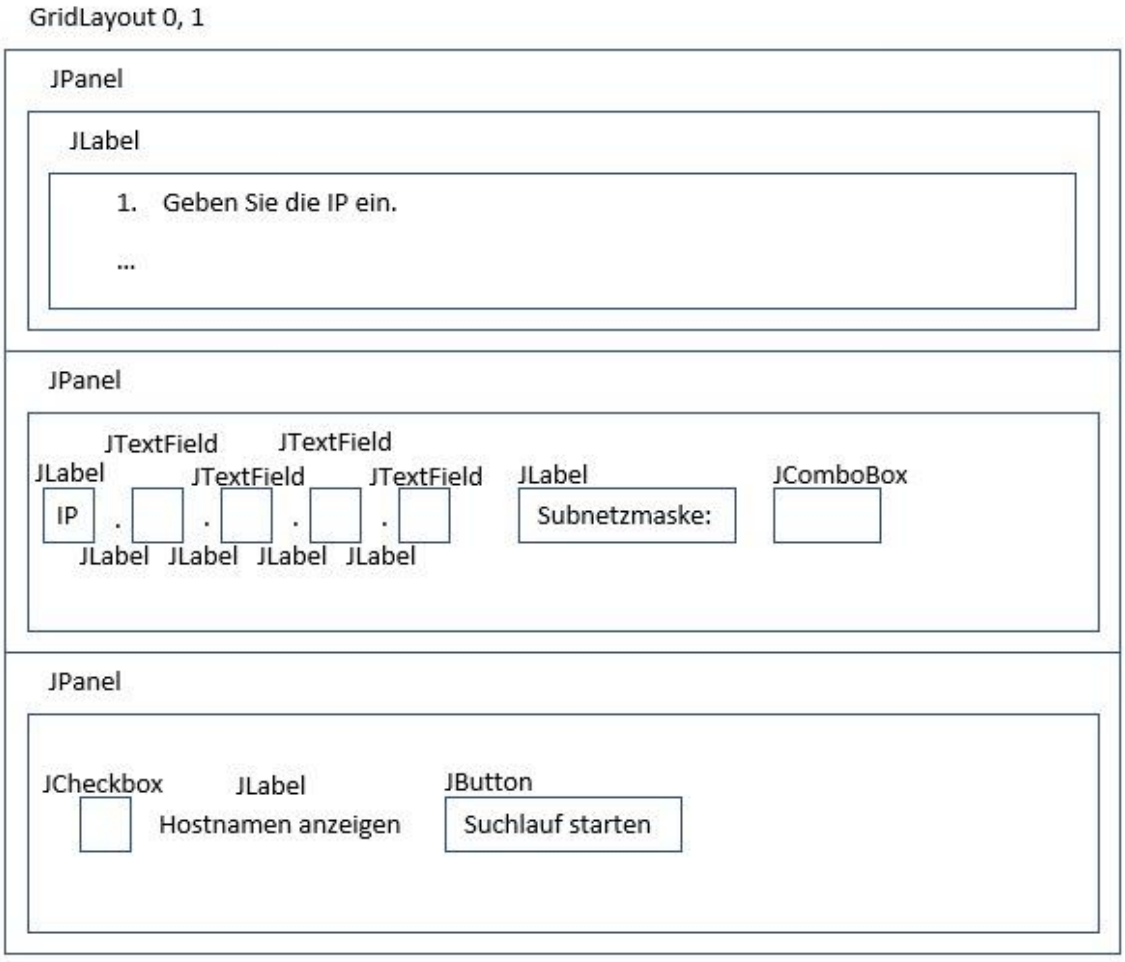


Figure 4: Aufbau des GUI

Damit nicht alle Elemente in einer Reihe erschienen, wurde eine Kombination aus dem Gridlayout[3] und einem FlowLayout[4] gewählt. Das GridLayout wurde mit der Zeilenanzahl 0 und der Spaltenanzahl 1 gewählt. Mit der Zahl 0 werden die Anzahl Zeilen automatisch evaluiert. Um zu verhindern, dass jedes Element nun eine eigene Zeile hat, wurden die Elemente, die auf die gleiche Zeile gehören in einen JPanel[5] gepackt. Somit entstanden drei Hauptelemente, die dann die Zeilen in Gridlayout wurden.

[3] GridLayout: Erstellt im GUI eine unsichtbare Tabelle.
[4] FlowLayout: Reiht alle Elemente im GUI aneinander.
[5] JPanel: Gruppiert mehre Elemente im GUI zu einem.

4.3 Lösung zum Auftrag

Folgende Screenshots zeigen den Gebrauch des Programms anhand eines Beispiel IP Ranges (213.144.129.0/24). Dieser Range wird verwendet, weil sich dieser sehr gut zur Demonstration der Funktion des Programmes eignet. Dies auf Grund der Tatsache, dass sich darin sowohl nicht antwortende, wie auch antwortgebende Ziele befinden.

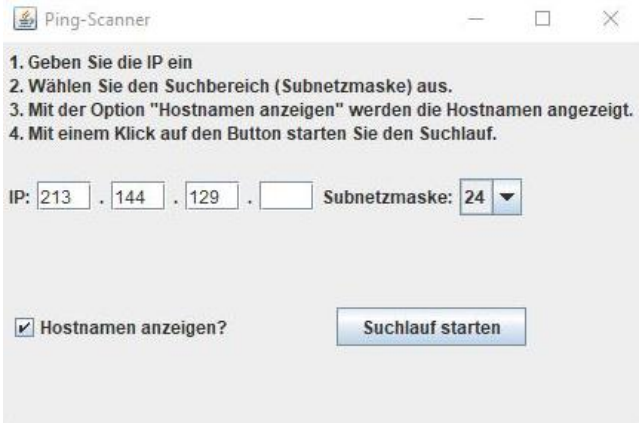


Figure 5: GUI ausgefüllt

Nach der Eingabe und Bestätigung durch den Button «Suchlauf starten», wird das Programm ausgeführt und gibt in diesem Beispiel das folgende Resultat in einem separaten Fenster aus:

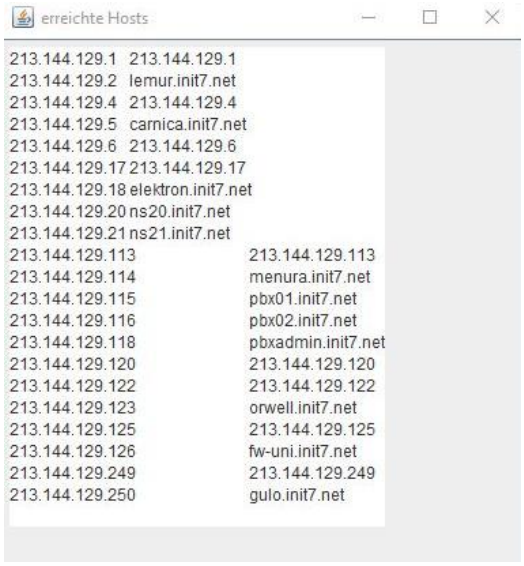


Figure 6: Ausgabe in separatem Fenster mit Namensauflösung

Das Weglassen der Option «Hostnamen anzeigen?» führt zu folgendem Ergebnis:

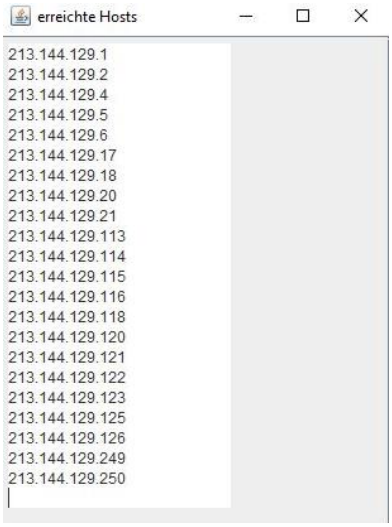


Figure 7: Ausgabe in separatem Fenster ohne Namensauflösung

5 Abschluss

5.1 Reflexion Weg zum Ziel

Die Zusammenarbeit war von Erhalt der Aufgabe über die Planung bis zur Realisierung und Abschluss des Projekts durchgehend positiv. Jeder der Beteiligten war stets mit Engagement am Werk. Allfällige Unklarheiten und Unterschiede im Wissensstand wurden rasch geklärt, sodass sich alle im Klaren sind, worum es geht. Dank der Online-Videokonferenz konnten alle Teilnehmer problemlos denselben Bildschirm sehen und ihre Vorschläge zu allen Funktionen einbringen. Diese Art der Kommunikation ist sehr gut für ein Projekt dieser Grösse geeignet. Eine Alternative hierzu wäre ein Offline-Meeting, jedoch muss auch hier sichergestellt werden, dass alle Teilnehmer denselben Bildschirm sehen. Dies ist ohne passende Infrastruktur wie einem Beamer und einer Leinwand etwas umständlich, weshalb die Online-Variante ein sehr viel effizienteres Arbeiten ermöglichte.

5.2 Ausblick

Das Programm erfüllt die Anforderungen und ist sehr nützlich, kann aber auch beliebig erweitert werden:

Gemäss Aufgabenstellung müssen nur die 8er, 16er und 24er Subnetze gescannt werden können. In der Realität existieren aber auch andere Subnetze, beispielsweise ein 17er Subnetz. Mit der aktuellen Version des Programms kann man ein solches nicht scannen, dies kann aber in einer zukünftigen Version implementiert werden.

Die Liste aller Hosts, die auf den Ping antworten, wird in der aktuellen Version in einem neuen Fenster in eine TextArea geschrieben. Mit einer zukünftigen Version des Programms kann die Liste in eine externe Datei geschrieben werden, wodurch die Weiterverarbeitung erleichtert wird. Zum Beispiel kann die Ausgabe in eine csv-Datei geschrieben werden, um sie später in eine Tabelle zu importieren.

Das Programm hat unter Windows 10 (Linux Debian 10 war bei Tests performanter) relativ lange für die Ausgabe. Vor allem die Abfrage eines 16er und 8er Subnetzes dauert mehrere Minuten bis Stunden. In einer zukünftigen Version wären Performance-Verbesserungen sehr erstrebenswert.

6 Quellenverzeichnis

Diverse JavaDocs von Oracle zu GUI-Elementen und Netzwerk-Elementen:
https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html
https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html
https://docs.oracle.com/javase/7/docs/api/java/net/InetAddress.html

Scrollbar für die TextArea im Ausgabefenster:
https://java2everyone.blogspot.com/2008/12/add-scrollbar-to-jframe.html

SimplifiedListener:
Teams unserer Klasse (__Z-TIN-18-a Programmieren 1) -> Allgemein -> Dateien -> 06_2D_Grafiken -> Simplified.zip

2D Grafiken Use Cases:
Teams unserer Klasse (__Z-TIN-18-a Programmieren 1) -> Allgemein -> Dateien -> Kapitel 6 & 7