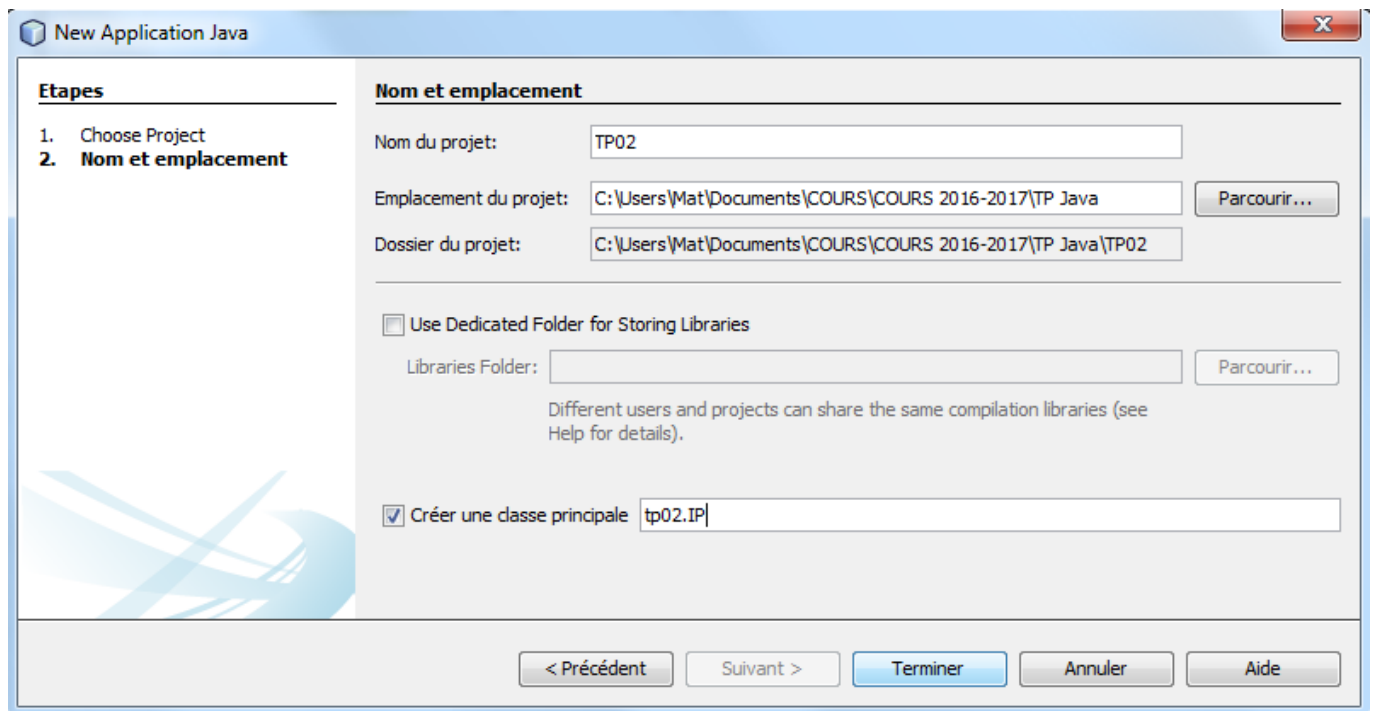


## PARTIE 1 – Classe IP

On désire construire une classe gérant une adresse IP. Cette classe doit permettre de donner son adresse réseau et sa classe (de réseau).

1. Démarrez Netbeans :

- choisissez le menu « Fichier », puis « Nouveau Projet ». Puis « **Java** » et « **Java Application** ».
- nommez la classe « **tp02.IP** » dans le projet « **TP02** », dans l'emplacement de vos projets.



2. Ajoutez à la classe « **IP** » les quatre attributs privés de type **int** : **octet1**, **octet**, **octet3**, **octet4**.  
Ajoutez un **constructeur**, à quatre paramètres, permettant de valoriser les attributs. Compilez afin de ne pas avoir d'erreur de syntaxe.

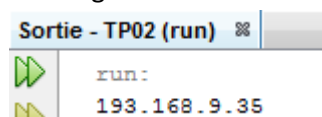
3. Cette classe doit pouvoir afficher ses attributs sous la forme décimale pointée (octets séparés par des points). Créez donc une méthode **versChaine()** qui a la signature suivante :

**public String versChaine()**

Ecrivez le code de test suivant dans la méthode **main** :

```
public static void main(String[] args)
{
    IP ip = new IP(193, 168, 9, 35);
    System.out.println(ip.versChaine());    ...
}
```

Lancez le projet (**F6**) ; ça doit produire l'affichage :



4. Ecrire dans la classe **IP** la méthode publique **getClasse()** qui retourne la classe de l'adresse (de type **char**) .

Rappel : la classe de l'adresse IP est :

- 'A' si le 1<sup>er</sup> octet est inférieur ou égal à 126 ;
- 'B' s'il est compris entre 128 et 191 ;
- 'C' s'il est compris entre 192 et 223 ;
- 'x' dans les autres cas.

5. Ecrire dans la classe **IP** la méthode publique **getAdresseReseau()** qui retourne l'adresse réseau (de type **IP**) de l'adresse IP. Elle a la signature suivante :

```
public IP getAdresseReseau()
```

Par exemple, le code :

```
IP ip = new IP(175, 168, 9, 35);           // adresse de classe B
System.out.println(ip.getAdresseReseau().versChaine());
// va afficher : 175.168.0.0
```

Autres exemples :

L'adresse IP 100.5.3.4 (classe A) aura pour adresse réseau : 100.0.0.0

L'adresse IP 192.168.8.4 (classe C) aura pour adresse réseau : 192.168.8.0

6. Ecrire dans la classe **IP** la méthode publique **estMemeReseau()** qui indique si l'adresse réseau passée en paramètre est dans le même réseau que l'adresse IP. Elle a la signature suivante :

```
public boolean estMemeReseau(IP ip)
```

Exemple d'utilisation :

```
IP ip = new IP(175, 168, 9, 35);           // son adresse réseau est 175.168.0.0
IP ip1 = new IP(175, 168, 6, 39);          // idem
if (ip.estMemeReseau(ip1))
    System.out.println("même réseau");      // c'est le cas ici
else
    System.out.println("pas le même réseau");
```

Vous pourrez écrire des accesseurs (get...) si nécessaire.

7. On s'intéresse maintenant aux entrées sorties (saisie/affichage). Dans la classe **IP**, ajoutez une méthode qui permet de saisir une adresse IP :

```
public static IP saisir()
```

Cette méthode doit permettre de saisir quatre octets, c. à d. quatre entiers compris entre 0 et 255. Elle retourne l'adresse IP constituée de ces 4 octets. Elle effectue un contrôle de saisie. Exemple d'utilisation dans **main** :

```
IP ip2 = saisir();
System.out.println(ip2.versChaine());
```

affichera :

```
run:
octet1 : 260
```

```
Recommencez : 265
Recommencez : 129
octet2 : 280
Recommencez : 175
octet3 : 120
octet4 : 90
129.175.120.90
```

Indications :

- pour afficher du texte sans retour à la ligne on utilise **print** : `System.out.print(texte)`
- pour saisir un nombre on utilise la classe **Scanner** :

```
package tp02;
```

```
// import de la classe Scanner (à écrire en dessous de package...)
```

```
import java.util.Scanner;
```

```
// etc.
```

```
// méthode saisir dans la classe IP :
```

```
public static IP saisir()
```

```
{
```

```
    int octet1, octet2, octet3, octet4;
```

```
    // instantiation du scanner (à faire une seule fois)
```

```
    Scanner sc = new Scanner(System.in);
```

```
    System.out.print("octet1 : ");
```

```
    // saisie du 1er octet :
```


```
    octet1 = sc.nextInt();
```

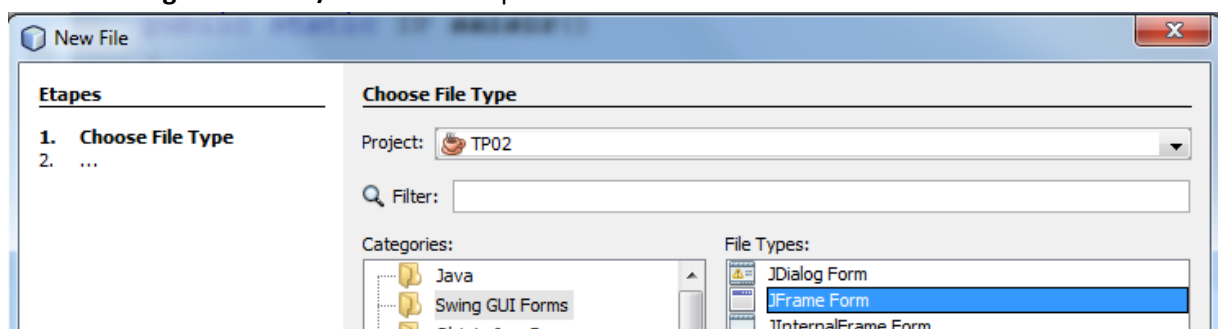
```
    // etc.
```

## PARTIE 2 – Interface graphique (GUI : graphical user interface)

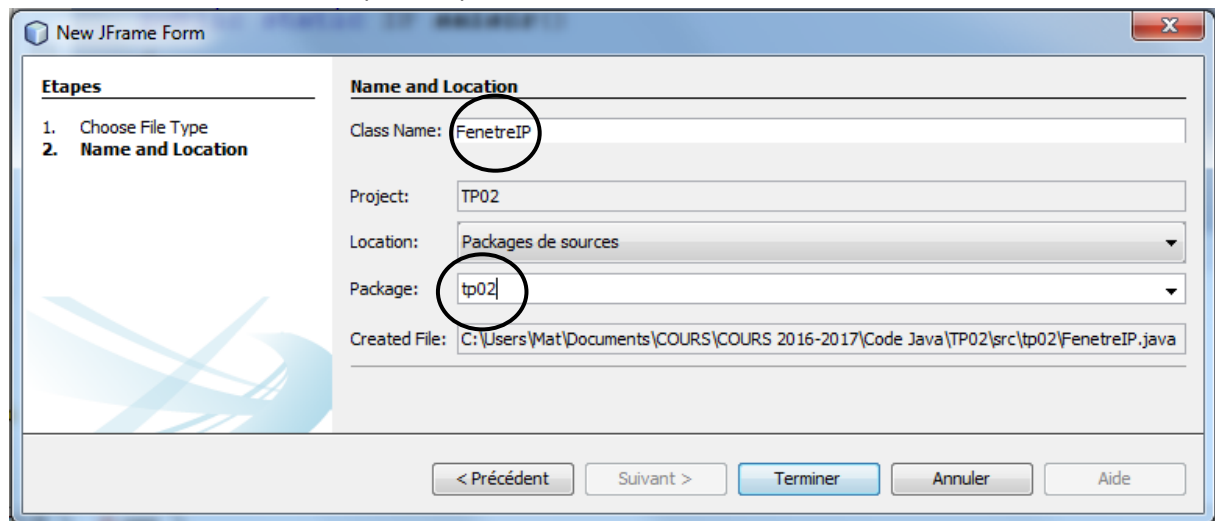
L'objectif de cette partie est de créer une interface graphique plus conviviale pour ce programme.

8. Ajoutez une nouvelle classe de fenêtre au projet

- clic droit sur le projet  puis **Nouveau / Autre...**
- choisir : **Swing GUI Forms / JFrame Form** puis **Suivant**

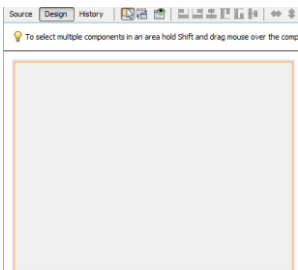


- Mettez comme nom de classe « **FenetreIP** » et comme **package** « **tp02** » (le même que celui de la classe IP. Le reste est pré rempli.



- puis « **Terminer** ».

Netbeans génère le code correspondant à la classe **FenetreIP** et ouvre l'éditeur d'interface graphique :

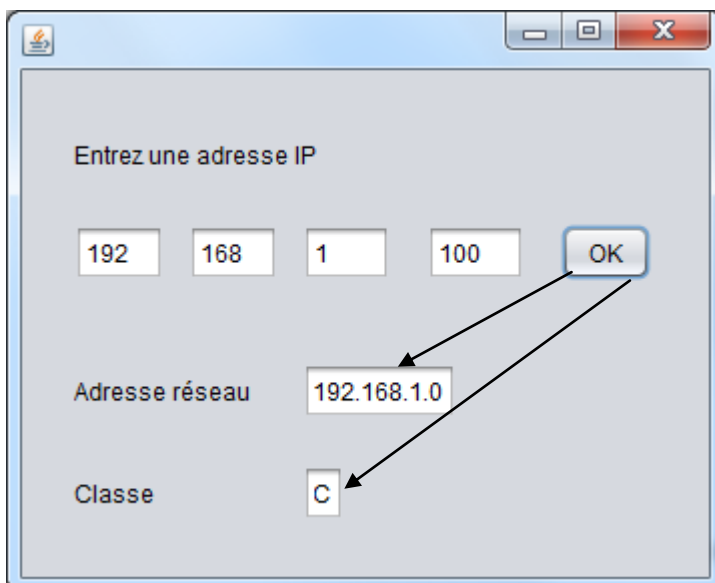


En mode « Source », vous pouvez voir le code généré :

```
public class FenetreIP extends javax.swing.JFrame { ...
```

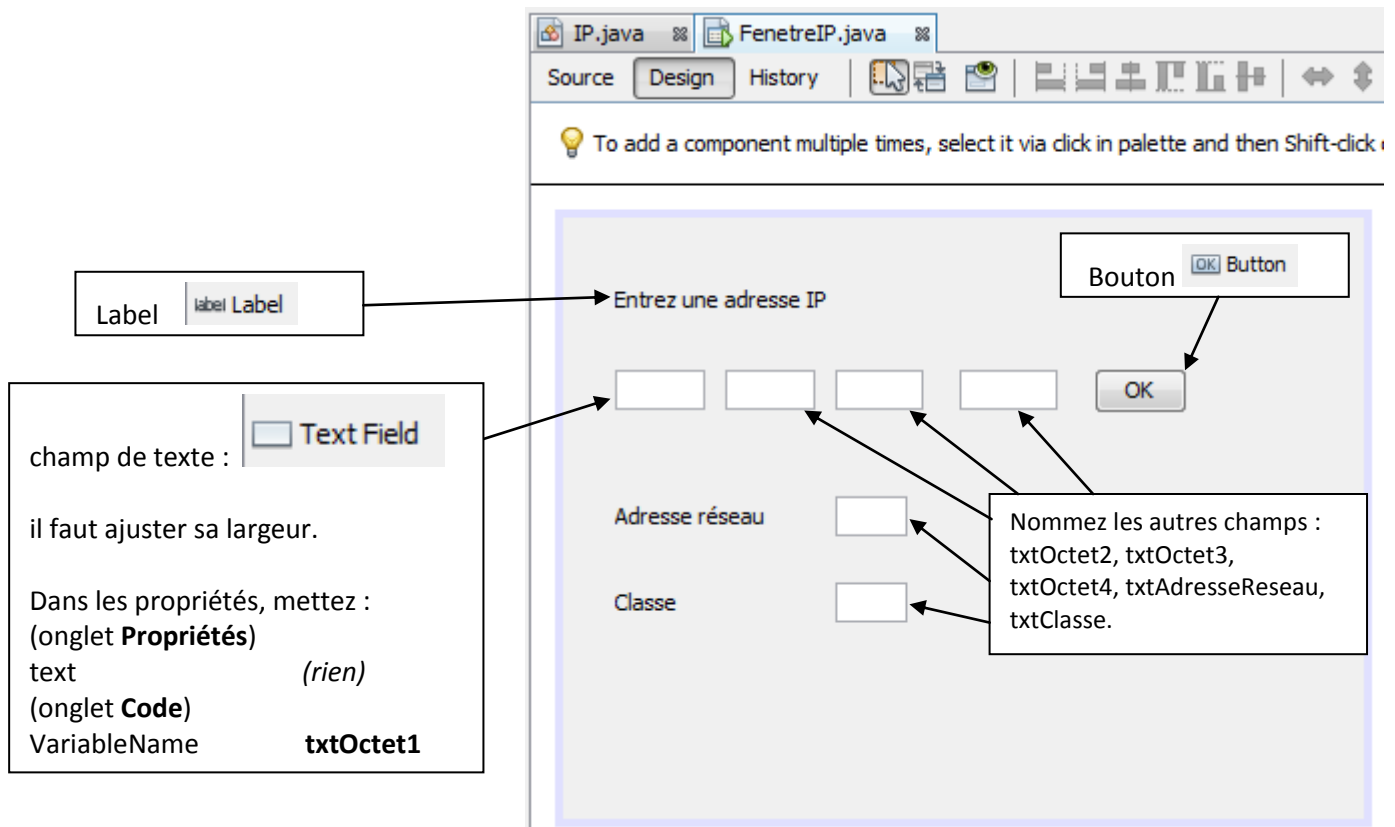
(Suite du cahier des charges)

Le fonctionnement sera le suivant :



on saisit une adresse IP et quand on clique sur OK l'adresse réseau et la classe s'affichent.

9. En mode « Design », concevez l'interface graphique suivante :



Testez le résultat avec le menu « **Exécuter / Exécuter le fichier** » (**maj + F6**). Fermez l'application à la fin.

10. Dans le code, il faut ensuite créer une méthode événementielle associée à l'événement du « clic » sur le bouton **OK**. L'événement associé est appelé en Java « **action performed** ».

En mode design, double-cliquez sur le bouton : .

La méthode événementielle **jButton1ActionPerformed** est créée dans la classe **FenetreIP** :

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { ...
```

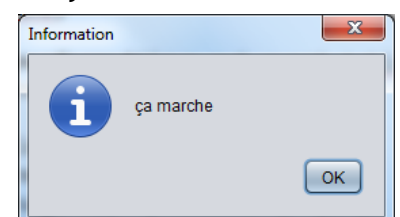
Modifiez le code comme indiqué ci-dessous :

```
// ajoutez un import en haut :
package tp02;
import javax.swing.JOptionPane;
```

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

    // Boîte d'information
    JOptionPane jop1 = new JOptionPane();
    //on affiche un message d'information
    jop1.showMessageDialog(null, "ça marche", "Information",
    JOptionPane.INFORMATION_MESSAGE);
}
```

Testez (menu « **Exécuter / Exécuter le fichier** » ou **maj + F6**). Cliquez sur le 1<sup>er</sup> bouton « OK ». Fermez l'application à la fin.



11. Continuons les tests. Quand on clique sur OK, on veut afficher dans une fenêtre le premier octet saisi. On va donc écrire dans la méthode événementielle :

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    // récupération du texte saisi dans le champ de texte txtOctet1  
    String octet1Saisi = txtOctet1.getText();  
    // affichage de l'octet saisi  
    JOptionPane jop1 = new JOptionPane();  
    jop1.showMessageDialog(null, octet1Saisi, "Information", JOptionPane.INFORMATION_MESSAGE);  
}
```

Testez (Maj + F6).

12. Terminez le programme. Il faut récupérer tous les octets saisis, instancier une adresse IP, obtenir son adresse de réseau et sa classe, et les afficher. Complétez la classe **Form1** :

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    // récupération du texte saisi dans le champ de texte txtOctet1  
    String octet1Saisi = txtOctet1.getText();  
  
    // conversion en entier (int)  
    int octet1 = Integer.parseInt(octet1Saisi);  
  
    // TODO : faire la même chose pour les trois autres...  
  
    // TODO : instanciation de l'objet IP  
    IP adresseIP = . . . . .  
  
    // TODO : obtenir l'adresse réseau de adresseIP  
    String adresseReseau = . . . . .  
  
    // Affichage de l'adresse réseau dans le champ de texte txtAdresseReseau  
    txtAdresseReseau.setText(adresseReseau);  
  
    // TODO : faire la même chose avec la classe...  
}
```