Key & Mouse Listeners

Introduction

In this unit you will learn how to build programs that interact with your JFrame or JPanel through the mouse and keyboard.

Key Listeners

Classes that implement the KeyListener interface are able to respond to keyboard events. A keyboard event is when keys are pressed, typed or released.

Import for KeyListener:

import java.awt.event.KeyListener;

KeyListener Overview:

Methods:

Return Type	Name	Description
void	keyPressed(KeyEvent e)	Method is called when a key is pressed down.
void	keyTyped(KeyEvent e)	Method is called when a key is typed. (When a key would normally appear on the screen)
void	keyReleased(KeyEvent e)	Method is called when a key goes up.

You most likely noticed that all the methods receive a KeyEvent object. This object stores information about the key event.

Import for KeyEvent:

import java.awt.event.KeyEvent;

KeyEvent Overview:

Methods:

Return Type	Name	Description
char	getKeyChar()	Returns the character value of the key.
int	getKeyCode()	Returns a numeric value of the key pressed. See the KeyEvent docs for the list of key codes. VK_UP is the code for the up arrow.

Using a KeyListener:

In a Frame	In a Panel
1. Make your frame implement KeyListener 2. Write the required methods from KeyListener 3. Let the program know your frame will be listening for key events Example: **Key Listener Moving Box (Frame)**	1. Make your panel implement KeyListener. 2. Write the required methods from KeyListener. 3. Let the program know your panel will be listening for key events. 4. Request Focus when the panel gets added to the frame (see example) Example: **Key Listener Moving Box (Panel)**

Mouse Listeners

Classes that implement the MouseListener interface are able to respond to mouse events. A mouse event is when mouse buttons are pressed, released, clicked or when the mouse enters/leaves the implanting object.

Import for MouseListener:

```
import java.awt.event.MouseListener;
```

MouseListener Overview:

Methods:

Return Type	Name	Description
void	mousePressed(MouseEvent e)	Method is called when a mouse key is pressed down.
void	mouseClicked(MouseEvent e)	Method is called when a mouse key is pressed down and then quickly released.
void	mouseReleased(MouseEvent e)	Method is called when a mouse key goes up.
void	mouseEntered(MouseEvent e)	Method is called when the mouse enters the area of the frame/panel.
void	mouseExited(MouseEvent e)	Method is called when the mouse leaves the area of the frame/panel.

You most likely noticed that all the methods receive a MouseEvent object. This object stores information about the mouse event.

Import for MouseEvent:

```
import java.awt.event.MouseEvent;
```

MouseEvent Overview:

Methods:

Return Type	Name	Description
Int	getX()	Returns the x position of the mouse during the event.
Int	gety()	Returns the y position of the mouse during the event.
Int	getButton()	Gets the button that was pressed. See the MouseEvent docs for the list of mouse codes. BUTTON1 is used for the left button of the mouse

Using a MouseListener:

In a Frame	In a Panel
1. Make your frame implement MouseListener 2. Write the required methods from MouseListener 3. Let the program know your frame will be listening for mouse events Example: **Frame Mouse Listener Example**	1. Make your frame implement MouseListener 2. Write the required methods from MouseListener 3. Let the program know your panel will be listening for mouse events Example: **Panel Mouse Listener Example**

Mouse Motion Listeners

Classes that implement the MouseMotionListener interface are able to respond to mouse motion. Mouse motion is when mouse is moved or dragged.

MouseMotionListener Overview:

Methods:

Return Type	Name	Description
void	mouseDragged(MouseEvent e)	Method is called when the mouse is moving while a mouse button is down.
void	mouseMoved(MouseEvent e)	Method is called when the mouse is moving.

Loading & Drawing Images

For images we will be using the BufferedImage class. For this unit all you need to know about BufferedImage is that it can store images.

We will load images using the below code:

```
try
{
   img1 = ImageIO.read((new File("Image1Location")));
   img2 = ImageIO.read((new File("Image2Location ")));
}
catch(Exception e)
{
   System.out.println("Error Loading Images: " + e.getMessage());
}
```

The code attempts to load 2 images. If it fails you will see an error message.

Once an image has been loaded we will use the following Graphics method to draw it to the screen:

Return Type	Name	Description
void	drawImage(Image img, int x, int y, ImageObserver observer)	Draws the given image to the screen at (x,y). Use null for the observer.

Example: Load and Draw 2 Images

Double Buffering

Sometimes when your program is running you may notice a flicker. The reason for this flicker is that the screen is cleared and then gets slowly redrawn. You see the flicker during the brief time the screen is empty, while all your data is being redrawn.

To fix this issue you have to use a double buffering to paint your screen. **Double buffering** is when you do all drawing/painting to a single image and then paint that image to screen. That single image is drawn so quickly you will not see any flickering.

Steps to use Double Buffering:

1. Create a BufferedImage that is the size of your drawing area
2. In your paint method do the following:
    a. Create a graphics object that stores the graphics of your image
    b. Do all your drawing to the image's graphics
    c. Draw the image to the graphics of your window at (0,0)

Example: Double Buffered Moving Box

Terms:

Term	Definition
Buffer	A temporary storage location for graphics
Double Buffering	Doing all your drawing to a buffer and then drawing the buffer to the screen.