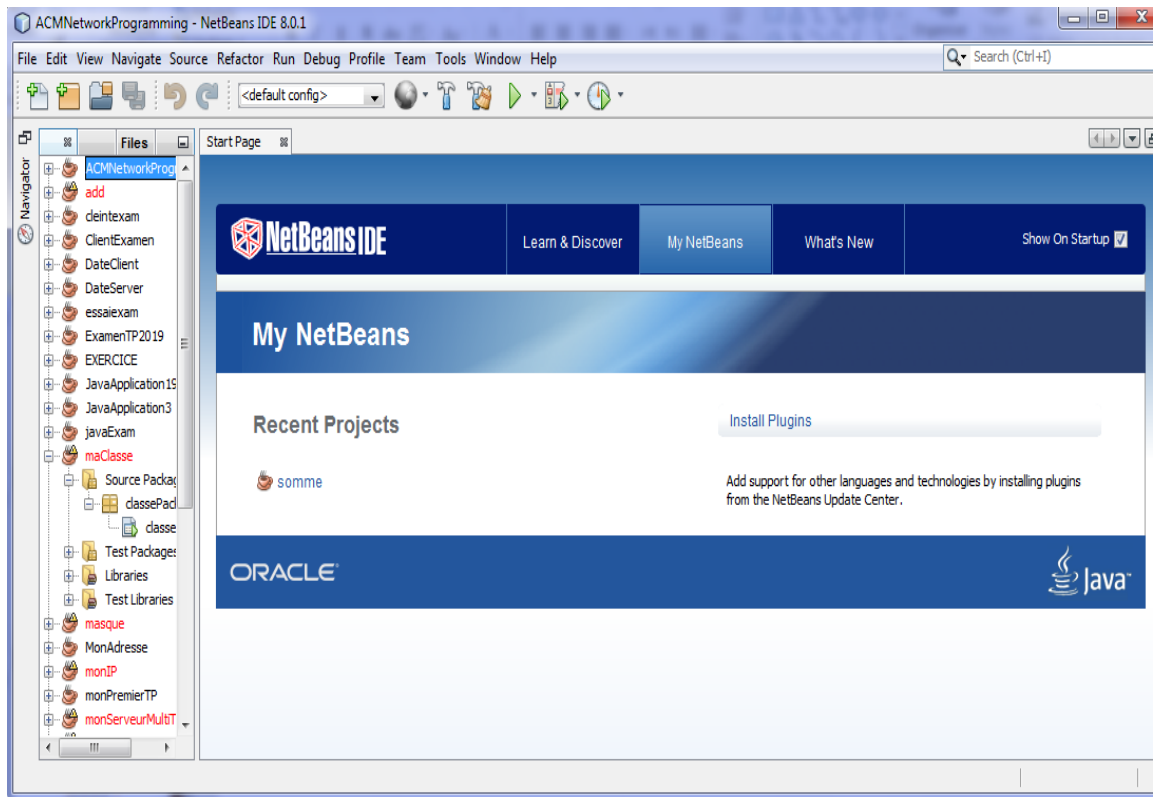


Séance 1

Classes d'adresse IP



Adresse IPV4

- Une @IP sert à **identifier** d'une manière unique une machine connectée sur un réseau (privé ou internet)
- Une @IP **Version 4** tient sur **32 bits** (4 octets).
- Notation **décimale** (4 décimaux séparés par des **points**).
- Chaque décimale prend donc une valeur entre **0- 255**.

Adresse IPV4

Exemple

Adresse IP en binaire :

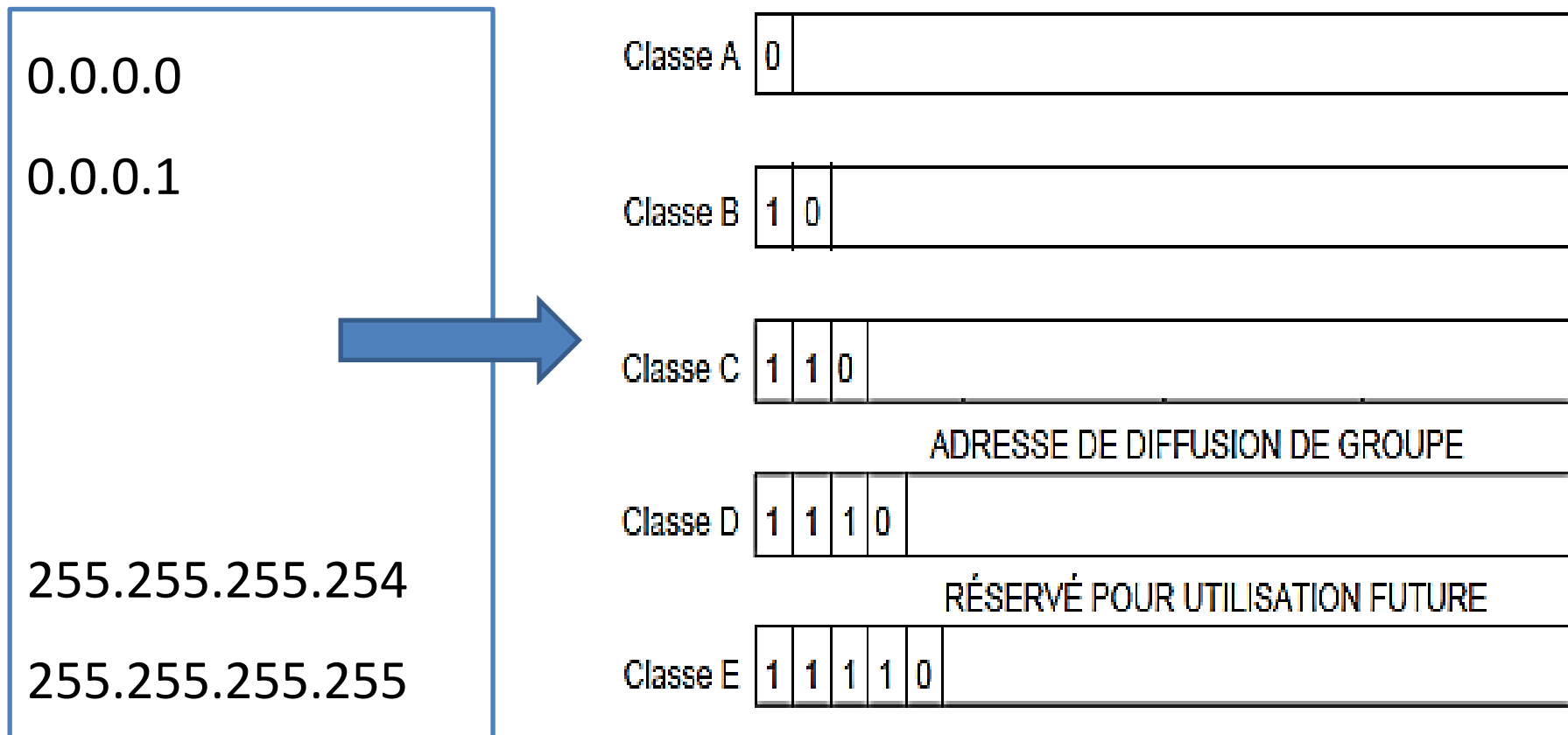
11000001 00011011 00101101 00100001

En notation décimale pointée:

193.27.45.33

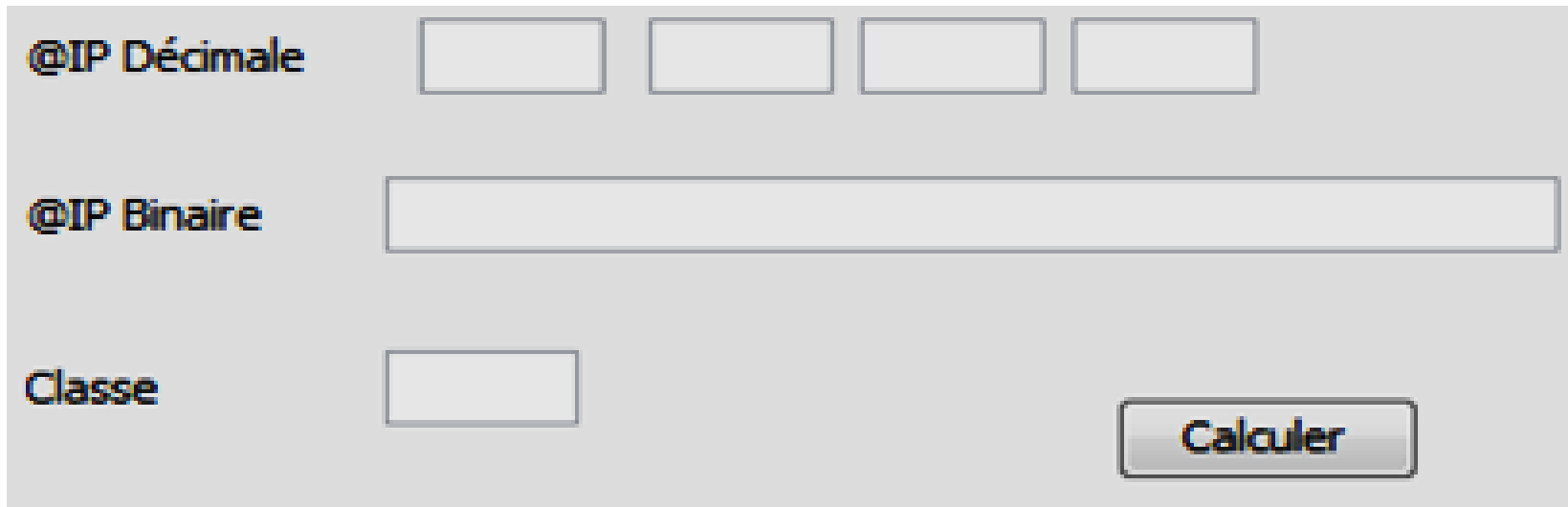
Classes d' Adresse IP

Comme une @IPV4 tient sur 32 bits, alors on peut avoir : 2^{32} @IP différentes.



Créer un programme avec GUI qui permet de :

- Saisir une @ IP en **notation décimale** (les 4 octets séparés)
- Convertir l'adresse en **Binaire**
- Trouver **la classe** de l'adresse



The image shows a graphical user interface (GUI) for IP address conversion and classification. It features three input fields and a calculation button. The first input field is labeled '@IP Décimale' and consists of four separate boxes for entering the four octets of an IP address. The second input field is labeled '@IP Binaire' and is a single wide box for entering the binary representation. The third input field is labeled 'Classe' and is a single box for entering the resulting class. A button labeled 'Calculer' is positioned to the right of the 'Classe' field.

@IP Décimale	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
@IP Binaire	<input type="text"/>			
Classe	<input type="text"/>	<input type="button" value="Calculer"/>		

Création du Projet

- File > **New Project**
- Categories / **Java** & Projects/ **Java Application** > Next
- Project NameNommer le programme (eg. **ClassIP**)
- **Décocher** (Create Main Class)
- Finish

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

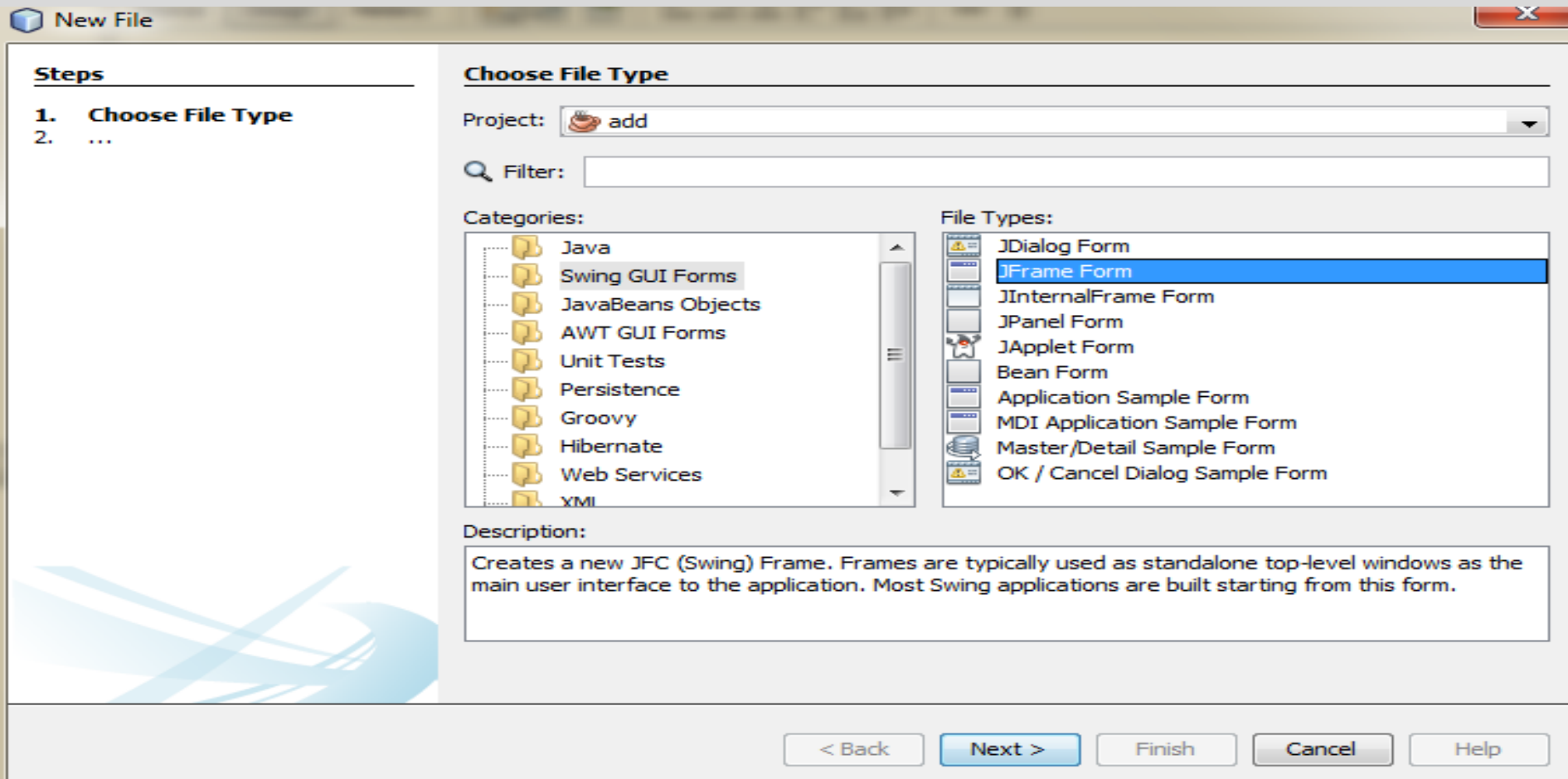
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

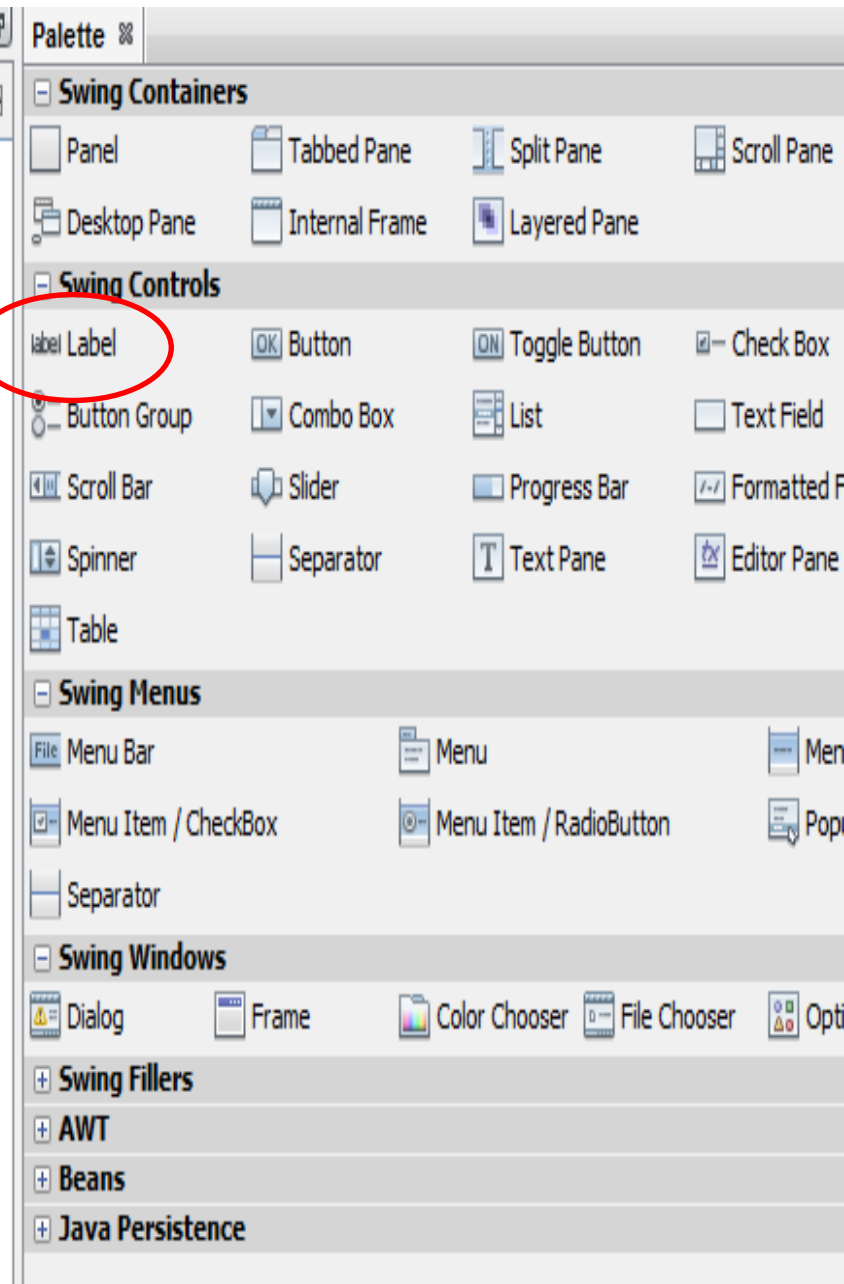
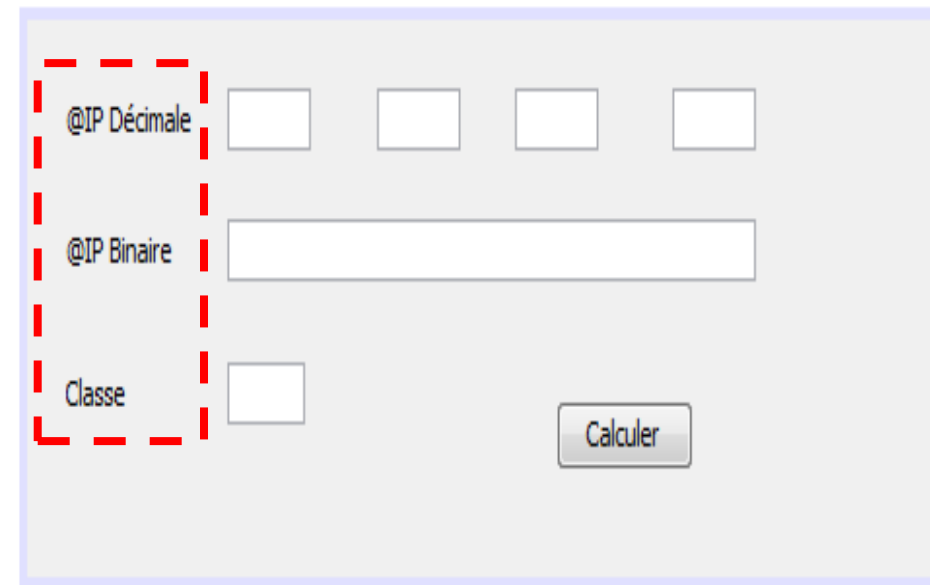
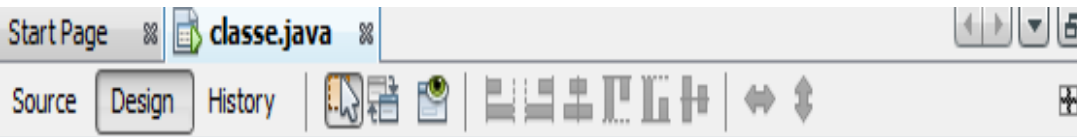
☒ **Create Main Class**

Création de l'interface

- File > **New File**
- Categories / **Swing Gui Forms** & File Types/ **Jframe Form** > Next
- Class Name Nommer la classe (eg, **ClassIPGUI**)
- Package Nommer le package (eg, **ClassIPack**)
- Finish



Etiquettes



Champs de Texte

The image displays a Java Swing IDE interface. The main window, titled "classe.java", shows a design view of a form. The form contains three labels: "@IP Décimale", "@IP Binaire", and "Classe". The "@IP Décimale" label is followed by four small text input fields. The "@IP Binaire" label is followed by a single wide text input field. The "Classe" label is followed by a single small text input field. A "Calculer" button is located at the bottom right of the form. A red dashed rectangle highlights the area containing the text input fields. The right side of the IDE shows a "Palette" window with various Swing components. The "Swing Controls" section is expanded, and the "Text Field" component is highlighted with a red circle. Other components visible in the palette include "Swing Containers" (Panel, Tabbed Pane, Split Pane, Scroll Pane, Desktop Pane, Internal Frame, Layered Pane), "Swing Menus" (Menu Bar, Menu, Menu Item / CheckBox, Menu Item / RadioButton, Separator), and "Swing Windows" (Dialog, Frame, Color Chooser, File Chooser, Op).

Start Page classe.java

Source Design History

@IP Décimale

@IP Binaire

Classe

Calculer

Palette

Swing Containers

- Panel
- Tabbed Pane
- Split Pane
- Scroll Pane
- Desktop Pane
- Internal Frame
- Layered Pane

Swing Controls

- Label
- OK Button
- ON Toggle Button
- Check Box
- Button Group
- Combo Box
- List
- Text Field
- Scroll Bar
- Slider
- Progress Bar
- Formatted
- Spinner
- Separator
- Text Pane
- Editor Pane
- Table

Swing Menus

- File Menu Bar
- Menu
- Menu Item / CheckBox
- Menu Item / RadioButton
- Separator

Swing Windows

- Dialog
- Frame
- Color Chooser
- File Chooser
- Op

Bouton

The image shows a Java Swing IDE interface. The main window displays a form with the following elements:

- Four text input fields for "@IP Décimale".
- A single text input field for "@IP Binaire".
- A single text input field for "Classe".
- A "Calculer" button, which is highlighted with a red dashed border.

The right side of the IDE shows a "Palette" of Swing components:

- Swing Containers:** Panel, Tabbed Pane, Split Pane, Scroll Pane, Desktop Pane, Internal Frame, Layered Pane.
- Swing Controls:** Label, OK Button (circled in red), Toggle Button, Check Box, Button Group, Combo Box, List, Text Field, Scroll Bar, Slider, Progress Bar, Formatted Text Area, Spinner, Separator, Text Pane, Editor Pane, Table.
- Swing Menus:** Menu Bar, Menu, MenuItem / CheckBox, MenuItem / RadioButton, Separator.
- Swing Windows:** Dialog, Frame, Color Chooser, File Chooser, Open File Dialog.

Code du bouton Calculer (double clic)

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}
```

```
int    oct1,oct2,oct3,oct4;
```

```
String oct1b, oct2b,oct3b,oct4b;
```

```
oct1  = Integer.parseInt(jTextField1.getText());
```

```
oct1b = Integer.toBinaryString(oct1);
```

```
while (oct1b.length() <8) { oct1b = "0" +oct1b ;}
```

```
// on fait de même pour le reste des octets
```

```
jTextField5.setText(oct1b+" "+oct2b+" "+oct3b+" "+oct4b);
```

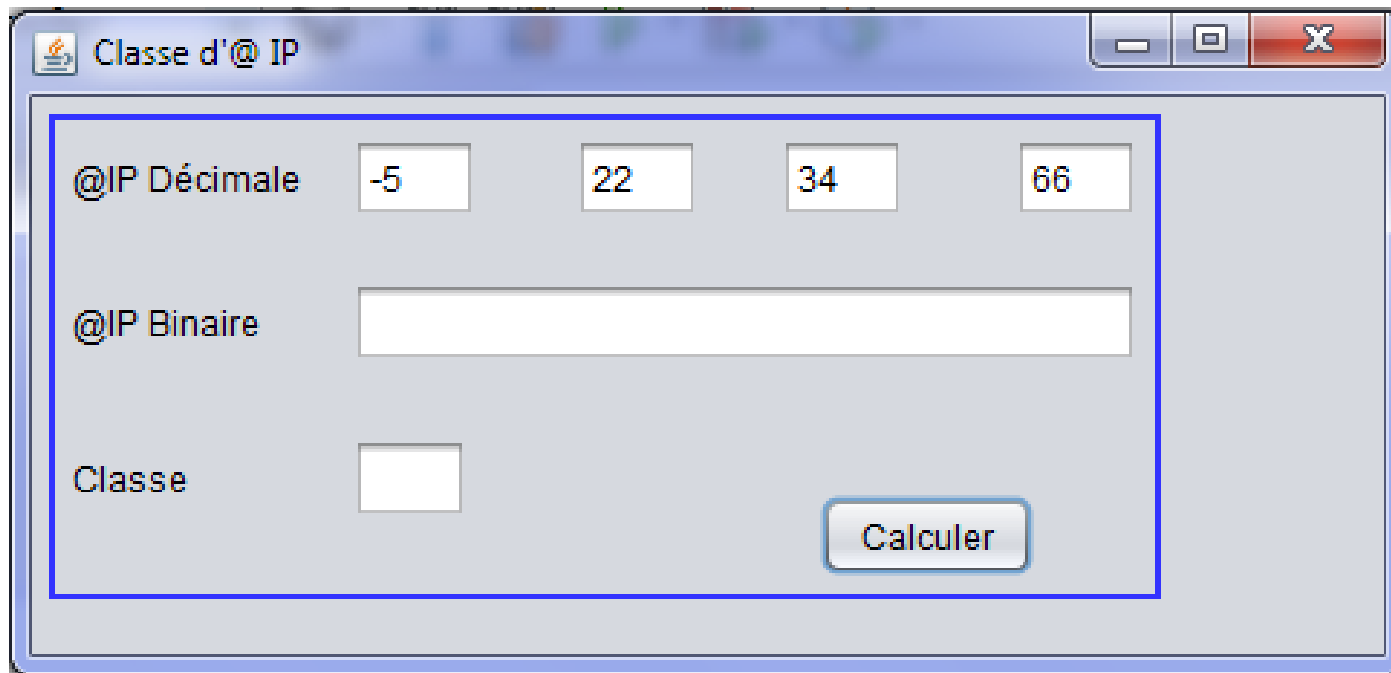
```
if (oct1b.startsWith("0")) {jTextField6.setText("A");}  
else{  
if (oct1b.startsWith("10")) {jTextField6.setText("B");}  
else{  
if (oct1b.startsWith("110")) {jTextField6.setText("C");}  
else{  
if (oct1b.startsWith("1110")) {jTextField6.setText("D");}  
else{  
if (oct1b.startsWith("1111")) {jTextField6.setText("E");}}}}}}}
```

Classe d'@ IP

@IP Décimale	195	12	0	7
@IP Binaire	11000011 00001100 00000000 00000111			
Classe	C			

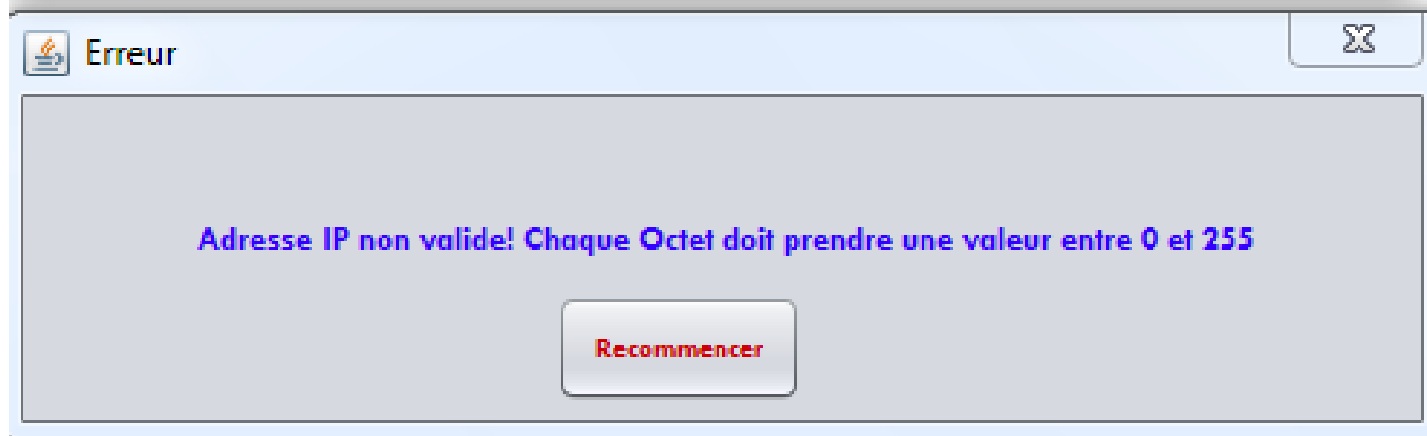
Calculer

...et si l'@IP est non valide ?



The screenshot shows a window titled "Classe d'@ IP". It contains three input fields: "@IP Décimale" with the value "-5", "@IP Binaire" which is empty, and "Classe" which is empty. A "Calculer" button is located at the bottom right of the input area. A blue rectangular box highlights the input fields and the "Calculer" button.

Field	Value
@IP Décimale	-5
@IP Binaire	
Classe	



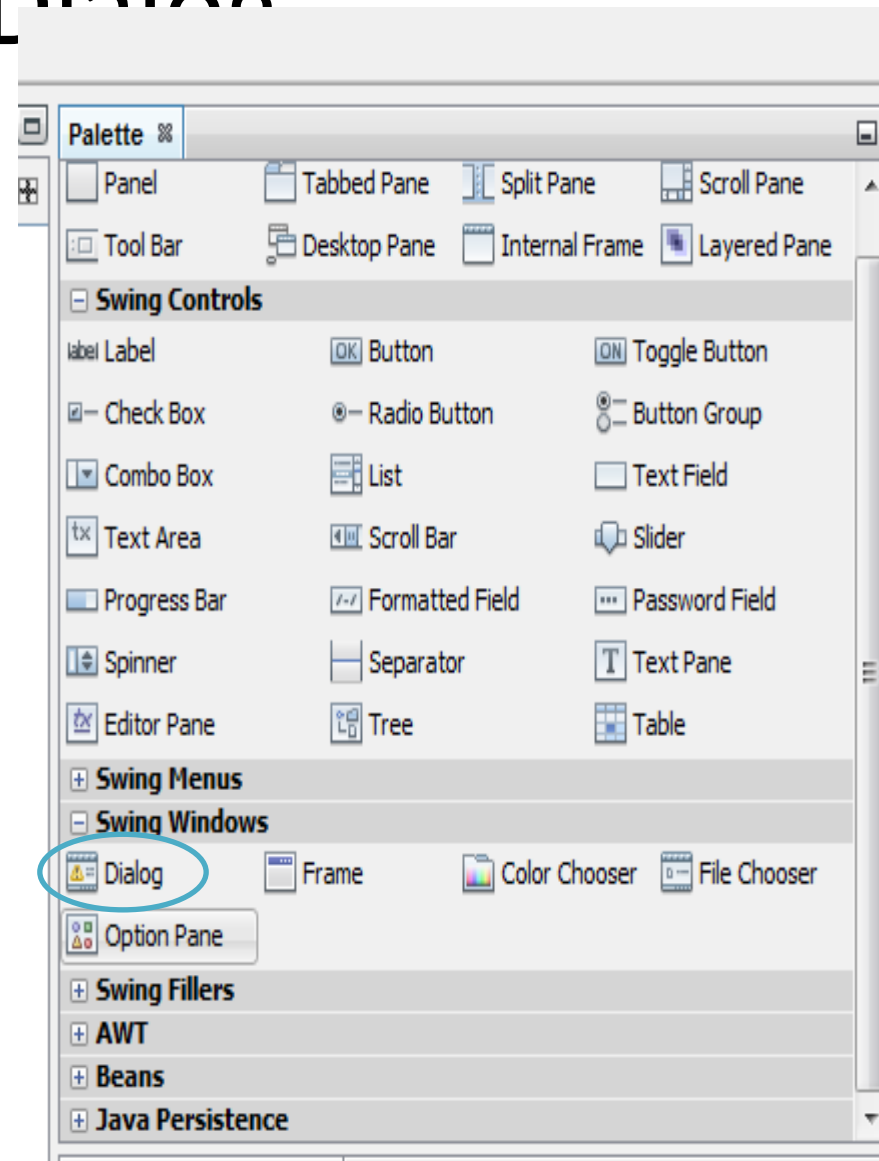
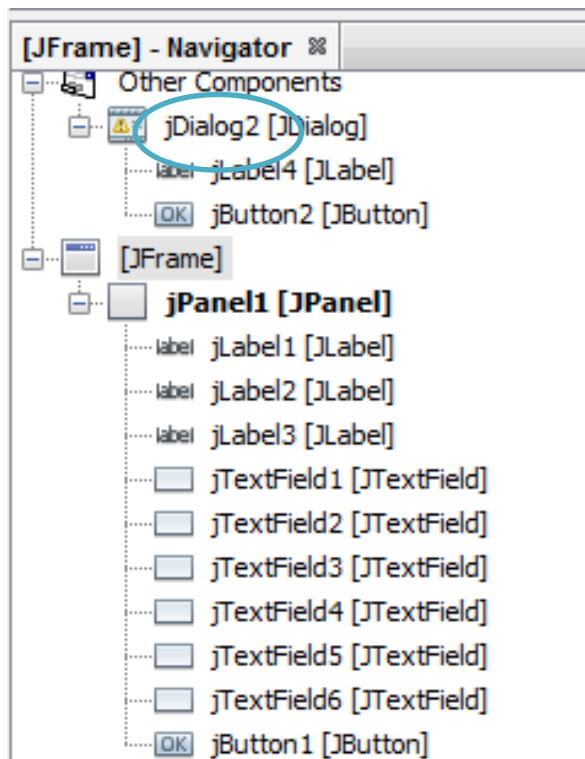
The screenshot shows an error dialog box titled "Erreur". It contains a message in blue text: "Adresse IP non valide! Chaque Octet doit prendre une valeur entre 0 et 255". A "Recommencer" button is located at the bottom center.

Adresse IP non valide! Chaque Octet doit prendre une valeur entre 0 et 255

Recommencer

Fenêtre Dialog

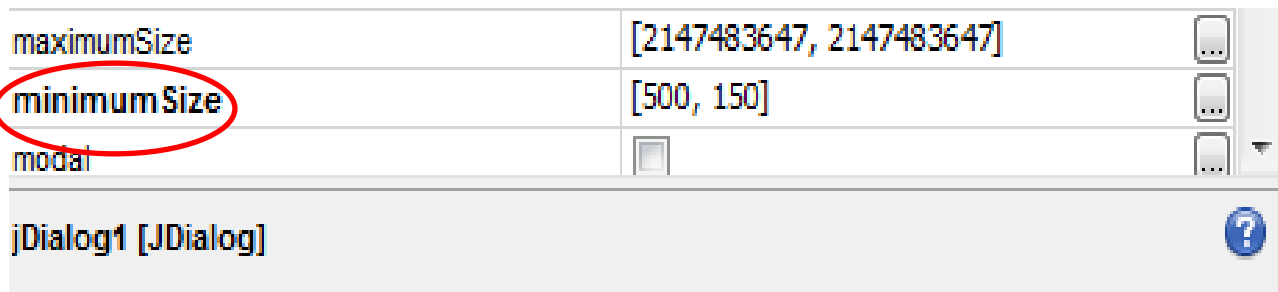
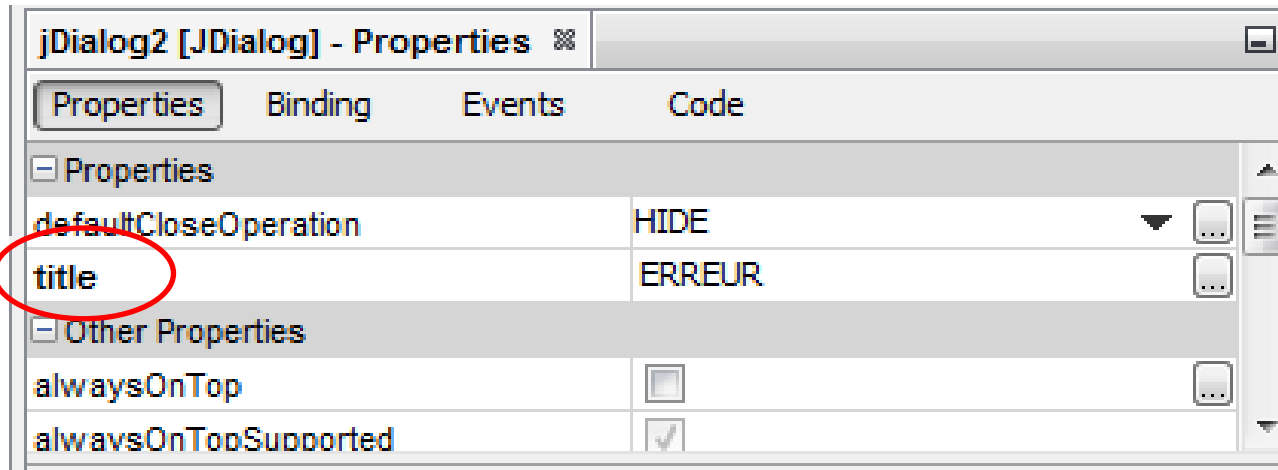
Design this Container



Ajouter des composants à Dialog

Adresse IP non valide! Chaque Octet doit prendre une valeur entre 0 et 255

Recommencer



Ajouter le Test

```
oct1= Integer.parseInt(jTextField1.getText());  
oct2= Integer.parseInt(jTextField2.getText());  
oct3= Integer.parseInt(jTextField3.getText());  
oct4= Integer.parseInt(jTextField4.getText());
```

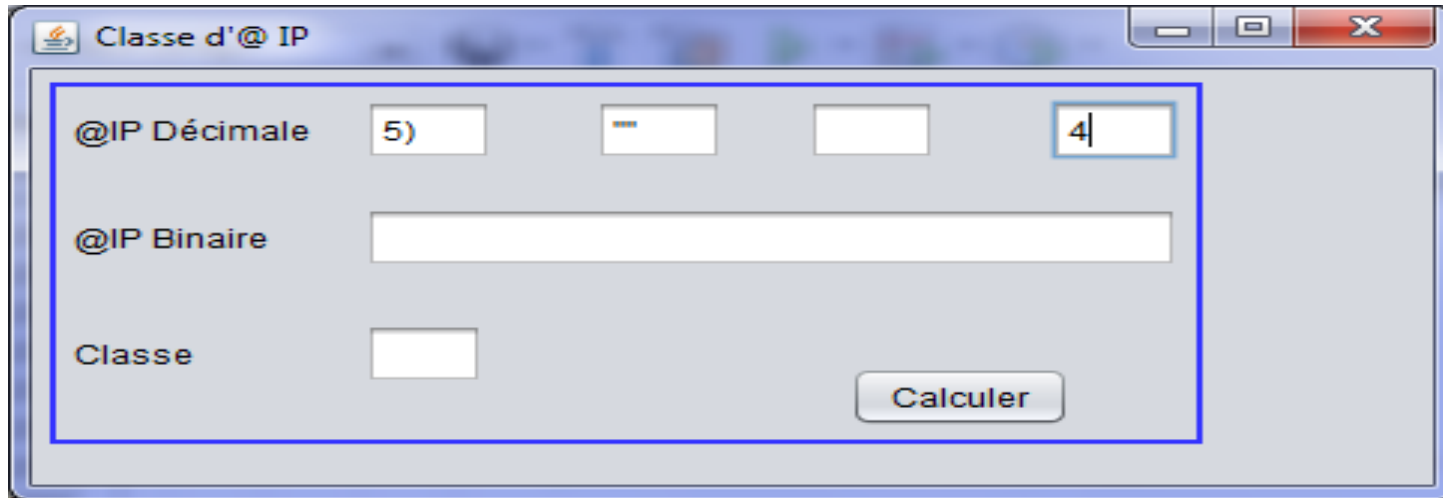
```
if ((oct1 > 255) || (oct1 < 0) || (oct2 > 255) || (oct2 < 0) ||  
    (oct3 > 255) || (oct3 < 0) || (oct4 > 255) || (oct4 < 0)) {  
    jDialog1.setVisible(true);  
} else {
```

```
// Reste du code  
}
```


Code pour le bouton Recommencer

```
private void jButton2ActionPerformed
(java.awt.event.MouseEvent evt) {
    jTextField1.setText("");
    jTextField2.setText("");
    jTextField3.setText("");
    jTextField4.setText("");
    jDialog1.setVisible(false);
}
```

Exception ..



```
try {  
    // tout le code  
} catch (NumberFormatException e){  
    jDialog1.setVisible(true);  
}
```