

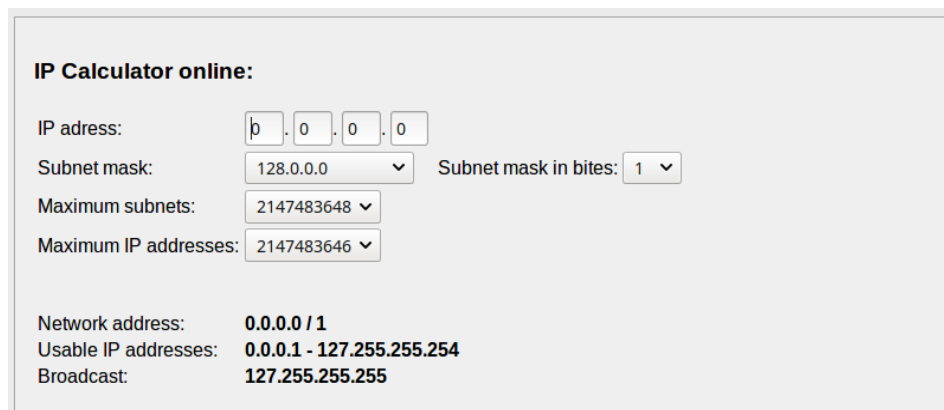


### TD 3 ~ CalculatriceIPv4

L'objectif de cet exercice est de concevoir une application graphique sous QT réalisant une calculatrice IPv4 permettant de déterminer l'adresse réseau dans laquelle elle se trouve ainsi que l'adresse de diffusion et l'ensemble des adresses utilisables du réseau.

Comme le montre la figure suivante, l'utilisateur donne une adresse IPv4 et au choix :

- un masque de réseau,
- un suffixe,
- un nombre de sous réseau
- un nombre d'hôte



**IP Calculator online:**

IP address:  .  .  .

Subnet mask:  Subnet mask in bites:

Maximum subnets:

Maximum IP addresses:

Network address: **0.0.0.0 / 1**

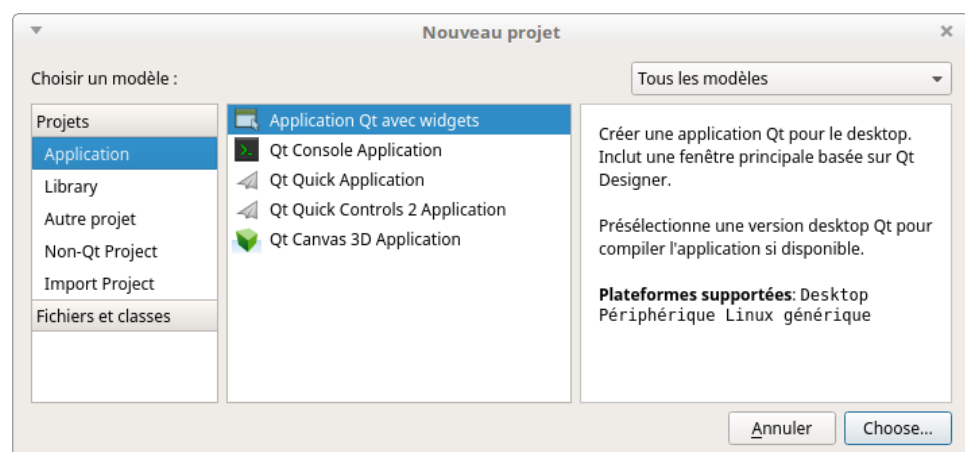
Usable IP addresses: **0.0.0.1 - 127.255.255.254**

Broadcast: **127.255.255.255**

L'application ressemblera à cette copie d'écran, obtenu à partir d'une calculatrice en ligne : <http://ipcalc.nmonitoring.com/> .

## 1 Création d'un projet

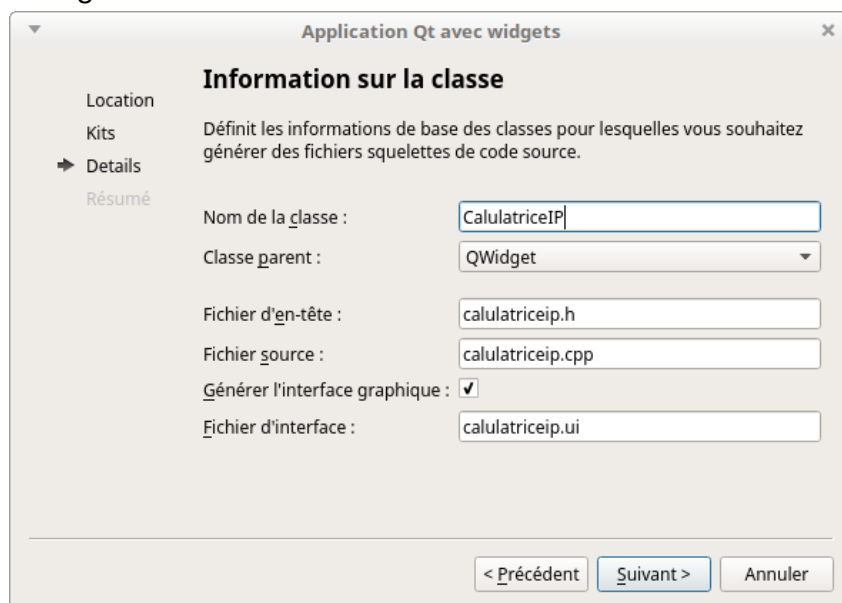
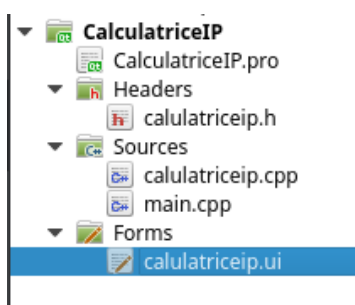
Après avoir lancé **Qt creator**, choisir dans le menu **fichier** l'option **nouveau fichier ou projet**. Choisir ensuite un **Projet** type **Application Qt avec widgets** comme le montre la figure.



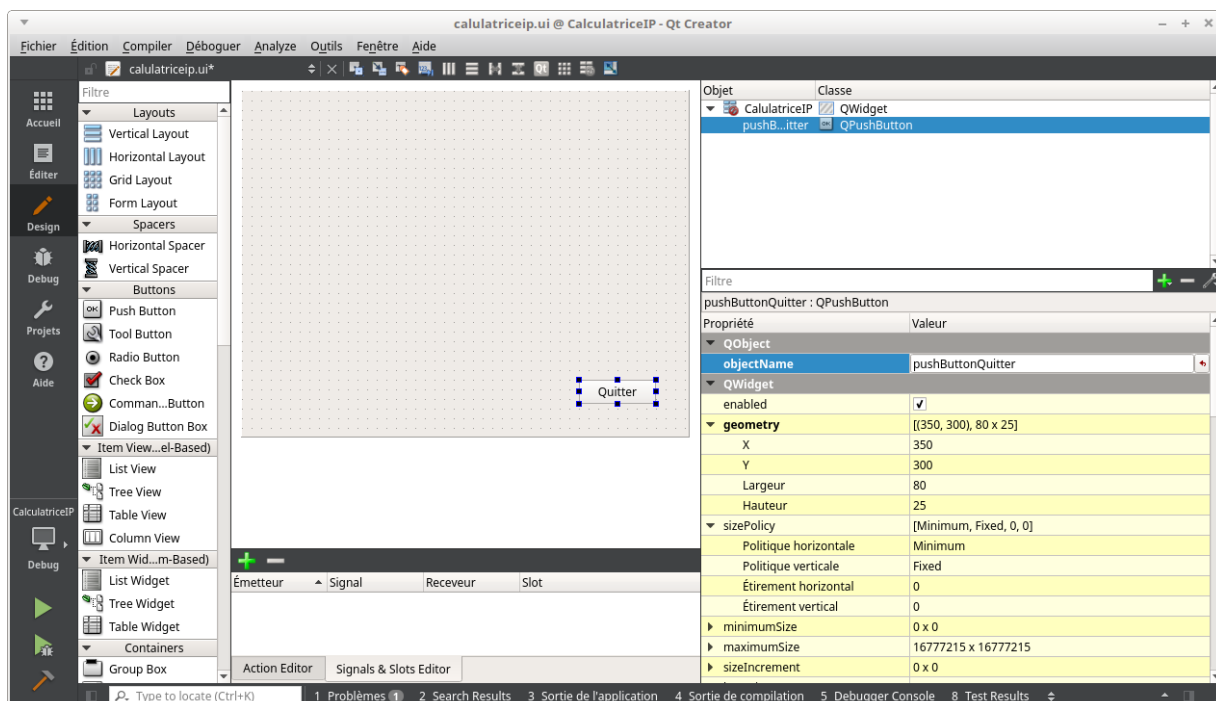
## 2 Création de la classe principale

Dans un premier temps réaliser la classe principale nommée CalculatriceIP. Cette classe hérite de QWidget comme le montre la figure

Le résultat obtenu dans l'onglet projet est donné par la figure ci-dessous.



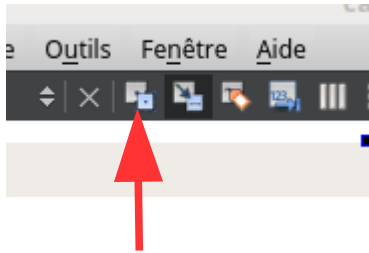
Après avoir cliquer sur calculatriceip.ui, l'IDE propose de dessiner l'interface graphique de l'application.



Ajouter le bouton « Quitter » et changer lui son nom dans la rubrique objectName

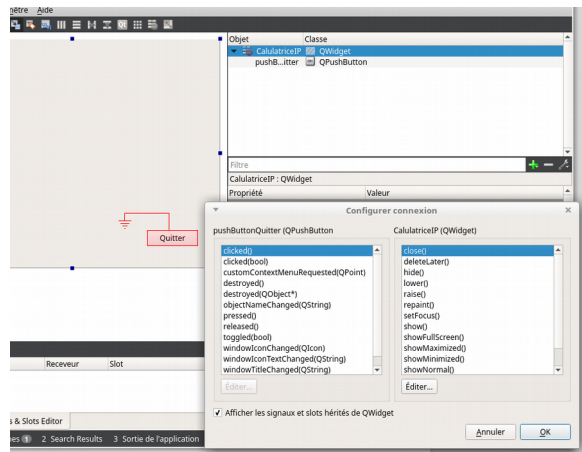


La fenêtre suivante apparaît :



l'appui sur F3 ou cet icône permet de revenir en mode édition du widget

Cliquer ici pour associé le signal **clicked()** du bouton au slot **close()** dont il a hérité.



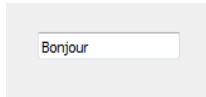
Compiler et exécuter l'application pour vérifier son fonctionnement.

### 3 Les zones d'édition :

Sous Qt les zones d'édition sont représentées par des **QLineEdit**. La méthode **setText()** permet d'afficher un **QString** à l'intérieur et **text()** affecte un **QString** avec le contenu de la zone.

Exemples :

```
ui->lineEdit->setText("Bonjour") ; // pour afficher
QString leTexte ;
leTexte = ui->lineEdit->text () ; // pour récupérer la valeur
```



Dans notre problème, Il est parfois nécessaire de convertir un entier en QString et vis et versa. Le site de développez.com <http://qt.developpez.com/doc/4.7/qstring/#toint> ou celui de QT <https://doc.qt.io/qt-5/qstring.html> nous propose la méthode de la classe QString : `int QString::toInt( bool *ok = 0, int base = 10).`

Cette méthode convertit une chaîne en entier en fonction de la base sélectionnée, par défaut en base 10, le booléen ok est mis à jour par la fonction si l'opération a pu être réalisé correctement.

Pour la fonctionnalité inverse, il existe deux méthodes possibles `setNum(int n, int base=10)` ou `number(int n, int base=10)`. La deuxième est une méthode **static** et n'a donc pas besoin d'une instance pour être utilisée.

L'appui sur F1 avec le curseur sur la classe **QString** dans votre source donne également accès à la documentation.

Exemples :

```
int a = 63;
QString s = QString::number(a, 16);           // s == "3f"
QString t = QString::number(a, 16).toUpper(); // t == "3F"

ou
QString str;
str.setNum(1234);    // str == "1234"
```

## 4 Les ComboBox

Les comboBox ou boîtes combinés permettent la saisie dans une zone d'édition ou la sélection d'un item dans une liste. Le remplissage de la liste est réalisé par la méthode :

```
void QComboBox::addItem (const QString &text, const QVariant &userData=QVariant())
```

Seul le premier paramètre est dans un premier temps utile. Il permet d'ajouter une chaîne de caractère sous la forme d'une **QString** à la liste. Les entiers devront être transformé en chaîne de caractères avant d'être ajoutés.

Le signal **currentIndexChanged** est émit chaque fois que l'utilisateur sélectionne un élément de la liste. Il existe sous deux formes, la première transmet l'index d'un item lors d'une sélection, la seconde transmet le texte de l'item sélectionné.

```
void currentIndexChanged ( int index )
void currentIndexChanged ( const QString & text )
```

Associé à un slot, ils peuvent chacun traité les choix de l'utilisateur. Les index varient de 0 à nb-1 éléments, la valeur -1 correspond à une valeur erronée ou vide. La valeur index ou text suivant le cas est transmise au slot qui réagit au signal.

Le slot public void **setCurrentIndex**(int index) est utilisé pour positionner l'élément visible de la liste.

Le slot public void **clear**() est utilisé pour vidé la liste de son contenu.

## 5 Réalisation de l'IHM

Compléter l'IHM avec les éléments suivants :

- L'adresse IPv4 est composé de 4 zones d'édition
- Le suffixe, le masque, le nombre maximum de sous réseaux et le nombre maximum d'adresses Ip sont représentés par des **comboBox**.
- Les trois autres zones d'édérations sont définies en lecture seule. L'utilisateur ne peut que visualiser les résultats.

Objet	Classe
CalculatriceIP	QWidget
verticalLayout_2	QVBoxLayout
horizontalLayout_5	QHBoxLayout
label_6	QLabel
lineEditAdresseReseau	QLineEdit
horizontalLayout_6	QHBoxLayout
label_7	QLabel
lineEditAdressesUtilisables	QLineEdit
horizontalLayout_7	QHBoxLayout
label_8	QLabel
lineEditAdresseDiffusion	QLineEdit
horizontalLayout	QHBoxLayout
comboBoxSuffixe	QComboBox
label_3	QLabel
lineEditAdresse1	QLineEdit
lineEditAdresse2	QLineEdit
lineEditAdresse3	QLineEdit
lineEditAdresse4	QLineEdit
verticalLayout	QVBoxLayout
horizontalLayout_2	QHBoxLayout
comboBoxMasque	QComboBox
label_2	QLabel
horizontalLayout_3	QHBoxLayout
comboBoxSousReseaux	QComboBox
label_4	QLabel
horizontalLayout_4	QHBoxLayout
comboBoxAdressesIP	QComboBox
label_5	QLabel
pushButtonQuitter	QPushButton

Le plus grand soin sera porté au nom de variable associé à chaque widget liés aux interactions avec les utilisateurs comme le montre par exemple le tableau ci-dessus.

**Le constructeur de la classe CalculatriceIPv4 doit initialiser les différents comboBox avec les valeurs appropriées :**

ComboBox	Val mini	Val maxi	Remarques
Le Suffixe	8	30	
Le Masque du réseau	255.0.0.0	255.255.255.252	
Nombre maximum de réseaux	4	16777416	2 <sup>n</sup> avec n le nombre de bits empruntés
Nombre d'Hôte maximum pour le réseau	2	16777414	2 <sup>n</sup> -2 avec n le nombre de bits pour définir les hôtes

Dans le constructeur de la classe CalculatriceIP, remplissez par calcul les différents éléments des comboBox en suivant les indications :

- Pour le suffixe les valeurs sont comprises entre 8 et 30.
- Pour le masque, on propose l'algorithme suivant :

<u>Début</u> leMasque ← 0xFF000000 <u>Pour</u> valeur allant de 8 à 30 fenetre ← 0xFF000000 <u>Pour</u> indice allant de 0 à 3 octet ← (leMasque ET fenetre) DécalerDroite (8 * (3 – indice)) masque = masque + octet <u>Si</u> indice < 3 <u>Alors</u> masque ← masque + '.' fenetre ← fenetre DécalerDroite( 8) <u>FinSi</u> <u>FinPour</u> // Ecriture du masque sur l'IHM leMasque ← leMasque DécalerDroite( 1) leMasque ← leMasque OU 0x80000000 <u>FinPour</u>	<u>Variables locales :</u>  leMasque entier non signé 32bits fenetre entier non signé 32bits octet entier non signé 8bit indice entier masque chaîne de caractères
--	--

- Chaque valeur ainsi obtenu est mis en forme dans une chaîne et ajouté aux items du comboBox.
  - Pour le nombre maximum de réseau et le nombre d'hôte la fonction **qPow** de la librairie **QtMath** peut être utilisée
3. Dans un premier temps, il est nécessaire de faire correspondre le suffixe et le masque, chaque fois que l'utilisateur agit sur le suffixe, le masque correspondant s'affiche. Créer un slot correspondant au signal indiquant que l'index courant de la comboBox à changé.
  4. Graphiquement effectuer les connexions entre les différents comboBox pour que la mise à jour se face automatiquement.
  5. Pour la saisie de l'adresse IP, il est nécessaire de restreindre la saisie aux entiers entre 0 et

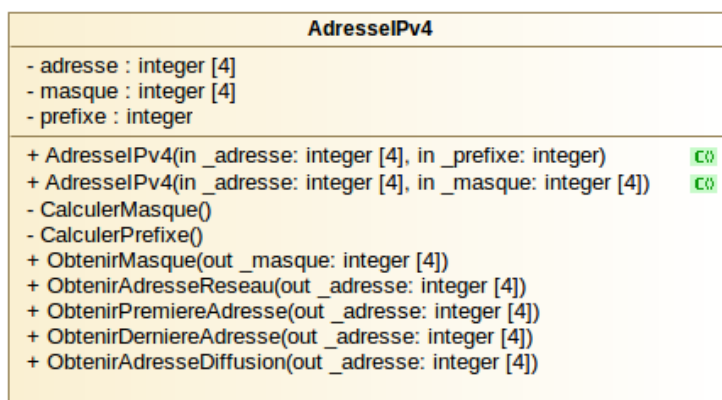
255 par exemple. Cela peut se faire de la manière suivante :

```
ui->lineEditAdresse->setValidator(new QIntValidator(0,255));
```

Procéder de manière équivalente pour les 4 zones d'éditations en tenant compte des valeurs possibles, on exclue les adresses de diffusions et il n'y a pas d'hôte avec une adresse se terminant par 0.

## 6 Classe AdresseIPv4

6. Réaliser la classe AdresseIPv4 dont la modélisation UML est la suivante :



Chacun des constructeurs appelle soit `CalculerMasque()` soit `CalculerPrefixe()` en fonction des paramètres reçus.

7. Réaliser le codage des deux méthodes privées de la classe AdresseIPv4

## 7 Intégration

8. Écrire le code de la méthode permettant de convertir un tableau d'octets en QString en ajoutant les points nécessaires entre les décimales dont le prototype est :

```
QString CalculatriceIP::ConvertirTabAdresseEnQString(const quint8 tabByte[])
```

9. Compléter le codage du slot de la question 3 afin de pouvoir renseigner les 3 zones d'éditations en lecture seule.
10. Créer un slot nommé `onAdresseIPChange()` dans la classe CalculatriceIP. Puis connecter dans le signal `textChange(QString)` de chaque zone d'édition à votre slot. Dans ce slot appeler le slot complété à la question précédente.