listeners (Java)

« event listeners » (Java)

variante des callebacks adaptée au Java:

methods de type AddListener spécifient non pas une fonction de callback, mais un objet (le *listener*)

lorsque le widget change d'état, il déclenche une méthode prédéfinie du *listener*

(par exemple *actionPerformed*)

« event listeners » (Java)

variante des callebacks adaptée au Java:

methods de type AddListener spécifient non pas une fonction de callback, mais un objet (le *listener*)

lorsque le widget change d'état, il déclenche une méthode prédéfinie du *listener*

(par exemple *actionPerformed*)

« event listeners » (Java)

Un composant (widget) qui crée des événements est appelé source

Le source délègue le traitement de l'événement au listener

Un listener doit s'inscrire auprès du composant source des événements qu'il veut traiter.

Un événement peut provenir :

- du clavier, un clique souris, un passage de la souris,..

A chaque type d'événement, une classe (existante)

A chaque type d'événement, son listener (à faire)

« event listeners » (Java)

```
public class ClickListener implements ActionListener {
 public void actionPerformed(ActionEvent e){
     JButton button = (JButton)e.getSource();
     …
 }
}

…
ClickListener listener = new ClickListener();
JButton button = new JButton(''Click me'');
button.addActionListener(listener);
…
```

« event listeners » (Java)

Anonymous Inner classes

“new <nom-de-classe> () { <corps> }”

cette construction fait deux choses :

- elle crée une nouvelle classe, sans nom, qui est une sous-classe de <nom-de-classe> définie par <corps>
- elle crée une instance (unique) de cette nouvelle classe et retourne sa valeur

cette class a accès aux variables et méthodes de la class dans la quelle elle est définie

« event listeners » (Java)

Anonymous Inner classes

```
…
button.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e){
           …
      }
});
…
panel.addMouseListener(new MouseAdapter(){
      public void mouseClicked(MouseEvent e){
           …
      }
});
```

Fonctions et évènements prédéfinis

« event listeners » (Java)

```
public class ClickListener implements ActionListener {
 public void actionPerformed(ActionEvent e){
     JButton button = (JButton)e.getSource();
     …
 }
}

…
ClickListener listener = new ClickListener();
JButton button = new JButton(''Click me'');
button.addActionListener(listener);
…
```

« event listeners » (Java)

Anonymous Inner classes

“new <nom-de-classe> () { <corps> }”

cette construction fait deux choses :

- elle crée une nouvelle classe, sans nom, qui est une sous-classe de <nom-de-classe> définie par <corps>
- elle crée une instance (unique) de cette nouvelle classe et retourne sa valeur

cette class a accès aux variables et méthodes de la class dans la quelle elle est définie

« event listeners » (Java)

Anonymous Inner classes

```
…
button.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e){
           …
      }
});
…
panel.addMouseListener(new MouseAdapter(){
      public void mouseClicked(MouseEvent e){
           …
      }
});
```

Fonctions et évènements prédéfinis