ECE 462
Object-Oriented Programming using C++ and Java

Key Inputs in Java Games

Yung-Hsiang Lu
yunglu@purdue.edu
Handle Key Events

• have the focus of the keyboard inputs by calling `setFocusable(true)`

• two types of events:
  – pressing and releasing
  – what key (Unicode) is pressed (such as 'A' or '<')

• implement three functions
  – `keyPressed(KeyEvent e)`
  – `keyReleased(KeyEvent e)`
  – `keyTyped(KeyEvent e)`
java.awt.event

Interface KeyListener

All Superinterfaces:
   EventListener

All Known Implementing Classes:
   AWTEventMulticaster, BasicComboBoxUI.KeyHandler, BasicComboBoxPopup.InvocationKeyHandler,
   BasicTableUI.KeyHandler, BasicTreeUI.KeyHandler, KeyAdapter

public interface KeyListener
extends EventListener

The listener interface for receiving keyboard events (keystrokes). The class that is interested in processing a keyboard event either implements this interface (and all the methods it contains) or extends the abstract KeyAdapter class (overriding only the methods of interest).
Since:
1.1

See Also:
KeyListener, KeyEvent, Tutorial: Writing a Key Listener

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void keyPressed(KeyEvent e)</code></td>
<td>Invoked when a key has been pressed.</td>
</tr>
<tr>
<td><code>void released(KeyEvent e)</code></td>
<td>Invoked when a key has been released.</td>
</tr>
<tr>
<td><code>void typed(KeyEvent e)</code></td>
<td>Invoked when a key has been typed.</td>
</tr>
</tbody>
</table>

---

### Method Detail

**keyTyped**

```java
void keyTyped(KeyEvent e)
```

Done
How to Write a Key Listener

Key events indicate when the user is typing at the keyboard. Specifically, key events are fired by the component with the keyboard focus when the user presses or releases keyboard keys. For detailed information about focus, see How to Use the Focus Subsystem.

Note: To define special reactions to particular keys, use key bindings instead of a key listener. For further information, see How to Use Key Bindings.

Notifications are sent about two basic kinds of key events:
- The typing of a Unicode character
- The pressing or releasing of a key on the keyboard
Copyright (c) 1995 - 2008 Sun Microsystems, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Sun Microsystems nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
package events;

/*
 * KeyEventDemo
 */

import java.awt.BorderLayout;

public class KeyEventDemo extends JFrame implements KeyListener, ActionListener {
    JTextArea displayArea;
    JTextField typingArea;
    static final String newline = System.getProperty("line.separator");

    public static void main(String[] args) {
        /* Use an appropriate Look and Feel */
        try {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.gtk.GTKLookAndFeel");
            UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
        } catch (UnsupportedLookAndFeelException ex) {
            ex.printStackTrace();
        } catch (IllegalAccessException ex) {
            ex.printStackTrace();
        } catch (InstantiationException ex) {
            ex.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }
    }
}
} */

Turn off metal's use of bold fonts */
UIManager.put("swing.boldMetal", Boolean.FALSE);

// Schedule a job for event dispatch thread:
// creating and showing this application's GUI.
javax.swing.SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        createAndShowGUI();
    }
});

/**
 * Create the GUI and show it. For thread safety, this method should be
 * invoked from the event-dispatching thread.
 */

private static void createAndShowGUI() {
    // Create and set up the window.
    KeyEventDemo frame = new KeyEventDemo("KeyEventDemo");
    frame setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Set up the content pane.
    frame.getContentPane();

    // Display the window.
    frame.pack();
    frame.setVisible(true);
```java
private void addComponentsToPane() {
    JButton button = new JButton("Clear");
    button.addActionListener(this);

    typingArea = new JTextField(20);
    typingArea.addKeyListener(this);

    // Uncomment this if you wish to turn off focus traversal. The focus
    // subsystem consumes focus traversal keys, such as Tab and Shift Tab.
    // If you uncomment the following line of code, this disables focus
    // traversal and the Tab events will become available to the key event
    // listener.
    // typingArea.setFocusTraversalKeysEnabled(false);

    displayArea = new JTextArea();
    displayArea.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(displayArea);
    scrollPane.setPreferredSize(new Dimension(375, 125));

    getContentPane().add(typingArea, BorderLayout.PAGE_START);
    getContentPane().add(scrollPane, BorderLayout.CENTER);
    getContentPane().add(button, BorderLayout.PAGE_END);
}
```
```java
public KeyEventDemo(String name) {
    super(name);
}

/** Handle the key typed event from the text field. */
public void keyTyped(KeyEvent e) {
    displayInfo(e, "KEY_TYPED: ");
}

/** Handle the key pressed event from the text field. */
public void keyPressed(KeyEvent e) {
    displayInfo(e, "KEY_PRESSED: ");
}

/** Handle the key released event from the text field. */
public void keyReleased(KeyEvent e) {
    displayInfo(e, "KEY_RELEASED: ");
}

/** Handle the button click. */
public void actionPerformed(ActionEvent e) {
    // Clear the text components.
    displayArea.setText(""");
    typingArea.setText("");

    // Return the focus to the typing area.
    typingArea.requestFocusInWindow();
}
```
typingArea.requestFocusInWindow();

/*
 * We have to jump through some hoops to avoid trying to print non-printing
 * characters such as Shift. (Not only do they not print, but if you put
 * them in a String, the characters afterward won't show up in the text
 * area.)
 */

private void displayInfo(KeyEvent e, String keyStatus) {

    // You should only rely on the key char if the event
    // is a key typed event.
    int id = e.getID();
    String keyString;
    if (id == KeyEvent.KEY_TYPED) {
        char c = e.getKeyChar();
        keyString = "key character = '" + c + '"';
    } else {
        int keyCode = e.getKeyCode();
        keyString = "key code = " + keyCode + " (",
            + KeyEvent.getKeyText(keyCode) + ")";
    }

    int modifiersEx = e.getModifiersEx();
    String modString = "extended modifiers = " + modifiersEx;
    String tmpString = KeyEvent.getModifiersExText(modifiersEx);
    if (tmpString.length() > 0) {

```java
if (tmpString.length() > 0) {
    modString += " (" + tmpString + ")";
} else {
    modString += " (no extended modifiers)";
}

String actionString = "action key? ";
if (e.isActionKey()) {
    actionString += "YES";
} else {
    actionString += "NO";
}

String locationString = "key location: ";
int location = e.getKeyLocation();
if (location == KeyEvent.KEY_LOCATION_STANDARD) {
    locationString += "standard";
} else if (location == KeyEvent.KEY_LOCATION_LEFT) {
    locationString += "left";
} else if (location == KeyEvent.KEY_LOCATION_RIGHT) {
    locationString += "right";
} else if (location == KeyEvent.KEY_LOCATION_NUMPAD) {
    locationString += "numpad";
} else { // (location == KeyEvent.KEY_LOCATION_UNKNOWN)
    locationString += "unknown";
}

displayArea.append(keyStatus + newline + " " + keyString + newline
```
String actionString = "action key? ";
if (e.isActionKey()) {
    actionString += "YES";
} else {
    actionString += "NO";
}

String locationString = "key location: ";
int location = e.getKeyLocation();
if (location == KeyEvent.KEY_LOCATION_STANDARD) {
    locationString += "standard";
} else if (location == KeyEvent.KEY_LOCATION_LEFT) {
    locationString += "left";
} else if (location == KeyEvent.KEY_LOCATION_RIGHT) {
    locationString += "right";
} else if (location == KeyEvent.KEY_LOCATION_NUMPAD) {
    locationString += "numpad";
} else { // (location == KeyEvent.KEY_LOCATION_UNKNOWN)
    locationString += "unknown";
}

displayArea.append(keyStatus + newline + "  " + keyString + newline
 + "  " + modString + newline + "  " + actionString
 + newline + "  " + locationString + newline);
displayArea.setCaretPosition(displayArea.getDocument().getLength());
Handle Key Events in a Full Screen Game
C:\yunglu\ece462\JavaGame\ch03src\build>java KeyTest
KeyInputTest. Pres
package com.brackeen.javagamebook.test;

import java.awt.*;

/**
 * Simple abstract class used for testing. Subclasses should
 * implement the draw() method.
 */
public abstract class GameCore {

    protected static final int FONT_SIZE = 24;

    private static final DisplayMode POSSIBLE_MODES[] = {
        new DisplayMode(800, 600, 32, 0),
        new DisplayMode(800, 600, 24, 0),
        new DisplayMode(800, 600, 16, 0),
        new DisplayMode(640, 480, 32, 0),
        new DisplayMode(640, 480, 24, 0),
        new DisplayMode(640, 480, 16, 0)
    };

    private boolean isRunning;
    protected ScreenManager screen;

    /**
     * Signals the game loop that it's time to quit
     */
    public void stop() {
    }
public void stop() {
    isRunning = false;
}

/**
 * Calls init() and gameLoop()
 */
public void run() {
    try {
        init();
        gameLoop();
    }
    finally {
        screen.restoreScreen();
    }
}

/**
 * Sets full screen mode and initiates and objects.
 */
public void init() {
    screen = new ScreenManager();
    DisplayMode displayMode =
        screen.findFirstCompatibleMode(POSSIBLE_MODES);
    screen.setFullScreen(displayMode);

    Window window = screen.getFullScreenWindow();
    window.setFont(new Font("Dialog", Font.PLAIN, FONT_SIZE));
    window.setBackground(Color.blue);
Window window = screen.getFullscreenWindow();
window.setFont(new Font("Dialog", Font.PLAIN, FONT_SIZE));
window.setBackground(Color.blue);
window.setForeground(Color.white);

isRunning = true;

public Image loadImage(String fileName) {
    return new ImageIcon(fileName).getImage();
}

/**
 * Runs through the game loop until stop() is called.
 */
public void gameLoop() {
    long startTime = System.currentTimeMillis();
    long currTime = startTime;

    while (isRunning) {
        long elapsedTime =
            System.currentTimeMillis() - currTime;
        currTime += elapsedTime;

        // update
        update(elapsedTime);
}
// draw the screen
Graphics2D g = screen.getGraphics();
draw(g);
g.dispose();
screen.update();

// take a nap
try {
    Thread.sleep(20);
}
catch (InterruptedException ex) {
}

/**
   * Updates the state of the game/animation based on the amount of elapsed time that has passed.
   *
   * @param elapsedTime the amount of time that has passed
   */
public void update(long elapsedTime) {
    // do nothing
}

/**
   * Draws to the screen. Subclasses must override this method.
   *
   * @param g Graphics context
   */
public abstract void draw(Graphics2D g);
```java
import java.awt.*;

/**<n *
 * A simple keyboard test. Displays keys pressed and released to the screen.
 * Useful for debugging key input, too.
 */

public class KeyTest extends GameCore implements KeyListener {

    public static void main(String[] args) {
        new KeyTest().run();
    }

    private LinkedList<String> messages = new LinkedList<String>();

    public void init() {
        super.init();
        Window window = screen.getFullScreenWindow();

        // allow input of the TAB key and other keys normally
        // used for focus traversal
        window.setFocusTraversalKeysEnabled(false);

        // register this object as a key listener for the window
        window.addKeyListener(this);
        addMessage("KeyPressTest. Press Escape to exit");
    }

    // a method from the KeyListener interface
    public void keyPressed(KeyEvent e) {
        int keyCode = e.getKeyCode();
    }

    // another method from the KeyListener interface
    public void keyReleased(KeyEvent e) {
        int keyCode = e.getKeyCode();
    }

    // another method from the KeyListener interface
    public void keyTyped(KeyEvent e) {
        int keyCode = e.getKeyCode();
    }
}
```
if (keyCode == KeyEvent.VK_ESCAPE) {
    stop();
} else {
    addMessage("Pressed: " + KeyEvent.getKeyText(keyCode));
    // make sure the key isn't processed for anything else
    e.consume();
}

// a method from the KeyListener interface
public void keyPressed(KeyEvent e) {
    int keyCode = e.getKeyCode();
    addMessage("Pressed: " + KeyEvent.getKeyText(keyCode));
    // make sure the key isn't processed for anything else
    e.consume();
}

// a method from the KeyListener interface
public void keyTyped(KeyEvent e) {
    // this is called after the key is released - ignore it
    // make sure the key isn't processed for anything else
    e.consume();
}

/**
 * Add a message to the list of messages.
 */
public synchronized void addMessage(String message) {
    messages.add(message);
    if (messages.size() >= screen.getHeight() / FONT_SIZE) {
        messages.remove(0);
    }
}

/**
 * Draw the list of messages
 */
public synchronized void draw(Graphics2D g) {
    Window window = screen.getFullScreenWindow();
    g.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
                       RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
    // draw background
    g.setColor(window.getBackground());
    g.fillRect(0, 0, screen.getWidth(), screen.getHeight());
    // draw messages
    g.setColor(window.getForeground());
    int y = FONT_SIZE;
    for (int i = 0; i < messages.size(); i++) {
        g.drawString((String) messages.get(i), 5, y);
        y += FONT_SIZE;
    }
}
Mouse Inputs in Java Games

Yung-Hsiang Lu
yunglu@purdue.edu
Java Mouse Listeners

- MouseListener: handle clicks
  - mouseClicked(MouseEvent e)
  - mouseEntered(MouseEvent e)
  - mouseExited(MouseEvent e)
  - mousePressed(MouseEvent e)
  - mouseReleased(MouseEvent e)

- MouseMotionListener: handle motion
  - mouseDragged(MouseEvent e)
  - mouseMoved(MouseEvent e)
java.awt.event

Interface MouseListener

All Superinterfaces:
   ActionListener

All Known Subinterfaces:
   MouseInputListener

All Known Implementing Classes:
   AWTEventMulticaster, BasicButtonListener, BasicComboBoxPopupInvocationMouseHandler,
   BasicComboBoxPopupListMouseHandler, BasicDesktopIconUI_MouseInputHandler,
   BasicFileChooserUI_DoubleClickListener, BasicInternalFrameUI_BorderListener,
   BasicInternalFrameUI_ClassPaneDispatcher, BasicListUI_MouseInputHandler, BasicMenuItemsUI_MouseInputHandler,
   BasicMenuUI_MouseInputHandler, BasicScrollBarUI_ArrowButtonListener, BasicScrollBarUI_TrackListener,
   BasicSliderUI_TrackListener, BasicSplitPaneDividerMouseHandler, BasicTabbedPaneUI_MouseHandler,
   BasicTitlePaneUI_TrackListener, BasicWindowPaneUI_MouseHandler,
   CenteredTitlePaneUI_TrackListener, DefaultButtonListener, DefaultCloseListener,
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mouseClicked</code></td>
<td>Invoked when the mouse button has been clicked (pressed and released) on a component.</td>
</tr>
<tr>
<td><code>mouseEntered</code></td>
<td>Invoked when the mouse enters a component.</td>
</tr>
<tr>
<td><code>mouseExited</code></td>
<td>Invoked when the mouse exits a component.</td>
</tr>
<tr>
<td><code>mousePressed</code></td>
<td>Invoked when a mouse button has been pressed on a component.</td>
</tr>
<tr>
<td><code>mouseReleased</code></td>
<td>Invoked when a mouse button has been released on a component.</td>
</tr>
</tbody>
</table>

## Method Detail

**mouseClicked**
java.awt.event

Interface MouseMotionListener

All Superinterfaces:
   EventListener

All Known Subinterfaces:
   MouseInputListener

All Known Implementing Classes:
   AWTEventMulticaster, BasicButtonListener, BasicComboBoxPopupInvocationMouseHandler,
   BasicComboBoxPopupInvocationMouseMotionHandler, BasicComboBoxPopupListMouseHandler,
   BasicComboBoxPopupListMouseMotionHandler, BasicDesktopIconUI.MouseInputHandler,
   BasicFileChooserUI.DoubleClickListener, BasicInternalFrameUI.BorderListener,
   BasicInternalFrameUIGlassPaneDispatcher, BasicListUI.MouseInputHandler, BasicMenuItemUI.MouseInputHandler,
   BasicMenuUI.MouseInputHandler, BasicScrollBarUI.ArrowButtonListener, BasicScrollBarUI.TrackListener.
1.1

See Also:

MouseMotionAdapter, MouseEvent, Tutorial: Writing a Mouse Motion Listener

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void mouseDragged(MouseEvent e)</td>
</tr>
<tr>
<td>Invoked when a mouse button is pressed on a component and then dragged.</td>
</tr>
<tr>
<td>void mouseMoved(MouseEvent e)</td>
</tr>
<tr>
<td>Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
</table>

mouseDragged

void mouseDragged(MouseEvent e)

Invoked when a mouse button is pressed on a component and then dragged. MOUSE_DRAGGED events will continue to be delivered to the component where the drag originated until the mouse button is released (regardless of whether the mouse
java.awt.event

Class MouseEvent

java.lang.Object
  | java.util.EventObject
    | java.awt.AWTEvent
      | java.awt.event.ComponentEvent
        | java.awt.event.InputEvent
          | java.awt.event.MouseEvent

All Implemented Interfaces:
  Serializable

Direct Known Subclasses:
  MenuDragMouseEvent, MouseWheelEvent
<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static int</strong> BUTTON1</td>
</tr>
</tbody>
</table>
| Indicates mouse button #1; used by `getButton()`.
| **static int** BUTTON2 |
| Indicates mouse button #2; used by `getButton()`.
| **static int** BUTTON3 |
| Indicates mouse button #3; used by `getButton()`.
| **static int** MOUSE_CLICKED |
| The "mouse clicked" event.
| **static int** MOUSE_DRAGGED |
| The "mouse dragged" event.
| **static int** MOUSE_ENTERED |
| The "mouse entered" event.
| **static int** MOUSE_EXITED |
| The "mouse exited" event.
| **static int** MOUSE_FIRST |
| The first number in the range of ids used for mouse events.
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>int getButton()</strong></td>
<td>Returns which, if any, of the mouse buttons has changed state.</td>
</tr>
<tr>
<td><strong>int getClickCount()</strong></td>
<td>Returns the number of mouse clicks associated with this event.</td>
</tr>
<tr>
<td><strong>Point getLocationOnScreen()</strong></td>
<td>Returns the absolute x, y position of the event.</td>
</tr>
<tr>
<td><strong>static String getModifiersText (int modifiers)</strong></td>
<td>Returns a String describing the modifier keys and mouse buttons that were down during the event, such as &quot;Shift&quot;, or &quot;Ctrl+Shift&quot;.</td>
</tr>
<tr>
<td><strong>Point getPoint()</strong></td>
<td>Returns the x,y position of the event relative to the source component.</td>
</tr>
<tr>
<td><strong>int getX()</strong></td>
<td>Returns the horizontal x position of the event relative to the source component.</td>
</tr>
<tr>
<td><strong>int getXOnScreen()</strong></td>
<td>Returns the absolute horizontal x position of the event.</td>
</tr>
<tr>
<td><strong>int getY()</strong></td>
<td>Returns the y position of the event.</td>
</tr>
</tbody>
</table>
MouseTest. Press Escape to exit.

Hello World!
```java
import java.awt.*;

/**
 * A simple mouse test. Draws a "Hello World!" message at the location of the
 * cursor. Click to change to "trail mode" to draw several messages. Use the
 * mouse wheel (if available) to change colors.
 */

public class MouseTest extends GameCore implements KeyListener,
        MouseMotionListener, MouseListener, MouseWheelListener {

    public static void main(String[] args) {
        new MouseTest().run();
    }

    private static final int TRAIL_SIZE = 10;
    private static final Color[] COLORS = { Color.white, Color.black,
            Color.yellow, Color.magenta };

    private LinkedList<Point> trailList;
    private boolean trailMode;
    private int colorIndex;

    public void init() {
        super.init();
        trailList = new LinkedList<Point>();
    }
```
public synchronized void draw(Graphics2D g) {
    int count = trailList.size();

    if (count > 1 && !trailMode) {
        count = 1;
    }

    Window window = screen.getFullScreenWindow();

    // draw background
    g.setColor(window.getBackground());
    g.fillRect(0, 0, screen.getWidth(), screen.getHeight());

    // draw instructions
    g.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
                     RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
    g.setColor(window.getForeground());
    g.drawString("MouseTest. Press Escape to exit.", 5, FONT_SIZE);

    // draw mouse trail
    for (int i = 0; i < count; i++) {
        window.addMouseWheelListener(this);
        window.addMouseMotionListener(this);
        window.addKeyListener(this);
    }
}
for (int i = 0; i < count; i++) {
    Point p = (Point) trailList.get(i);
    g.drawString("Hello World!", p.x, p.y);
}

// from the MouseListener interface
public void mousePressed(MouseEvent e) {
    trailMode = !trailMode;
}

// from the MouseListener interface
public void mouseReleased(MouseEvent e) {
    // do nothing
}

// from the MouseListener interface
public void mouseClicked(MouseEvent e) {
    // called after mouse is released - ignore it
}

// from the MouseListener interface
public void mouseEntered(MouseEvent e) {
    mouseMoved(e);
}

// from the MouseListener interface
public void mouseExited(MouseEvent e) {
// from the MouseMotionListener interface

public void mouseDragged(MouseEvent e) {
    mouseMoved(e);
}

// from the MouseMotionListener interface
public synchronized void mouseMoved(MouseEvent e) {
    Point p = new Point(e.getX(), e.getY());
    trailList.addFirst(p);

    while (trailList.size() > TRAIL_SIZE) {
        trailList.removeLast();
    }
}

// from the MouseWheelListener interface
public void mouseWheelMoved(MouseWheelEvent e) {
    colorIndex = (colorIndex + e.getWheelRotation()) % COLORS.length;

    if (colorIndex < 0) {
        colorIndex += COLORS.length;
    }

    Window window = screen.getFullScreenWindow();
    window.setForeground(COLORS[colorIndex]);
}

// from the KeyListener interface
public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_ESCAPE) {
}
```java
public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_ESCAPE) {
        // exit the program
        stop();
    }
}

public void keyReleased(KeyEvent e) {
    // do nothing
}

public void keyTyped(KeyEvent e) {
    // do nothing
}

Window window = screen.getFullScreenWindow();
window.setForeground(COLORS[colorIndex]);
```

```java
int colorIndex = (colorIndex + e.getWheelRotation()) % COLORS.length;

if (colorIndex < 0) {
    colorIndex += COLORS.length;
}
```
ECE 462
Object-Oriented Programming using C++ and Java

Multiple Inheritance in C++

Yung-Hsiang Lu
yunglu@purdue.edu
Multiple Inheritance

- One of the most (if not the most) controversial features in C++.
- Not supported in Java. Java uses interfaces.
- Most C++ books do not explain the concept and the problems.
- Understand the advantages and the problems and you can decide whether to use it.
Multiple Inheritance

- Student
- Teacher
- TeachingAssistant
- Car
- Truck
- SportUtilityVehicle
- Person
- Woman
- Employee
- FemaleEmployee
Design Issues

- A class provides interface and implementation.
- Code reuse is good but a class, once declared, is hard to change because of the code depending on this class.
- **If you have any doubt, do not create a class.**
- Avoid the proliferation of classes because they make code reuse harder.
Too Many Classes

- In many payroll systems, a person's status (for example tax withholding) depends on the person's role. If a person has children, the person's tax withholding is less.
- If an employee is also a mother, a company may send a gift to the children on their birthdays.
- Should you create one class for each possible status of employee?
- Or, use attributes to distinguish their status?
What Does a Class Give You?

- interface (public inheritance) and implementation
- polymorphism
- object creation

- If you are not using polymorphism, think twice (or more) before creating a class.
- It is usually easier to change the behavior of an attribute (encapsulation) than changing the interface (code reuse).
Repeated Inheritance

class Vehicle
{
    int v_engineSize;
    int v_numberWheel;
};

Does SportUtilityVehicle have one v_engineSize or two?
⇒ two, unless you use virtual inheritance
⇒ ambiguous, compilation error
```cpp
#include <iostream>
using namespace std;

class A {
    protected:
        int a_val;
};
class B : public A {
    protected:
        int b_val;
};
class C : public A {
    protected:
        int c_val;
};
class D : public B, public C {
public:
    D(int av, int bv, int cv) {
        a_val = av;
        b_val = bv;
        c_val = cv;
    }
};

int main(int argc, char * argv[]) {
    return 0;
}
```
```cpp
#include <iostream>
using namespace std;

class A {
    protected:
        int a_val;
};
class B : public A {
    protected:
        int b_val;
};
class C : public A {
    protected:
        int c_val;
};
class D : public B, public C {
public:
    D(int av, int bv, int cv) {
        E::a_val = av;
        F::a_val = av;
        b_val = bv;
        c_val = cv;
    }

    int main(int argc, char * argv[]) {
        return 0;
    }
};
```
Virtual Inheritance
```cpp
#include <iostream>
using namespace std;

class A {
public:
    int eval;
    A(int aa) :
        eval(aa) {
        cout << "A::A" << endl;
    }
    virtual void print() {
        cout << "A::print " << eval << endl;
    }
};

class B : virtual public A {
public:
    int bval;
    B(int aa, int bb) :
        A(aa), bval(bb) {
        cout << "B::B" << endl;
    }
    void print() {
        A::print();
        cout << "B::print " << bval << endl;
    }
};

class C : virtual public A {
public:
    int cval;
    C() :
        A(), cval() {
        cout << "C::C" << endl;
    }
    void print() {
        A::print();
        cout << "C::print " << cval << endl;
    }
};
```
```cpp
int cval;
C(int aa, int cc) :
    A(aa), cval(cc) {
    cout << "C::C" << endl;
}

void print() {
    A::print();
    cout << "C::print " << cval << endl;
}
};
class D : public B, public C {
public:
    int dval;
D(int aa, int bb, int cc, int dd) :
    C(aa, cc), B(aa + 100, bb), A(aa + 1000), dval(dd) {
    cout << "D::D" << endl;
}

void print() {
    B::print();
    C::print();
    cout << "D::print " << dval << endl;
}
};
int main() {
    D u_obj_1(1, 2, 3, 4);
    u_obj_1.print();
    return 0;
}
```
Virtual Inheritance and Copy Constructor
//VirtualBaseCopyConstruct.cc

#include <iostream>
#include <string>

using namespace std;

class X {
  int x;
public:
  X(int xx) :
    x(xx) {
  }
  //copy constructor:
  X(const X& other) :
    x(other.x) {
    //(&)
    virtual void print() {
      cout << "x of X subobject: " << x << endl;
    }
  }
};

class Y : virtual public X {
  int y;
public:
  Y(int xx, int yy) :
    X(xx), y(yy) {
  }
  //copy constructor:
  Y(const Y& other) :
    x(other.x), y(other.y) {
  }
};
Y(const Y & other) :
    X(other), y(other.y) {
} // (B)

void print() {
    X::print();
    cout << "y of Y subobject: " << y << endl;
}

class T : virtual public X {
    int t;

    public:
        T(int xx, int tt) :
            X(xx), t(tt) {
    }

    // copy constructor:
    T(const T & other) :
        X(other), t(other.t) {
    } // (C)

    void print() {
        X::print();
        cout << "t of T subobject: " << t << endl;
    }
};

class Z : public Y {
    int z;

    public:
        Z const & operator=(Z & other) {
            Z(o)
    } // (C)

    void print() {
        Z::print();
        cout << "z of Z subobject: " << z << endl;
    }
}
```cpp
public:
    Z(int xx, int yy, int zz):
        Y(xx, yy), X(xx), z(zz) {
    } // copy constructor:
    Z(const Z& other) :
        Y(other), X(other), z(other.z) {
    } // (D)
    void print() {
        Y::print();
        cout << "x of Z subobject: " << x << endl;
    }
};

class U : public Z, public T {
    int u;
public:
    U(int xx, int yy, int zz, int tt, int uu):
        Z(xx, yy, zz), T(tt, uu), X(xx), u(uu) {
    } // copy constructor:
    U(const U& other) // copy constructor
        :
            Z(other), T(other), X(other), u(other.u) {
    } // (E)
    void print() {
        Z::print();
        T::print();
        cout << "x of U subobject: " << u << endl;
    }
};
```
```cpp
int main() {
    cout << "Z object: " << endl;
    Z z_obj_1(1110, 1120, 1130);
    z_obj_1.print(); // (F)
    cout << endl;

    cout << "Z's duplicate object: " << endl;
    Z z_obj_2 = z_obj_1;
    z_obj_2.print(); // (G)
    cout << endl;

    cout << "U object: " << endl;
    U u_obj_1(9100, 9200, 9300, 9400, 9500);
    u_obj_1.print(); // (H)
    cout << endl;

    //call U's copy constructor:
    cout << "U's duplicate object: " << endl;
    U u_obj_2 = u_obj_1;
    u_obj_2.print(); // (I)
    cout << endl;

    return 0;
}
```
Virtual Inheritance and Assignment Operator
```cpp
#include <iostream>
using namespace std;

class X {
   int x;
public:
   X(int xx) : x(xx) {
   } // assignment op:
   X& operator=(const X& other) {
      if (this == &other)
         return *this;
      x = other.x;
      return *this;
   }
   virtual void print() {
      cout << "x of X subobject: " << x << endl;
   }
};

class Y : virtual public X {
   int y;
};
```
```cpp
int y;
public:
Y(int xx, int yy):
    X(xx), y(yy) {}  
Y(const Y& other):
    X(other), y(other.y) {} 
//assignment op:
Y& operator=(const Y& other) {
    if (this == &other)
        return *this;
    X::operator=(other);
    y = other.y;
    return *this;
}
void print()
    X::print();
    cout << "y of Y subobject: " << y << endl;
};

class T : virtual public X {
    int t;
public:
    T(int xx, int tt):
        X(xx), t(tt) {}  
};
```
```cpp
public:
    T(int xx, int tt) :
        X(xx), t(tt) {}

    T(const T& other) :
        X(other), t(other.t) {}

    // assignment op:
    T& operator=(const T& other) {
        if (this == &other)
            return *this;
        X::operator=(other);
        t = other.t;
        return *this;
    }

    void print() {
        X::print();
        cout << "t of T subobject: " << t << endl;
    }

};

class Z : public Y {
    int z;

    public:
    Z(int xx, int yy, int zz) :
        Y(xx, yy), X(xx), z(zz) {
    }
    Z(const Z& other) :
```
```cpp
Z(const Z& other) :
    Y(other), X(other), z(other.z)
{}
// assignment op:
Z& operator=(const Z& other) {
    if (this == &other)
        return *this;
    Y::operator=(other);
    z = other.z;
    return *this;
}

void print() {
    Y::print();
    cout << "z of Z subobject: " << z << endl;
}

class U : public Z, public T {
    int u;
    public:
    U(int xx, int yy, int zz, int tt, int uu) :
        Z(xx, yy, zz), T(xx, tt), X(xx), u(uu)
    {}
    U(const U& other) :
        Z(other), T(other), X(other), u(other.u)
    {}
    // assignment op:
    U& operator=(const U& other) {
```
```cpp
class U : public Z, public T {
    int u;

public:
    U(int xx, int yy, int zz, int tt, int uu) :
        Z(xx, yy, zz), T(xx, tt), X(xx), u(uu) { }
    U(const U& other) :
        Z(other), T(other), X(other), u(other.u) { }
    // assignment op:
    U& operator=(const U& other) {
        if (this == &other)
            return *this;
        Z::operator=(other); // (A)
        T::operator=(other); // (B)
        u = other.u;
        return *this;
    } 
    void print() {
        Z::print();
        T::print();
        cout << "U of U subobject: " << u << endl;
    }
};

int main() {
    cout << "U object coming up: " << endl;
    U u_obj1(9100, 9200, 9300, 9400, 9500);
    cout << "U object after assignment: " << endl;
    U u_obj2(9100, 9200, 9300, 9400, 9500);
    u_obj1 = u_obj2;
    u_obj1.print();
};
```
return *this;
Z::operator=( other ); // (A)
T::operator=( other ); // (B)
u = other.u;
return *this;

void print() {
Z::print();
T::print();
cout << "u of U subobject: " << u << endl;
}

int main() {
cout << "U object coming up: " << endl;
U u_obj_1(9100, 9200, 9300, 9400, 9500);
u_obj_1.print(); // (C)
cout << endl;
U u_obj_2(7100, 7200, 7300, 7400, 7500);

u_obj_2 = u_obj_1; // (D)
cout << "U object after assignment: " << endl;
u_obj_2.print(); // (E)
return 0;
}
ECE 462
Object-Oriented Programming
using C++ and Java

Qt Layout

Yung-Hsiang Lu
yunglu@purdue.edu
Qt Examples and Demos

Qt is supplied with a number of example applications and demonstrations that have been written to provide developers with examples of the Qt API in use, highlight good programming practice, and showcase features found in each of Qt's core technologies.

The example and demo launcher can be used to explore the different categories available. It provides an overview of each example, lets you view the documentation in Qt Assistant, and is able to launch examples and demos.

Documentation for examples can be found in the Tutorial and Examples section of the Qt documentation.
Qt uses a layout-based approach to widget management. Widgets are arranged in the optimal positions in windows based on simple layout rules, leading to a consistent look and feel.

Custom layouts can be used to provide more control over the positions and sizes of child widgets.

The example launcher provided with Qt can be used to explore each of the examples in this directory.
The Basic Layouts example shows how to use the standard layout managers that are available in Qt: QBoxLayout and QGridLayout.
Basic Layouts

File

Horizontal layout

Button 1  Button 2  Button 3  Button 4

Grid layout

This widget takes up about two thirds of the grid layout.

Line 1: 

Line 2:  

Line 3:  

This widget takes up all the remaining space in the top-level layout.

OK  Cancel
Layouts

Qt uses a layout-based approach to widget management. Widgets are arranged in the optimal positions in windows based on simple layout rules, leading to a consistent look and feel.

Custom layouts can be used to provide more control over the positions and sizes of child widgets.

The example launcher provided with Qt can be used to explore each of the examples in this directory.
Flow Layout

The Flow Layout example demonstrates a custom layout that arranges child widgets from left to right and top to bottom in a top-level widget.
Menubar and Vertical Layout
C:\yunglu\workspace\VLayout>qmake -project
C:\yunglu\workspace\VLayout>qmake

mingw32-make -f Makefile.Debug
mingw32-make[1]: Entering directory 'C:/yunglu/workspace/VLayout'
g++ -c -g -frtti -fexceptions -mtthreads -Wall -DUNICODE -DQT_LARGEFILE_SUPPORT -DQT_NEEDS_QMAI

programs/Qt/include/QtCore" -I".../programs/Qt/include/QtCore" -I".../programs/Qt/include/QtGui" -I".../
include" -I"" -I"c:/yunglu/programs/Qt/include/ActiveQt" -I"debug
programs/Qt/mkspecs/win32-g++" -o debug\main.o main.cpp

g++ -c -g -frtti -fexceptions -mtthreads -Wall -DUNICODE -DQT_LARGEFILE_SUPPORT -DQT_NEEDS_QMAI

programs/Qt/include/QtCore" -I".../programs/Qt/include/QtCore" -I".../programs/Qt/include/QtGui" -I".../
include" -I"" -I"c:/yunglu/programs/Qt/include/ActiveQt" -I"debug
programs/Qt/mkspecs/win32-g++" -o debug\vlayout.o vlayout.cpp

C:/yunglu/programs/Qt/bin/moc.exe -DUNICODE -DQT_LARGEFILE_SUPPORT

_GLIBC -DQT_FUNCTIONAL -DQT_THREAD_SUPPORT -DQT_NEEDS_QMAIN -I".../include/QtCore" -I".../programs/Qt/include/QtCore" -I".../pr
de/QtGui" -I"".../programs/Qt/include/QtGui" -I"".../programs/Qt/include/ActiveQt" -I""debug\moc_vlayout.h -o debug\moc_vl

g++ -c -g -frtti -fexceptions -mtthreads -Wall -DUNICODE -DQT_LARGEFILE_SUPPORT -DQT_NEEDS_QMAI

programs/Qt/include/QtCore" -I".../programs/Qt/include/QtCore" -I".../programs/Qt/include/QtGui" -I".../
include" -I"" -I"c:/yunglu/programs/Qt/include/ActiveQt" -I"debug
programs/Qt/mkspecs/win32-g++" -o debug\moc_vlayout.o debug\moc_vlayout.o -L"c:/yunglu/programs/Qt/lib" -L"c:/yunglu/programs/Qt/lo
C:\yunglu\workspace\VLayout\debug\VLayout.exe
C:\yunglu\workspace\VLayout
#ifndef VLAYOUT_H
#define VLAYOUT_H
#include <QtGui>

class VLayout : public QDialog {
  Q_OBJECT

public:
  VLayout();

private:
  void createMenu();
  void createVerticalGroupBox();
  enum (NumButtons = 4);
  QMenuBar *menuBar;
  QGroupBox *verticalGroupBox;
  QPushButton *buttons[NumButtons];
  QMenu *fileMenu;
  QAction *exitAction;
};

#endif /*VLAYOUT_H */
```cpp
#include <QLayout>
#include "vlayout.h"

VLayout::VLayout() {
    createMenu();
    createVerticalGroupBox();
    QVBoxLayout *mainLayout = new QVBoxLayout;
    mainLayout->setMenuBar(menuBar);
    mainLayout->addWidget(vertialGroupBox);
    setLayout(mainLayout);
    setWindowTitle(tr("Vertical Layout"));
}

void VLayout::createMenu() {
    menuBar = new QMenu;
    fileMenu = new QMenu(tr("File"), this);
    exitAction = fileMenu->addAction(tr("Exit"));
    menuBar->addMenu(fileMenu);
    connect(exitAction, SIGNAL(triggered()), this, SLOT(accept()));
}

void VLayout::createVerticalGroupBox() {
    verticalGroupBox = new QGroupBox(tr("Vertical Layout"));
    QVBoxLayout *layout = new QVBoxLayout;
    for (int i = 0; i < NumButtons; ++i) {
        QPushButton buttons[i] = new QPushButton(tr("Button %1"), arg(i + 1));
        layout->addWidget(buttons[i]);
    }
    verticalGroupBox->setLayout(layout);
}
```
Horizontal Layout
```cpp
#include <QGui>
#include "vlayout.h"

VLayout::VLayout() {
    createMenu();
    createHorizontalGroupBox();
    QBoxLayout *mainLayout = new QBoxLayout;
    mainLayout->setMenuBar(menuBar);
    mainLayout->addWidget(horizontalGroupBox);
    setLayout(mainLayout);
    setWindowTitle(tr("Horizontal Layout"));
}

void VLayout::createMenu() {
    menuBar = new QMenuBar;
    fileMenu = new QMenu(tr("File"), this);
    exitAction = fileMenu->addAction(tr("Exit"));
    menuBar->addMenu(fileMenu);
    connect(exitAction, SIGNAL(triggered()), this, SLOT(accept()));
}

void VLayout::createHorizontalGroupBox() {
    horizontalGroupBox = new QGroupBox(tr("Horizontal layout"));
    QBoxLayout *layout = new QBoxLayout;
    for (int i = 0; i < NumButtons; ++i) {
        buttons[i] = new QPushButton(tr("Button %1").arg(i + 1));
        layout->addWidget(buttons[i]);
    }
    horizontalGroupBox->setLayout(layout);
}
```
Grid Layout
Calculator Example

Files:

- widgets/calculator/button.cpp
- widgets/calculator/button.h
- widgets/calculator/calculator.cpp
- widgets/calculator/calculator.h
- widgets/calculator/main.cpp

The example shows how to use signals and slots to implement the functionality of a calculator widget, and how to use QGridLayout to place child widgets in a grid.
The example shows how to use signals and slots to implement the functionality of a calculator widget, and how to use QGridLayout to place child widgets in a grid.

The example consists of two classes:
QGridLayout *mainLayout = new QGridLayout;
mainLayout->setFixedSize(QLayout::SetFixedSize);

mainLayout->addWidget(display, 0, 0, 1, 6);
mainLayout->addWidget(backspaceButton, 1, 0, 1, 2);
mainLayout->addWidget(clearButton, 1, 2, 1, 2);
mainLayout->addWidget(clearAllButton, 1, 4, 1, 2);

mainLayout->addWidget(clearMemoryButton, 2, 0);
mainLayout->addWidget(readMemoryButton, 3, 0);
mainLayout->addWidget(setMemoryButton, 4, 0);
mainLayout->addWidget(addToMemoryButton, 5, 0);

for (int i = 1; i < NumDigitButtons; ++i) {
    int row = ((9 - i) / 3) + 2;
    int column = ((i - 1) % 3) + 1;
    mainLayout->addWidget(digitButtons[i], row, column);
}

mainLayout->addWidget(digitButtons[0], 5, 1);
mainLayout->addWidget(pointButton, 5, 2);
mainLayout->addWidget(changeSignButton, 5, 3);
Public Functions

- `QGridLayout ( QWidget * parent )`
- `QGridLayout ()`
- `~QGridLayout ()`
- void `addItem ( QLayoutItem * item, int row, int column, int rowspan = 1, int columnSpan = 1, Qt::Alignment alignment = 0 )`
- void `addLayout ( QLayout * layout, int row, int column, Qt::Alignment alignment = 0 )`
- void `addLayout ( QLayout * layout, int row, int column, int rowspan, int columnSpan, Qt::Alignment alignment = 0 )`
- void `addWidget ( QWidget * widget, int row, int column, Qt::Alignment alignment = 0 )`
- void `addWidget ( QWidget * widget, int fromRow, int fromColumn, int rowSpan, int columnSpan, Qt::Alignment alignment = 0 )`
- `QRect cellRect ( int row, int column ) const`
- `int columnCount () const`
- `int columnMinimumWidth ( int column ) const`
- `int columnStretch ( int column ) const`
- void `getGeometry ( int index, int * row, int * column, int * rowSpan, int * columnSpan )`
- `int horizontalSpacing () const`
Combination of Layouts
Basic Layouts Example

Files:

- layouts/basiclayouts/dialog.cpp
- layouts/basiclayouts/dialog.h
- layouts/basiclayouts/main.cpp

The Basic Layouts example shows how to use the standard layout managers that are available in Qt: QBoxLayout and QGridLayout.
The **QBoxLayout** class lines up widgets horizontally or vertically. **QHBoxLayout** and **QVBoxLayout** are convenience subclasses of **QBoxLayout**.
buttonBox = new QDialogButtonBox(QDialogButtonBox::Ok
                               | QDialogButtonBox::Cancel);

connect(buttonBox, SIGNAL(accepted()), this, SLOT(accept()));
connect(buttonBox, SIGNAL(rejected()), this, SLOT(reject()));

QVBoxLayout *mainLayout = new QVBoxLayout;
mainLayout->setMenuBar(menuBar);
mainLayout->addWidget(horizontalGroupBox);
mainLayout->addWidget(gridGroupBox);
mainLayout->addWidget(bigEditor);
mainLayout->addWidget(buttonBox);
setLayout(mainLayout);

setTitle(tr("Basic Layouts"));

void Dialog::createMenu()
{
    menuBar = new QMenuBar;

    QMainWindow fileMenu = new QMenu(tr("&File"), this);
    exitAction = fileMenu->addAction(tr("E&xit"));
}
connect(exitAction, SIGNAL(triggered()), this, SLOT(accept()));

void Dialog::createHorizontalGroupBox()
{
    horizontalGroupBox = new QGroupBox(tr("Horizontal layout"));
    QBoxLayout *layout = new QBoxLayout;

    for (int i = 0; i < NumButtons; ++i) {
        buttons[i] = new QPushButton(tr("Button %1").arg(i + 1));
        layout->addWidget(buttons[i]);
    }
    horizontalGroupBox->setLayout(layout);
}

void Dialog::createGridGroupBox()
{
    gridGroupBox = new QGroupBox(tr("Grid layout"));
    QGridLayout *layout = new QGridLayout;

    for (int i = 0; i < NumGridRows; ++i) {
        labels[i] = new QLabel(tr("Line %1:").arg(i + 1));
        layout->addWidget(labels[i], i, 0);
    }
    gridGroupBox->setLayout(layout);
}