

CS2110. GUI: Listening to Events

Also

Example of Stepwise Refinement
and
Anonymous classes

Download the demo zip file from
course website and look at the demos
of GUI things: sliders, scroll bars,
combobox listener, etc

1

Developing the prelim string problem

*/** s is a sequence of words with each pair of words separated
* by one or more blanks. Return a list of the Pig-Latin
* translations of the words, with no duplicates */*
public static ArrayList<String> m(String s) {

A few points to be constantly aware of

- Focus on one thing at a time.
- Use abstraction.
- Keep things simple
- Avoid case analysis where possible
- Don't introduce a variable unless you need it.

Word: a sequence of ≥ 1 lowercase letters

2

Use a loop to process string s

*/** s is a sequence of words with each pair of words separated
* by one or more blanks. Return a list of the Pig-Latin
* translations of the words, with no duplicates */*
public static ArrayList<String> m(String s) {

Which kind of loop?

```
for (int k= 0; k < s.length(); k= k+1) {}
```

```
int k= 0;  
while ( ) {}
```

Problem is stated in
terms of a sequence of
words. Therefore, the
loop is best written
with each iteration
processing one word.
For-loop leads to
disaster!

Word: a sequence of ≥ 1 lowercase letters

3

Use abstraction to allow focus on one thing

For now, forget about what to do with each word and
concentrate on just "processing" each word, using a loop.
Later, figure out what "processing" means.

```
while ( ) {  
    Find first word in s, process it, and remove it from s  
}
```

But: what about blanks before and after first word. Best if
we get rid of blanks before the word.

use `s= s.trim();` // don't know about it? Write a loop

Word: a sequence of ≥ 1 lowercase letters

4

Outline the while-loop

```
s= s.trim();
```

```
// inv: All processed words have been removed from s,  
// and s has no surrounding blanks
```

```
while ( s.length() > 0 ) {  
    Process first word of s and remove it from s
```

```
}
```

Word: a sequence of ≥ 1 lowercase letters

5

Outline the while-loop

```
s= s.trim();  
// inv: All processed words have been removed from s,  
// and s has no surrounding blanks
```

```
while (s.length() > 0) {  
    // Process first word of s and remove it from s
```

```
    int k= s.indexOf(" "); // # of chars in first word
```

```
    if (k < 0) k= s.length();
```

```
    String word= s.substring(0, k);
```

```
    s= s.substring(k).trim();
```

```
    Process word
```

```
}
```

Problem: the
last word has
no blank after
it!

Whenever you write
`b[k]` or `s.charAt[k]` or `s.substring(h,k)` or `list.get(k)`, etc.
ask yourself whether index `k` is in bounds.

6

Stepwise refinement

```

s= s.trim();
// inv: All processed words have been removed from s,
// and s has no surrounding blanks
while (s.length() > 0) {
    // Get first word of s into word and remove it from s
    int k= s.indexOf(" ");
    if (k < 0) k= s.length();
    String word= s.substring(0, k);
    s= s.substring(k).trim();

    // Process word
    ...
}

```

Now we can work on processing a word, which has to do with constructing the ArrayList and adding the Pig Latin of non-duplicate words.

Stepwise refinement: Take one (small) step at a time. Focus on the most important one at the moment.

7

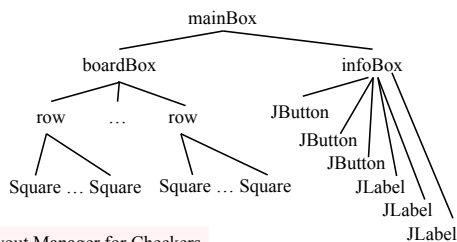
Stepwise refinement

Stepwise refinement: Take one (small) step at a time. **Focus on the most important one at the moment.**

Examples of steps:

- Implement an English statement by a sequence of statements
- Decide on using a loop
- Stub in a new method (Specification and header, with empty body) because of duplicate code or to remove complexity
- Add a local variable or field
- Replace an English statement by an equivalent Java statement

8



Layout Manager for Checkers game has to process a tree

pack(): Traverse the tree, determining the space required for each component

boardBox: vertical Box
row: horizontal Box
Square: Canvas or JPanel
infoBox: vertical Box

9

Listening to events: mouse click, mouse movement into or out of a window, a keystroke, etc.

- An **event** is a mouse click, a mouse movement into or out of a window, a keystroke, etc.
- To be able to “listen to” a kind of event, you have to:
 1. Have some class C implement an interface IN that is connected with the event.
 2. In class C, override methods required by interface IN; these methods are generally called when the event happens.
 3. Register an object of class C as a *listener* for the event. That object’s methods will be called when event happens.

We show you how to do this for clicks on buttons, clicks on components, and keystrokes.

10

What is a JButton?

Instance: associated with a “button” on the GUI, which can be clicked to do something

```

jb1= new JButton() // jb1 has no text on it
jb2= new JButton("first") // jb2 has label "first" on it
jb2.isEnabled() // true iff a click on button can be
// detected
jb2.setEnabled(b); // Set enabled property
jb2.addActionListener(object); // object must have a method,
// which is called when button jb2 clicked (next page)

```

At least 100 more methods; these are most important

JButton is in package javax.swing

11

Listening to a JButton

1. Implement interface ActionListener:

```

public class C extends JFrame implements
ActionListener {
    ...
}

```
2. In class C override actionPerformed, which is to be called when button is clicked:

```

/** Process click of button */
public void actionPerformed(ActionEvent e) {
    ...
}

```
3. Add an instance of class C an “action listener” for button:

```

button.addActionListener(this);

```

12

```

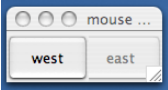
/** Object has two buttons. Exactly one is enabled. */
class ButtonDemo1 extends JFrame
    implements ActionListener {
    /** Class inv: exactly one of eastB, westB is enabled */
    JButton westB= new JButton("west");
    JButton eastB= new JButton("east");

    public ButtonDemo1(String t) {
        super(t);
        Container cp= getContentPane();
        cp.add(westB, BorderLayout.WEST);
        cp.add(eastB, BorderLayout.EAST);
        westB.setEnabled(false);
        eastB.setEnabled(true);
        westB.addActionListener(this);
        eastB.addActionListener(this);
        pack(); setVisible(true);
    }

    public void actionPerformed
        (ActionEvent e) {
        boolean b= eastB.isEnabled();
        eastB.setEnabled(!b);
        westB.setEnabled(b);
    }
}

```

red: listening
blue: placing




Listening to a Button

13

A JPanel that is painted

- The JFrame content pane has a JPanel in its CENTER and a "reset" button in its SOUTH.
- The JPanel has a horizontal box b, which contains two vertical Boxes.
- Each vertical Box contains two instances of class Square.
- Click a Square that has no pink circle, and a pink circle is drawn. Click a square that has a pink circle, and the pink circle disappears. Click the reset button and all pink circles disappear.
- This GUI has to listen to:
 - a click on Button reset
 - a click on a Square (a Box)

these are different kinds of events, and they need different listener methods




14

```

/** Instance: JPanel of size (WIDTH, HEIGHT).
    Green or red: */
public class Square extends JPanel {
    public static final int HEIGHT= 70;
    public static final int WIDTH= 70;
    private int x, y; // Panel is at (x, y)
    private boolean hasDisk= false;
    /** Const: square at (x, y). Red/green? Parity of x+y. */
    public Square(int x, int y) {
        this.x= x; this.y= y;
        setPreferredSize(new Dimension(WIDTH,HEIGHT));
    }
    /** Complement the "has pink disk" property */
    public void complementDisk() {
        hasDisk= ! hasDisk;
        repaint(); // Ask the system to repaint the square
    }
}

```

Class Square



continued on later

15

Class Graphics

An object of abstract class Graphics has methods to draw on a component (e.g. on a JPanel, or canvas).

Major methods:

```

drawString("abc", 20, 30); drawLine(x1, y1, x2, y2);
drawRect(x, y, width, height); fillRect(x, y, width, height);
drawOval(x, y, width, height); fillOval(x, y, width, height);
setColor(Color.red); getColor();
setFont(Font f);

```

More methods

You won't create an object of Graphics; you will be given one to use when you want to paint a component

Graphics is in package java.awt

16

continuation of class Square

```

/** paint this square using g. System calls
    paint whenever square has to be redrawn.*/
public void paint(Graphics g) {
    if ((x+y)%2 == 0) g.setColor(Color.green);
    else g.setColor(Color.red);
    g.fillRect(0, 0, WIDTH-1, HEIGHT-1);
    if (hasDisk) {
        g.setColor(Color.pink);
        g.fillOval(7, 7, WIDTH-14, HEIGHT-14);
    }
    g.setColor(Color.black);
    g.drawRect(0, 0, WIDTH-1, HEIGHT-1);
    g.drawString(""+x+" "+y+",", 10, 5+HEIGHT/2);
}
}


```

Class Square

```

/** Remove pink disk (if present) */
public void clearDisk() {
    hasDisk= false;
    // Ask system to repaint square
    repaint();
}

```



17

Listen to mouse event

(click, press, release, enter, leave on a component)

```

public interface MouseListener {
    void mouseClicked(MouseEvent e);
    void mouseEntered(MouseEvent e);
    void mouseExited(MouseEvent e);
    void mousePressed(MouseEvent e);
    void mouseReleased(MouseEvent e);
}

```

In package java.awt.event

Having to write all of these in a class that implements MouseListener, even though you don't want to use all of them, can be a pain. So, a class is provided that implements them in a painless.

18

Listen to mouse event (click, press, release, enter, leave on a component)

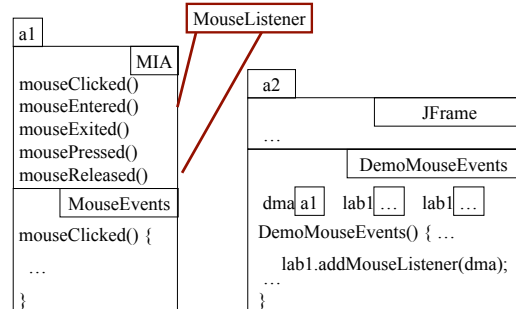
In package java.swing.event

```
public class MouseInputAdaptor
    implements MouseListener, MouseInputListener {
    public void mouseClicked(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
    ... others ...
}
```

So, just write a subclass of MouseInputAdaptor and
} override only the methods appropriate for the application

19

javax.swing.event.MouseInputAdapter
implements MouseListener

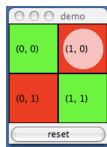


20

A class that listens to a mouseclick in a Square

import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;

red: listening
blue: placing



/** Contains a method that responds to a
mouse click in a Square */

```
public class MouseEvents
    extends MouseInputAdapter {
    // Complement "has pink disk" property
    public void mouseClicked(MouseEvent e) {
        Object ob = e.getSource();
        if (ob instanceof Square) {
            ((Square)ob).complementDisk();
        }
    }
}
```

This class has several methods
(that do nothing) that process
mouse events:
mouse click
mouse press
mouse release
mouse enters component
mouse leaves component
mouse dragged beginning in
component

Our class overrides only the method that processes mouse clicks

21

```
public class MD2 extends JFrame
    implements ActionListener {
    Box b= new Box(...X_AXIS);
    Box leftC= new Box(...Y_AXIS);
    Square b00, b01= new squares;
    Box riteC= new Box(..Y_AXIS);
    Square b10, b11= new squares;
    JButton jb= new JButton("reset");
```

jb.addActionListener(this);
b00.addMouseListener(me);
b01.addMouseListener(me);
b10.addMouseListener(me);
b11.addMouseListener(me);

```
    public void actionPerformed (
        ActionEvent e) {
        call clearDisk() for  
b00, b01, b10, b11
    }
    MouseEvents me=
    new MouseEvents();
```

/** Constructor: ... */

```
public MouseDemo2() {
    super(t);
    place components on content pane;
    pack, make unresizable, visible;
```

red: listening
blue: placing



Class MouseDemo2

22

Listening to the keyboard

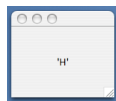
import java.awt.*; import java.awt.event.*; import javax.swing.*;

```
public class AllCaps extends KeyAdapter {
    JFrame capsFrame= new JFrame();
    JLabel capsLabel= new JLabel();
```

red: listening
blue: placing

```
public AllCaps() {
    capsLabel.setHorizontalAlignment(SwingConstants.CENTER);
    capsLabel.setText("");
    capsFrame.setSize(200,200);
    Container c= capsFrame.getContentPane();
    c.add(capsLabel);
    capsFrame.addKeyListener(this);
    capsFrame.show();
}
```

```
public void keyPressed (KeyEvent e) {
    char typedChar= e.getKeyChar();
    capsLabel.setText("" + typedChar + "" .toUpperCase());
}
```



23

```
public class BDemo3 extends JFrame implements ActionListener {
    private JButton wButt, eButt ...;
    public ButtonDemo3() {
        Add buttons to content pane, enable  
ne, disable the other
        wButt.addActionListener(this);
        eButt.addActionListener(new BeListener()); }
    public void actionPerformed(ActionEvent e) {
        boolean b= eButt.isEnabled();
        eButt.setEnabled(!b); wButt.setEnabled(b); }
}
```

Have a different
listener for each
button

```
A listener for eastButt
class BeListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        boolean b= eButt.isEnabled();
        eButt.setEnabled(!b); wButt.setEnabled(b);
    }
}
```

Doesn't work!
Can't
reference
eButt, wButt

24

