//BinaryObserver

public class BinaryObserver extends Observer{

public BinaryObserver(Subject subject){

this.subject = subject;

this.subject.attach(this);

}

@Override

public String update() {

return "Binary String: " + Integer.toBinaryString( subject.getState()) +"\n";

}

}

// HexaObserver

public class HexaObserver extends Observer{

public HexaObserver(Subject subject){

this.subject = subject;

this.subject.attach(this);

}

@Override

public String update() {

return "Hex String: " + Integer.toHexString( subject.getState() ).toUpperCase() + "\n";

}

}

// OctalObserver

public class OctalObserver extends Observer{

public OctalObserver(Subject subject){

this.subject = subject;

this.subject.attach(this);

}

@Override

public String update() {

return "Octal String: " + Integer.toOctalString( subject.getState() ) +"\n";

}

}

//Observer

public abstract class Observer {

protected Subject subject;

public abstract String update();

}

//Subject

import java.util.ArrayList;

import java.util.List;

public class Subject {

private List<Observer> observers = new ArrayList<Observer>();

private int state;

public int getState() {

return state;

}

public void setState(int state) {

this.state = state;

notifyAllObservers();

}

public void attach(Observer observer){

observers.add(observer);

}

public void notifyAllObservers(){

for (Observer observer : observers) {

observer.update();

}

}

}

//ObserverPatternDemo

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

public class ObserverPatternDemo {

private Frame fm;

private Label l1,s1,s2,s3;

private TextField tf;

private Button btn;

ObserverPatternDemo(){

prepareGUI();

}

public void prepareGUI(){

fm = new Frame("Observer Pattern");

fm.setBounds(450,200,350, 350);

fm.setLayout(null);

fm.setBackground(Color.pink);

fm.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent we){

System.exit(0);}});

Font myFont = new Font("Arial",Font.PLAIN,17);

l1 = new Label(" State: ");

l1.setAlignment(Label.RIGHT);

l1.setFont(myFont);

l1.setBounds(20, 50,100,50);

tf = new TextField();

tf.setBounds(150, 50, 50, 40);

btn = new Button("Get all");

btn.setActionCommand("C");

btn.setFont(myFont);

btn.setBounds(100, 100, 100, 40);

s1 = new Label("");

s1.setBounds(100, 150, 200, 50);

s1.setFont(myFont);

s2 = new Label("");

s2.setBounds(100, 200, 200, 50);

s2.setFont(myFont);

s3 = new Label("");

s3.setBounds(100, 250, 200, 50);

s3.setFont(myFont);

fm.add(l1);

fm.add(tf);

fm.add(btn);

fm.add(s1);

fm.add(s2);

fm.add(s3);

fm.setVisible(true);

ActionListener al = new MyActionListener();

btn.addActionListener(al);

}

private class MyActionListener implements ActionListener{

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

if( command.equals( "C" )){

int v = Integer.parseInt(tf.getText());

Subject subject = new Subject();

HexaObserver h = new HexaObserver(subject);

OctalObserver o = new OctalObserver(subject);

BinaryObserver b = new BinaryObserver(subject);

subject.setState(v);

s1.setText(h.update());

s2.setText(o.update());

s3.setText(b.update());

}

}

}

public static void main(String[] args) {

ObserverPatternDemo d = new ObserverPatternDemo();

}

}

