

CALCULAODRA BINARIA

Alumne: Jordi Ribellas Ramos

Professor: Francis Real

Curs: 2022/23

1- Enunciat

- o En aquesta activitat es demana:

Crea una calculadora de IPs.

A partir de una IP decimal y de su mascara en formato reducido, calcula:

- La IP binaria
- La mascara binaria
- La IP de la red

The screenshot shows a web-based application titled "IP Calculadora". It features four input sections, each with a label and four text boxes separated by dots, followed by a slash and a final text box. The "IP Decimal:" section contains the values 192, 186, 100, 25, and 20. The "IP Binaria:" section contains the binary values 11000000, 10111010, 01100100, and 00011001. The "Mascara de Subred:" section contains 11111111, 11111111, 11110000, and 00000000. The "Red:" section contains 11000000, 10111010, 01100000, and 00000000. Below these sections is a "Calcular" button. At the bottom, there is a text area with the placeholder text "Espai per mostrar missatges d'error".

- El campo inferior se utilizará para mostrar mensajes de error que puedan surgir, como por ejemplo si faltan datos, si hay algún número superior a 255, y los que se os ocurran.
- En el caso de que el usuario no informase la máscara de subred no se considerará error, sino que se entenderá que deberán aplicarse las máscaras de subred asociadas a las tipologías estándares A, B, C, D y E, teniendo en cuenta que en las dos últimas (D y E) no se debe informar ninguna máscara de subred.

2- Codi

- o En aquest apartat es veurà el meu codi.

```
import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class CalculadoraBinaria extends JFrame {
    private JTextField[] ipDecimal = new JTextField[5];
    private JTextField ipMascara = new JTextField(3);
    private JTextField[] ipBinari = new JTextField[5];
    private JTextField[] ipMasBinari = new JTextField[5];
    private JTextField[] xarxaBinari = new JTextField[5];
    private JTextField Error = new JTextField(10);

    public CalculadoraBinaria() {

        super("Calculadora IP Binaria");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(900, 300);

        setTextIpDecimal();
        setTextBi();
        setTextMascara();
        setTextXarxaBinari();

        // Decimal
        JPanel panelIP = new JPanel();
        panelIP.setLayout(new FlowLayout());
        panelIP.add(ipDecimal[1]);
        panelIP.add(new JLabel("."));
        panelIP.add(ipDecimal[2]);
        panelIP.add(new JLabel("."));
        panelIP.add(ipDecimal[3]);
        panelIP.add(new JLabel("."));
        panelIP.add(ipDecimal[4]);
        panelIP.add(new JLabel("/"));
        panelIP.add(ipMascara);

        // Binari
        JPanel panelBinari = new JPanel();
        panelBinari.setLayout(new FlowLayout());
        panelBinari.add(ipBinari[1]);
        panelBinari.add(new JLabel("."));
        panelBinari.add(ipBinari[2]);
        panelBinari.add(new JLabel("."));
        panelBinari.add(ipBinari[3]);
        panelBinari.add(new JLabel("."));
        panelBinari.add(ipBinari[4]);
```

```

// Mascara Binari
JPanel panelSubred = new JPanel();
panelSubred.setLayout(new FlowLayout());
panelSubred.add(ipMasBinari[1]);
panelSubred.add(new JLabel("."));
panelSubred.add(ipMasBinari[2]);
panelSubred.add(new JLabel("."));
panelSubred.add(ipMasBinari[3]);
panelSubred.add(new JLabel("."));
panelSubred.add(ipMasBinari[4]);

// XarxaBinari
JPanel panelRed = new JPanel();
panelRed.setLayout(new FlowLayout());
panelRed.add(xarxaBinari[1]);
panelRed.add(new JLabel("."));
panelRed.add(xarxaBinari[2]);
panelRed.add(new JLabel("."));
panelRed.add(xarxaBinari[3]);
panelRed.add(new JLabel("."));
panelRed.add(xarxaBinari[4]);

// Grid organitzat
JPanel panelDatos = new JPanel();
GridLayout gl = new GridLayout(5, 2);
panelDatos.setLayout(gl);
panelDatos.add(new JLabel("IP Decimal: "));
panelDatos.add(panelIP);
panelDatos.add(new JLabel("IP Binaria: "));
panelDatos.add(panelBinari);
panelDatos.add(new JLabel("Mascara Subred: "));
panelDatos.add(panelSubred);
panelDatos.add(new JLabel("Red: "));
panelDatos.add(panelRed);
Error.setEditable(false);
panelDatos.add(Error);

// Boto de calcular
JButton boton = new JButton("Calcular");
boton.setBounds(10, 10, 10, 10);
boton.addActionListener(new EventBoto());

// Panell Sud
JPanel panelSud = new JPanel();
panelSud.setLayout(new FlowLayout());
panelSud.add(boton);

Container cp = getContentPane();
cp.add(panelDatos, BorderLayout.CENTER);
cp.add(panelSud, BorderLayout.SOUTH);
}

public class EventBoto implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {

        int a = 0;
        boolean errors = false;

```

```

        int mascara = 0;

        for (int i = 1; i < ipDecimal.length; i++) {

            if (ipDecimal[i].getText().isEmpty() &&
ipMascara.getText().isEmpty()) {
                Error.setText("Hi ha algun camp buit");
                errors = true;

            } else if
(!isNumero(ipDecimal[i].getText().toString())) {
                Error.setText("Algun camp de la ip no te
numeros");

                errors = true;

            }else {
                a = Integer.parseInt(ipDecimal[i].getText());
                mascara =
Integer.parseInt(ipMascara.getText());

            }

            if (a > 255 || a < 0) {
                Error.setText("La Ip no es correcta");
                errors = true;

            } else if (mascara > 32 || mascara < 0) {
                Error.setText("La mascara no es correcta");
                errors = true;

            }

        }

        if (errors == false) {
            calcularIP();
            calcularMascara();
            calcularXarxa();
            Error.setText("");
        } else {
            for (int i = 1; i < 5; i++) {
                ipBinari[i].setText("");
                ipMasBinari[i].setText("");
                xarxaBinari[i].setText("");

            }

        }

    }

}

public void calcularIP() {
    int a, operador;
    String intString = "", provisional = "";

    for (int i = 1; i < ipDecimal.length; i++) {

        intString = "";
        a = Integer.parseInt(ipDecimal[i].getText());

```

```

        do {
            operador = a % 2;
            intString = operador + intString;
            a = a / 2;
        } while (a > 0);

        while (intString.length() <= 7) {
            intString = "0" + intString;
        }

        StringBuilder sb = new StringBuilder(provisional);
        sb.reverse();

        ipBinari[i].setText(intString);
    }

}

public void calcularMascara() {
    int a = 0, operador;
    String intString = "";
    int mascara = Integer.parseInt(ipMascara.getText());

    for (int i = 0; i < mascara; i++) {

        intString = "";
        do {
            operador = a % 2;
            intString = operador + intString;
            a = a / 2;
        } while (a > 0);

        intString = "1";

        for (i = 1; i < mascara; i++) {
            intString = intString + "1";
        }

        do {
            intString = intString + "0";
        } while (intString.length() < 33);

    }

    for (int j = 1; j <= 4; j++) {
        ipMasBinari[j].setText(intString.substring((j - 1) * 8, (j
- 1) * 8 + 8));
    }

}

public void calcularXarxa() {
    for (int i = 1; i <= 4; i++) {
        StringBuilder sb = new StringBuilder();
        for (int j = 0; j < 8; j++) {
            ipMasBinari[i].getText().substring(j, j + 1);

```

```

        if (ipMasBinari[i].getText().substring(j, j +
1).equals("1")) {
            sb.append(ipBinari[i].getText().substring(j,
j + 1));
        } else {
            sb.append("0");
        }
    }
    xarxaBinari[i].setText(sb.toString());
}

private static boolean isNumero(String cadena) {
    try {
        Integer.parseInt(cadena);
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

private void setTextIpDecimal() {
    for (int i = 1; i < 5; i++) {
        ipDecimal[i] = new JTextField(3);
    }
}

private void setTextBi() {
    for (int i = 1; i < 5; i++) {
        ipBinari[i] = new JTextField(8);
        ipBinari[i].setEditable(false);
    }
}

private void setTextMascara() {
    for (int i = 1; i < 5; i++) {
        ipMasBinari[i] = new JTextField(8);
        ipMasBinari[i].setEditable(false);
    }
}

private void setTextXarxaBinari() {
    for (int i = 1; i < 5; i++) {
        xarxaBinari[i] = new JTextField(8);
        xarxaBinari[i].setEditable(false);
    }
}
}

```

3- Documentació del codi:

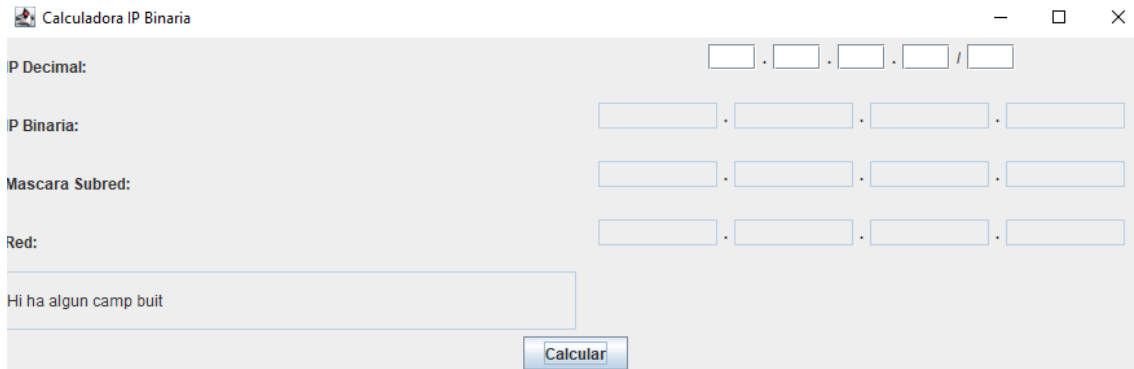
- o En aquest apartat comentaré el codi funció per funció

- 3.1. El primer que podem veure al codi es la declaració dels JTextField que farem servir en el codi. Aquests son arrays ja que tractarem les dades com arrays y no com a variables individuals.
- 3.2. Després podem veure com declarem la nova finestra en la que treballarem i li donem les seves mides, sota d'això podem veure com criem a les funcions que estan al final del codi que se n'encarreguen d'omplir cada JTextField amb un JTextField de 8 posicions i en el cas dels camps que corresponen al resultat de la calculadora els posem en que no es puguin editar.
- 3.3. Sota d'aquest codi (sobre la línia 34) podem veure com declarem un JPanel i li apliquem l'estructura d'un FlowLayout i dins d'aquest panell li anem afegint els camps i els "." o "/" depenent del cas.
- 3.4. Sobre la línia 80, podem veure com creem el panell que serà la base de l'aplicació i li assignem l'estructura del GridLayout i li afegim uns JLabel i els panells anteriors. Al final d'aquest bloc podem veure com declarem on anirà el camp on es mostraran els errors.
- 3.5. Tot seguit podem veure com declarem el botó de calcular i li assignem les mides, i com li diem que quan es cliqui cridi a la funció "EventBoto"
 - a. Aquesta funció el que fa es: recorre tota la llargada de la variable "ipDecimal" i aquí dins comprovem:
 - i. Si hi ha algun camp buit
 - ii. Si el que li hem introduït son numeros
 - iii. Si els numeros introduïts estan entre 255 i 0
 - iv. Si la mascara esta entre 32 i 0
 - b. Totes aquestes comprovacions anteriors es per a fer que l'usuari introdueixi numeros correctes, si ho fa llavors passem a la línia 145 on executem les funcions per a fer la conversió de la IP i la mascara a binari. Si aquesta funció no es compleix assignarem tots els JTextField a "res" per a mostrar en blanc els camps en cas de que hi hagués algún error.

4- Proves

- Aquí farem les proves de que l'aplicació funciona correctament.

1) Comprovació de que si no omplim cap camp controlem l'error



Calculadora IP Binaria

P Decimal: . . . /

P Binaria: . . .

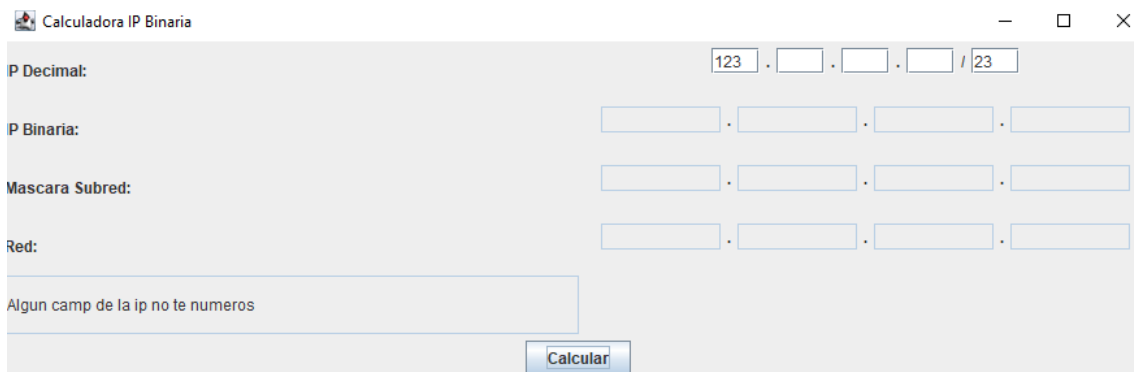
Mascara Subred: . . .

Red: . . .

Hi ha algun camp buit

Calcular

2) Comprovació de que si omplim algun camp de la IP i la mascara, ens mostra error



Calculadora IP Binaria

P Decimal: . . . /

P Binaria: . . .

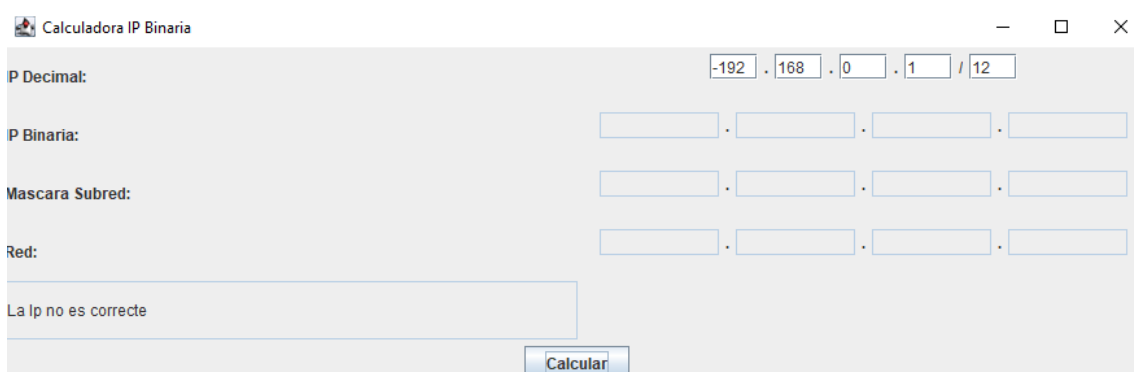
Mascara Subred: . . .

Red: . . .

Algun camp de la ip no te numeros

Calcular

3) Comprovació de que si introduïm numeros negatiu ens dona error



Calculadora IP Binaria

P Decimal: . . . /

P Binaria: . . .

Mascara Subred: . . .

Red: . . .

La Ip no es correcte

Calcular

4) Comprovació de que si introduïm lletres ens mostra error

Calculadora IP Binaria

IP Decimal: asd . 168 . 0 . 1 / 12

IP Binaria:

Mascara Subred:

Red:

Algun camp de la ip no te numeros

Calcular

5) Comprovació de que si no posem una mascara correcte ens dona error

Calculadora IP Binaria

IP Decimal: 192 . 168 . 0 . 15 / 90

IP Binaria:

Mascara Subred:

Red:

La mascara no es correcte

Calcular

6) Comprovacions del funcionament correcte

Calculadora IP Binaria

IP Decimal: 192 . 168 . 5 . 10 / 20

IP Binaria: 11000000 . 10101000 . 00000101 . 00001010

Mascara Subred: 11111111 . 11111111 . 11110000 . 00000000

Red: 11000000 . 10101000 . 00000000 . 00000000

Calcular

Calculadora IP Binaria

IP Decimal: 127 . 0 . 0 . 0 / 17

IP Binaria: 01111111 . 00000000 . 00000000 . 00000000

Mascara Subred: 11111111 . 11111111 . 10000000 . 00000000

Red: 01111111 . 00000000 . 00000000 . 00000000

Calcular