Template Week 6 – Networking

Student number: 522880

Bonus point assignment – week 6

Remember that bitwise java application you've made in week 2? Expand that application so that you can also calculate a network segment as explained in the PowerPoint slides of week 6. Use the bitwise & AND operator. You need to be able to input two Strings. An IP address and a subnet.

IP: 192.168.1.100 and subnet: 255.255.255.224 for /27

```
Example: 192.168.1.100/27
Calculate the network segment
IP Address:   11000000.10101000.00000001.01100100
Subnet Mask:  11111111.11111111.11111111.11100000
-----------------------------------------------
Network Addr: 11000000.10101000.00000001.01100000

This gives 192.168.1.96 in decimal as the network address.
For a /27 subnet, each segment (or subnet) has 32 IP addresses (2^5).
The range of this network segment is from 192.168.1.96 to 192.168.1.127.
```

Paste source code here, with a screenshot of a working application.

```
Please enter a valid IP address, i don't do edge-case handling:
167.162.52.67
Please enter a valid subnet mask, i don't do edge-case handling:
28
IPv4:           167.162.52.67
Subnet:         255.255.255.240
-------------------------------------------------
IPv4:           10100111.10100010.00110100.01000011
Subnet:         11111111.11111111.11111111.11110000
-------------------------------------------------
Network start:  10100111.10100010.00110100.01000000
Network end:    10100111.10100010.00110100.01001111
Network start:  167.162.52.64
Network end:    167.162.52.79

Process finished with exit code 0
```

```
import java.util.Scanner;

public class Main {
    static Scanner in = new Scanner(System.in);

    public static void main(String[] args) {
        String ipAddress;
        String subnet;

        // Get ip and mask
        System.out.println("Please enter a valid IP address, i don't do
edge-case handling:");
        ipAddress = in.nextLine();
        System.out.println("Please enter a valid subnet mask, i don't do
edge-case handling:");
        subnet = in.nextLine();

        var ip = stringToIntArray(ipAddress.split("\\."));
        var sub = stringToIntArray(subnet.split("\\."));

        // If only the number of bits is given, create the mask
        if (sub.length == 1) {
            sub = maskToSubnet(sub[0]);
        }

        // Print both in decimal
        System.out.print("IPv4:\t\t\t");
        writeAddress(ip);
        System.out.print("Subnet:\t\t\t");
        writeAddress(sub);
```

```
        System.out.println("-----------------------------------------------
----");

        // Print both in binary
        System.out.print("IPv4:\t\t\t");
        writeAddressAsBinary(ip);
        System.out.print("Subnet:\t\t\t");
        writeAddressAsBinary(sub);

        System.out.println("-----------------------------------------------
----");

        // Get the upper and lower bound of the network
        Integer[] netLowerBound = getNetwork(ip, sub);
        Integer[] netUpperBound = getNetworkUpper(ip, sub);

        // Print everything
        System.out.print("Network start:\t");
        writeAddressAsBinary(netLowerBound);
        System.out.print("Network end:\t");
        writeAddressAsBinary(netUpperBound);
        System.out.print("Network start:\t");
        writeAddress(netLowerBound);
        System.out.print("Network end:\t");
        writeAddress(netUpperBound);
    }

    // Returns the network address of a given ip and subnet mask
    // uses bitwise AND operator
    private static Integer[] getNetwork(Integer[] ip, Integer[] sub) {
        Integer[] net = new Integer[4];
        for (int i = 0; i < 4; i++) {
            net[i] = (ip[i] & sub[i]);
        }
        return net;
    }

    // Returns the upper bound of the network address given ip and subnet
maks
    // we first get the 'opposite' subnet mask and use bitwise OR to get
the upper bound
    private static Integer[] getNetworkUpper(Integer[] ip, Integer[] sub) {
        Integer[] net = new Integer[4];
        // 'flip'
        for (int i = 0; i < 4; i++) {
            sub[i] = 255 - sub[i];
        }
        // add 1s to the end
        for (int i = 0; i < 4; i++) {
            net[i] = (ip[i] | sub[i]);
        }
        return net;
    }
```

```
    //takes a given number and creates a subnet mask from it
    //eg 27 to 11111111.11111111.11111111.11100000
    private static Integer[] maskToSubnet(int mask) {
        Integer[] res = new Integer[4];
        // fill the array with the required amount of 1s
        for (int i = 0; i < mask; i++) {
            int oct = i / 8;
            if (res[oct] == null) {
                res[oct] = 0;
            }
            res[oct] = res[oct] * 10 + 1;
        }
        // fill the array with the required 0s if needed
        for (int i = 0; i < 4; i++) {
            while (res[i].toString().length() < 8) {
                res[i] *= 10;
            }
            res[i] = Integer.parseInt(res[i].toString(), 2);
        }
        return res;
    }

    //Creates an array of integers from a string, split on ','
    private static Integer[] stringToIntArray(String[] ips) {
        Integer[] res = new Integer[ips.length];
        for (int i = 0; i < ips.length; i++) {
            res[i] = Integer.parseInt(ips[i]);
        }
        return res;
    }

    //Writes a given address block to console
    private static void writeAddress(Integer[] ipAddress) {
        for (int i = 0; i < 4; i++) {
            System.out.print(ipAddress[i]);
            if (i != 3) {
                System.out.print(".");
            }
        }
        System.out.println();
    }

    //Writes a given address block as binary to console
    private static void writeAddressAsBinary(Integer[] ipAddress) {
        for (int i = 0; i < 4; i++) {
            System.out.print(String.format("%8s",
Integer.toBinaryString(ipAddress[i])).replace(' ', '0'));
            if (i != 3) {
                System.out.print(".");
            }
        }
        System.out.println();
    }
}
```

Ready? Save this file and export it as a pdf file with the name: **week6.pdf**