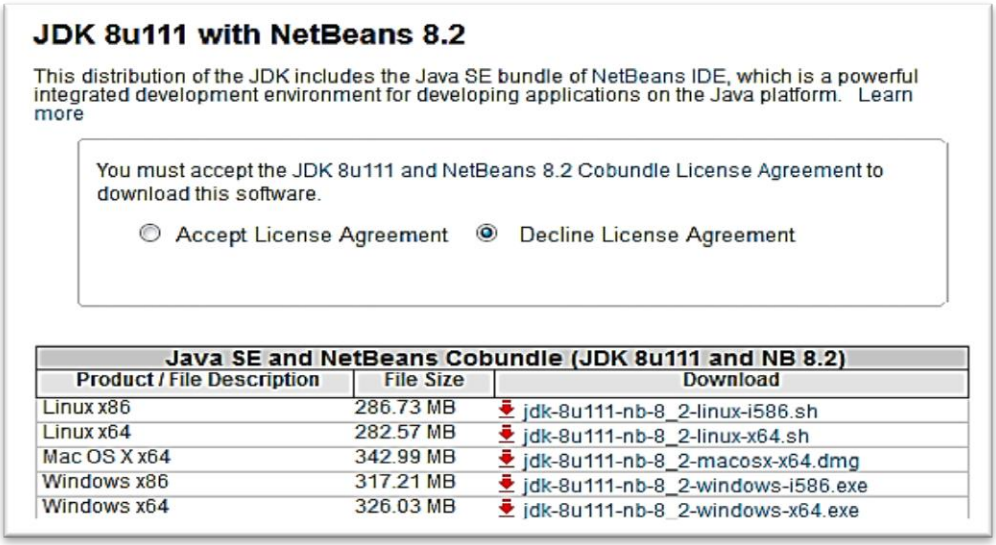
Interfaces Graphiques avec NetBeans

Installation de l'IDE[1] NetBeans

1. Consulter le lien ci-dessous :

https://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-3413139-esa.html

2. Choisir ***Accept License Agreement***
3. Télécharger l'exécutable approprié : à savoir ***Windows x86*** correspond à un système Windows 32 bits et ***Windows x64*** correspond à un système Windows 64 bits



JDK 8u111 with NetBeans 8.2

This distribution of the JDK includes the Java SE bundle of NetBeans IDE, which is a powerful integrated development environment for developing applications on the Java platform. Learn more

You must accept the JDK 8u111 and NetBeans 8.2 Cobundle License Agreement to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Java SE and NetBeans Cobundle (JDK 8u111 and NB 8.2)		
Product / File Description	**File Size**	**Download**
Linux x86	286.73 MB	jdk-8u111-nb-8_2-linux-i586.sh
Linux x64	282.57 MB	jdk-8u111-nb-8_2-linux-x64.sh
Mac OS X x64	342.99 MB	jdk-8u111-nb-8_2-macosx-x64.dmg
Windows x86	317.21 MB	jdk-8u111-nb-8_2-windows-i586.exe
Windows x64	326.03 MB	jdk-8u111-nb-8_2-windows-x64.exe

La principale caractéristique d'un programme avec GUI (***G**raphical **U**ser **I**nterface*) réside dans la notion de *programmation événementielle*. Le programme doit détecter et répondre aux *actions* de l'utilisateur, comme par exemple, un clic de souris, la frappe d'une touche du clavier, etc. Cela est en opposition avec *un programme à interface console* où on a l'impression que c'est le programme qui pilote l'utilisateur en le sollicitant au moment voulu pour qu'il fournisse des informations. A savoir, ce dialogue se fait en *mode texte* et de façon séquentielle dans une fenêtre nommée *console*.

Objectif du TP: En utilisant les classes Swing de JAVA, nous allons construire un programme à interface graphique qui permet d'additionner deux nombres réels.

Partie I

D'abord, lancer *NetBeans IDE*

Etape 1 : Création du Projet

- File > New Project
- Categories / Java & Projects/ Java Application > Next
- Project Name ……Nommer le programme
- Décocher (Create Main Class)
- Finish

[1] *Integrated Development Environment*

Etape 2 : Création d'une fenêtre

- File > New File
- Categories /Swing Gui Forms & File Types/ Jframe Form > Next
- Class Name …….. Nommer la classe
- Package …….. Nommer le package
- Finish

Jusqu'ici, nous avons créé le *conteneur principal* « Frame » qui va encapsuler toutes les entités de notre interface graphique.

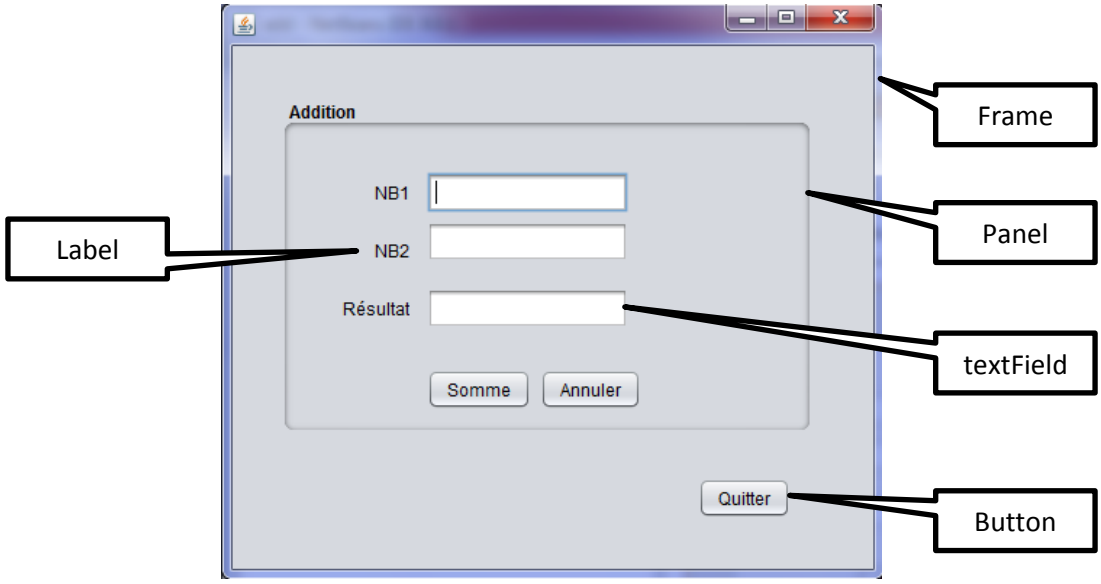


Figure 1. Interface graphique.

Pour construire l'interface graphique, nous allons travailler sous l'onglet *Design* (Conception) où nous nous servirons, principalement, de la *palette des composants*.

Etape 3: Insérer un *conteneur intermédiaire* de type *Panel* (Panneau). Son rôle est de regrouper d'autres composants. Utiliser la propriété « *border* » pour choisir « *Titled Border* » (ici le titre est Addition). Il est possible, entre autres, de changer la couleur du fond en utilisant la propriété « *background* ».

Etape 4: En se référant à la **Figure 1**, insérer le reste des *composants atomiques* et ce à partir de la partie *Swing Controls* de la palette.

- Pour créer une étiquette, insérer un ***Label***.
- Pour créer un champ de texte (une zone de saisie de texte sur une ligne), insérer un ***TextField***.
- Pour créer un bouton, insérer un ***Button.***

NB. Pour le champ de texte résultat, on peut désactiver la saisie en décochant la propriété *editable*.

Partie II

Dans cette partie, nous allons attribuer des fonctionnalités aux boutons : Somme, Annuler et Quitter. Par un double clic sur un bouton, on se trouve devant le code source :

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

C'est à nous de programmer la tâche que nous souhaitons que le bouton fasse lorsque le bouton est enfoncé (soit par un clic de souris ou via le clavier).

- **Bouton Quitter :** Lorsque l'utilisateur clique sur Quitter, l'application doit se fermer (le code 0 indique une sortie normal). Alors, on doit ajouter l'instruction :

```
System.exit(0) ;
```

- **Bouton Annuler :** Lorsque l'utilisateur clique sur Annuler, le contenu des champs de texte doit être réinitialisé ou simplement être effacé. Concrètement, nous affectons la chaine vide aux champs de saisie.

```
jTextField1.setText("");
jTextField2.setText("");
jTextField3.setText("");
```

- **Bouton Somme :** Lorsque l'utilisateur clique sur Somme, le contenu de type String des champs jTextField1 et jTextField2 doit être récupéré (grâce à a méthode ***getText()***) et puis converti au type Float (grâce à la méthode ***parseFloat***). Enfin, la somme res est convertie au type String (grâce à la méthode ***valueOf()***) et placé dans jTextField3.

```
float n1,n2,res;
n1=Float.parseFloat(jTextField1.getText());
n2=Float.parseFloat(jTextField2.getText());
res=n1+n2;
jTextField3.setText(String.valueOf(res));
```

Remarque sur l'exécution des programmes

- Dans l'IDE
  Choisir Run > Run Project
- En dehors de l'IDE
  Choisir Run > Clean and Build Project afin de créer le fichier JAR correspondant.
  Un double clic sur le fichier .jar (généralement placé sous le répertoire \Documents\NetBeansProjects) suffit pour lancer l'application.

Partie III (Gestion des exceptions)

Le terme *exception* désigne tout événement arrivant durant l'exécution d'un programme interrompant son fonctionnement normal. Autrement, les exceptions sont des erreurs survenant lors de l'exécution d'un programme. Voici des exemples :

- Trouver des lettres quand une variable de type numérique est attendue.
- Une division par zéro.
- L'indexation d'un composant de tableau en dehors du domaine de définition du type des indices.
- Allocation mémoire impossible car Espace mémoire insuffisant.
- Etc.

Pour *traiter une exception* produite par l'exécution d'une action A, il faut la placer dans une clause ***try*** (qui veut dire essayer). Suivie obligatoirement d'une clause ***catch*** (qui veut dire attraper) qui contient le traitement de l'exception

```
try                          {  A  }
catch (uneException e) {   B }
```

Dans notre exemple, l'utilisateur peut lancer le calcul alors qu'un parmi les champs est vide. Aussi, il peut fournir des données non numériques. Dans les deux cas, il s'agit de l'exception prédéfinie « ***NumberFormatException*** » *levée* ici par la méthode *parseFloat().*

Pour ce Tp, on peut par exemple créer une *boite de dialogue* pour demander à l'utilisateur de « saisir des données numérique ». Pour ce faire, aller dans la partie *Swing Windows* et insérer un composant *Dialog*. Au sein de cette boite de dialogue, insérer un message adéquat avec un bouton ok. Enfin, le code correspondant au bouton Somme devient comme suit :

```
try{
      float n1,n2,res;
      n1=Float.parseFloat(jTextField1.getText());
      ….
      jTextField3.setText(String.valueOf(res));
      } catch (NumberFormatException e){
      jDialog1.setVisible(true); }
```

En appuyant sur le bouton ok, la boite de dialogue doit juste se fermer. D'où le code :

```
private void jButton4ActionPerformed (java.awt.event.ActionEvent evt) {
     jDialog1.setVisible(flase); }
```

Exercice supplémentaire

On veut construire un programme avec interface graphique qui permet de calculer quelques métriques de performance réseau.

- Temps d'émission (TE): délai entre le début et la fin de la transmission d'un message.

  Te= Taille / Débit

- Temps de propagation (TP): temps nécessaire au signal pour passer d'un bout à l'autre du support.

  TP = Distance / Vitesse de Propagation du signal

- Délai d'acheminement (Da): temps nécessaire pour que les données arrivent au récepteur.

  DA= TE+TP

Pour guider l'utilisateur, insérer les unités comme des Label à côté de chaque champ :

Mesure	**Unité**
Temps / Délai	Seconde
Taille	Mbit
Débit	Mbit/s
Longueur	KM
Vitesse	KM/s

On doit prendre en considération les différentes exceptions qui peuvent se produire (entrées non numériques, division par zéro, etc.)