Introduction to GUI Building

Contenu

Comment fonctionne la gestion des évènements

Exercice 1: Créer un projet

ÉTAPE 1. **Créer un projet IDE** pour **l'application** à développer. Ex. Nom de projet: **Addition_de_Nombres**

1. Choisir **File > New Project**, Alternativement, on peut cliquer sur l'icone **New Project** icon dans la toolbar IDE.
2. Dans la palette des **Categories**, selectionner **Java node**,
3. Dans la palette **Projects**, choisir **Java Application**. Click **Next**.
4. Taper **Addition_de_Nombres** dans le champs **Project Name** et spécifier un chemin pour l'emplacement du projet,
5. Assuer vous que the Set as Main Project checkbox is selectionnée,
6. Décocher la case de Create Main Class checkbox.
7. Click Finish.

Exercice 2: Construire l'interface utilisateur

Pour procéder dans la construction de l'interface utilisateur, nous avons besoin de créer un conteneur Java (**Java container)** dans lequel nous placerons les autres éléments graphiques de l'interface .

Dans cette étape,

1. Nous allons créer un conteneur en utilisant le composant **JFrame** ,
2. Nous plaçons le conteneur dans un nouveau **package** apparaissant dans le noeud des packages sources.

Exercice 2: Construire l'interface utilisateur

2.1. Créer un conteneur JFrame

1. Dans la fenêtre **Projects**, click droit sur le noeud Addition_de_Nombres,
2. choisr **New > JFrame Form**.
3. Taper **IUAdditionNombres** comme nom de **classe**,
4. Taper **Mon_AdditionNombres** comme nom de **package**.
5. Click Finish.

L'IDE crée le formulaire UIAdditionNombres ainsi que la classe IUAdditionNombres dans l'application Addition_de_Nombres, et ouvre le formulaire IUAdditionNombres dans le GUI Builder. Le package Mon_AdditionNombres remplace le package par défaut.

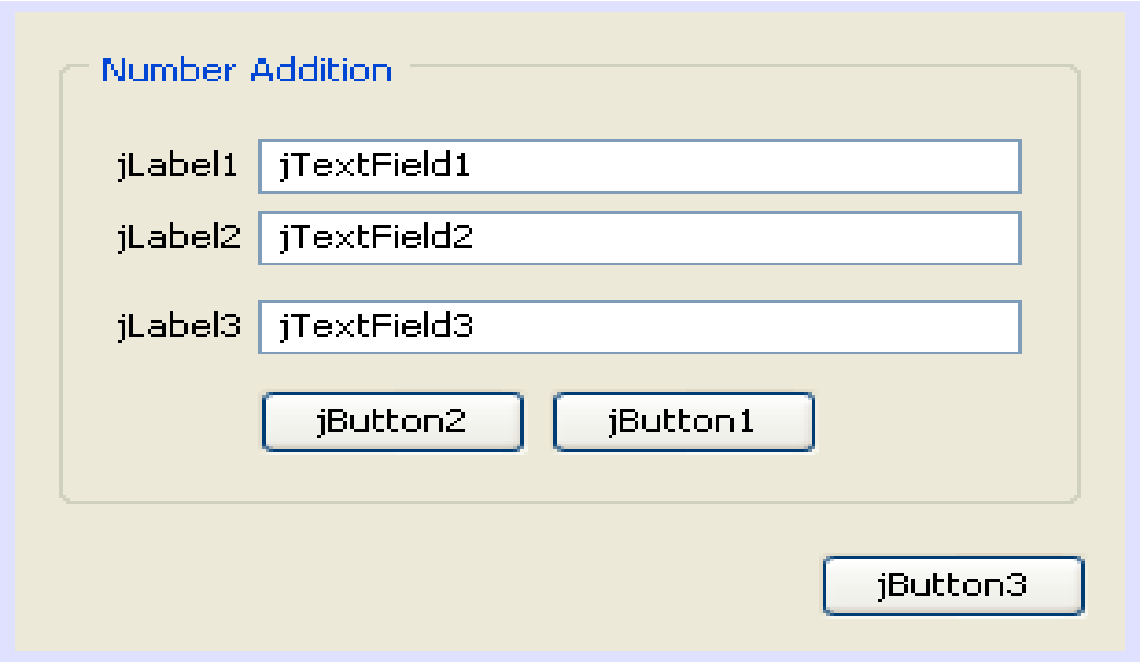
Exercice 2: Construire l'interface utilisateur

2.2. Ajouter des composants

On utilise la palette d'outils pour garnir l'interface de notre application avec Jpanel.

Nous ajoutons 3 **JLabels**, 3 **JTextFields**, et 3 **JButtons**.

- Après positionnement des différents composants rajoutés, le **Jframe** peut ressembler à ceci.



Exercice 2: Construire l'interface utilisateur

- Si la fenetre des palettes n'est apparente, Choisir **Windows > Palette.**

1. Selectionner un **JPanel** à partir de la palette et placer le dans le **Jframe,**
2. Pendant que Jpanel est en surbrillance, choisir la fenetre des **propriétés** et choisr le bouton ellipse (…) sur la bordure pour selectionner un style de bordure du bouton,
3. Dans la boite de dialogue bordure, selectionner **TitledBorder** à partir de la liste,
4. Taper **Addition de Nombres** dans le champs de **titre**,
5. OK pour enregistrer les changements puis exit,
6. Nous obtenons un Jframe vierge intitulé **Addition de Nombres**
7. Puis rajouter les 3**Jlabels,** 3 **JTextFields** et 3 **Jbuttons,**

Exercice 2: Construire l'interface utilisateur

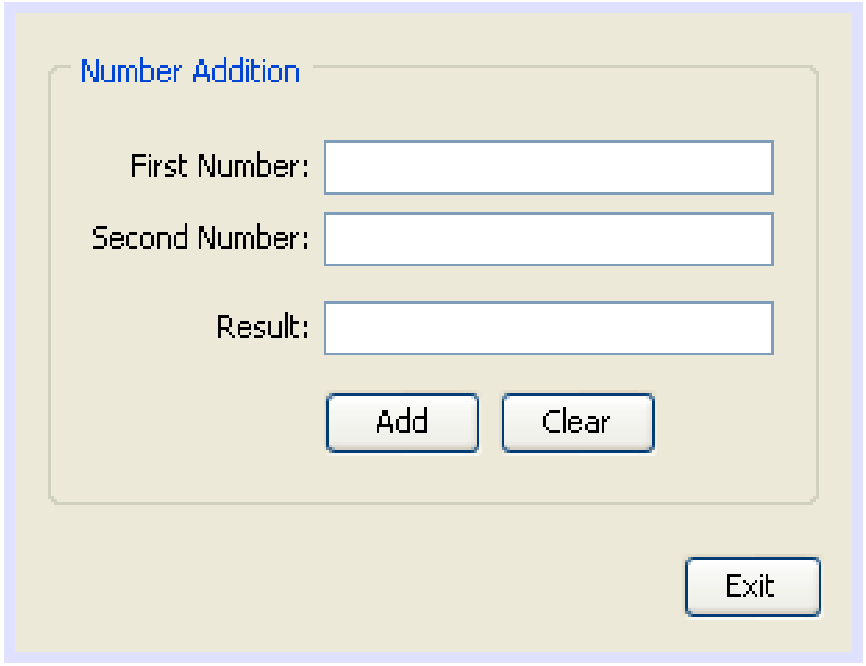
2.3. Renommer les composants

- Nous allons renommer l'affichage des textes des composants ajoutés dans le JFrame,

1. Double-click jLabel1 et changer la propriété texte à **First Number**
2. Double-click jLabel2 et changer la propriété texte à **Second Number**
3. Double-click jLabel3 et changer la propriété texte à **Result**
4. Effacer les textes par défaut des jTextField1. jTextField2 etjTextField3.
5. Renommer le texte affiché de jButton1 à **Clear**.
6. Renommer le texte affiché de jButton2 à **Add**.
7. Renommer le texte affiché de jButton3 à **Exit**.

Exercice 2: Construire l'interface utilisateur

- Notre IU ressemble à ceci :



Exercise 3: Ajout de fonctionnalités

- Nous allons ajouter des fonctionnalités aux boutons **Add**, **Clear**, et **Exit**.
- Les boites **jTextField1** et **jTextField2** seront utilisées pour les entrées utilisateur et
- La boite **jTextField3** pour le result du programme

3.1. Rendre le bouton Exit fonctionner

Nous assignons un gestionnaire d'évènements pour chaque bouton:
nous voulons savoir quand le bouton est pressé, soit par un click de souris ou par clavier.
Nous utilisons **ActionListener** correspondant à **ActionEvent**.

1. Click droit sur le bouton Exit,
2. A partir du menu Pop-up, chosir **Events > Action > ActionPerformed**.

- Remarque: Le menu contient pluisuers évenements pouvant être assignés. Quand, ActionPerformed est choisi, l'IDE ajoute automatiquement un ActionListener au bouton Exit et génére une méthode handler pour executer la méthode ActionPerformed du Listener.

1. L'IDE ouvre la fenetre de code sources en choisissant l'emplacement de la méthode à executer. Le code source doit contenir les lignes suivantes de code suivantes :

```
2. private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
3. //TODO: Ajouter le code ici :                }
```

4. Ajouter le code pour le bouton Exit, type System.exit(0); to the above code, replacing the TODO line. La fin du code :

```
5. private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
6. System.exit(0);                }
```

Exercise 3: Ajout de fonctionnalités

3.2. Rendre le bouton Clear fonctionner

2. Design tab at the top of your work area to go back to the Form Design
3. Click droit sur le bouton Clear (jButton1).
4. pop-up menu select Events > Action > actionPerformed.
5. button Clear: Efface les textes des champs jTextFields.
6. Le code suivant :
7. private void
8. jButton1ActionPerformed(java.awt.event.ActionEvent evt){
9. jTextField1.setText(""); jTextField2.setText("");
10. jTextField3.setText(""); }

Exercise 3: Ajout de fonctionnalités

3.3. Rendre le bouton Add fonctionner

Trois actions:

1. It is going to accept user input from jTextField1 and jTextField2 acceptent des entrées utilisateur et convertit les données de type texte en réel.
2. Faire l'addition des deux nombres et convertit la somme au type texte et placer le résultat dans jTextField3.

3. Design tab at the top of your work area to go back to the Form Design.
4. Click droit sur le bouton Add (jButton2). Puis selctionner Events > Action > actionPerformed
5. On ajoute du code au bouton. Le code suivant :
6. private void jButton2ActionPerformed(java.awt.event.ActionEvent evt){
7. // First we define float variables.
8. float num1, num2, result;
9. // We have to parse the text to a type float.
10. num1 = Float.parseFloat(jTextField1.getText());
11. num2 = Float.parseFloat(jTextField2.getText()); /
12. / Faire l'addition.
13. result = num1+num2;
14. // Passer le résultat dans jTextField3.
15. // changer la valeur du resultat du réel en texte jTextField3.setText(String.valueOf(result)); }

Exercise 4: Executer le programme

- L'étape finale est de construire et executer le programme.

1. Choisir **Build > Build Main Project**.
2. Choisir **Run > Run Main Project**
3. Si on a une fenêtre informant que le projet n'a pas de classe principale choisie, on selectionne **Mon_AdditionNombres.IUAdditionNombres** comme classe principale,
4. Le programme maintenant est en execution

How Event Handling Works

- This tutorial showed how to respond to a simple button event. There are many more events you can have your application respond to. The IDE can help you find the list of available events your GUI components can handle:
- Go back to the file NumberAdditionUI.java in the Editor. Click the Design tab to see the GUI's layout in the GUI Builder.
- Right-click any GUI component, and select Events from the pop-up menu. For now, just browse the menu to see what's there, you don't need to select anything.
- Alternatively, you can select Properties from the Window menu. In the Properties window, click the Events tab. In the Events tab, you can view and edit events handlers associated with the currently active GUI component.
- You can have your application respond to key presses, single, double and triple mouse clicks, mouse motion, window size and focus changes. You can generate event handlers for all of them from the Events menu. The most common event you will use is an Action event. (Learn best practices for Event handling from Sun's Java Events Tutorial.)

Gestion des évènements

1. Dans le fichier **IUAdditionNombres.java** dans l'éditeur.
2. Regarder les methodes **jButton1ActionPerformed(), jButton2ActionPerformed(), etjButton3ActionPerformed()** déjà implementées. cesmethodes sont appelées **event handlers**.
3. Regarder la methode appelée **initComponents()**. Si vous ne voyez pas cette méthode, chercher la ligne generated code; click sur le signe + pour etendre la methode initComponents() condensée.
4. Naviguer dasn la méthode initComponents(). Ce code est généré et mis ç jour automatically quand on édite et place des composants dans la vue concption.
5. Dans initComponents(), retrouver les lignes suivantes:
6. jButton3.setText("Exit");
7. jButton3.addActionListener(new java.awt.event.ActionListener() {
8. public void actionPerformed(java.awt.event.ActionEvent evt) { jButton3ActionPerformed(evt);
9. }
10. });

Gestion des évènements

- This is the spot where an event listener object is added to the GUI component; in this case, you register an ActionListener to the jButton3.
- The ActionListener interface has an actionPerformed method taking ActionEvent object which is implemented simply by calling your jButton3ActionPerformed event handler.
- The button is now listening to action events. Everytime it is pressed an ActionEvent is generated and passed to the listener's actionPerformed method which in turn executes code that you provided in the event handler for this event.
- Generally speaking, to be able to respond, each interactive GUI component needs to register to an event listener and needs to implement an event handler.
- As you can see, NetBeans IDE handles hooking up the event listener for you, so you can concentrate on implementing the actual business logic that should be triggered by the event.