CSCI 2113 Lab 10

Bo Mei

Event Handling

- Difficulty: We don’t know when an event will be triggered.
- Solution: Separate the logic of the code that processes the event from the main logic of the program.
- This solution is the core idea behind the *delegation event model*.

- Java event handling uses delegation event model.
- Definition of the model: A source generates an event and sends it to one or more listeners.
- Three roles/pieces of code: Sources, Listeners, and Events.

Event Sources

- An object that generates an event.
- The class of the object (generally already) provides methods that can add and remove listeners.
- Example 1: Component (A superclass of JPanel)
  - addKeyListener(KeyListener l)
  - addMouseListener(MouseListener l)
  - addMouseMotionListener(MouseMotionListener l)
  - addMouseWheelListener(MouseWheelListener l)
  - …
- Example 2: AbstractButton (A superclass of JButton)
  - addActionListener(ActionListener l)
  - …

Event Listeners

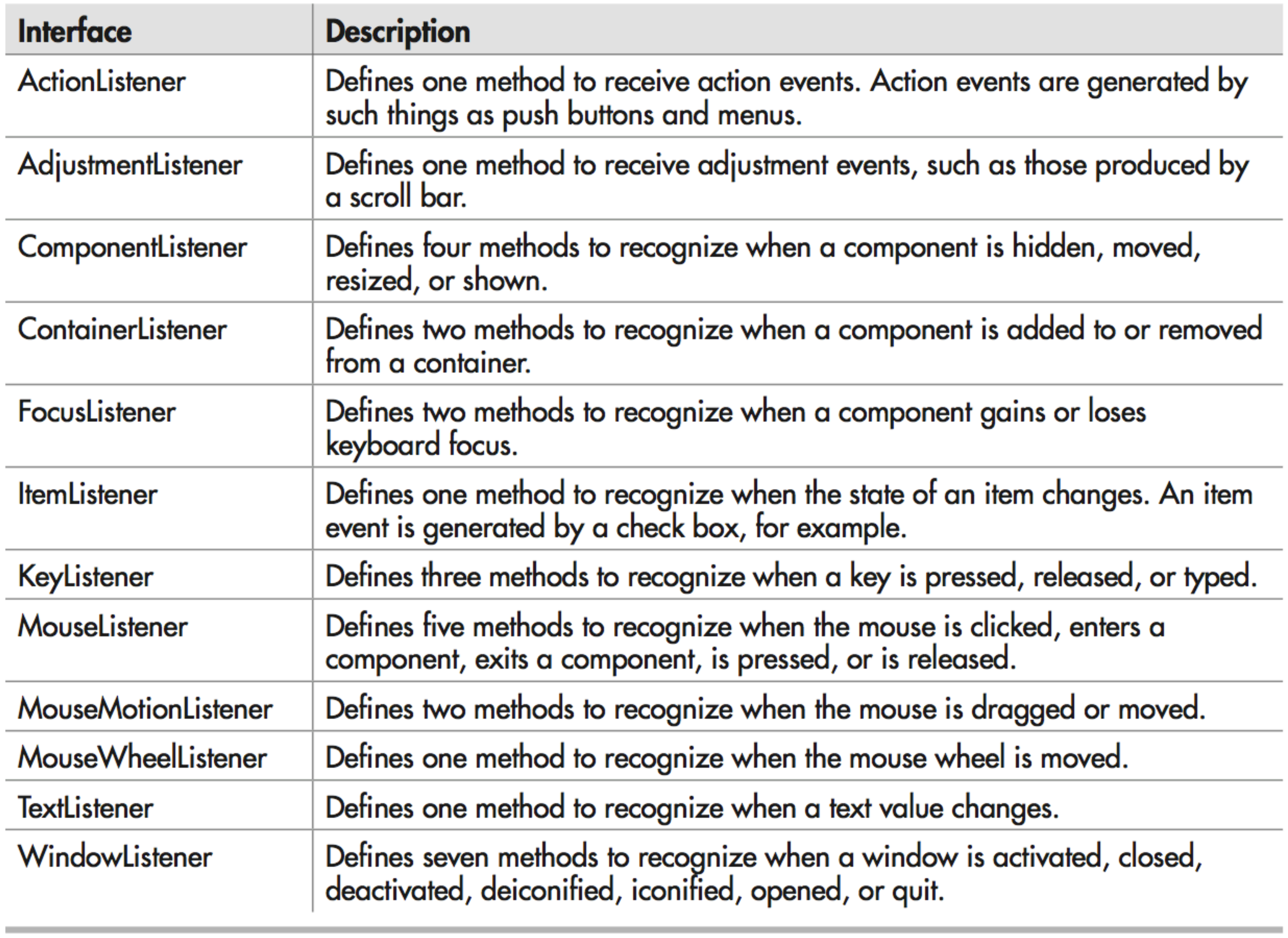
- An object that is notified when an event occurs. It will then process the event.
- The class (probably created by yourself) of the object implements certain Java-defined listener interfaces.

- Example:

```
public class AnyClass implements KeyListener {
        …
        public void keyPressed(KeyEvent e) {…}
        public void keyReleased(KeyEvent e) {…}
        public void keyTyped(KeyEvent e) {…}
        …
}
```

Event Listeners

- There are several listener interfaces.
- If a class implements certain interfaces, it must implements all the methods defined by the interfaces. It’s common that the contents of some methods are empty.



Interface	Description
ActionListener	Defines one method to receive action events. Action events are generated by such things as push buttons and menus.
AdjustmentListener	Defines one method to receive adjustment events, such as those produced by a scroll bar.
ComponentListener	Defines four methods to recognize when a component is hidden, moved, resized, or shown.
ContainerListener	Defines two methods to recognize when a component is added to or removed from a container.
FocusListener	Defines two methods to recognize when a component gains or loses keyboard focus.
ItemListener	Defines one method to recognize when the state of an item changes. An item event is generated by a check box, for example.
KeyListener	Defines three methods to recognize when a key is pressed, released, or typed.
MouseListener	Defines five methods to recognize when the mouse is clicked, enters a component, exits a component, is pressed, or is released.
MouseMotionListener	Defines two methods to recognize when the mouse is dragged or moved.
MouseWheelListener	Defines one method to recognize when the mouse wheel is moved.
TextListener	Defines one method to recognize when a text value changes.
WindowListener	Defines seven methods to recognize when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

Events

- Event classes are well-defined by Java already.
- The root class of all the Java event classes is EventObject, whose superclass is still Object.
- Different event classes contain different useful methods.
- Example: MouseEvent
    - int getX()
    - int getY()
    - int getXOnScreen()
    - int getYOnScreen()
    - …

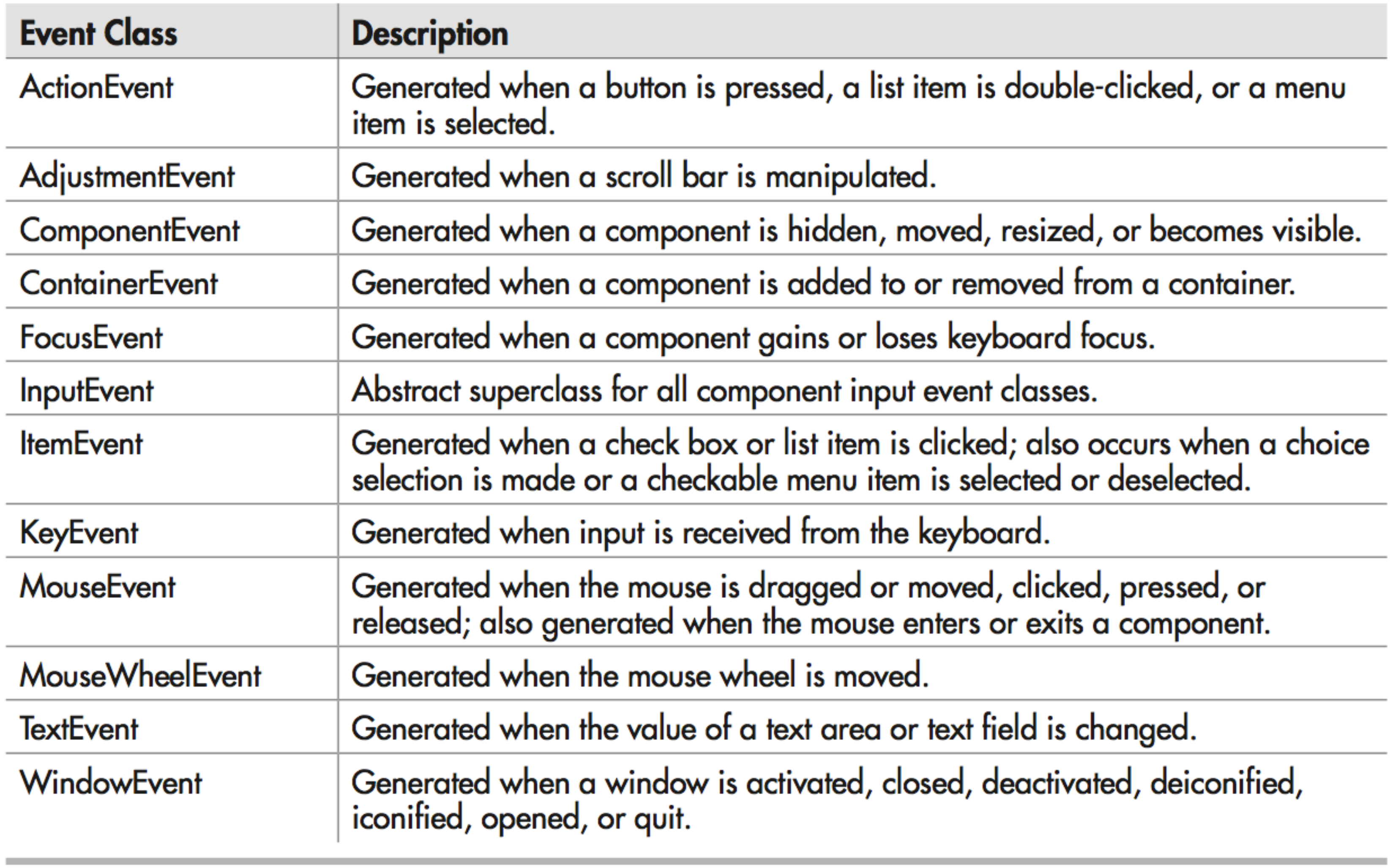
Events

- Example: How to use MouseEvent

```
public class AnyClass implements MouseListener {
	public void mouseClicked(MouseEvent e) {
		System.out.println(“Mouse clicked at (” + e.getX() + “, ” + e.getY() + “)”);
	}
	public void mouseEntered(MouseEvent e) {}
	public void mouseExited(MouseEvent e) {}
	public void mousePressed(MouseEvent e) {}
	public void mouseReleased(MouseEvent e) {}
}
```

Events

- There are several event classes.



Event Class	Description
ActionEvent	Generated when a button is pressed, a list item is double-clicked, or a menu item is selected.
AdjustmentEvent	Generated when a scroll bar is manipulated.
ComponentEvent	Generated when a component is hidden, moved, resized, or becomes visible.
ContainerEvent	Generated when a component is added to or removed from a container.
FocusEvent	Generated when a component gains or loses keyboard focus.
InputEvent	Abstract superclass for all component input event classes.
ItemEvent	Generated when a check box or list item is clicked; also occurs when a choice selection is made or a checkable menu item is selected or deselected.
KeyEvent	Generated when input is received from the keyboard.
MouseEvent	Generated when the mouse is dragged or moved, clicked, pressed, or released; also generated when the mouse enters or exits a component.
MouseWheelEvent	Generated when the mouse wheel is moved.
TextEvent	Generated when the value of a text area or text field is changed.
WindowEvent	Generated when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

DotDrawer Example

- Clone using IntelliJ: https://github.com/cs2113f16/lec-10-guis.git
- DotDrawer class deals with the main logic as well as acts as event listeners.
- *button*.addActionListener(*this*);
  - *button*, which is an instance of JButton class, is an event source. *button* calls addActionListener method because JButton provides the method.
  - We can use *this* because DotDrawer implements ActionListener interface.

DotDrawer Example

- *dp*.addMouseListener(*this*);
  - *dp*, which is an instance of DotPanel class, is an event source. *dp* calls addMouseListener method because DotPanel is a subclass of JPanel, which provides the method.
  - We can use *this* because DotDrawer implements MouseListener interface.
- Since DotDrawer implements ActionListener and MouseListener interfaces, DotDrawer must implements all the methods defined by both interfaces. Notice that the contents of some methods are empty.

Summary — Basic Steps

1. sourceObject.addXxxListener(*AnyClass object that implements XxxListener interface*)
2. public class AnyClass implements XxxListener {…}
3. Within AnyClass, complete the methods that are required by XxxListener.