Klassendiagramm Dokumentation

Converter

Die statische Klasse Converter, enthält die hauptsächliche Logik des Programmes, sei es die Umwandlung von IP-Addressen oder auch die Überprüfung der Netzwerke.
Da der Converter nur statische Methoden enthält, können diese Methoden ohne Instanzierung der Klasse aufgerufen werden.
Die Methoden *convertIpToBinary* und *convertIpToHex* dienen dazu dezimale IPAddressen in Hex oder auch Binär umzuwandeln. Die Methode *checkIfPossibleNewNetwork* wird vom *NetworkPanel* genutzt um zu überprüfen ob die Netzwerke sich überlappen, dazu wird die Methode *isColliding* verwendet, welche zwei Netzwerke vergleicht. Um die Überprüfung zu erleichtern gibt es die Methode *IpToLong*, welche die IP-Addressen als Strings in long Werte umwandelt, so können die IP-Adressen von den Methoden besser verwaltet werden. Die Methode *getAllIPsInNetwork* hat zwei verschiedene Anwendungsfälle einmal um die Die Host-IP-Adressen anzuzeigen und einmal um die Anzahl der freien IP-Addressen für das jeweilige Netzwerk zu ermitteln. Die Subnetzmaske bilden wir aus den Präfix, dieser wiederum kann mittels zweier Methoden ermittelt werden, *getPrefixFromAmountOfHosts* und *getPrefixFromCompleteNetwork*. Die Methode *getNewFreeIPAfterNetwork* wird dazu verwendet um im SubnetPanel zu ermitteln welche IP-Addresse für das nächste Subnetz frei wäre. Damit die Subnetze auch richtig geordnet angezeigt werden, gibt es hierfür die Methode *sortNetworksInModel*, sodass die Netzwerke nach ihrer Größe sortiert

werden.

NetworkAddressValidator

Die Klasse prüft, ob die Netzwerkadressen korrekt sind. NETWORK_ADDRES wird als private static final deklariert, da diese Variable nur innerhalb der Klasse verfügbar ist. Die Variable enthält den validen REGEX für eine Netzwerkadresse. Um die eingegebene Netzwerkadresse zu prüfen, gibt es die Methode *validate*. Der prüft dann mit dem REGEX NETWORK_ADDRESS den eingebenen String und gibt ein Boolean zurück

NetworkCalculator

Der NetworkCalculator erbt von JFrame, in welchem die einzelnen JPanels hinzugefügt werden. Als Variable enthält der NetworkCalculator ein tabbedPanel vom Typ JTabbedPane, welches direkt instanziert wird. Der Kostruktor ertwartet ein String und ein JSONArray, der String enthält den Titel und das JSONArray die einzelnen Netzwerke. Zudem wird im Konstruktor ein ChangeListener hinzugefügt, welcher überprüft ob ein Tab geschlossen wurde. Außerdem werden hier die grundlegenden Design-Einstellungen getätigt.

Desweiteren wurde hier die Speichermöglichkeit implementiert. Wenn das Programm geschlossen wird, werden alle eingetragenen Netzwerke gespeichert, sodass diese später wieder eingelesen werden können.

Damit auf die einzelnen Tabs des tabbedPane einfacher wieder zugegriffen werden kann, wurde die Methode *getTabIndexFromTitle* implementiert, welcher ein JTabbedPane und ein String erwartet. Mittels des Strings wird dann der jeweilige Tab ermittelt und zurückgegeben.

Die Methoden *showMoreInformationAboutNetwork* und *showMoreInformationAboutTheHostIP* erwarten jeweils einen String und sind vom Typ void. Diese Methoden geben alle relevanten Infos aus für das Netzwerk oder auch der HostAdresse, zudem wird hier der jeweilige binäre und hexadezimale Wert angezeigt.

NetworkPanel

Dieses JPanel dient dazu um einzelne Netzwerke hinzuzufügen, dafür gibt es eine Textfelder Maske in der ein Netzwerk eingetragen werden kann. Zusätzlich können relevante Informationen zum Netzwerk in einem neuen Fenster angezeigt werden.

Das NetworkPanel erbt von JPanel und enthält diverse Attribute. Ein JTabbedPane, ein JSONArray, eine DefaultListModel und eine ArrayList. Der DefaultListModel und die ArrayList sind jeweilis vom Typ static, damit auf diese direkt zugegriffen werden kann. Der DefaultListModel enthält Strings und die ArrayList SubnetPanels. Der Konstruktor vom NetworkPanel hat als Parameter ein NetworkCalculator und ein JSONArray. Damit auf die Methoden des NetworkCalculator zugegriffen werden kann wird dieser übergeben, das JSONArray enthält die bereits vorhandenen Netzwerke. In dem NetworkPanel wird ein MouseListener implementiert welcher ein Doppelklick auf die Netzwerke ermöglicht, um ein SubnetPanel zu öffnen, welcher als Tab angezeigt wird.
Die Methode *openNewSubnet* öffnet ein neues Subnetz und erwartet als Parameter eine JList und ein NetworkCalculator. Die Jlist enthält die Netzwerke und mittels des NetworkCalculators wird überprüft ob, ein neues Tab geöffnet werden muss oder nicht. Um einen Wert aus ArrayList zu entfernen gibt es die statische Methode *removeEntryFromArrayList* welche ein int erwartet. Damit ein Value zur ArrayList hinzugefügt werden kann gibt es die statische Methode *addEntryToArrayList* welche auch ein int erwartet.

Das NetworkPanel enthält vier anonyme ActionListener, einer wird verwendet um ein neues Netzwerk hinzuzufügen mit Überprüfung ob dies möglich ist, ein anderer um ein Netzwerk zu löschen, ein weiterer um zusätzliche Informationen anzuzeigen, zudem einer um ein SubnetPanel zu öffnen

SubnetPanel

Die Klasse SubnetPanel erbt vom JPanel. Das Panel enthält die Subnetzadressen mit dem jeweilligen Prefix, drei Buttons um mit dem Subnetzen zu interagieren und es besteht die Möglichkeit eine neues Subnetz mit einer x Anzahl an Hosts zu erstellen. Als Variable enthält das SubnetPanel ein tabbedPane vom Typ JTabbedPane, welches ebenfalls direkt instanziiert wird. Zudem enthält es ein String, welches mit den Netzwerktitel, ein JSONArray, ein JLabel mit den freien Addressen, ein DefaulListModel vom Typ String und eine ArrayList vom Typ HostPanel. Die ArrayList ist static, damit direkt darauf zugegriffen werden kann.
Der Konstruktor enthält als Parameter ein String mit dem Netzwerk, ein NetworkCalculator und ein JSONArray. Mit dem JSONArray werden schon vorhandene Subnetze vom Netzwerk eingefügt. Die Methode *openNewHostPanel* erwartet eine JList und ein NetworkCalculator um ein HostPanel für das Subnetz zu erstellen.
Um Werte zu der ArrayList hinzuzufügen gibt es die Methode *addEntryToArrayList*, welche ein HostPanel erwartet. Um einen Wert zu löschen *removeEntryFromArrayList*, welche ein int erwartet. Damit man weiß welche IP-Addressen im Netzwerk noch frei sind gibt es die Methode *updateFreeAddressesLabel*.
Das SubnetPanel enthält vier verschiedene anonyme ActionListener. Ein Listener führt die Methode *openNewHostPanel* aus, einer löscht ein Subnetz, der nächste dient dazu zusätzliche Informationen anzuzeigen, der

andere überprüft beim anlegen eines Subnetztes ob dies möglich ist und fügt es dann hinzu oder nicht.

Hostpanel

Die Klasse Hostepanel erbt vom JPanel. Wie der Name schon sagt, enthält das Panel nur die verfügbaren Hosts. Die Hosts und die dazugehörigen Notizen werden in einer DefaulListModel als static gespeichert. Der Konstruktor hat als Parameter zwei Strings und ein JSONArray, die beiden Strings enthalten jeweils das Netzwerk und das Subnetz, mittels des JSONArray's werden die bisher vorhandenen Hosts mit deren Notizen angezeigt. Falls keine Hosts bisher vorhanden sind, werden diese neue erstellt. Das HostPanel implementiert zudem ein MouseListener, welcher bei Doppelklick auf eine HostAdresse zusätzliche Informationen anzeigt.