D[i] Département Informatique

UNIVERSITÉ TOULOUSE III PAUL SABATIER
Université de Toulouse
Faculté Sciences et Ingénierie

Programmation Orientée Objet

Librairie Java Swing

Gestion des événements

Mathieu RAYNAL

mathieu.raynal@irit.fr

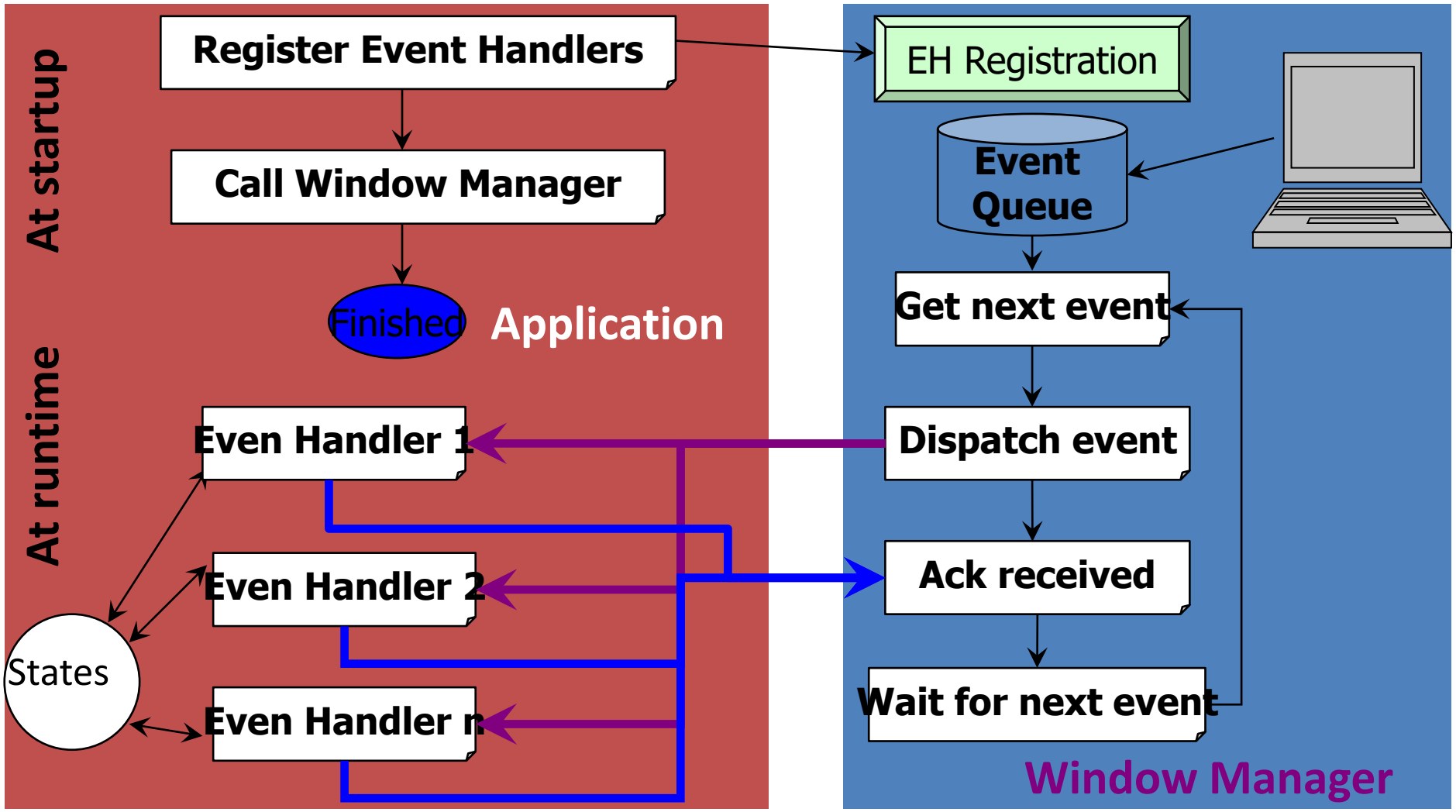
http://www.irit.fr/~Mathieu.Raynal

Fonctionnement par événements

- Evénement = Action extérieure au programme se produisant sur l'interface
  - Généralement effectué par l'utilisateur

- Les sources
  - Matériel
    - Clavier,
    - Souris,
    - Ecran tactile,
    - …
  - Logiciel (composants graphiques)
    - Fenêtre,
    - Bouton,
    - Liste,
    - …

Gestion des événements



Les événements

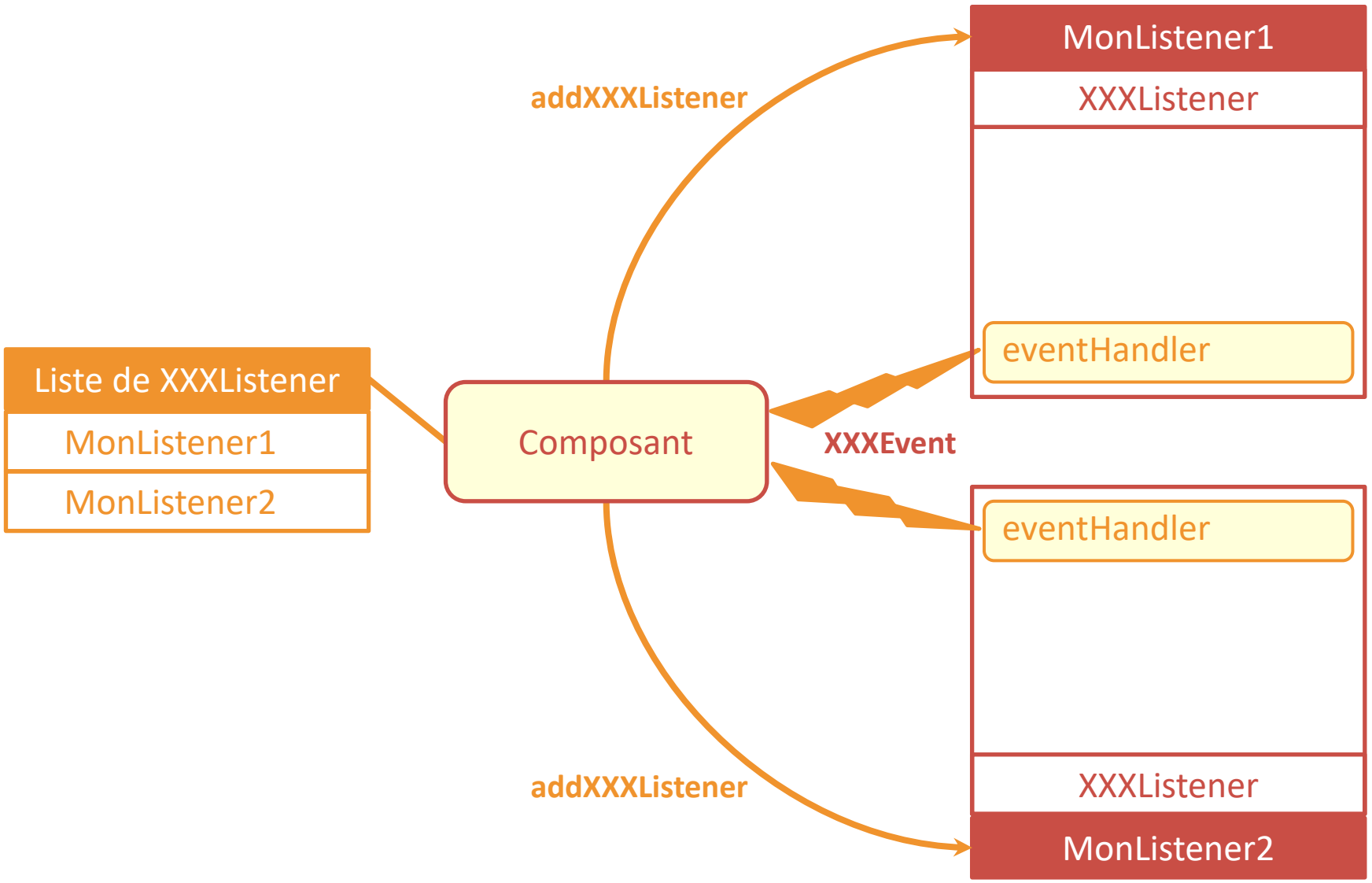
- Deux niveaux d'événements
  - Bas niveau
    - Déplacer la souris
    - Appuyer sur un bouton d'un dispositif
    - Taper sur une touche du clavier
  - Sémantique
    - Appuyer sur un bouton de l'interface
    - Prendre ou perdre le focus
    - Entrer ou sortir d'un composant

Les événements

- Les événements sont caractérisés par des classes
  - Chaque type d'événement a une classe spécifique
    - Clic bouton : **ActionEvent**
    - Evénements liés à la souris : **MouseEvent**
    - Evénements liés au clavier : **KeyEvent**
  - Cette classe contient les caractéristiques de l'événement
    - Position, moment, …

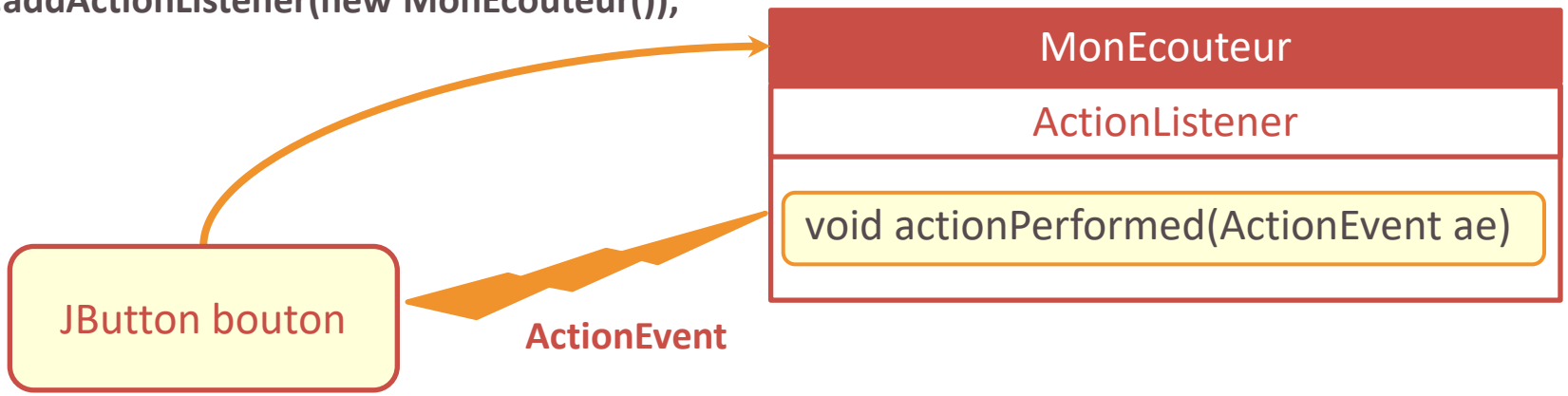
- Les événements sont gérés par les composants graphiques
  - Un composant peut gérer plusieurs types d'événements
  - Un type d'événement peut se produire sur différents types de composants

Gestion des événements



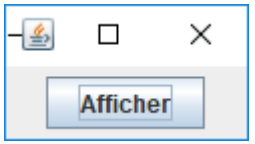
Cas du bouton

bouton.addActionListener(new MonEcouteur());



Exemple 1 – gestion d'un bouton

- Afficher Bonjour dans la console suite à un appui sur le bouton

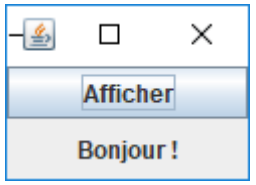



```
6  public class MonEcouteur implements ActionListener{
7      @Override
8      public void actionPerformed(ActionEvent e) {
9          System.out.println("Bonjour !");
10     }
11 }
```

```
17         bouton = new JButton("Afficher");
18         bouton.addActionListener(new MonEcouteur());
```

Exemple 2 – gestion d'un bouton

- Afficher Bonjour dans un label sur l'interface suite à un appui sur le bouton

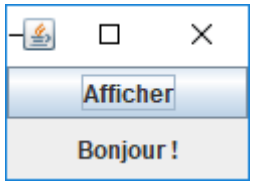



```
14 public class ExempleEvenementBouton1 implements ActionListener{
15    JLabel label;
16    JButton bouton;
17
18    ExempleEvenementBouton1() {
19       JFrame frame = new JFrame("Gestion événement bouton 1");
20       frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
21
22       label = new JLabel("",JLabel.CENTER);
23
24       bouton = new JButton("Afficher");
25       bouton.addActionListener(this);
```

```
45    @Override
46    public void actionPerformed(ActionEvent e) {
47       label.setText("Bonjour !");
48    }
```

Exemple 3 – gestion d'un bouton

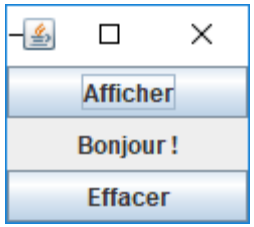
- Afficher Bonjour dans un label sur l'interface et dans la console suite à un appui sur le bouton




```
24      bouton = new JButton("Afficher");
25      bouton.addActionListener(this);
26      bouton.addActionListener(new MonEcouteur());
```

Exemple 4 – gestion d'un bouton

- 2 boutons
  - Afficher Bonjour dans un label sur l'interface suite à un appui sur le bouton « Afficher »
  - Effacer le contenu du label suite à un appui sur le bouton « Effacer »

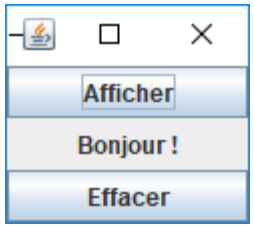



```
22      boutonAfficher = new JButton("Afficher");
23      boutonAfficher.addActionListener(this);
24
25      boutonEffacer = new JButton("Effacer");
26      boutonEffacer.addActionListener(this);
```

```
47    public void actionPerformed(ActionEvent e) {
48       if(e.getActionCommand().equals("Afficher"))
49          label.setText("Bonjour !");
50
51       if(e.getActionCommand().equals("Effacer"))
52          label.setText("");
53    }
```

Exemple 4 – gestion d'un bouton

- 2 boutons
  - Afficher Bonjour dans un label sur l'interface suite à un appui sur le bouton « Afficher »
  - Effacer le contenu du label suite à un appui sur le bouton « Effacer »

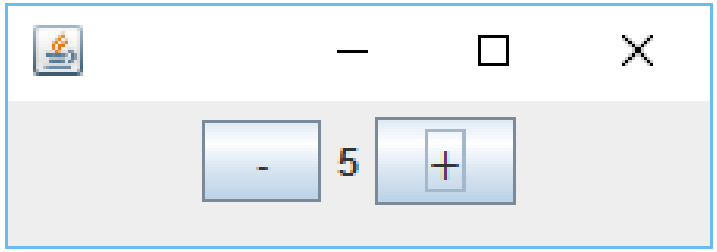



```
22      boutonAfficher = new JButton("Afficher");
23      boutonAfficher.addActionListener(new ActionListener() {
24          @Override
25          public void actionPerformed(ActionEvent arg0) {
26              afficher();
27          }
28      });
```

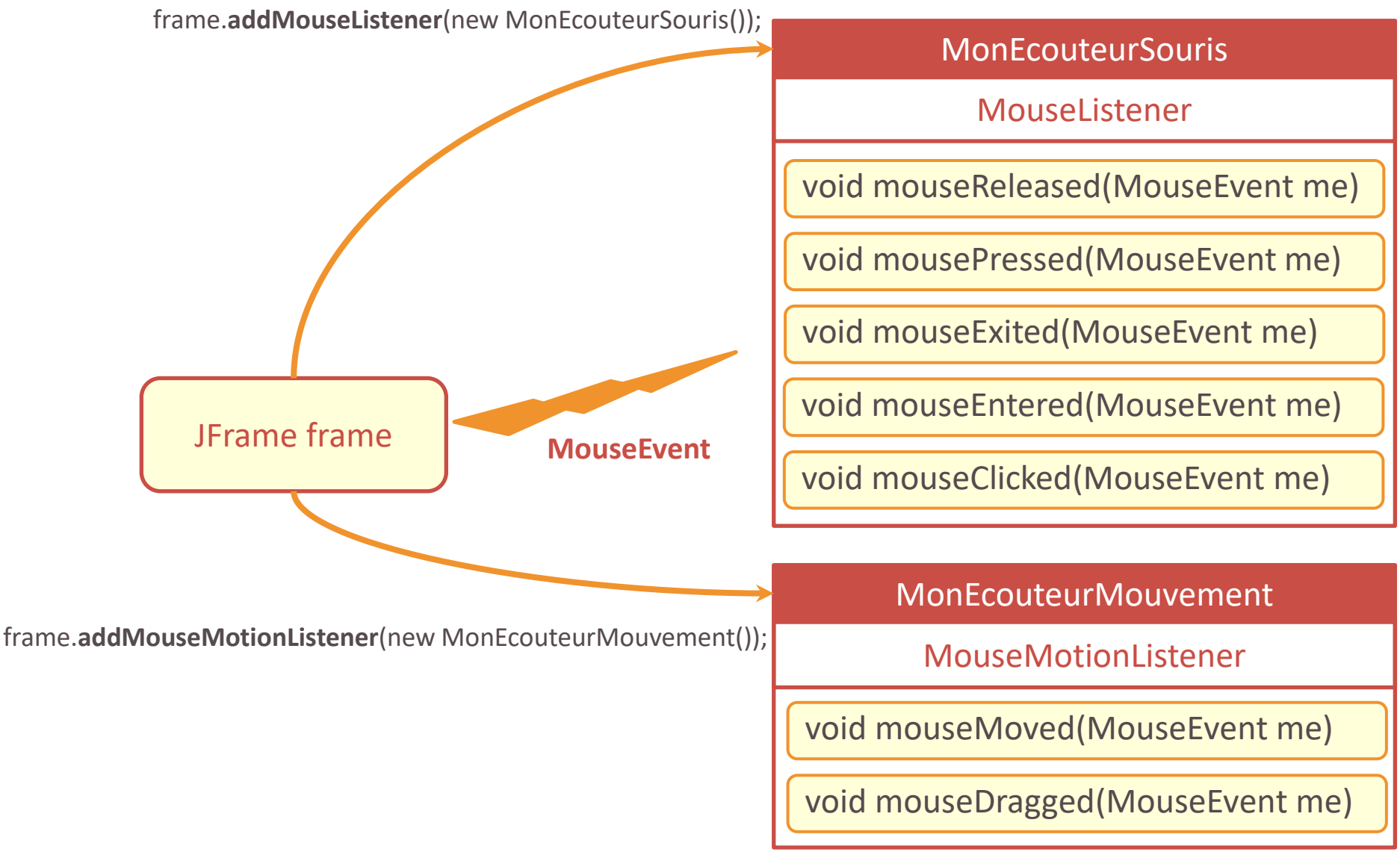
```
51      public void afficher()
52      {
53          label.setText("Bonjour !");
54      }
```

Exercice

- Les boutons + et – permettent d'incrémenter et décrémenter le compteur



Cas de la souris



Listener versus Adapter

- **Listener** : Interface
  - Obligation d'avoir toutes les méthodes de l'interface !
- **Adapter** : Classe abstraite qui implémente le Listener
  - On ne masque que les méthodes qui nous intéressent

  → Attention de masquer correctement la méthode

Exemple – gestion de la souris

- Afficher les coordonnées du pointeur au moment où l'on presse un bouton de la souris

```
25      frame.addMouseListener(new MouseAdapter() {
26          @Override
27          public void mousePressed(MouseEvent e) {
28              System.out.println("Pression du bouton de la souris en ("+e.getX()+", "+e.getY()+")");
29          }
30      });
```

En résumé - Gestion des événements

- Lorsqu'un événement se produit sur un **composant**
  - Création d'une instance de l'événement
  - Envoi à tous ceux qui sont intéressés

- Qui est « intéressé » par un événement **XXXEvent** ?
  - Les écouteurs (**XXXListener**)
    - Spécifiques à chaque type d'événement
  - Les écouteurs doivent se faire connaitre auprès du composant
    - **addXXXListener**(XXXListener)