

# Template Week 6 – Networking

Student number:

## **Assignment 6.1: Working from home**

Screenshot installation openssh-server:

Screenshot successful SSH command execution:

Screenshot successful execution SCP command:

Screenshot remmina:

## **Assignment 6.2: IP addresses websites**

Relevant screenshots nslookup command:

Screenshot website visit via IP address:

## **Assignment 6.3: subnetting**

How many IP addresses are in this network configuration 192.168.110.128/25?

What is the usable IP range to hand out to the connected computers?

Check your two previous answers with this calculator:

<https://www.calculator.net/ip-subnet-calculator.html>

Explain the above calculation in your own words.

## Assignment 6.4: HTML

Screenshot IP address Ubuntu VM:

Screenshot of Site directory contents:

Screenshot python3 webserver command:

Screenshot web browser visits your site

## 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.

```
Week6Bonus Version control
Project
Sandbox C:\Users\kayla\OneDrive - Sax
src
Application
Sandbox.iml
External Libraries
Scratches and Consoles
Application.java
1 import nl.saxion.app.SaxionApp;
2
3 public class Application implements Runnable {
4
5     public static void main(String[] args) {
6         SaxionApp.start(new Application(), width: 1024, height: 768);
7     }
8
9     public void run() {
10         int menuOption = -1;
11         while (menuOption != 0) {
12
13             SaxionApp.println(text: "1. Is number odd?");
14             SaxionApp.println(text: "2. Is number a power of 2?");
15             SaxionApp.println(text: "3. Two's complement of number?");
16             SaxionApp.println(text: "4. Calculate network segment");
17             SaxionApp.print(text: "Choose a menu option: ");
18             menuOption = SaxionApp.readInt();
19
20             SaxionApp.println();
21             if (menuOption == 1) { //Is number odd?
22                 isNumberOdd(askForInput());
23             } else if (menuOption == 2) { //Is number a power of 2?
24                 isPowerOfTwo(askForInput());
25             } else if (menuOption == 3) { //Two's complement of number
26                 twoComplement(askForInput());
27             } else if (menuOption == 4) { //calculate network segment
28                 calculateNetworkSegment();
29             }
30             SaxionApp.pause();
31             SaxionApp.clear();
32         }
33     }
34
35     public int askForInput() { 3 usages
```

```
Week6Bonus Version control
Project
Sandbox C:\Users\kayla\OneDrive - Sax
src
Application
Sandbox.iml
External Libraries
Scratches and Consoles
Application.java
35 public class Application implements Runnable {
36
37     public int askForInput() { 3 usages
38         SaxionApp.print(text: "Enter a number: ");
39         return SaxionApp.readInt();
40     }
41
42     public void isNumberOdd(int input) { 1 usage
43         if ((input & 1) == 1) {
44             SaxionApp.println(text: "number is odd");
45         } else {
46             SaxionApp.println(text: "number is not odd");
47         }
48     }
49
50     public void isPowerOfTwo(int input) { 1 usage
51         if (((input - 1) & input) == 0) {
52             SaxionApp.println(text: "number is a power of 2");
53         } else {
54             SaxionApp.println(text: "number is not a power of 2");
55         }
56     }
57
58     public void twoComplement(int input) { 1 usage
59         int inputTwoComplement = -input + 1;
60         SaxionApp.println(text: "Two's complement: " + inputTwoComplement);
61     }
62
63     public void calculateNetworkSegment() { 1 usage
64         SaxionApp.print(text: "Enter IP address: ");
65         String ipAddress = SaxionApp.readString();
66         SaxionApp.print(text: "Enter subnet mask: ");
67         String subnetMask = SaxionApp.readString();
68
69         int ip = convertIpToBinary(ipAddress);
70         int mask = convertIpToBinary(subnetMask);
71
72         // Calculate network address using bitwise AND
73         int networkAddress = ip & mask;
74
75         // Display network address in decimal format
76         String networkAddrString = convertBinaryToIp(networkAddress);
77         SaxionApp.println(text: "Network Address: " + networkAddrString);
78
79         // Calculate range
80         int numberOfAddresses = ~mask + 1; // Nr of addresses in the subnet
81         int firstIp = networkAddress;
82         int lastIp = networkAddress + numberOfAddresses - 1;
83
84         SaxionApp.println(text: "Range: " + convertBinaryToIp(firstIp) + " - " + convertBinaryToIp(lastIp));
85     }
86
87     private int convertIpToBinary(String ipAddress) { 2 usages
88         String[] parts = ipAddress.split(regex: "\\.");
89         int binary = 0;
90         for (String part : parts) {
91             binary = (binary << 8) | Integer.parseInt(part);
92         }
93         return binary;
94     }
```

```
Week6Bonus Version control
Project
Sandbox C:\Users\kayla\OneDrive - Sax
src
Application
Sandbox.iml
External Libraries
Scratches and Consoles
Application.java
35 public class Application implements Runnable {
36
37     public int askForInput() { 3 usages
38         SaxionApp.print(text: "Enter a number: ");
39         return SaxionApp.readInt();
40     }
41
42     public void isNumberOdd(int input) { 1 usage
43         if ((input & 1) == 1) {
44             SaxionApp.println(text: "number is odd");
45         } else {
46             SaxionApp.println(text: "number is not odd");
47         }
48     }
49
50     public void isPowerOfTwo(int input) { 1 usage
51         if (((input - 1) & input) == 0) {
52             SaxionApp.println(text: "number is a power of 2");
53         } else {
54             SaxionApp.println(text: "number is not a power of 2");
55         }
56     }
57
58     public void twoComplement(int input) { 1 usage
59         int inputTwoComplement = -input + 1;
60         SaxionApp.println(text: "Two's complement: " + inputTwoComplement);
61     }
62
63     public void calculateNetworkSegment() { 1 usage
64         SaxionApp.print(text: "Enter IP address: ");
65         String ipAddress = SaxionApp.readString();
66         SaxionApp.print(text: "Enter subnet mask: ");
67         String subnetMask = SaxionApp.readString();
68
69         int ip = convertIpToBinary(ipAddress);
70         int mask = convertIpToBinary(subnetMask);
71
72         // Calculate network address using bitwise AND
73         int networkAddress = ip & mask;
74
75         // Display network address in decimal format
76         String networkAddrString = convertBinaryToIp(networkAddress);
77         SaxionApp.println(text: "Network Address: " + networkAddrString);
78
79         // Calculate range
80         int numberOfAddresses = ~mask + 1; // Nr of addresses in the subnet
81         int firstIp = networkAddress;
82         int lastIp = networkAddress + numberOfAddresses - 1;
83
84         SaxionApp.println(text: "Range: " + convertBinaryToIp(firstIp) + " - " + convertBinaryToIp(lastIp));
85     }
86
87     private int convertIpToBinary(String ipAddress) { 2 usages
88         String[] parts = ipAddress.split(regex: "\\.");
89         int binary = 0;
90         for (String part : parts) {
91             binary = (binary << 8) | Integer.parseInt(part);
92         }
93         return binary;
94     }
```

```
1 public class Application implements Runnable {
2     public void calculateNetworkSegment() { // usage
3
4         // Calculate range
5         int numberOfAddresses = -mask + 1; // Nr of addresses in the subnet
6         int firstIp = networkAddress;
7         int lastIp = networkAddress + numberOfAddresses - 1;
8
9         SaxionApp.printlnf("Range: " + convertBinaryToIp(firstIp) + " - " + convertBinaryToIp(lastIp));
10    }
11
12    private int convertIpToBinary(String ipAddress) { // 2 usages
13        String[] parts = ipAddress.split("\\.");
14        int binary = 0;
15        for (String part : parts) {
16            binary = (binary << 8) | Integer.parseInt(part);
17        }
18        return binary;
19    }
20
21    private String convertBinaryToIp(int binary) { // 3 usages
22        String part1 = String.valueOf((binary >> 24) & 0xFF);
23        String part2 = String.valueOf((binary >> 16) & 0xFF);
24        String part3 = String.valueOf((binary >> 8) & 0xFF);
25        String part4 = String.valueOf(binary & 0xFF);
26
27        return part1 + "." + part2 + "." + part3 + "." + part4;
28    }
29 }
30
31 }
```

Run Application

Saxion Drawingboard

```
1. Is number odd?
2. Is number a power of 2?
3. Two's complement of number?
4. Calculate network segment
Choose a menu option: 4

Enter IP address: 192.168.1.100
Enter subnet mask: 255.255.255.224
Network Address: 192.168.1.96
Range: 192.168.1.96 - 192.168.1.127

PRESS ANY KEY TO CONTINUE
```

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