javax.swing....
java.awt.....

Swings and Inner Classes

Hsuan-Tien Lin

Department of CSIE, NTU

OOP Class, June 14-15, 2010

Getting Started with GUI

```
1  import javax.swing.*;
2  public class GUIDemo{
3    public static void main(String[] argv){
4      JFrame frm = new JFrame();
5      frm.setVisible(true);
6    }
7  }
```

Setting Size of Frame

```
1  import javax.swing.*;
2  public class GUIDemo{
3    public static void main(String[] argv){
4      JFrame frm = new JFrame();
5      frm.setSize(640, 480);
6      frm.setVisible(true);
7    }
8  }
```

Inheriting from JFrame

```
1  import javax.swing.*;
2  public class GUIDemo{
3    public static void main(String[] argv){
4      JFrame frm = new MyFrame("lalala");
5    }
6  }
7  class MyFrame extends JFrame{
8    MyFrame(String title){
9      super(title);
10     setSize(640, 480);
11     setVisible(true);
12   }
13 }
```

Adding Components to JFrame

```
1   class MyFrame extends JFrame{
2     JLabel lbl;
3     MyFrame(String title){
4       super(title);
5       setSize(640, 480);
6       lbl = new JLabel("undefined");
7       add(lbl);
8       setVisible(true);
9     }
10  }
```

Using CounterLabel as both JLabel and Runnable

```
1  import java.awt.*;
2  class MyFrame extends JFrame{
3    MyFrame(String title){
4      super(title);
5      setSize(640, 480); setVisible(true);
6      for(int i=0;i<10;i++) new CounterLabel(this);
7      setLayout(new FlowLayout());
8    }
9  }
10 class CounterLabel extends JLabel implements Runnable{
11   int count = 0;
12   CounterLabel(Container container){
13     container.add(this);
14     (new Thread(this)).start();
15   }
16   public void run(){
17     while(true){
18       try{ Thread.sleep(1000); }
19       catch(InterruptedException e){}
20       count++;
21       this.setText("" + count);
22     }
23   }
```

Writing CounterLabel by Inner Thread Class

```
1   class CounterLabel extends JLabel{
2     int count = 0;
3     CounterLabel(Container container){
4       container.add(this);
5       (new CounterThread()).start();
6     }
7     class CounterThread extends Thread{
8       public void run(){
9         while(true){
10          try{
11            Thread.sleep(1000);
12          }
13          catch(InterruptedException e){}
14          count++;
15          setText("" + count);
16        }
17      }
18    }
19  }
```

Writing CounterLabel by Anonymous Thread Class

```
1   class CounterLabel extends JLabel{
2     int count = 0;
3     CounterLabel(Container container){
4       container.add(this);
5       (new Thread(){
6         public void run(){
7           while(true){
8             try{
9               Thread.sleep(1000);
10            }
11            catch(InterruptedException e){}
12            count++;
13            setText("" + count);
14          }
15        }
16      }).start();
17    }
18  }
```

Inner Class

- easily share (private) data of outer-class instances —usually as instance-specific, private utility class
- often also do-able with interfaces, but clearer if used appropriately
- anonymous class: a extreme if the specific class is only needed "temporarily"

A CounterLabel that Stops

```
1   class CounterLabel extends JLabel{
2     int count = 0;
3     boolean stopping = true;
4     CounterLabel(Container container){
5       container.add(this);
6       (new CounterThread()).start();
7     }
8     void start(){ stopping = false; }
9     void stop(){ stopping = true; }
10    class CounterThread extends Thread{
11      public void run(){
12        while(true){
13          try{
14            Thread.sleep(1000);
15          }
16          catch(InterruptedException e){}
17          if (!stopping){
18            count++;
19            setText("" + count);
20          }
21        }
22      }
23    }
```

A CounterLabel that Comes with a Control Button

```
1   class CounterLabel extends JLabel{
2     ControlButton but = new ControlButton();
3     CounterLabel(Container container){
4       container.add(this);
5       container.add(but);
6       (new CounterThread()).start();
7     }
8     void start(){ stopping = false; but.resetText(); }
9     void stop(){ stopping = true; but.resetText(); }
10    class ControlButton extends JButton{
11      ControlButton(){ resetText(); }
12      void resetText(){ setText(stopping ? "Start" : "Stop"); }
13    }
14  }
```

A Control Button that Can Be Pressed

```
1   import java.awt.event.*;
2   class CounterLabel extends JLabel{
3     class ControlButton extends JButton{
4       ControlButton(){
5         resetText();
6         addActionListener(new ActionListener(){
7           public void actionPerformed(ActionEvent e){
8             stopping = !stopping;
9             resetText();
10          }
11        });
12      }
13      void resetText(){ setText(stopping ? "Start" : "Stop"); }
14    }
15  }
```

Event Driven Programming

- throw-exception-catch
  - event: any Throwable
  - event generator: throw
  - event handler: catch
  - handler registration: write after try
- component-action-listener
  - event: ActionEvent
  - event generator: some component
  - event handler: ActionListener
  - handler registration: addActionListener
- component-mouse-listener
  - event: MouseEvent
  - event generator: some component
  - event handler: MouseListener
  - handler registration: addMouseListener
- .....

Event Driven Programming

- throw-exception-catch
  - event: any Throwable
  - event generator: throw
  - event handler: catch
  - handler registration: write after try
- component-action-listener
  - event: ActionEvent
  - event generator: some component
  - event handler: ActionListener
  - handler registration: addActionListener
- component-mouse-listener
  - event: MouseEvent
  - event generator: some component
  - event handler: MouseListener
  - handler registration: addMouseListener
- .....

Event Driven Programming

- throw-exception-catch
  - event: any Throwable
  - event generator: throw
  - event handler: catch
  - handler registration: write after try
- component-action-listener
  - event: ActionEvent
  - event generator: some component
  - event handler: ActionListener
  - handler registration: addActionListener
- component-mouse-listener
  - event: MouseEvent
  - event generator: some component
  - event handler: MouseListener
  - handler registration: addMouseListener
- .....