# Subnet Calculator

## Rövid leírás / felhasználói dokumentáció:

A cél egy olyan eszköz készítése volt, amiben a számomra ismert eszközöknél gyorsabban lehet alhálózat kiosztásokat számolni.

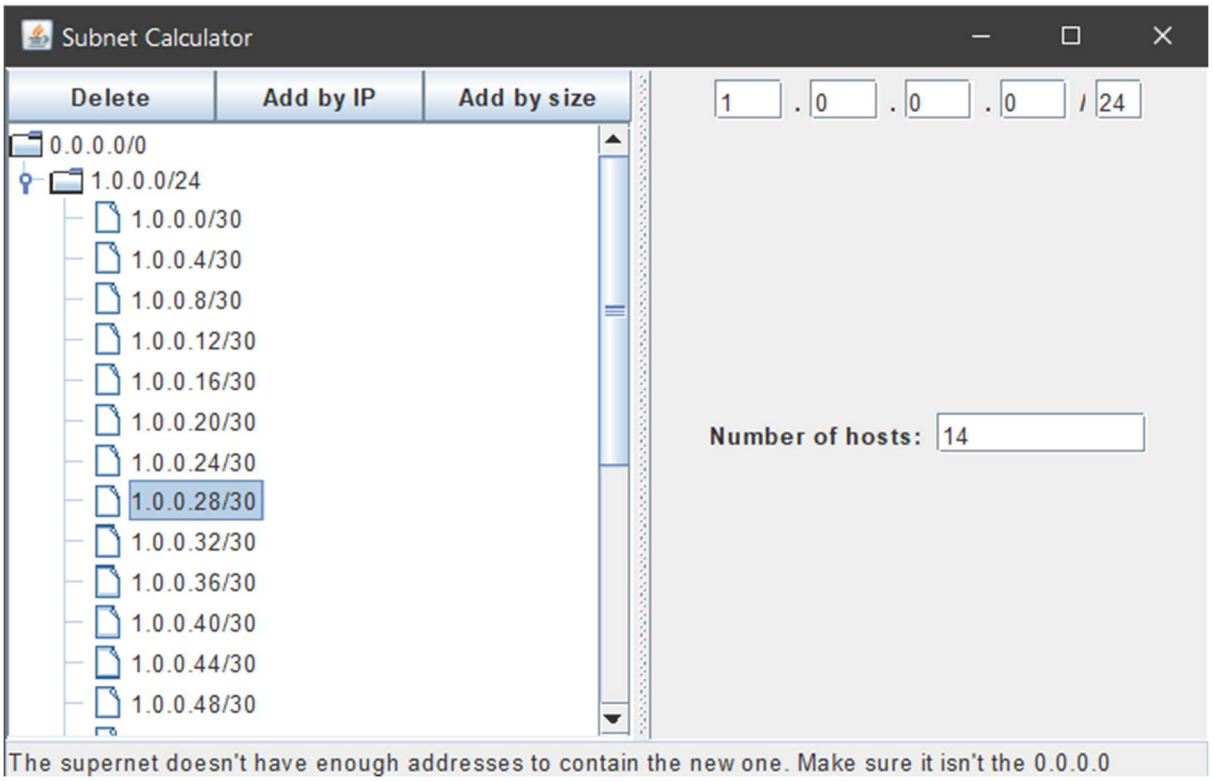
Maga az alkalmazás elég egyszerű. Kétféle módon lehet új hálózatot bevinni.

- IP cím alapján (ezt csak a gyökérnél ajánlott használni, mert nincs validálás, hogy tényleg belefér-e a kiválasztottba)
- A szükséges állomáscímek száma alapján.

A baloldali fában ki lehet választani, hogy melyik hálózathoz kell új alhálózatot adni. Ha hiba lép fel azt alul közli az alkalmazás a felhasználóval.

Ha törlünk egy hálózatot érdemes a fában utána következő testvéreitől is megszabadulni, mert a hálózathoz adás állomások alapján mindig a legutolsó alhálózat alapján számolódik ki.

Az alkalmazás bezárása a networks.dat-ba menti az aktuális állapotokat. Ha megsérülne, akkor nyugodtan lehet törölni, de akkor visszaáll az alkalmazás az alapállapotába. (Csak a 0.0.0.0/0 hálózat marad meg.)



## Use-case lista

Ha nincs kiválasztva hálózat akkor a program feltételezni, hogy a felhasználó a gyökérre gondolt.

| | |
|---|---|
| Új hálózat hozzáadása a megadott IP cím alapján | A baloldalon kiválasztott hálózathoz ad validálás nélkül egy új hálózatot |
| Új hálózat hozzáadása a megadott állomásszám alapján | A baloldalon kiválasztott hálózathoz ad validálás nélkül egy új hálózatot |
| Meglévő hálózat törlése | Törli a kiválasztott hálózatot és összes alhálózatát.<br>A gyökéren nem működik. Alul hibát ír ki. |

## Implementáció

**JSwing elemek hierarchiája:**

```
SubnetCalcFrame: Frame
      mainPane: SplitPane
            treePanel: JPanel[BorderLayout]
                  buttonPanel: JPanel[GridLayout]
                        removeButton: JButton
                        addByIpButton: JButton
                        addBySizeButton: JButton
                  <anonymous>: JScrollPanel
                        networkTree: JTree
            rightPanel: JPanel[GridLayout]
                  addressPanel: JPanel
                        octet1: JTextField
                        <anonymous>: JLabel
                        octet2: JTextField
                        <anonymous>: JLabel
                        octet3: JTextField
                        <anonymous>: JLabel
                        octet4: JTextField
                        <anonymous>: JLabel
                        mask: JTextField
                  addBySizePanel: JPanel
                        <anonymous>: JLabel
                        size: JTextField
      messages: JTextField
```

Létrehoztam egy Network osztályt (erre még visszatérek), ami a `javax.swing.tree.DefaultMutableTreeNode`-ból származik és e fölé hoztam létre a NetworkData osztályt, ami a `javax.swing.tree.DefaultTreeModel`-ből származik, így a `SubnetCalcFrame`-ben található JTree automatikusan újra renderelődik, ha valami változás éri. Maga a NetworkData az elég üres, de a bővíthetőség kedvéért megtartottam. Létrehoztam egy saját kivétel típust `InvalidIPv4AddressException` néven.

## Network

Az örökölt mezőkön kívül van 2 saját mezője. A short[] típusú `octets` és az int típusú `mask`. A `mask` azért lett int mert az IDE minden byte-omat át castolta és olvashatatlan lett az egész.

Az `octets` az azért áll 4 shortból mert a java byte-ok nem tudnak +255-ig tárolni és a longban tárolás meg a `toString` és a felhasználói bevitel esetén nem szerencsés.

Sokszor jön elő a `chidren` Vektor, amit a `DefaultMutableTreeNode`-tól örököl.

A Network osztály aljában található 5 statikus metódus csak arra van, hogy a short[4]->long konverziókat meg a bitmaszkolást kényelmesebben lehessen csinálni. A `makeWildcard` meg az `isValidNetwork` tesztelését nem éreztem szükségesnek mert amíg a `makeMask` meg a `getLongFromOctets` működik addig ezeknek is illene az egy sorukkal.

Network(Network *superNet*, long *size*) a szuperhálózat és legutóbbi testvérhálózata alapján létrehoz (vagy nem hoz létre) egy hálózatot, amibe `size` db állomás belefér. Ha ez nem jött össze hibát dob amit az adott gombra rakott callback (ami a new Networkot is meghívta) elkap és az interface message elemébe írja az értékét.

*Egyéb függvények:*
long getNAddrs()

Az aktuális hálózat összes állomáscímének a száma + 2 (network, broadcast)

long nextFreeNetworkAddress()

Kiszámolja az első címet ami nincs ebben a hálózatban (avagy a következő elérhető hálózati címet)

long nextFreeSubnetAddress()

A következő hálózat cím (utolsó alhálózat alapján) ami elérhető a hálózatban vagy utána. Hibát dobhat, ha valami túlindexelés vagy hasonló lép fele a `chidren` (subnet) vektorban.

void add(Network *net*)

Ez ad a `children` vektorhoz új alhálózatot.