

Programmation Orientée Objets - JAVA

Erick STATNER

Maître de Conférences en Informatique

Université des Antilles

erick.stattner@univ-ag.fr

www.erickstattner.com



Chapitre V. Interfaces Graphiques

1. Introduction
2. Fenêtre
3. Conteneurs et composants
4. Gestionnaires de mise en forme
5. Gestion des évènements
6. Dessins dans un composant



3

V. Interfaces Graphiques

1. Introduction

V. Interfaces graphiques

1) Introduction

Contexte

- ▶ JAVA permet de mettre en place des IHM
- ▶ Trois principales API
 - ▶ AWT (JAVA 1.0)
 - ▶ SWING (JAVA 1.2)
 - ▶ JAVAFX (JAVA 8)
- ▶ Dans ce cours: SWING
 - ▶ Packages: `java.awt.*` et `javax.swing.*`

V. Interfaces graphiques

1) Introduction

Structure d'une interface graphique

- ▶ Conteneurs:
 - ▶ Fenêtre (**JFrame**)
 - ▶ Panneau (**JPanel**)
 - ▶ Onglets (**JTabbedPane**)
 - ▶ ...
- ▶ Composants
 - ▶ Boutons (**JButton**)
 - ▶ Liste déroulante (**JComboBox**)
 - ▶ Boite à cocher (**JCheckBox**)
 - ▶ ...
- ▶ Gestionnaires de mise en forme
- ▶ Gestionnaire d'évènements



6

V. Interfaces Graphiques

2. Fenêtre

V. Interfaces graphiques

2) Fenêtre

La fenêtre

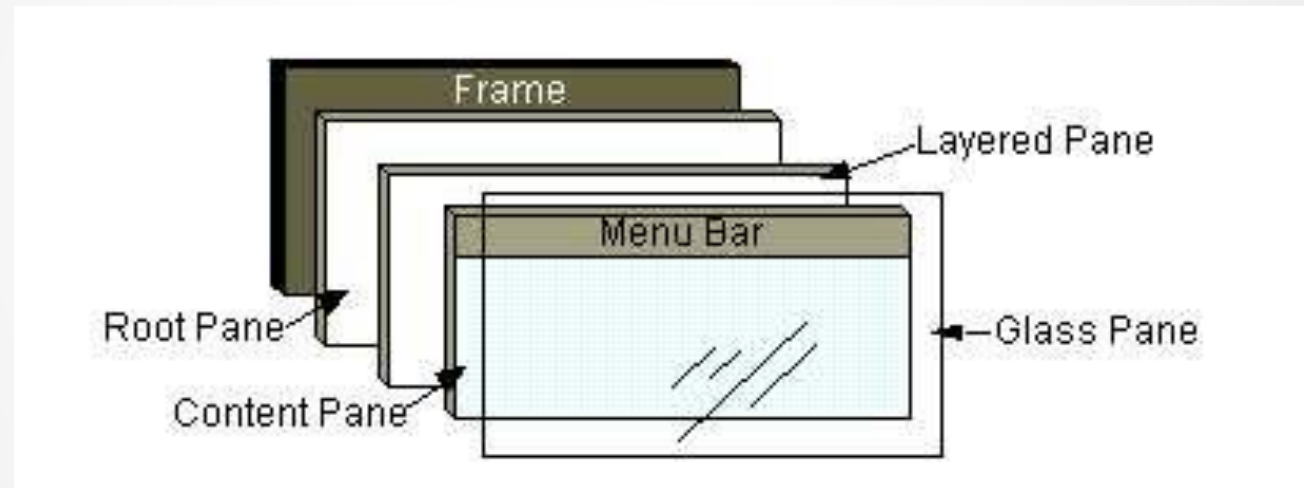
- ▶ Classe **JFrame**
 - ▶ Constructeur vide
- ▶ Plusieurs méthodes
 - ▶ **setSize**
 - ▶ **setTitle**
 - ▶ **setVisible**
- ▶ Par défaut, la fermeture d'une fenêtre n'arrête pas le programme
 - ▶ **setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);**



V. Interfaces graphiques

2) Fenêtre

Structure d'un JFrame



- Les composants sont ajoutés au contenu
- On récupère le contenu avec la méthode **Container contenu = getContentPane()**
- On ajoute des éléments au contenu avec la méthode **contenu.add()**

V. Interfaces graphiques

2) Fenêtre

Exercice

1. Créer une fenêtre
 1. Lui donner une taille de 800px par 600px
 2. Lui donner un titre
 3. Permettre que le programme s'arrête quand on la ferme
 4. Récupérer une référence vers son contenu
 5. La rendre visible

V. Interfaces Graphiques

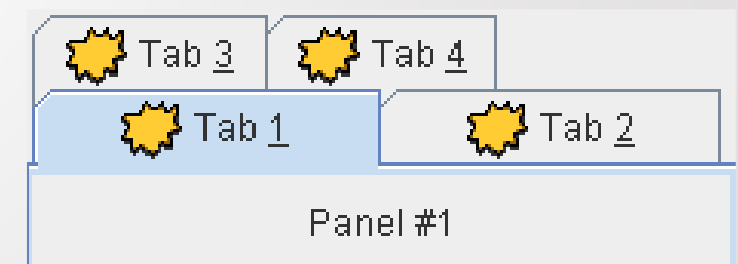
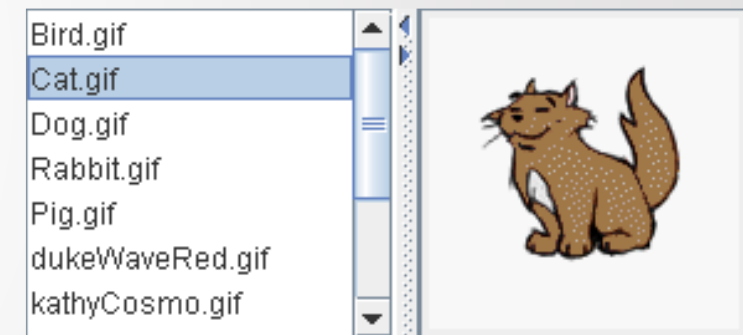
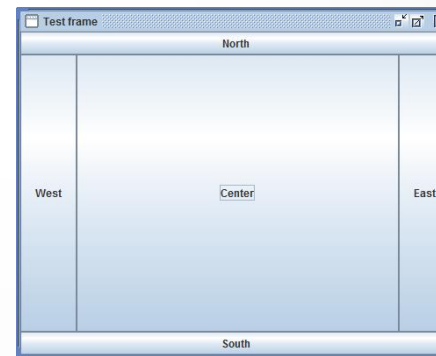
3. Conteneurs et composants

V. Interfaces graphiques

3) Conteneurs et composants

Conteneurs

- ▶ Permettent de structurer l'interface
- ▶ Plusieurs classes
 - ▶ **JPanel**
 - ▶ **JScrollPane**
 - ▶ **JSplitPane**
 - ▶ **JTabbedPane**
 - ▶ **JToolBar**
 - ▶ ...
- ▶ Pour ajouter des éléments au conteneur, utiliser la méthode
 - ▶ **add**
- ▶ Un conteneur peut en contenir un autre



V. Interfaces graphiques

3) Conteneurs et composants

Exercice

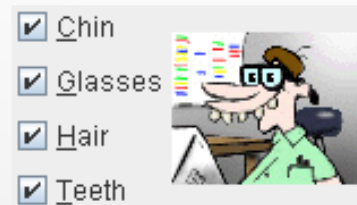
1. Ajouter à la fenetre un JPanel P1

V. Interfaces graphiques

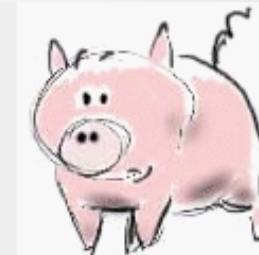
3) Conteneurs et composants

Composants: Les boutons

- ▶ JButton
- ▶ JCheckbox
- ▶ JRadioButton



- Bird
- Cat
- Dog
- Rabbit
- Pig



V. Interfaces graphiques

3) Conteneurs et composants

Composants: Les textes

- ▶ JTextField
- ▶ JPasswordField
- ▶ JTextArea

Sur chacun des objets

- ▶ `setText`
- ▶ `getText`

City:

Enter the password:

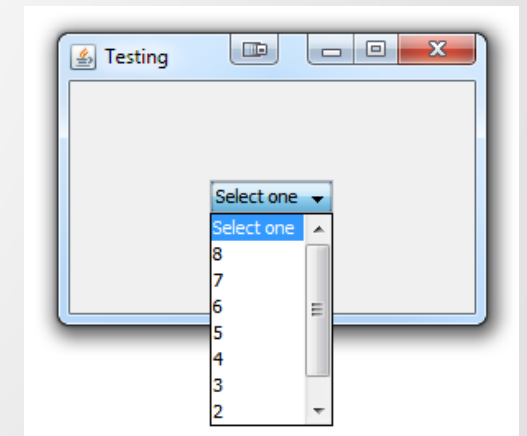
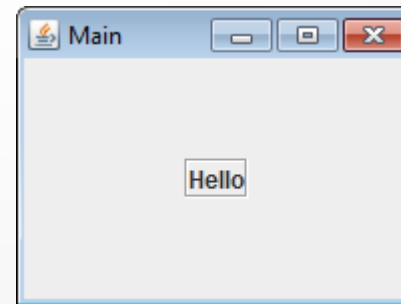
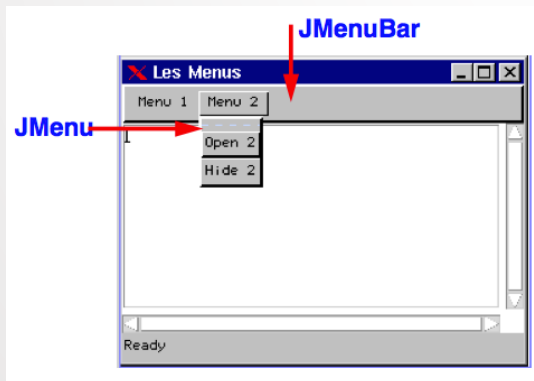
This is an editable JTextArea. A text area is a "plain" text component, which means that although it can display text in any font, all of the text is in the same font.

V. Interfaces graphiques

3) Conteneurs et composants

De nombreux autres composants

- ▶ JMenu
- ▶ JLabel
- ▶ JComboBox



V. Interfaces graphiques

3) Conteneurs et composants

Exercice

1. Ajouter a P1 un JLabel avec l'étiquette « Nom »
2. Ajouter à P1 un JTextField vide
3. Ajouter à P1 un JButton avec le message « valider »

V. Interfaces Graphiques

4. Gestionnaires de mise en forme

V. Interfaces graphiques

4) Gestionnaires de mise en forme

Comment disposer les éléments ?

- Pour chaque conteneur, choisir un gestionnaire de mise en forme
- Chargé de la disposition des éléments
- Garanti que la disposition reste cohérente, même en cas de redimensionnement de la fenêtre
- La méthode **setLayout** est utilisée sur un conteneur pour définir un gestionnaire
- Les plus courant sont
 - **BoderLayout**
 - **FlowLayout**
 - **GridLayout**

V. Interfaces graphiques

4) Gestionnaires de mise en forme

BorderLayout

- Layout par défaut des **JFrame**
- Dispose les éléments selon les 4 points cardinaux
- Préciser l'emplacement au moment de l'ajout

Exemple:

```
JPanel p = new JPanel();  
p.setLayout(new BorderLayout());  
p.add(new JButton(« b1»), BorderLayout.NORTH);
```



V. Interfaces graphiques

4) Gestionnaires de mise en forme

FlowLayout

- Layout par défaut des **JPanel**
- Dispose les éléments les uns à la suite des autres sur une même ligne
- Si la ligne n'a pas plus de place, les éléments sont disposés sur la ligne suivant
- Fonctionne come un éditeur de texte
- A la construction, on peut préciser si les éléments sont ajoutés
 - De gauche à droite (choix par défaut), de droite à gauche, ou à partir du milieu

Exemple:

```
JPanel p = new JPanel();  
p.setLayout(new FlowLayout());  
p.add(new JButton("b1"));
```



V. Interfaces graphiques

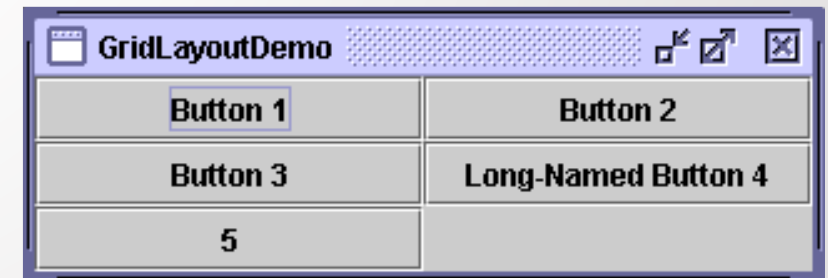
4) Gestionnaires de mise en forme

GridLayout

- Dispose les éléments dans un tableau bidimensionnel
- Les deux dimensions doivent être définies à la création
- Chaque composant occupe une cellule
- Taille uniformément répartie pour tous les composants
- Composant sont ajoutés au tableau dans l'ordre des appels

Exemple:

```
JPanel p = new JPanel();  
p.setLayout(new GridLayout(3, 2));  
p.add(new JButton(« Boutton 2»));  
p.add(new JButton(« Boutton 2»));
```



V. Interfaces graphiques

4) Gestionnaires de mise en forme

En général

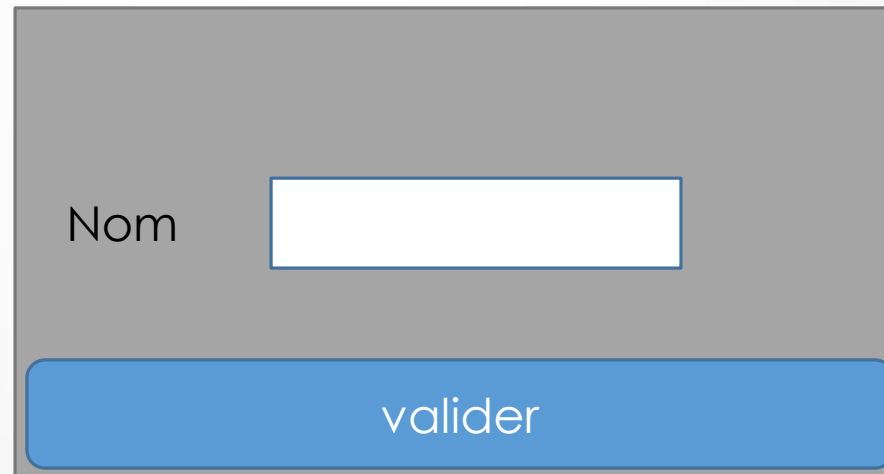
- **FlowLayout** et **BorderLayout** permettent de mettre en place des interfaces complexes
- Mais selon les besoins:
 - BorderLayout
 - GridBagLayout
 - SpringLayout
 - ...

V. Interfaces graphiques

4) Gestionnaires de mise en forme

Exercice

1. Modifier le programme précédent pour qu'il affiche les composants comme suit



The diagram shows a rectangular window with a gray background. On the left side, the text "Nom" is displayed. To its right is a white rectangular text input field with a thin blue border. Below the input field is a blue button with rounded corners and the text "valider" centered on it.

V. Interfaces Graphiques

5. Gestion des évènements

V. Interfaces graphiques

5) Gestion des événements

Évènement

- ▶ Action de l'utilisateur sur les composants (pression d'une touche, clic sur bouton, déplacement de la souris, etc.)

Gestion des événements

- ▶ Stratégies pour traiter les événements
- ▶ Basée sur le **programmation événementielle (POE)**
- ▶ Modèle émetteur/récepteur
 1. Un composant déclenche un événement, représenté sous la forme d'un objet
 2. Un (ou plusieurs) objet(s) écouteurs détecte l'évènement
 3. Mette(ent) en place des actions
- ▶ L'écouteur doit au préalable s'enregistrer auprès du composant

V. Interfaces graphiques

5) Gestion des évènements

Différents types d'évènements

- Liés à la souris, au clavier, etc.
- Deux familles
 - Evènements de haut niveau (**Action**)
 - Gérer une action sur un composant sans se soucier des détails
 - Ex. Un clique sur un bouton
 - Interface: **ActionListener**
 - Evènements de bas niveau (**Listeners**)
 - Gérer plus finement les interactions de l'utilisateur
 - Ex. bouton enfoncé mais pas relâché, souris en mouvement, etc.
 - Chaque évènement **XXX** à son Interface dédiée **XXXListeners**
 - Interface: **MouseListener, MouseMotionListener, KeyListener, FocusListener, etc.**

V. Interfaces graphiques

5) Gestion des événements

S'enregistrer auprès d'un composant

- ▶ Tous les composants sont des sources d'évènements possibles
- ▶ Chaque composant possède des méthodes permettant à un objet écouteur de s'enregistrer
 - ▶ `addXXXListener`
- ▶ Exemple pour un bouton

```
JButton b = new JButton(« Valider »);  
b.addActionListener(new monEcouteur());  
b.addMouseListener(new monEcouteur());
```
- ▶ En fonction du type d'évènement des fonctions doivent être implémentées pour traiter les interactions sur les composants

V. Interfaces graphiques

5) Gestion des évènements

Exemple avec la classe qui gère elle-même les évènements

Fenetre.java

```
Public class Fenetre extends JFrame implements ActionListener{  
    Jbutton b1;  
    Public Fenetre(){  
        setSize(800, 600);  
        setTitle(« test »);  
        JPanel p = new JPanel()  
        Container contenu = this.getContentPane();  
        contenu.add(p);  
  
        b1 = new JButton(« Valider »);  
        p.add(b1);  
        b1.addaActionListener(this);  
        setVisible(true);  
    }  
  
    public void actionPerformed(ActionEvent e){  
        System.out.println(« Action sur le bouton »);  
    }  
}
```

V. Interfaces graphiques

5) Gestion des événements

Exemple avec écouteur dédié

Fenetre.java

```
Public class Fenetre extends JFrame {  
    Jbutton b1;  
    Public Fenetre(){  
        setSize(800, 600);  
        setTitle(« test »);  
        JPanel p = new JPanel()  
        Container contenu = this.getContentPane();  
        contenu.add(p);  
  
        b1 = new JButton(« Valider »);  
        p.add(b1);  
        b1.addaActionListener(new MonEcouleur());  
  
        setVisible(true);  
    }  
}
```

monEcouleur.java

```
public class monEcouleur implements ActionListener{  
    public void actionPerformed(ActionEvent e){  
        System.out.println(« Action sur le bouton »);  
    }  
}
```

V. Interfaces graphiques

5) Gestion des évènements

Exemple avec écouteur qui gère plusieurs composants ???

Fenetre.java

```
Public class Fenetre extends JFrame {  
    JButton b1, b2;  
    Public Fenetre(){  
        setSize(800, 600);  
        setTitle(« test »);  
        JPanel p = new JPanel()  
        Container contenu = this.getContentPane();  
        contenu.add(p);  
  
        b1 = new JButton(« Valider »);  
        p.add(b1);  
        b1.addaActionListener(...  
  
        b2 = new JButton(« Annuler »);  
        p.add(b2);  
        b1.addaActionListener(...  
  
        setVisible(true);  
    }  
}
```

V. Interfaces graphiques

5) Gestion des évènements

Exemple avec écouteur qui gère plusieurs composants ???

Fenetre.java

```
Public class Fenetre extends JFrame implements ActionListener{  
    JButton b1, b2;  
    Public Fenetre(){  
        setSize(800, 600);  
        setTitle(« test »);  
        JPanel p = new JPanel()  
        Container contenu = this.getContentPane();  
        contenu.add(p);  
  
        b1 = new JButton(« Valider »);  
        p.add(b1);  
        b1.addaActionListener(this);  
  
        b2 = new JButton(« Annuler »);  
        p.add(b2);  
        b1.addaActionListener(this);  
  
        setVisible(true);  
    }  
}
```

```
public void actionPerformed(ActionEvent e){  
    if(e.getSource() == b1){  
        System.out.println(« Appui sur Valider »);  
    }  
    else if(e.getSource() == b2){  
        System.out.println(« Appui sur Annuler »);  
    }  
}
```

V. Interfaces graphiques

5) Gestion des évènements

Exemple avec écouteur qui gère plusieurs composants ???

Fenetre.java	MonEcouteur.java
<pre>Public class Fenetre extends JFrame { JButton b1, b2; Public Fenetre(){ setSize(800, 600); setTitle(« test »); JPanel p = new JPanel() Container contenu = this.getContentPane(); contenu.add(p); b1 = new JButton(« Valider »); p.add(b1); b1.addaActionListener(new MonEcouteur(1)); b2 = new JButton(« Annuler »); p.add(b2); b1.addaActionListener(new MonEcouteur(2)); setVisible(true); } }</pre>	<pre>public class monEcouteur implements ActionListener{ Private int action; public monEcouteur(int id){ action = id; } public void actionPerformed(ActionEvent e){ if(action == 1){ System.out.println(« Appui sur Valider »); } else if(action == 2){ System.out.println(« Appui sur Annuler »); } } }</pre>

V. Interfaces graphiques

5) Gestion des évènements

Evènement de bas niveau: `KeyListener`

Modifier and Type	Method and Description
void	<code>keyPressed(KeyEvent e)</code> Invoked when a key has been pressed.
void	<code>keyReleased(KeyEvent e)</code> Invoked when a key has been released.
void	<code>keyTyped(KeyEvent e)</code> Invoked when a key has been typed.

- ▶ L'objet `KeyEvent` propose de nombreuses méthodes
 - ▶ `getKeyCode()`
 - ▶ `getKeyChar()`
 - ▶ ...

V. Interfaces graphiques

5) Gestion des évènements

Evènement de bas niveau: `MouseListener`

Modifier and Type	Method and Description
void	<code>mouseClicked(MouseEvent e)</code> Invoked when the mouse button has been clicked (pressed and released) on a component.
void	<code>mouseEntered(MouseEvent e)</code> Invoked when the mouse enters a component.
void	<code>mouseExited(MouseEvent e)</code> Invoked when the mouse exits a component.
void	<code>mousePressed(MouseEvent e)</code> Invoked when a mouse button has been pressed on a component.
void	<code>mouseReleased(MouseEvent e)</code> Invoked when a mouse button has been released on a component.

L'objet `MouseEvent` propose de nombreuses méthodes

- `getPoint()`
- `getX()`
- `getY()`
- ...

V. Interfaces graphiques

5) Gestion des évènements

Exemple d'écouteurs bas niveau géré par la classe elle-meme

Fenetre.java

```
Public class Fenetre extends Frame implements KeyListener {
    JButton b1, b2;
    Public Fenetre(){
        setSize(800, 600);
        setTitle(« test »);
        JPanel p = new JPanel()
        Container contenu = this.getContentPane();
        contenu.add(p);
        p.addKeyListener(this);

        b1 = new JButton(« Valider »);
        p.add(b1);
        b1.addaActionListener(this);

        b2 = new JButton(« Annuler »);
        p.add(b2);
        b1.addaActionListener(this);

        setVisible(true);
    }
}
```

```
public void keyPressed (KeyEvent e){
    System.out.println(« une touche est enfoncée » +
e. getKeyCode();)
}

public void keyReleased (KeyEvent e){
    System.out.println(« une touche est relachée » +
e. getKeyCode();)
}

public void keyTyped (KeyEvent e){
    System.out.println(« la touche est enfoncée puis
relachée » + e. getKeyCode();)
}
}
```

V. Interfaces graphiques

5) Gestion des évènements

Exemple d'écouteurs bas niveau dédié

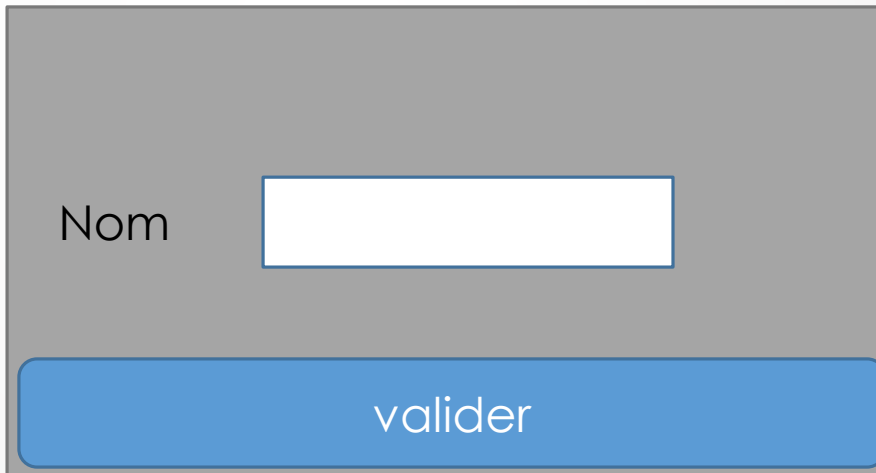
Fenetre.java	MonEcouteur.java
<pre>Public class Fenetre extends Frame { JButton b1, b2; Public Fenetre(){ setSize(800, 600); setTitle(« test »); JPanel p = new JPanel() Container contenu = this.getContentPane(); contenu.add(p); p.addKeyListener(new MonEcouteurClavier()); b1 = new JButton(« Valider »); p.add(b1); b1.addaActionListener(this); b2 = new JButton(« Annuler »); p.add(b2); b1.addaActionListener(this); setVisible(true); } }</pre>	<pre>public monEcouteurClavier implements KeyListener public void keyPressed (KeyEvent e){ System.out.println(« une touche est enfoncée » + e.getKeyCode();) } public void keyReleased (KeyEvent e){ System.out.println(« une touche est relachée » + e.getKeyCode();) } public void keyTyped (KeyEvent e){ System.out.println(« la touche est enfoncée puis relachée » + e.getKeyCode();) } }</pre>

V. Interfaces graphiques

5) Gestion des évènements

Exercice

1. Modifier le programme précédent pour qu'il affiche dans la console « bonjour » suivi du nom saisi par l'utilisateur, après un clique sur le bouton



The image shows a graphical user interface (GUI) window with a gray background. On the left side, the text "Nom" is displayed. To its right is a white rectangular text input field with a thin blue border. Below the input field is a blue button with rounded corners and the text "valider" written in white.

V. Interfaces Graphiques

6. Dessins dans un composant

V. Interfaces graphiques

6) Dessins dans un composant

Dessiner dans un composant

- Tous les composants possède une méthode **paintComponent(Graphics G)** qui leur est propre
 - **Graphics G** correspond au contexte graphique
 - Par exemple la méthode **PaintComponent** de la classe **JButton** permet d'afficher un bouton dans conteneur
- Redéfinir la méthode pour avoir un affichage particulier
- L'objet **Graphics** offre de nombreuses méthodes de dessin
 - `setColor(Color c)`
 - `drawLine (int x1, int y1, int x2, int y2)`
 - `drawOval (int x, int y, int width, int height)`
 - `drawRect (int x, int y, int width, int height)`
 - ...
- Toutes les fonctions de dessin disponibles ici: <https://goo.gl/3d17yc>

V. Interfaces graphiques

6) Dessins dans un composant

Exemple du dessin dans un JPanel

```
Public monPanel extends JPanel {  
    public void paintComponent(Graphics g){  
        super.paintComponent(g);  
        graphics.setColor(Color.RED);  
        graphics.fillOval(0, 0, 30, 30);  
        graphics.drawOval(0, 50, 30, 30);  
        graphics.fillRect(50, 0, 30, 30);  
        graphics.drawRect(50, 50, 30, 30);  
    }  
}
```

