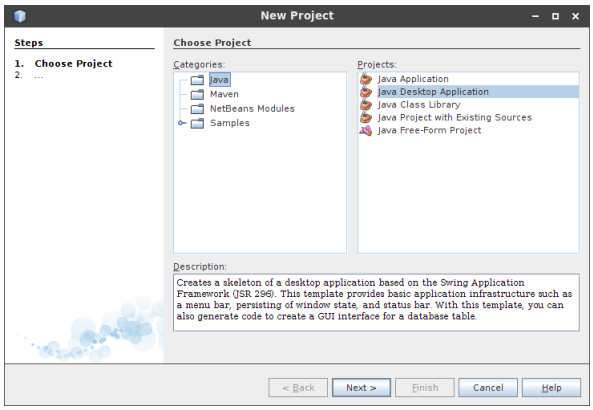
Annexe - TP1

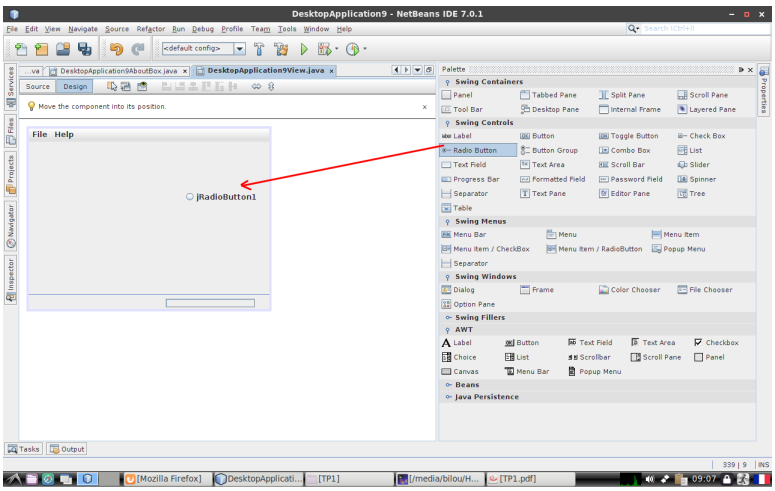
Graphical User Interface (GUI): Une interface graphique est moyen visuel d'interaction entre l'homme et la machine. Elle est composée de plusieurs éléments graphiques qui assurent cette interaction: Boutons, zones de texte, menus, etc.
Swing: est un module (classe) Java qui fournit des fonctions permettant de créer les éléments graphiques d'une interface, par exemple jTextField, jButton, jLabel, etc. Il faut savoir que toute interface graphique Swing doit comporter au moins un conteneur (fenêtre). Il en existe trois types: jPanel (une fenêtre), jFrame (une fenêtre séparée), jDialog (Une fenêtre indépendante qui affiche un message).
Un exemple de programme en Swing: Soit le programme suivant qui affiche une simple fenêtre contenant un boutton. Crée un nouveau projet (java application), puis copier le code ci-dessous.

```
import javax.swing.*;
import java.awt.FlowLayout;

class gui{
        public static void main(String args[]){
                JFrame frame = new JFrame("My First GUI");  //création de l'interface
                frame.setBounds(100,100,300,300);  // Déinir la position et la taille
                JButton button = new JButton("Valider");    // Crée un boutton
                button.setBounds(110,110,30,20);
                frame.getContentPane().add(button); // Ajouter le boutton à la fenêtre
                frame.setVisible(true);  // afficher la fenêtre
        }
}
```

La palette de l'IDE (Netbeans par exemple): La palette est une boite à outils intégrée dans Neatbeans qui permet d'insérer les éléments graphiques sans avoir besoin d'écrire le code source de ces éléments. On a donc deux moyen de créer une interface graphique, soit en écrivant le code soit même, soit en utilisant la palette. Elle se situe dans la partie droite de la fenêtre Netbeans; et elle contient des éléments tels que Label, Button, CheckBox, etc. Pour insérer un élément tel qu'un bouton, il suffit juste de le glisser vers l'interface. La figure ci-dessous illustre les étapes à suivre pour créer la même interface décrite par le code ci-dessus.



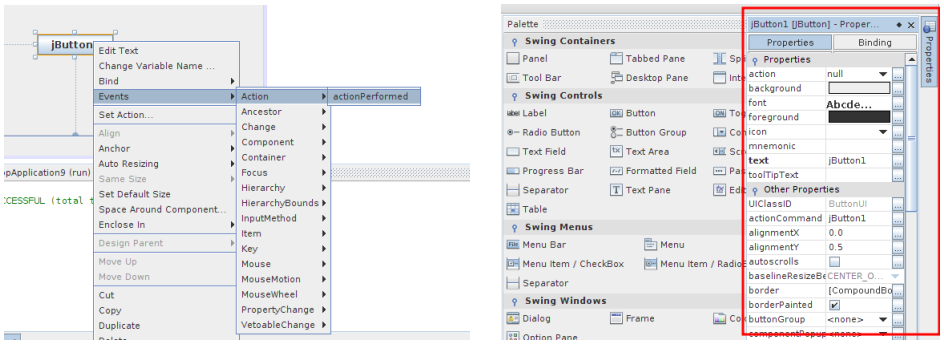


La gestion des événements: Un événement est une action qui se produit suite à un actionnement d'un composant graphique, un clic souris, touche clavier, changement d'élément dans une liste déroulante, etc. La classe qui reçoit l'événement reste en écoute jusqu'à ce quelle reçoit le signal de la source. Par exemple, l'appui sur le bouton ESC produit une boite de dialogue pour confirmer la fermeture de la fenêtre, ou par exemple en saisissant une lettre dans un champ destiné exclusivement aux chiffres un message d'erreur apparaîtra.
Voici un exemple qui illustre ce principe. Un message est affiché dans une boite de dialogue lorsque un bouton est actionné.

```
b_ok.addActionListener(new ActionListener()
{
        @Override
        public void actionPerformed(ActionEvent e)
        {
                // ... Action du bouton OK ici
                System.out.println("Clic du bouton OK");
```

```
        }
});
```

Le bouton est en mode écoute, l'événement d'affichage du message "click du boutton ok" est déclenché lorsque le bouton est appuyé.
Noter qu'on peut aussi associer un événement à une composante (bouton par exemple) en faisant un clique droit, puis choisir Events, puis choisir l'action adéquate (Figure ci-dessous - (a)). Il y a plusieurs événements disponibles, le plus courant est *actionPerformed* qui signifie que la composante est actionnée.



(a) **(b)**

Propriétés des éléments: Chaque éléments graphique a des propriétés tels que sa taille, le texte qui s'affiche dessus, la position, etc. Ces propriétés pourront d'être modifiés directement via l'outils *properties* dans l'environnement de développement (voir (b) dans la figure ci-dessus).

Fonctions utiles

Voici une liste de quelques fonctions qui pourront vous aider à la réalisation de ce TP [1].

- **jRadioButton1.setSelected(true);** permet de sélectionner le bouton jRadioButton1 par défaut.
- **jbuttonGroup1.add(jRadioButton1);** permet de rajouter le bouton radio jRadioButton1 à un groupe jbuttonGroup1. Les boutons mutuellement exclusifs doivent appartenir au même groupe.
- **jComboBox1.setEnabled(false);** permet de désactiver la liste jComboBox1.
- **jComboBox1.removeAllItems();** permet de supprimer les éléments (items) dans une liste jComboBox1.
- **jComboBox2.addItem("item text");** permet de rajouter un élément (item) à une liste jComboBox2.
- **jTextField2.setEditable(false);** permet d'interdire l'écriture dans une zone de texte (jTextField2).
- **jTextField1.setText("une string");** permet d'insérer une valeur dans une zone de texte (jTextField1).
- **String val = jTextField1.getText();** permet de récupérer le contenu d'une zone de texte et le mettre dans une variable (val).
- **Integer val = Integer.parseInt(input);** permet de convertir une string (input) en un entier (val).
- **String resS = String.valueOf(res);** permet de convertir un entier (res) en une string (resS).
- **Integer val = jSlider1.getValue();** permet de récupérer la valeur d'un *jSlider*, et la mettre dans une variable (ici val).
- **JOptionPane.showMessageDialog(this, "message", " titre ", JOptionPane.WARNING_MESSAGE);** permet d'afficher une boite de dialogue pour fournir une information.
- **JOptionPane d = new JOptionPane().YES_NO_OPTION; int retour = d.showConfirmDialog(laFrame, "le message", "le titre", messageType);** Boite de dialogue pour confirmation.
- **System.exit();** pour fermer une fenêtre.

[1] Le code Java fournie correspond aux fonctions Swing. Vous pouvez aussi utiliser la palette (fournie dans Netbeans, Eclipse) pour créer les éléments graphiques plus facilement.