CS205 Object Oriented Programming in Java

Module 4 - **Advanced features of Java** (Part 7)

Prepared by

Renetha J.B.

AP

Dept.of CSE,

Lourdes Matha College of Science and Technology

Topics

☑ **Event handling:**

- ❑ Event Classes

Event Classes

- The classes that represent events(Event classes) are at the core of Java's event handling mechanism.
- The most widely used events are those defined by the AWT and those defined by Swing.
- At the **root of the Java event class hierarchy** is **EventObject**, which is in **java.util.**
  - **EventObject is the** superclass for all events.
  - Its one constructor is :

    EventObject(Object *src)*

  - Here, *src is the object that generates this event.*

Event Classes(contd.)

- **EventObject** contains two methods:

  getSource( )

  toString( ).

- The getSource( ) method returns the source of the event.

  Its general form is :

  ```
  Object getSource( )
  ```

- **toString( )** returns the string equivalent of the event.

Event Classes(contd.)

- The class **AWTEvent**, defined within the **java.awt** package, is a subclass of EventObject.
  - It is the **superclass** (either directly or indirectly) of all AWT-based events used by the delegation event model.
  - Its **getID( )** method can be used to determine the type of the event.
  - The signature of **getID( )** method is

```
int getID( )
```

Event Classes(contd.)

- **EventObject** is a **superclass** of **all events.**
- **AWTEvent is a superclass of all AWT events that are handled by the delegation event model.**

Event Class (contd.)

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, 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.

Event classes

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.

Event classes(contd.)

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, 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**.

The ActionEvent Class

- An **ActionEvent** is generated when
  - a button is pressed,
  - a list item is double-clicked,
  - menu item is selected.
- The **ActionEvent** class defines four integer constants that can be used to identify any modifiers associated with an action event:
  - **ALT_MASK**
  - **CTRL_MASK**
  - **META_MASK**
  - **SHIFT_MASK**.
- Integer constant **ACTION_ PERFORMED,** can be used to identify action events.

The ActionEvent Class(contd.)

- **ActionEvent has these three constructors:**

ActionEvent(Object *src, int type, String cmd)*

ActionEvent(Object *src, int type, String cmd, int modifiers)*

ActionEvent(Object *src, int type, String cmd, long when,*

int modifiers)

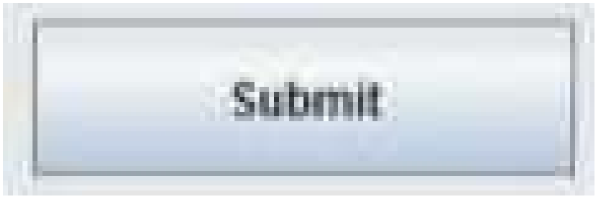
- Here, *src* is a reference to the object that generated this event.
- The type of the event is specified by *type*, and its command string is *cmd*.
- The argument *modifiers* indicates which modifier keys (ALT, CTRL, META, and/or SHIFT) were pressed when the event was generated.
- The *when* parameter specifies when the event occurred.

The ActionEvent Class(contd.)

- To obtain the **command name** for the invoking **ActionEvent** object **getActionCommand( )** method can used**:**

String **getActionCommand( )**

- For example, when a button is pressed, an action event is generated that has a command name equal to the **label on that button**.
- E.g.



The command name of button is **Submit.**

The ActionEvent Class(contd.)

- The **getModifiers( )** method
  - returns a value that indicates **which modifier keys** (ALT, CTRL, META, and/or SHIFT) **were pressed** when the event was generated. Its form is :

```
int getModifiers( )
```

- The method **getWhen( )**
  - returns the time at which the event took place. This is called the **event's *timestamp***. *The getWhen( ) method is :*

```
long getWhen( )
```

The AdjustmentEvent Class

- An AdjustmentEvent is generated by a **scroll bar**.
- There are five types of adjustment events.
- The AdjustmentEvent class defines **integer constants** that can be used to identify them.

BLOCK_DECREMENT	• The user clicked inside the scroll bar to decrease its value.
BLOCK_INCREMENT	• The user clicked inside the scroll bar to increase its value.
TRACK	• The slider was dragged.
UNIT_DECREMENT	• The button at the end of the scroll bar was clicked to decrease its value.
UNIT_INCREMENT	• The button at the end of the scroll bar was clicked to increase its value

The AdjustmentEvent Class(contd.)

- An integer constant, **ADJUSTMENT_VALUE_CHANGED,** that indicates that a change has occurred.
- One **AdjustmentEvent constructor:**

AdjustmentEvent(Adjustable *src, int id, int type, int data)*

  - Here, *src* is a reference to the object that generated this event.
  - The *id* specifies the event.
  - The type of the adjustment is specified by *type*, and its associated data is *data*.

The AdjustmentEvent Class(contd.)

- The **getAdjustable( )** method returns the object that generated the event. Its form is**;**

```
Adjustable getAdjustable( )
```

- The type of the adjustment event may be obtained by the **getAdjustmentType( )** method. It returns one of the constants defined by **AdjustmentEvent.** The general form is :

```
int getAdjustmentType( )
```

- The amount of the adjustment can be obtained from the **getValue( )** method is**:**

```
int getValue( )
```

  – For example, when a scroll bar is manipulated, this method returns the value represented by that change.

The ComponentEvent Class

- A ComponentEvent is generated when the size, position, or visibility of a component is changed.
- There are four types of component events.
  - The ComponentEvent class defines integer constants for this.

COMPONENT_HIDDEN	• The component was hidden.
COMPONENT_MOVED	• The component was moved.
COMPONENT_RESIZED	• The component was resized.
COMPONENT_SHOWN	• The component became visible.

The ComponentEvent Class(contd.)

- **ComponentEvent has the constructor:**

```
ComponentEvent(Component src, int type)
```

  - Here, *src* is a reference to the object that generated this event. The type of the event is specified by *type*.

- **ComponentEvent** is the **superclass** either directly or indirectly of *ContainerEvent, FocusEvent, KeyEvent, MouseEvent, and WindowEvent.*

- The **getComponent( )** method returns the component that generated the event

```
Component getComponent( )
```

The ContainerEvent Class

- A ContainerEvent is generated when a component is added to or removed from a container.
- There are **two types of container events**.
- The ContainerEvent class defines int constants that can be used to identify them:
  - COMPONENT_ADDED
  - COMPONENT_REMOVED.

The ContainerEvent Class(contd.)

- **ContainerEvent** is a **subclass of ComponentEvent.**
- Constructor:

ContainerEvent(Component *src, int type, Component comp)*

  - Here, *src* is a reference to the container that generated this event. The type of the event is specified by *type*, and the component that has been added to or removed from the container is *comp*.

The ContainerEvent Class(contd.)

- A reference to the container that generated this event by using the **getContainer( ) method.**

```
Container getContainer( )
```

- The **getChild( )** method returns a reference to the **component** that was added to or removed from the container.

```
Component getChild( )
```

The FocusEvent Class

- A FocusEvent is generated when a component gains or loses input focus.
- These events are identified by the integer constants
  - FOCUS_GAINED
  - FOCUS_LOST.
- **FocusEvent** is a **subclass** of **ComponentEvent** and has these constructors:

FocusEvent(Component *src, int type)*
FocusEvent(Component *src, int type, boolean temporaryFlag)*
FocusEvent(Component *src, int type, boolean temporaryFlag, Component other)*

The argument *temporaryFlag is set to* ***true if the focus event is temporary.*** Otherwise, it is set to **false.**
The other component involved in the focus change, called the ***opposite component**, is passed* in *other.*

The FocusEvent Class(contd.)

- A temporary focus event occurs as a result of another user interface operation.
  - For example, assume that the focus is in a text field. If the user moves the mouse to adjust a scroll bar, the focus is temporarily lost.
- if a FOCUS_GAINED event occurred, *other* will refer to the component that lost focus.
- Conversely, if a FOCUS_LOST event occurred, *other* will refer to the component that gains focus.

The FocusEvent Class(contd.)

- To determine the other component call **getOppositeComponent( ):**

```
Component getOppositeComponent( )
```

  – The opposite component is returned.

- The **isTemporary( )** method indicates if this focus change is temporary.

```
boolean isTemporary( )
```

  – The method returns **true** **<u>if the change is temporary.</u>**
  – Otherwise, it returns **false.**

The InputEvent Class

- The abstract class **InputEvent** is a **subclass** of **ComponentEvent** and is the **superclass** for component input events.
  - Its **subclasses** are **KeyEvent** and **MouseEvent.**
- **InputEvent** defines several integer constants that represent any modifiers, such as the control key being pressed.
- **InputEvent** class defined the following eight values to **represent the modifiers**:

•ALT_MASK

•ALT_GRAPH_MASK

•BUTTON1_MASK

•BUTTON2_MASK

•BUTTON3_MASK

•CTRL_MASK

•META_MASK

•SHIFT_MASK

The InputEvent Class(contd.)

- The **extended modifier** values to avoid conflict between keybaord and mouse event modifiers are:

•ALT_DOWN_MASK

•ALT_GRAPH_DOWN_MASK

•BUTTON1_DOWN_MASK

•BUTTON2_DOWN_MASK

•BUTTON3_DOWN_MASK

•CTRL_MASK

•META_DOWN_MASK

•SHIFT_DOWN_MASK

The InputEvent Class(contd.)

- To test if a modifier was pressed at the time an event is generated, use the **isAltDown( ), isAltGraphDown( ), isControlDown( ), isMetaDown( ),** and **isShiftDown( ) methods.**

boolean isAltDown( )
boolean isAltGraphDown( )
boolean isControlDown( )
boolean isMetaDown( )
boolean isShiftDown( )

The InputEvent Class(contd.)

- To obtain a value that contains all of the original modifier flags call **getModifiers( ) method**

```
int getModifiers( )
```

- We can obtain the extended modifiers by calling **getModifiersEx( ), which is shown here:**

```
int getModifiersEx( )
```

The ItemEvent Class

- An **ItemEvent** is generated when
  - a **check box or a list item is clicked** or
  - when a **checkable menu item is selected or deselected.**
- There are two types of **item events**, which are identified by the following integer constants:

DESELECTED The user deselected an item.

SELECTED The user selected an item.

The ItemEvent Class(contd.)

- **ItemEvent** defines one integer constant, ITEM_STATE_CHANGED, that signifies a change of state.
- ItemEvent has this constructor:

ItemEvent(ItemSelectable *src, int type, Object entry, int state)*

  - Here, *src* is a reference to the component that generated this event.
    - For example, this might be a list or choice element.
  - The type of the event is specified by *type*.
  - The specific item that generated the item event is passed in *entry*.
  - The current state of that item is in *state*

The ItemEvent Class(contd.)

- The getItem( ) method can be used to obtain a reference to the item that generated an event.

```
Object getItem( )
```

- The getItemSelectable( ) method can be used to obtain a reference to the ItemSelectable object that generated an event.

```
ItemSelectable getItemSelectable( )
```

  - Lists and choices are examples of user interface elements that implement the ItemSelectable interface.

- The getStateChange( ) method returns the state change (that is, SELECTED or DESELECTED) for the event.

```
int getStateChange( )
```

The KeyEvent Class

- A **KeyEvent** is generated <u>when keyboard input occurs.</u>
- There are three types of key events, which are identified by these integer constants:

  KEY_PRESSED

  KEY_RELEASED

  KEY_TYPED.

  - The first two events are generated when any key is pressed or released.
  - The last event occurs only when a character is generated.
- Some keypresses does not result in characters.
  - For example, pressing SHIFT does not generate a character.

The KeyEvent Class(contd.)

- There are many other integer constants that are defined by **KeyEvent.**
  - For example, VK_0 through VK_9 define the ASCII equivalents of the numbers.
  - VK_A through VK_Z define the ASCII equivalents of the letters.

VK_ALT	VK_DOWN	VK_LEFT	VK_RIGHT
VK_CANCEL	VK_ENTER	VK_PAGE_DOWN	VK_SHIFT
VK_CONTROL	VK_ESCAPE	VK_PAGE_UP	VK_UP

- The VK constants specify **virtual key codes** and are independent of any modifiers, such as control, shift, or alt.

The KeyEvent Class(contd.)

- **KeyEvent** is a **subclass** of **InputEvent.**
- **Constructor:**

KeyEvent(Component *src, int type, long when,*
int modifiers, int code, char ch)

- Here, *src* is a reference to the component that generated the event.
- The type of the event is specified by *type*.
- The system time at which the key was pressed is passed in *when*.
- The *modifiers* argument indicates which modifiers were pressed when this key event occurred.
- The virtual key code, such as **VK_UP, VK_A**, and so forth, is passed in code.
- The character equivalent (if one exists) is passed in *ch.*
  - If no valid character exists, then ch contains CHAR_UNDEFINED.
  - For KEY_TYPED events, code will contain VK_UNDEFINED.

The KeyEvent Class(contd.)

- The KeyEvent class defines several methods,
  - **getKeyChar( ),** which returns the character that was entered,
  - **getKeyCode( ),** which returns the key code.

char **getKeyChar( )**
int **getKeyCode( )**

  - If no valid character is available, then getKeyChar( ) returns CHAR_UNDEFINED.
  - When a KEY_TYPED event occurs, getKeyCode( ) returns VK_UNDEFINED.

The MouseEvent Class

- There are eight types of mouse events.
- The **MouseEvent class** defines the following integer constants that can be used to identify them:
  - ✓ MOUSE_CLICKED The user clicked the mouse.
  - ✓ MOUSE_DRAGGED The user dragged the mouse.
  - ✓ MOUSE_ENTERED The mouse entered a component.
  - ✓ MOUSE_EXITED The mouse exited from a component.
  - ✓ MOUSE_MOVED The mouse moved.
  - ✓ MOUSE_PRESSED The mouse was pressed.
  - ✓ MOUSE_RELEASED The mouse was released.
  - ✓ MOUSE_WHEEL The mouse wheel was moved.

The MouseEvent Class(contd.)

- **MouseEvent** is a s**ubclass of InputEvent.** ***Constructor:***

MouseEvent(Component *src, int type, long when, int modifiers,* int *x, int y, int clicks, boolean triggersPopup)*

- Here, *src* is a reference to the component that generated the event. The type of the event is specified by *type*. The system time at which the mouse event occurred is passed in *when*.
- The *modifiers* argument indicates which modifiers were pressed when a mouse event occurred.
- The coordinates of the mouse are passed in *x* and *y.*
- The click count is passed in *clicks*.
- The *triggersPopup* flag indicates if this event causes a pop-up menu to appear on this platform.

The MouseEvent Class(contd.)

- Two commonly used methods in this class are
  - **getX( ) -**return the X coordinate
  - **getY( ) -** return the Y coordinate
    - (within the component when the event occurred.)

```
int getX( )
int getY( )
```

- **getPoint( ) method -** to obtain the **coordinates** of the mouse.

```
Point getPoint( )
```

  - returns a Point object that contains the X,Y coordinates in its integer members: x and y.

The MouseEvent Class(contd.)

- The **translatePoint( )** method **changes the location of the event**.

```
void translatePoint(int x, int y)
```

  - Here, the arguments *x and* y are ***added to the coordinates*** of the event.

- The **getClickCount( )** method obtains the **number of mouse clicks** for this event.

```
int getClickCount( )
```

- The **isPopupTrigger( )** method **tests if this event causes a pop-up menu to appear** on this platform.

```
boolean isPopupTrigger( )
```

The MouseEvent Class(contd.)

- **getButton( )** method- **returns a value that represents the button** that caused the event

```
int getButton( )
```

- The return value will be one of these constants defined by **MouseEvent:**

NOBUTTON - indicates that no button was pressed or released.

BUTTON1

BUTTON2

BUTTON3

The MouseEvent Class(contd.)

- Java SE 6 added three methods to **MouseEvent that obtain the coordinates of the mouse <u>relative to the screen</u>** rather than the component.

Point **getLocationOnScreen**( )

- The **getLocationOnScreen( )** method returns a Point object that contains both the X and Y coordinate.

int **getXOnScreen( )** –

- return the X coordinate.

int **getYOnScreen( )**

- return the Y coordinate.

The MouseWheelEvent Class

- The **MouseWheelEvent** class encapsulates a mouse wheel event.
  - It is a **subclass of MouseEvent**.
- Not all mice have wheels.
- If a mouse has a wheel, it is located between the left and right buttons.
- Mouse wheels are used for scrolling.
- MouseWheelEvent defines these two integer constants:

WHEEL_BLOCK_SCROLL	A **page-up** or **page-down** scroll event occurred.
WHEEL_UNIT_SCROLL	A line-up or line-down scroll event occurred.

The MouseWheelEvent Class(contd.)

- Constructor defined by **MouseWheelEvent:**

  MouseWheelEvent(Component *src*, int *type*, long *when*, int *modifiers*, int *x*, int *y*, int *clicks*, boolean *triggersPopup*, int *scrollHow*, int *amount*, int *count)*

  - Here, *src* is a reference to the object that generated the event. The type of the event is specified by *type*. The system time at which the mouse event occurred is passed in *when*. The *modifiers* argument indicates which modifiers were pressed when the event occurred. The coordinates of the mouse are passed in *x* and *y*. The number of clicks the wheel has rotated is passed in *clicks*. The *triggersPopup* flag indicates if this event causes a pop-up menu to appear on this platform.
  - The *scrollHow* value must be either **WHEEL_UNIT_SCROLL or WHEEL_BLOCK_SCROLL.**
  - The number of units to scroll is passed in *amount*.
  - The *count* parameter indicates the number of rotational units that the wheel moved

The MouseWheelEvent Class(contd.)

- MouseWheelEvent defines methods that give you access to the wheel event. To obtain the number of rotational units, call getWheelRotation( ), :

  int **getWheelRotation**( )

  - If the value is **positive**, the wheel moved counterclockwise.
  - If the value is negative, the wheel moved clockwise.

- To obtain the type of scroll, call getScrollType( ), shown next:

  int **getScrollType**( )

  - It returns either WHEEL_UNIT_SCROLL or WHEEL_BLOCK_SCROLL.

- If the scroll type is WHEEL_UNIT_SCROLL, we can obtain the number of units to scroll by calling getScrollAmount( ).

  int **getScrollAmount**( )

The TextEvent Class

- Instances of TextEvent class describe text events.
- These are generated by **text fields** and **text areas** when ***characters are entered by a user or program***.
- **TextEvent** defines the integer constant **TEXT_VALUE_CHANGED.**
- The one constructor for this class is :

TextEvent(Object *src, int type)*

  – Here, *src* is a reference to the object that generated this event. The type of the event is specified by *type*

The TextEvent Class(contd.)

- The **TextEvent object** **does not include the characters currently in the text component** that generated the event
  - Our program must use other methods associated with the text component to retrieve that information.
- Text event notification is like as a signal to a listener that it should retrieve information from a specific text component

The WindowEvent Class

- There are ten types of window events.
- The **WindowEvent class defines integer constants** that can be used to
  - **WINDOW_ACTIVATED** The window was activated.
  - **WINDOW_CLOSED** The window has been closed.
  - **WINDOW_CLOSING** The user requested that the window be closed.
  - **WINDOW_DEACTIVATED** The window was deactivated.
  - **WINDOW_DEICONIFIED** The window was deiconified.
  - **WINDOW_GAINED_FOCUS** The window gained input focus.
  - **WINDOW_ICONIFIED** The window was iconified.
  - **WINDOW_LOST_FOCUS** The window lost input focus.
  - **WINDOW_OPENED** The window was opened.
  - **WINDOW_STATE_CHANGED** The state of the window changed.

The WindowEvent Class(contd.)

- **WindowEvent** is a **subclass of ComponentEvent.**
- It defines several constructors.

WindowEvent(Window *src, int type)*

- Here, *src is a reference to the object that generated this event. The type of the event is type.*

- The next three constructors offer more detailed control:

WindowEvent(Window *src, int type, Window other)*

WindowEvent(Window *src, int type, int fromState, int toState)*

WindowEvent(Window *src, int type, Window other, int fromState, int toState)*

- Here, *other* specifies the opposite window when a focus or activation event occurs. The *fromState* specifies the prior state of the window, and *toState* specifies the new state that the window will have when a window state change occurs.

The WindowEvent Class(contd.)

- A commonly used method in this class is getWindow( ).
- getWindow( ) returns the Window object that generated the event.

```
Window getWindow( )
```

- WindowEvent also defines methods that
  - return the opposite window (when a focus or activation event has occurred),

```
Window getOppositeWindow( )
```

  - the previous window state,

```
int getOldState( )
```

  - and the current window state.

```
int getNewState( )
```

Reference

- Herbert Schildt, Java: **The Complete Reference, 8/e, Tata McGraw Hill, 2011**.