Guide d'utilisation d'Eclipse pour créer des applications en Java

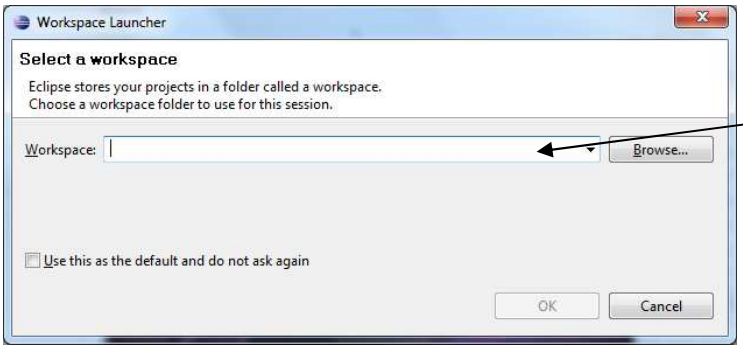
M. Dalmau, IUT de Bayonne-Pays Basque

NOTE

Ce guide n'a pas été conçu dans un simple but décoratif (bien qu'il en ait les qualités), vous devez toujours l'avoir avec vous lors des TPs de même que les polycopiés Java et Swing cela vous évitera de chercher inutilement ailleurs des renseignements qui y sont.

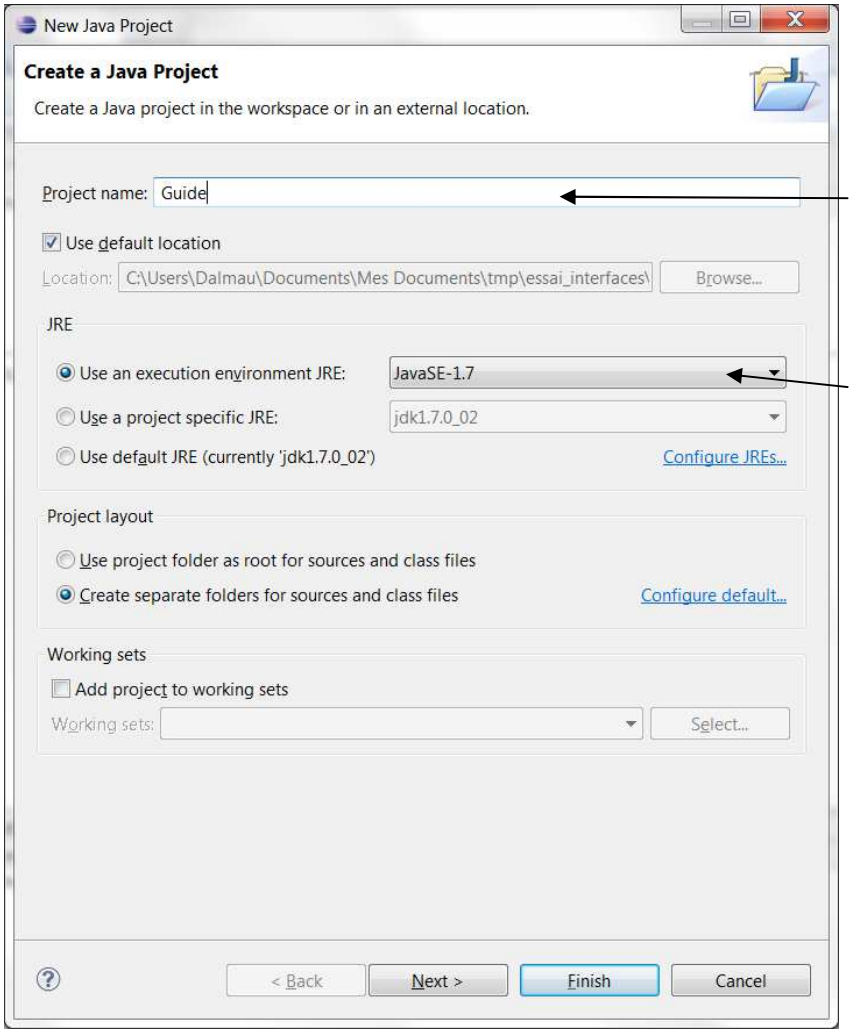
Créer un projet Java avec Eclipse

1. Lancer Eclipse



Choisir un répertoire de travail

2. Quand 'Eclipse a démarré aller dans le menu : File → New → Java Project

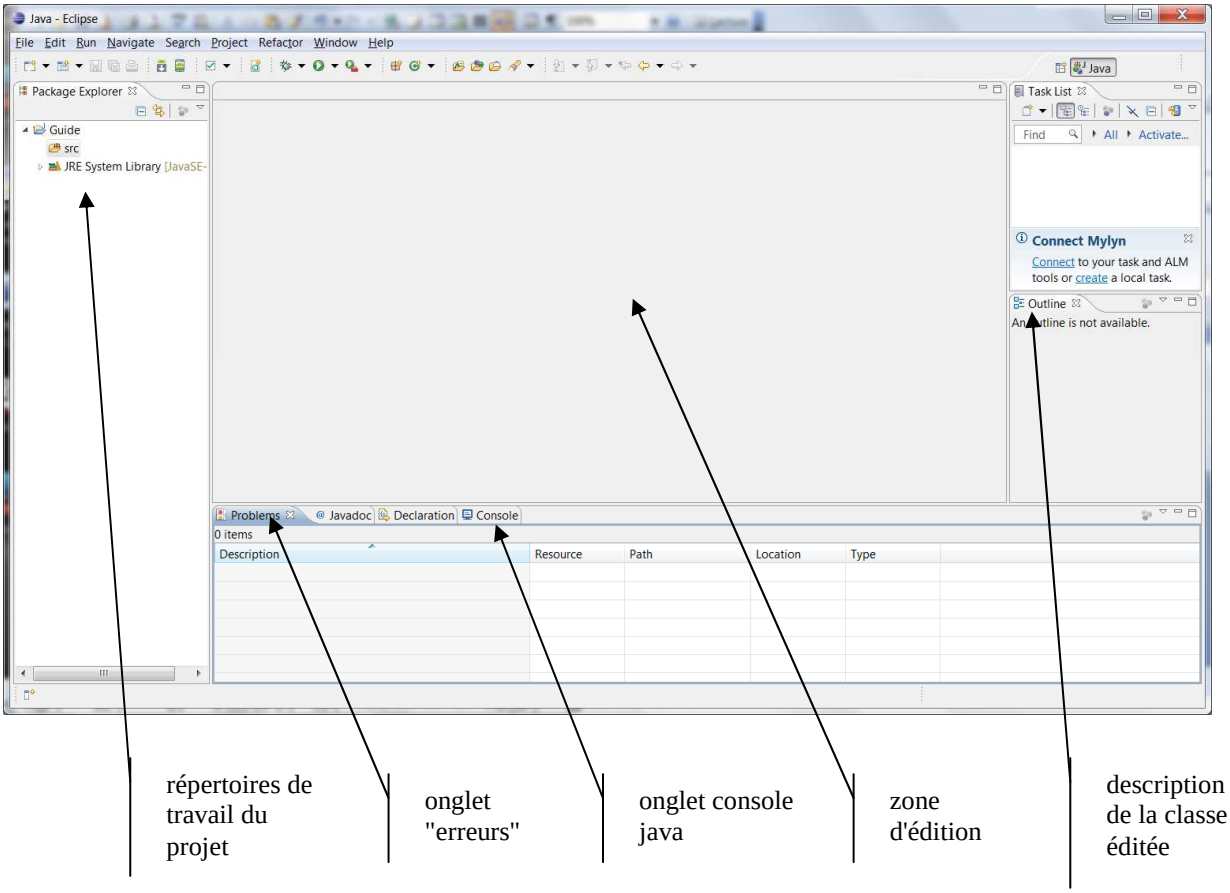


Choisir un nom pour le projet.

Choisir la version de java

5. Cliquer sur Finish, le projet est créé, on y trouve le répertoire src pour les fichiers sources (.java)

L'écran de travail d'Eclipse :



Configurer l'éditeur d'Eclipse

Pour afficher les numéros de lignes dans votre code
Windows → Preferences → General → Editors → TextEditors et cocher "Show Line numbers"

Pour enlever la correction orthographique
Windows → Preferences → General → Editors → TextEditors → Spelling et dé-cocher "Enable Spell Checking"

Pour ajouter des vues
Windows → Show View choisir la vue à ajouter. On peut ensuite déplacer la vue par glisser/déposer pour la mettre en onglet à droite, à gauche ou en bas.

Quelques commandes utiles d'Eclipse

Si vous ajoutez, supprimez ou modifiez des fichiers en dehors d'Eclipse
Il faut resynchroniser le projet dans Eclipse par : clic droit sur le projet → Refresh

Eclipse ne recompile que les fichiers modifiés, si vous voulez l'obliger à tout recompiler
Il faut nettoyer le projet dans Eclipse par : Projet → Clean puis cocher le projet à recompiler et cliquer OK

Ajouter un répertoire : clic droit sur le projet → New → Folder

Ajouter une classe : clic droit sur le paquetage → New → Class

Renommer un paquetage ou une classe : clic droit sur le paquetage ou le fichier de la classe → Refractor → Rename

Renommer une variable dans un fichier source : sélectionner la variable puis clic droit → Refractor → Rename

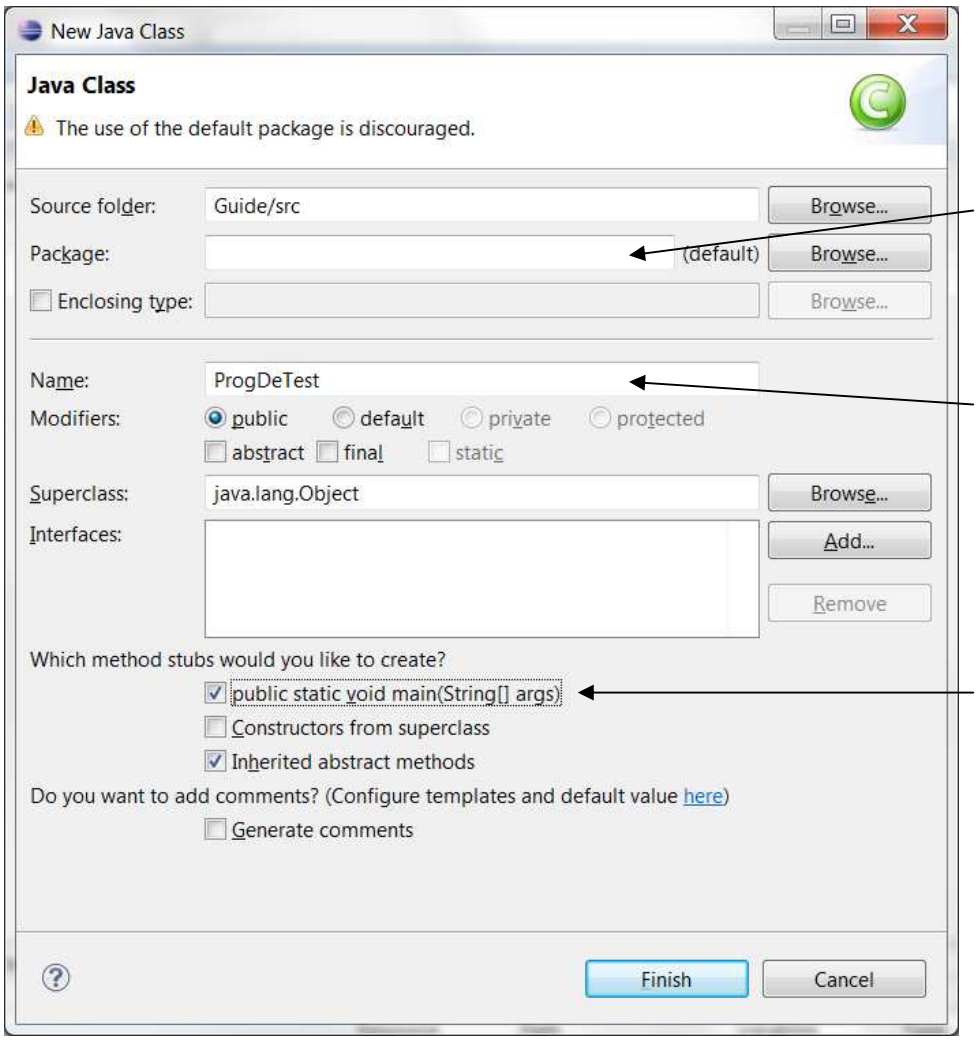
Créer une application java avec Eclipse

Si dans le menu "projet" d'Eclipse "Buid Automatically" est coché, le projet sera recompilé à chaque enregistrement de l'un des fichiers du projet.

Attention : modifier un fichier sans l'enregistrer ne provoquera pas de recompilation et c'est toujours l'ancienne version qui sera exécutée !

Si vous dé-cochez la case "Buid Automatically" il faudra lancer la compilation par le menu "projet" en choisissant "Build Project".

1. Création du programme principal :
   clic droit sur src : New → Class

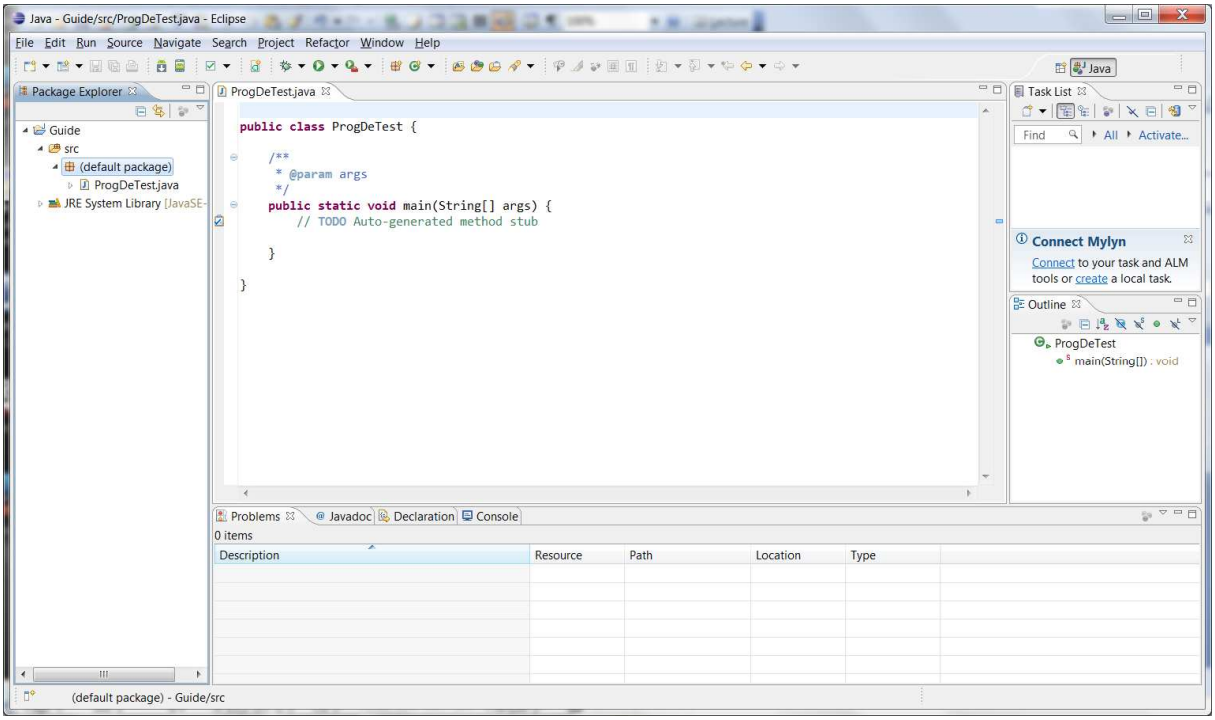


On peut donner un nom de paquetage et c'est préférable (par défaut ce sera "default package")

Donner un nom à la classe du programme principal

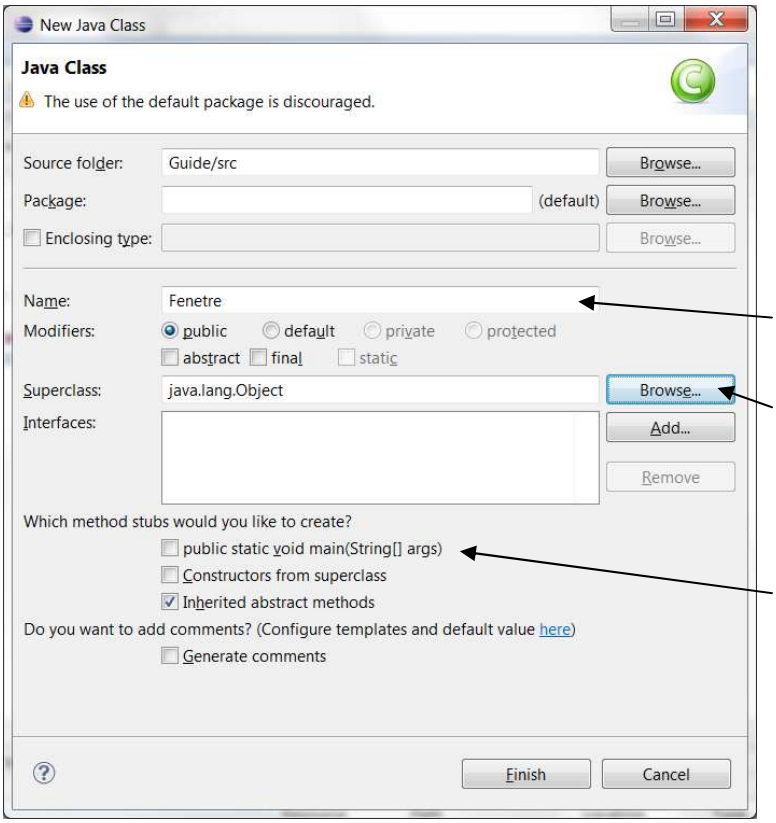
Cocher la case pour créer une méthode de lancement (main)

Cliquer sur Finish. La classe est créée et contient une méthode "main" pour le moment vide :



2. Création de la fenêtre d'interface :

clic droit sur default package (ou le nom de votre paquetage si vous en avez mis un) → New → class

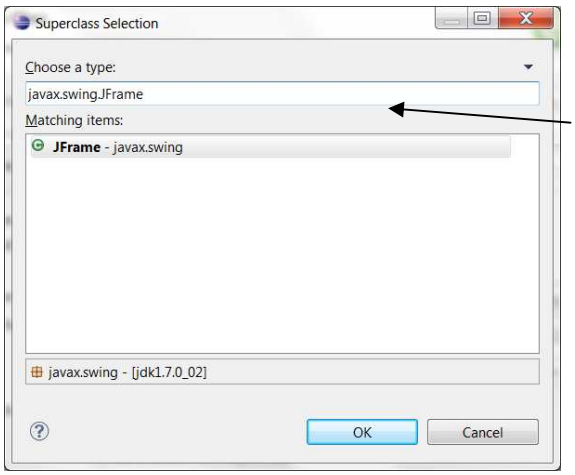


Donner un nom à la classe

La classe doit hériter de JFrame (voir ci-dessous)

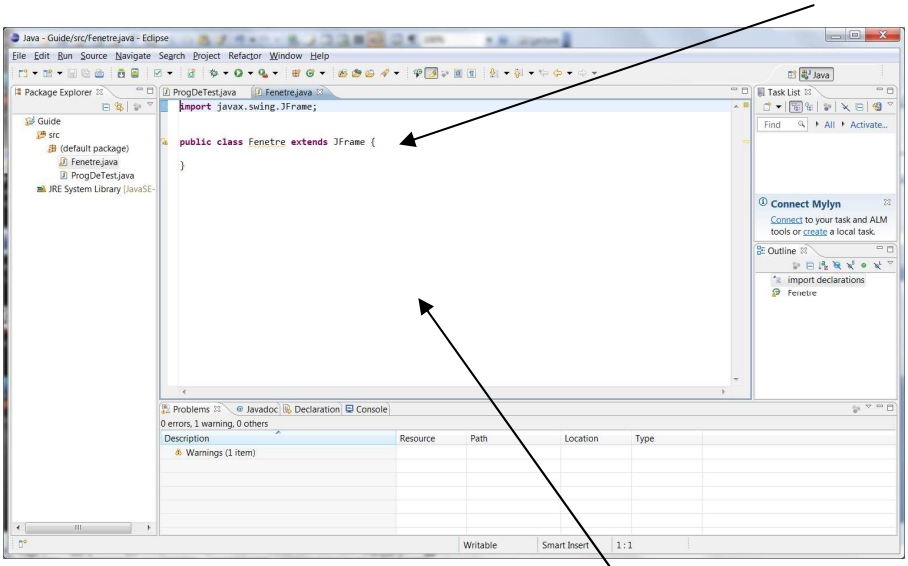
Pas de méthode "main" dans cette classe

La classe doit hériter de JFrame :



Fenêtre ouverte par le bouton "Browse" de la fenêtre précédente => choisir que la classe hérite de *javax.swing.JFrame*

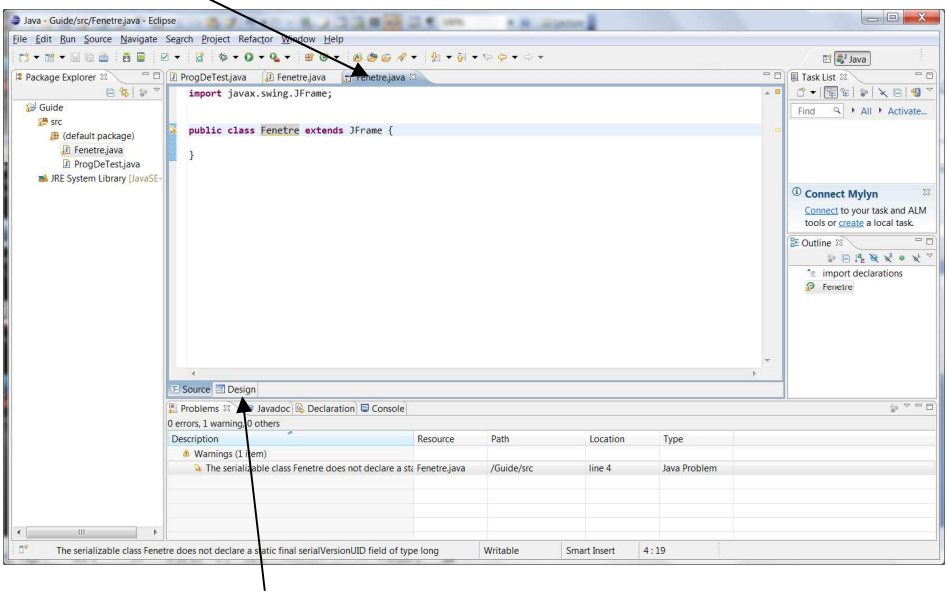
Cliquer OK dans cette fenêtre puis Finish dans la précédente. La classe est créée (vide) :



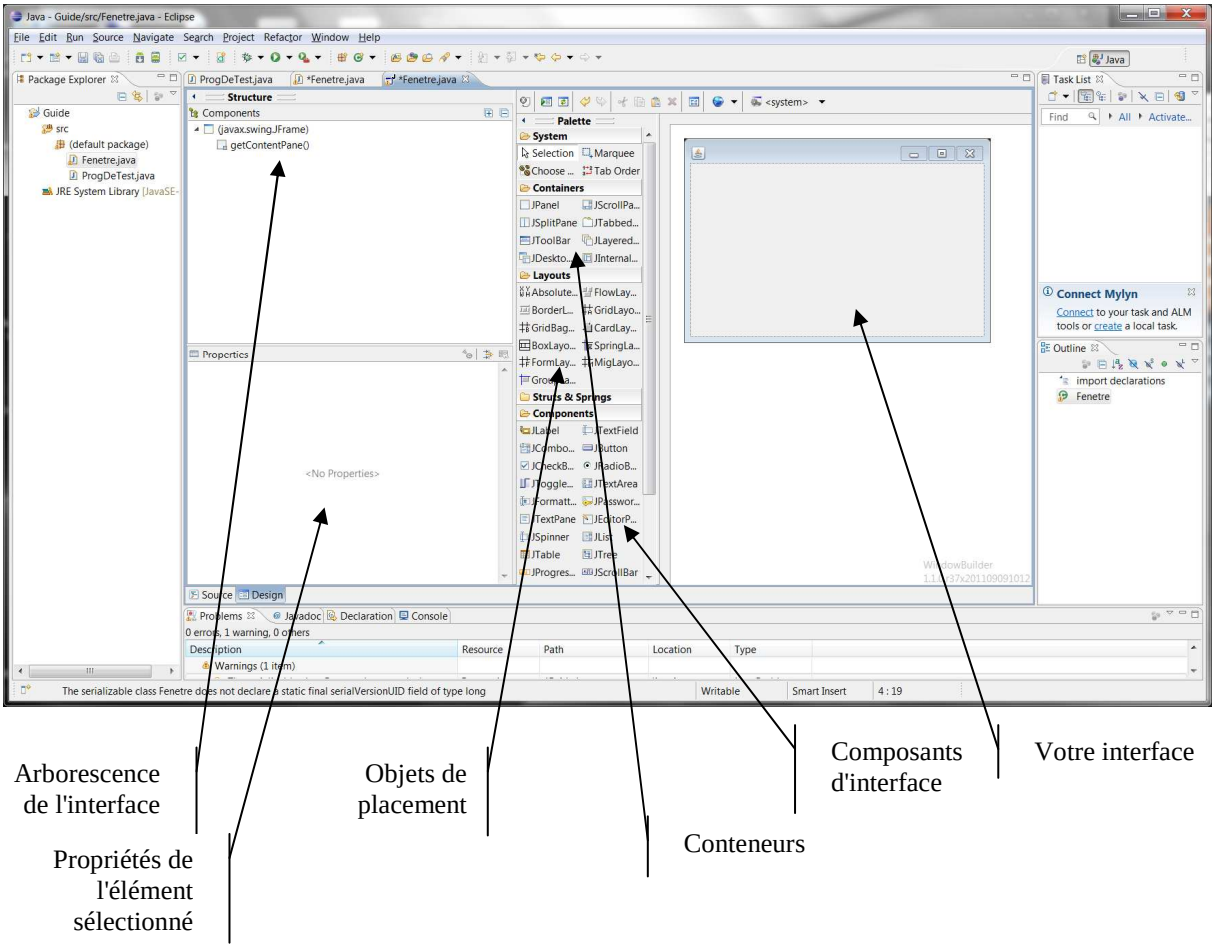
3. Utiliser l'éditeur graphique pour créer l'interface.

Clic droit dans la zone d'édition de la classe (fichier .java) → OpenWith → WindowBuilder Editor

Un nouvel onglet apparaît contenant la classe de l'interface graphique :

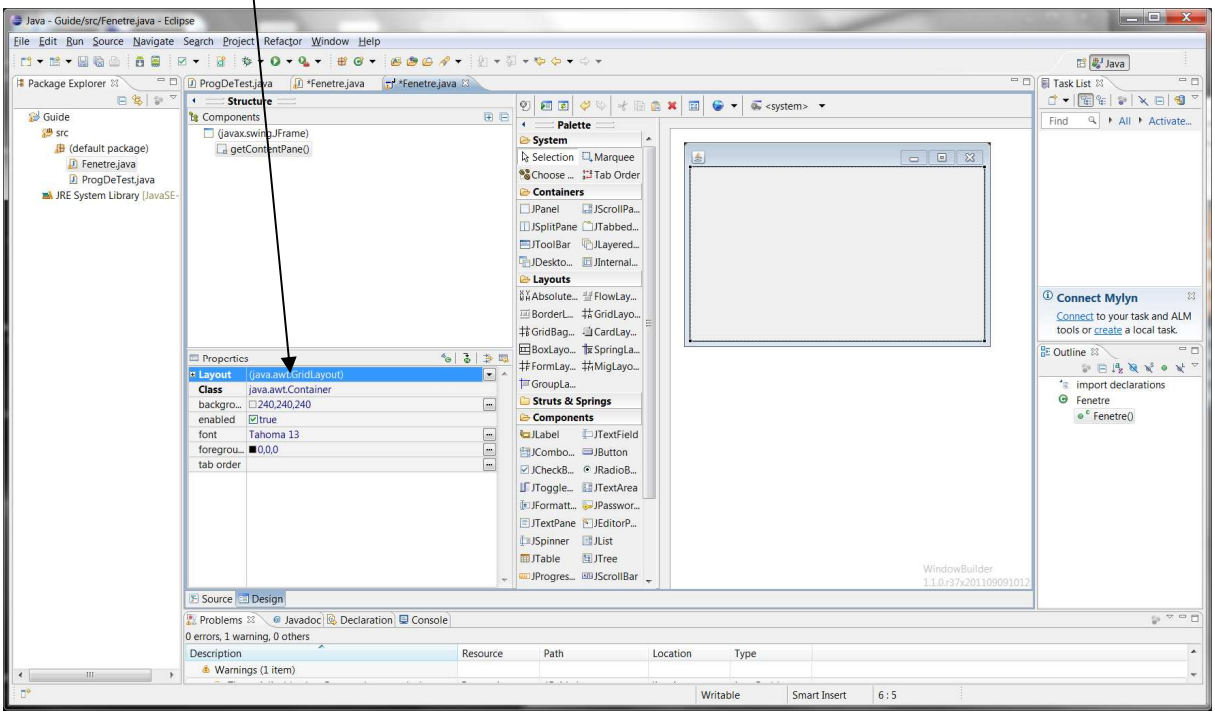


Choisir l'onglet "Design" pour accéder à l'éditeur graphique dont l'apparence est la suivante :



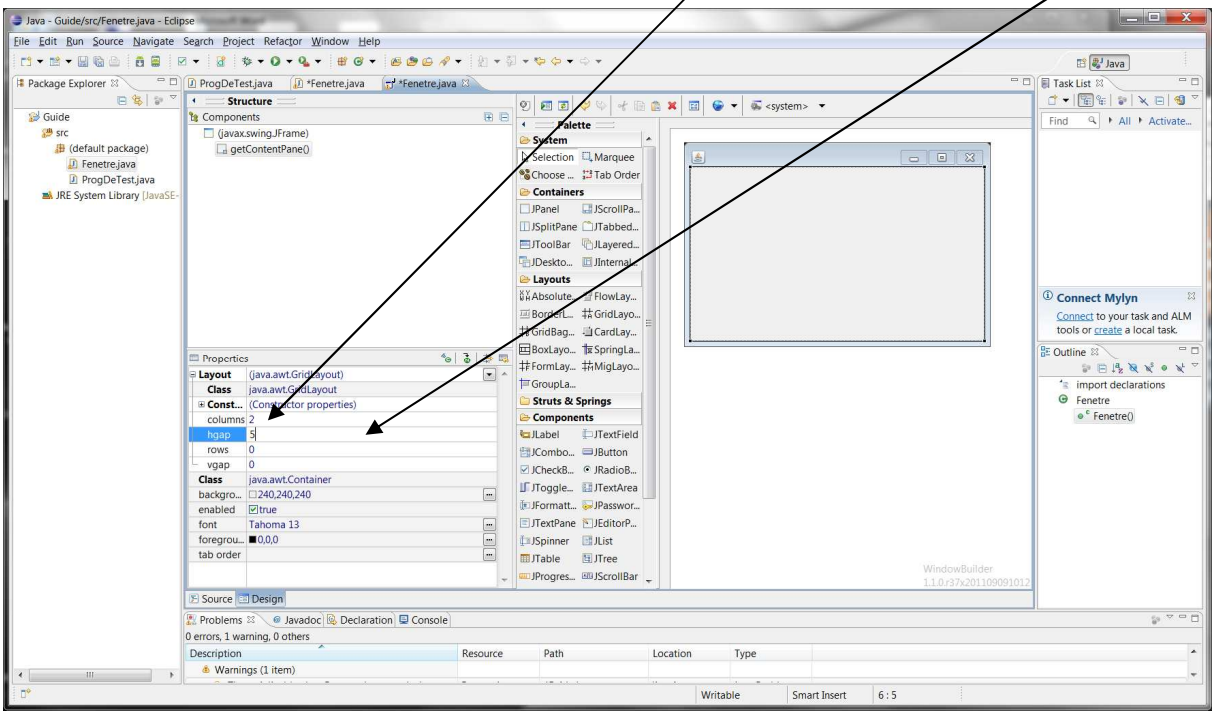
3. Ajout d'un objet de placement au conteneur de base (celui obtenu par la méthode *getContentPane()*):
   Dans l'arborescence faire un clic droit sur *getContentPane()* puis sélectionner *SetLayout* et, dans la fenêtre qui s'ouvre, choisir l'objet de placement désiré (par exemple GridLayout)

Il est associé au conteneur



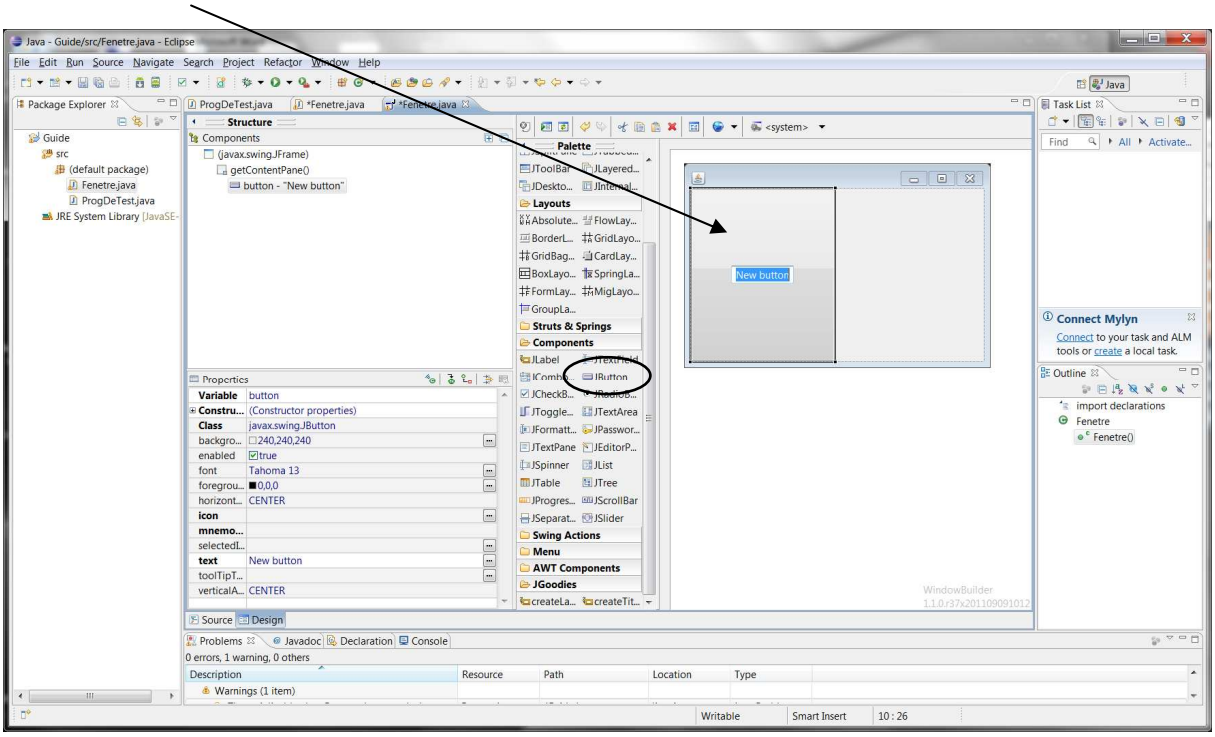
4. Modifier les paramètres de cet objet de placement :

Dans l'arborescence sélectionner *getContentPane()* puis cliquer sur le + à gauche de *Layout* pour modifier les paramètres du GridLayout (par exemple 2 colonnes et espacement horizontal de 5) :



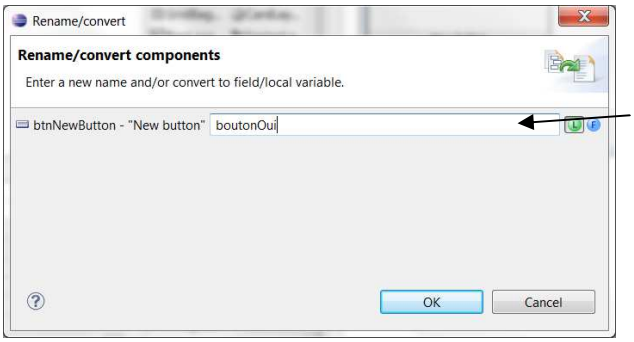
5. Ajout d'un composant d'interface (widget) à gauche de la fenêtre :

Dans la liste des composants d'interface choisir un widget (par exemple un JButton) puis déplacer la souris vers la case de gauche de la fenêtre de votre interface (un trait rouge apparaît), cliquer, le bouton est ajouté :



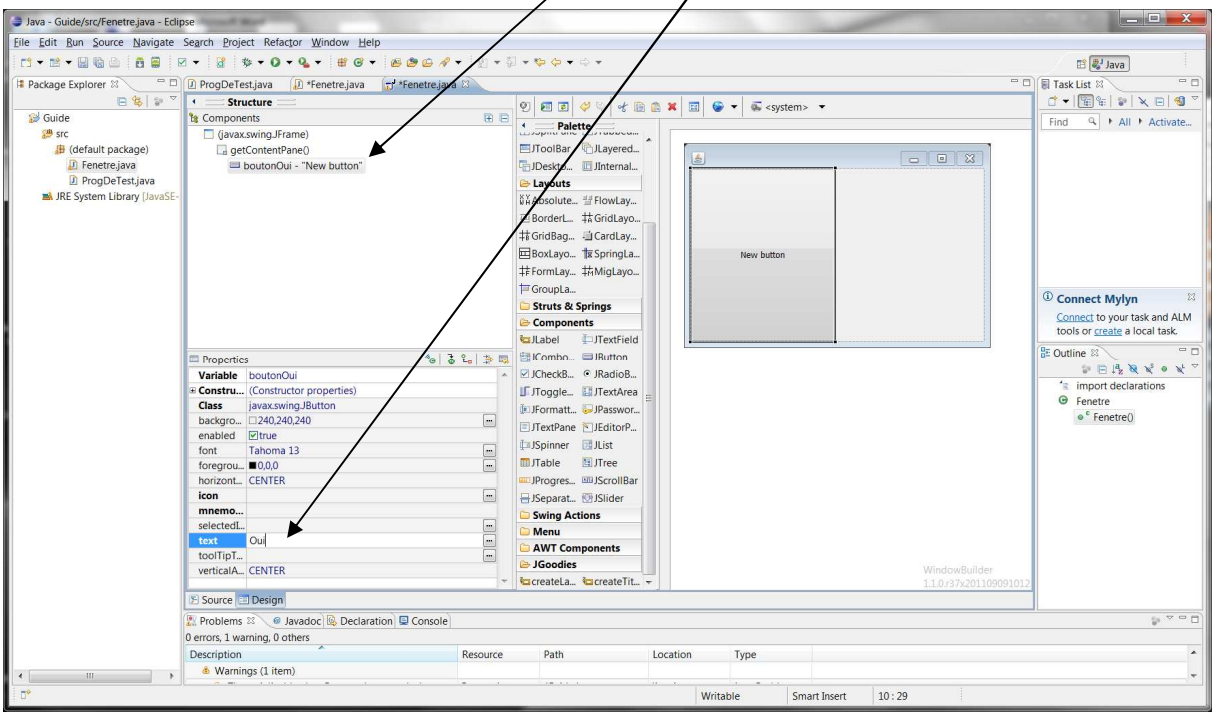
6. Donner un nom à l'objet (par défaut Eclipse l'appelle btnNewButton) :

clic droit sur *button : "NewButton"* → *Rename ...*

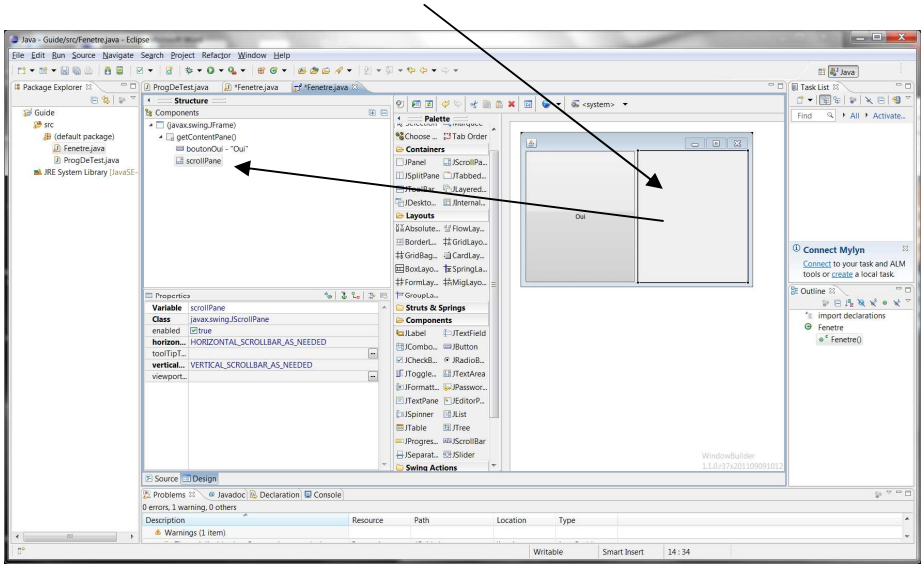


Définir le nom de l'objet (celui sous lequel il sera désigné dans le code de l'application)

7. Définir les paramètres du bouton (par exemple le texte qu'il contient, par défaut c'est "NewButton") :
Dans l'arborescence faire un clic sur le bouton et dans *text* mettre le texte à afficher :



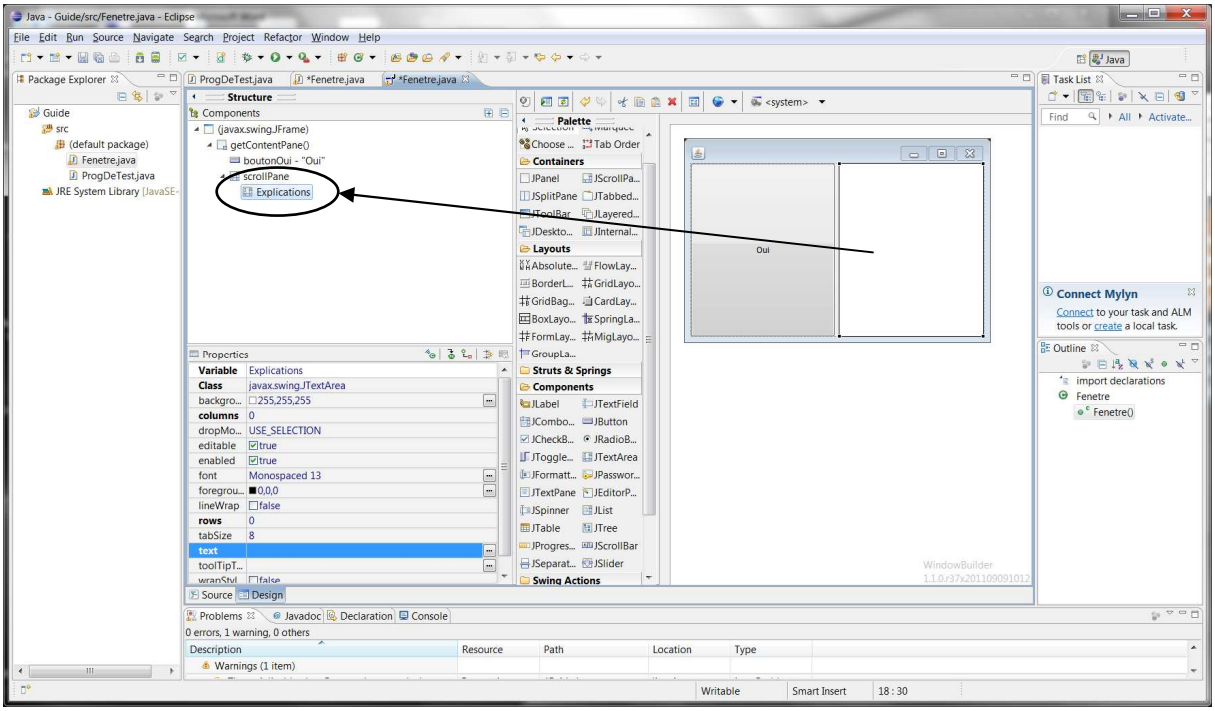
8. Ajout d'un conteneur dans la case de droite (exemple un JScrollPane) :
Dans la liste des conteneurs d'interface choisir JScrollPane puis déplacer la souris vers la partie droite de votre interface (un trait rouge apparaît), cliquer :



9. Ajout d'un widget dans ce conteneur :

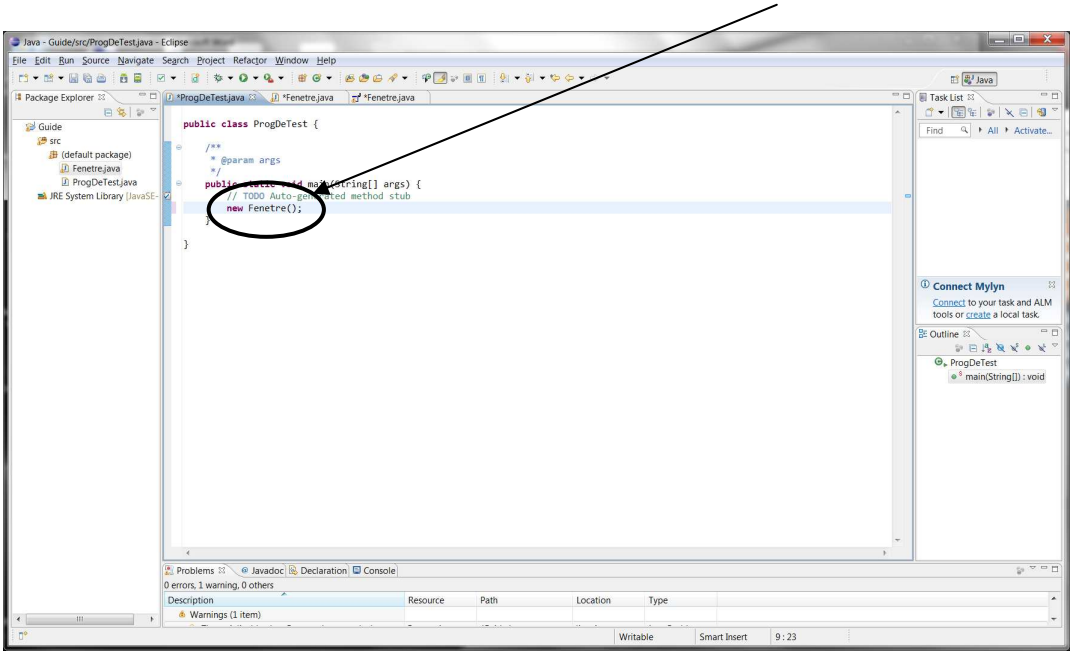
Dans la liste des composants d'interface choisir un widget (par exemple un JTextArea) puis déplacer la souris vers la partie droite de votre interface (un trait rouge apparaît), cliquer.
Puis lui donner un nom par Rename ... et en modifier les propriétés (voir ci-dessus).

L'interface est terminée :



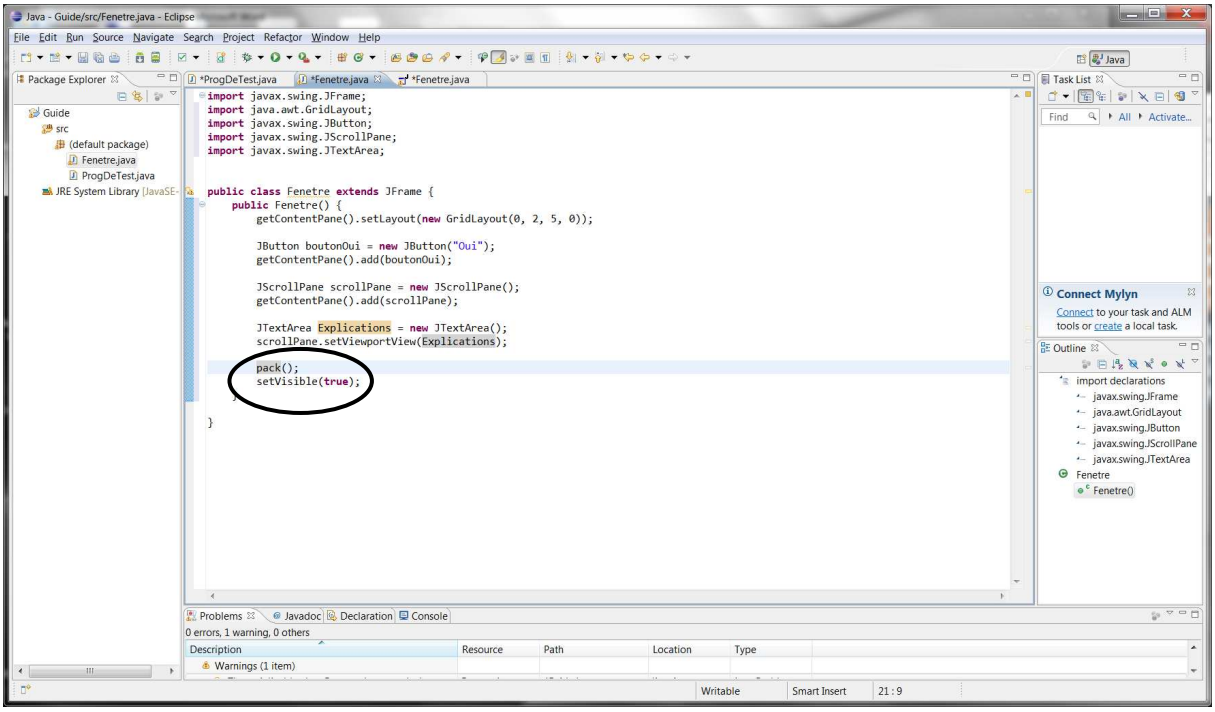
10. Tester votre application :

1°) Dans la méthode *main* ajouter la ligne : new Fenetre();

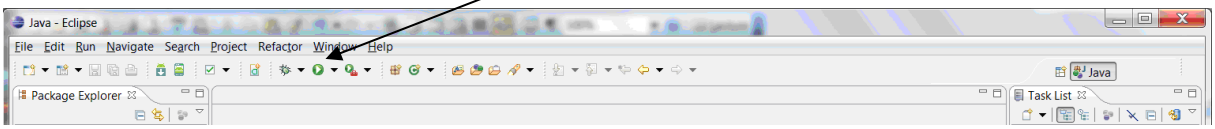


2°) Dans le fichier .java de la classe de l'interface à la fin du constructeur ajouter les lignes :

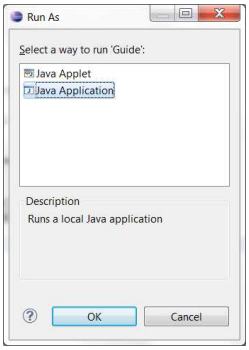
pack();
setVisible(true);

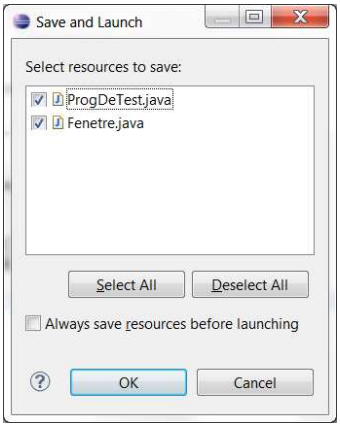


Pour lancer le programme cliquer sur le bouton *run* de la barre d'outils d'Eclipse



Dans la fenêtre qui apparaît (si elle apparaît) choisir Java Application
(il ne sera plus nécessaire de le faire à l'avenir) :

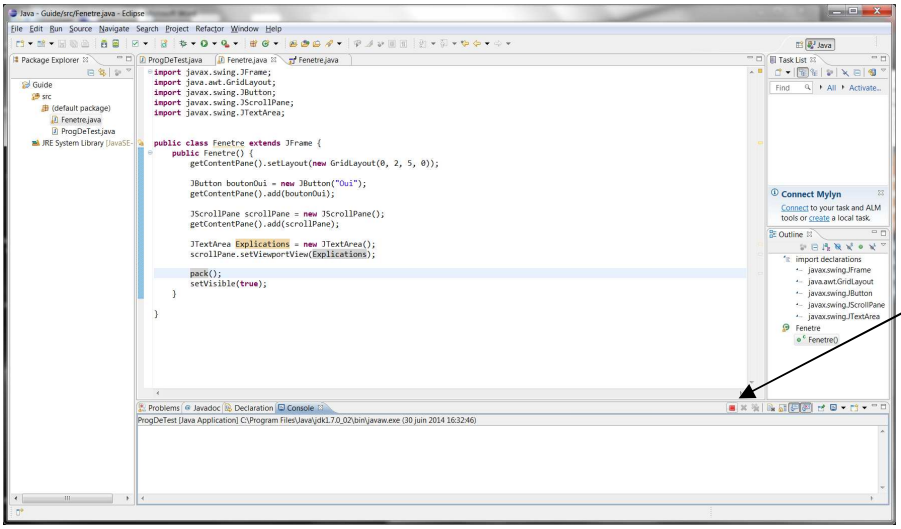




Si certains fichiers .java n'ont pas été sauvegardés cocher les cases correspondantes (en fait elles le sont déjà) et répondre OK

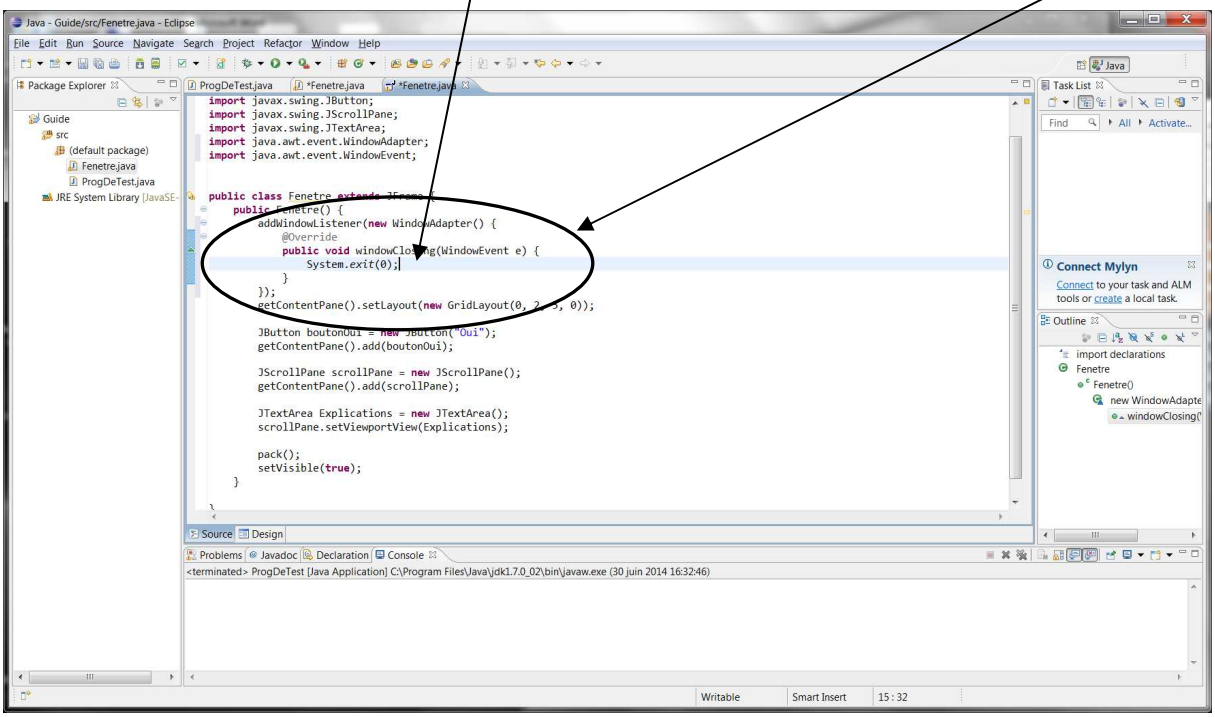
Votre application démarre et l'interface apparaît. Si elle ne convient pas utiliser l'éditeur graphique pour la modifier.

11. Problème de terminaison du programme :



Quand on ferme la fenêtre créée par ce programme le programme ne se termine pas => il faudra le faire par le bouton *terminate* (carré rouge) de l'onglet Console

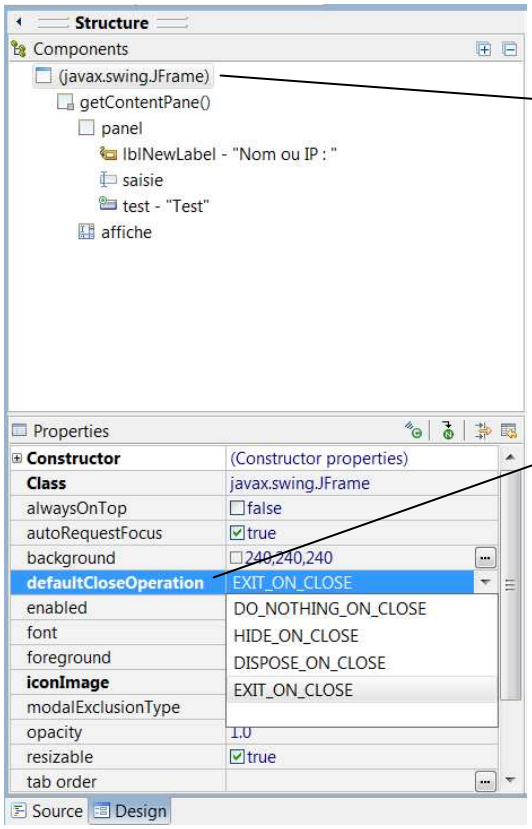
Pour avoir un arrêt automatique du programme quand la fenêtre est fermée, revenir dans l'éditeur graphique d'IHM et, dans l'arborescence de l'interface, faire un clic droit sur javax.swing.JFrame → add event handler → window → windowClosing, le code d'un écouteur d'événements associé à la fermeture de la fenêtre est généré et il faut le compléter par la ligne : System.exit(0);



Maintenant si on relance le programme il se termine bien quand on ferme la fenêtre (le bouton *terminate* de la console est grisé).

La méthode décrite précédemment est très générale, elle permet d'associer n'importe quelle action à la fermeture de la fenêtre par exemple si on doit sauvegarder quelque chose avant de quitter. Dans le cas où on veut seulement que le programme se termine quand on ferme la fenêtre on peut utiliser une façon de procéder plus simple :

1. Dans la liste des composants d'interface choisir la fenêtre (JFrame)
2. Puis dans ses propriétés choisir : DefaultCloseOperation
3. Et dans la liste sélectionner EXIT_ON_CLOSE



Sélectionner la fenêtre

Dans DefaultCloseOperation
choisir EXIT_ON_CLOSE

Eclipse ajoutera dans le constructeur de la fenêtre la ligne suivante :

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

Faire du code propre

Eclipse génère du code peu lisible il est préférable de séparer mieux les choses pour la maintenance.

Ici, par exemple si on a choisi la première solution pour l'arrêt du programme, l'écouteur d'événements lié à la fermeture de la fenêtre a été codé directement dans le constructeur par Eclipse => on va le séparer :

remplacer la ligne : addWindowListener(new WindowAdapter() {
par addWindowListener(new FermerFenetre());

puis déplacer le code entre les {} après le constructeur et le modifier comme suit :

```
private class FermerFenetre extends WindowAdapter
{
        @Override
        public void windowClosing(WindowEvent e) {
                System.exit(0);
        }
}
```

Voila c'est tout mais c'est bien mieux.

De même, en général, les widgets de votre interface sont déclarés en local dans le constructeur or ils devraient être mis en propriétés de la classe => les ajouter en propriétés :

```
public class Fenetre extends JFrame {

    private JButton boutonOui;
    private JTextArea explications;

    public Fenetre() {
    ...
```

Puis supprimer leurs déclarations dans le constructeur :

~~JButton~~ boutonOui = **new** JButton("Oui");

et ~~JTextArea~~ explications = new JTextArea();

Tester et mettre au point

Les erreurs de compilation apparaissent dans l'onglet "Problems". En double cliquant sur une erreur on est renvoyé dans l'éditeur à la ligne de cette erreur, il n'y plus qu'à la corriger.
Le survol d'une ligne erronée dans le code donne des indications sur l'erreur.

Un clic droit propose des solutions pour résoudre l'erreur mais le plus souvent ces solutions ne sont pas les bonnes : elles se contentent de faire en sorte qu'il n'y ait plus d'erreur de compilation.

Soyez donc vigilants et n'acceptez pas sans réfléchir les propositions que fait Eclipse.

Exemple :
Dans le constructeur de l'interface nous avons ajouté la ligne : pack();
Si, par erreur, vous aviez saisi : getContentPane().pack();

Cette méthode n'existe pas et, au survol, Eclipse signale une erreur :

La solution proposée (changer la classe de l'objet renvoyé par *getContentPane*) va modifier le code comme suit :

```
((Window) getContentPane()).pack();
```

Ce qui est syntaxiquement correct mais n'a aucun sens et ne fonctionnera pas : au lancement du programme on obtient dans l'onglet console le message suivant :

```
Exception in thread "main" java.lang.ClassCastException: javax.swing.JPanel cannot be cast to
java.awt.Window
    at Fenetre.<init>(Fenetre.java:31)
    at ProgDeTest.main(ProgDeTest.java:9)
```