Network Computing Introduction and Course Administrative Details

Instructor: *Louise Moser*

TA: *Yung-Ting Chuang*

Quarter: Spring 2012

Contact Information

- TA: Yung-Ting Chuang
- Email: ece155b.s12@gmail.com (PREFFERED)
- Office Hour: Wed from 2-3:30 pm & Fri from 11-12:30 pm
- Office: ECI Lab Harold Frank Hall
- Visit class web for all the handouts & project templates: http://www.ece.ucsb.edu/courses/ECE155/155B_S12Moser/

Grading

- There will be 6 projects in this course(70% of the grading):
  - 1st project: 1 week
  - 2nd project: 2 weeks
  - 3rd project: 1 week
  - 4th project: 1 weeks
  - 5th project: 2 weeks
  - 6th project: 1 week
- 15% from the class and discussion participation
- 15% from the term paper (professor Moser will discuss this toward the end of the quarter)

How projects are graded

- Do demo to me on the due date (which should be on Fridays) between my office hour (11-12:30 pm)
- Contact me for the actual demo time on Friday
- I will have a list which ask for the certain functions/tasks(and how many points they worth), and I will ask you to demo them to me.
- You need to have projects run correctly on the linux machines in ECI lab.
- Email me your codes with your short paper describing what you have done(in rar/tar format) before your demo time.

How projects are graded(cont)

- Late work is accepted for 10 points deducted.
- Programs will be worth 90% of the grading, and short term papers will be worth 10% of the grading.
- There will be extra credits for the 3rd project and also toward the end of the quarter. If you implement something cool for a particular project, show me during the demo and I might consider them for the extra credits as well ;)

Overall Project

Patient

Login patient
Request appointment
View prescription status

Doctor

Create patient profile
Create patient diagnosis
Create prescription
Submit prescription

Pharmacy

Inform prescription status

Projects Overview

- **Project 1: Applet Programming**
- **Project 2: Socket Programming**
- **Project 3: Remote Method Invocation**
- **Project 4: Java Database Connectivity**
- **Project 5: Java Servlets**
- **Project 6: Web Services**

Network Computing Lecture on IDEs, Swing, Applet, and XML

Instructor: *Louise Moser*

TA: *Yung-Ting Chuang*

Quarter: Spring 2012

Network Computing
Lecture 1.1: XML

Instructor: *Louise Moser*

TA: *Yung-Ting Chuang*

Quarter: Spring 2012

What is XML?

- XML stands for Extensible Markup Language
- XML is a ***markup language*** like html
- XML was designed to ***describe data***
- XML tags are not predefined; you must ***define your own tags***

XML versus HTML

- XML is not a replacement for HTML
- XML and HTML were designed with different goals:
  - XML: Describe the data and focus on *what the data* is
  - HTML: Display the data and focus on *how it looks*
- HTML is about displaying information, while XML is about describing information
- XML DOES NOT do anything, since it was not designed to do anything
- XML was created to structure, store and send information.

Example XML file for project1

```
<?xml version='1.0' ?>
<Doctor>
   <DoctorName>Yung-Ting</DoctorName>
   <DoctorLastName>Chuang</DoctorLastName>
   <Patients>
      <Patient>
         <PatientSSN>123-45-2355</PatientSSN>
         <PatientFirstName>Po-Yo</PatientFirstName>
         <Notes></Notes>
      </Patient>
      <Patient>
         <PatientSSN>236-34-0292</PatientSSN>
         <PatientFirstName>Shu-Ha</PatientFirstName>
         <Notes>
            <Note>
               <Author>Yung-Ting Chuang</Author>
               <Date>2011/3/24</Date>
               <Content>Has foot ingury</Content>
            </Note>
         </Notes>
      </Patient>
   </Patients>
</Doctor>
```

XML versus HTML (continued)

- XML is free and extensible
- XML allows you to define your own tags and structure of XML documents. XML tags are not predefined; you must "invent" your own tags. The XML tags are not defined by the XML standard.
- In contrast, tags and the structure of HTML documents are predefined by the HTML standard. The author of an HTML document can use only tags that are predefined in the HTML standard, such as <p>, <h1>, etc.
- It is most likely that XML will be used to describe the data, while HTML will be used to format and display the same data.
- We strongly believe that XML will be as important to the Web of the future as HTML has been to the Web of the past and that XML will be the most common format for data manipulation and transmission.

XML (continued)

- The first line of an XML file describes the XML version. We don't need to worry about this, just assume it's always **<?xml version="1.0" encoding="ISO-8859-1"?>**
- XML elements have opening and closing tags
- XML tags are **case-sensitive**
- XML tags must be **properly nested**
- Example which XML is not properly nested:

```
<email>
        <to>Yung-Ting Chuang</to>
        <from>Professor Moser</from>
        <body>
                <message>Your discussion notes need some work!</message>
                <signature>--Professor Moser
        </body>
                </signature>
```

- *XML attributes* are optional parts of an opening tag. Example:

<user name="Yung-Ting" major="CE">TA</user>

How to write XML in project1

- In project1, when the doctor need to retrieve his/her patient information, we need to use the toXML() method to write the information in XML format.

- This method will be similar to the *toString()* method, but will return the String in XML format.

Example for Doctor code:

Doctor class

```
public class Doctor
{
    public String name = "MyName";
    public String lastname = "MyLastName";

    Vector <Patient> patients; // doctor keep patient information in a vector

    public String toXML()
    {
        String xml = "<Doctor>";
        xml += "<DoctorName>"+name+"</DoctorName>";
        xml += "<DoctorLastName>"+lastname+"</DoctorLastName>";
        xml += "<Patients>";
        for (int i = 0; i < patients.size(); i++)
            xml += patients.elementAt(i).toXML(); // add xml code for each patient
        xml += "</Patients>";
        xml += "</Doctor>";

        return xml;
    }
}
```

How to read XML in project1

- We use import javax.xml.parsers and org.xml.sax packages to read XML data. Example:
  SAXParserFactory spf = SAXParserFactory.newInstance()
  SAXParser sp = spf.newSAXParser()

- We first create a parser object. We need to provide a parser for a HandlerBase object which has the following methods for us to overwrite:
  - **void startElement**(*String name, AttributeList list*)
  - **void endElement**(*String name*)

- When the parser recognizes a tag opening, it calls the startElement method of the handler and, similarly, the endElement for a tag closing. If there are attributes, this will be provided in the attribute list. For our project we don't need this, we will just use the *name* parameter.

- In the ***startElement*** method, if the object name such as Patient, Note, orDoctor , we create an object for it and insert it into the vector.

- In the ***endElement*** method, we set the values of Objects. If it is the object name such as Patient, Note, Doctor we do not need to do anything about them.

- It is important that the tag names that you use when writing and when parsing are the same. Otherwise, you will have trouble reading what you have written.

Example of XMLParser reading XML file we had in slide 5:

```
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import javax.xml.parsers.*;

public class XMLParser extends DefaultHandler
{
	public Doctor doctor;
	private Patient patient;
	private Note note;
	private StringBuffer accumulator = new StringBuffer();		// Buffer used during parsing

	public XMLParser()
	{
		doctor = new Doctor();
	}
	public void characters(char[] buffer, int start, int length)
	{
		accumulator.append(buffer, start, length);
	}

	//Each time a new tag is opened, this method is called
	public void startElement(String uri,String lname, String name, Attributes attributes)
	{
```

Example of XMLParser (continued):

```
        // - - - - - - - - - - - - - - - - - - - - -
        //              New Note starts
        if(name.equals ("Note"))
        {
                        note = new Note();              // we will set values of this later
                        patient.addNote(note);          // add to patient
        }
        // - - - - - - - - - - - - - - - - - - - - -
        //              New Patient starts
        else if(name.equals ("Patient"))
        {
                        patient = new Patient();        // we will set values of patient later
                        doctor.addPatient(patient);     // add to doctor
        }
        // - - - - - - - - - - - - - - - - - - - - -
        // Reset accumulator
        accumulator.setLength(0);
}
```

Example of XMLParser (continued):

```
    public void endElement(String uri,String lname,String name)
    {
            String value = accumulator.toString();
            // - - - - - - - - - - - - - - - - - - - - -
            // Doctor information
            if(name.equals ("DoctorName"))
                    doctor.name= value;
            else if(name.equals ("DoctorLastName"))
                    doctor.lastname = value;
            // - - - - - - - - - - - - - - - - - - - - -
            // Patient information
                    else if(name.equalsIgnoreCase("PatientSSN"))
                            patient.ssn = value;
                    else if(name.equalsIgnoreCase("PatientFirstName"))
                            patient.name = value;
            // - - - - - - - - - - - - - - - - - - - - -
            // Note information
                    else if(name.equalsIgnoreCase("Author"))
                            note.author = value;
                    else if(name.equalsIgnoreCase("Date"))
                            note.date = value;
    }
}
```

Network Computing
Lecture 1.2: Swing

Instructor: *Louise Moser*

TA: *Yung-Ting Chuang*

Quarter: Spring 2012

Java Swing

- Java Swing is a useful set of tools for building simple graphical user interfaces (GUIs) that includes a number of basic control components
  - *JApplet*
  - *JFrame*
  - *JLabel*
  - *JButton*
- You must also know about two aspects of a GUI that add some complications
  - Layout Managers: gives position the basic control components
  - Event Listeners: tells your program when a user invokes one of the basic controls

Information about Swing

- To use swing, you need to do:

  import javax.swing.*;

- **To find out more information about Swing, go to:**

  http://www.apl.jhu.edu/~hall/java/Swing-Tutorial/

  http://www.thejavatutorial.com

  http://java.sun.com (the definitive reference for API's)

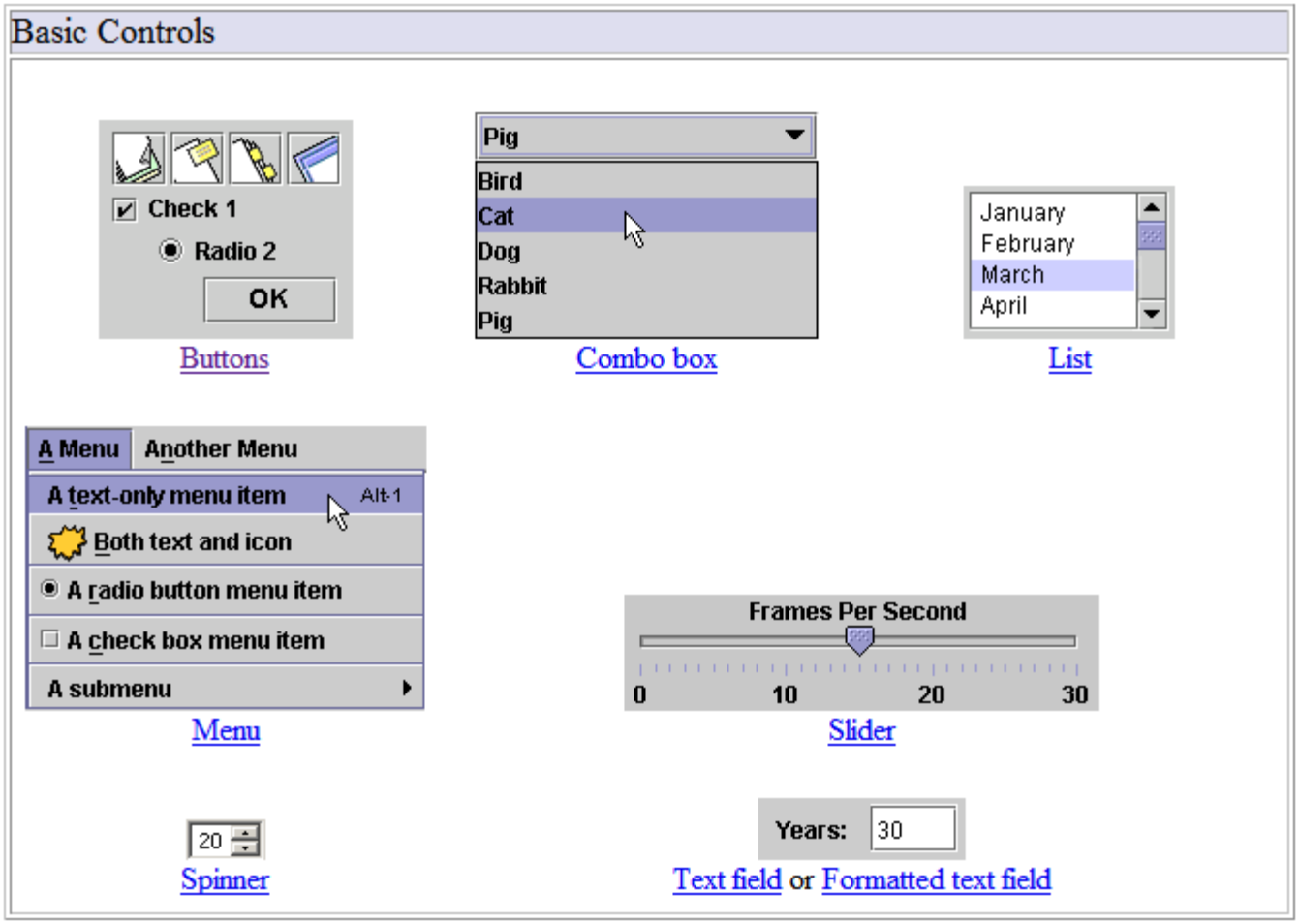
Swing Control Components

- Basic Compoments:
  a) *JApplet*
  b) *JFrame*
  c) *JButton*
  d) *JLabel*
  e) *JPanel*
  f) *JCheckbox*
  g) *JComboBox*
  h) *JList*
  i) *JMenu*
  j) *JScrollBar*
  k) *JTextArea*
  l) *JTextField*

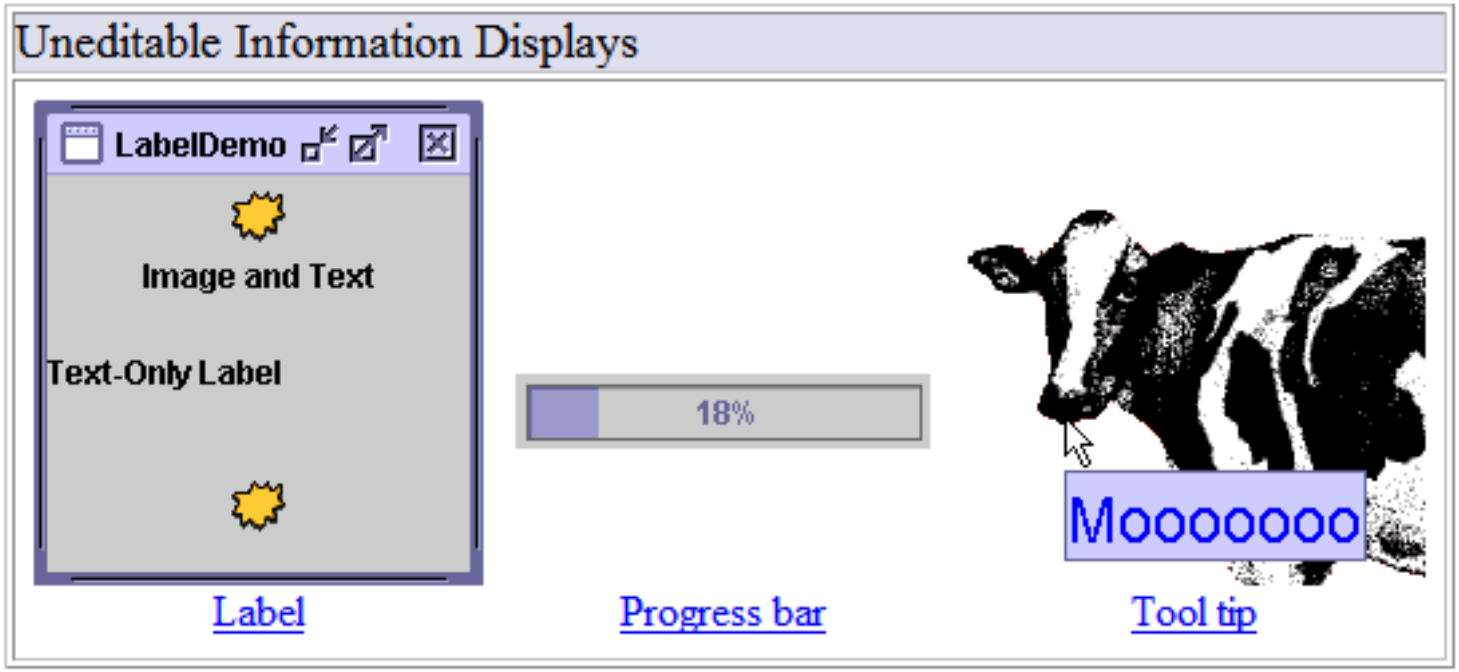
- There are more but these will do at present…

- Please see handout for the complete example codes with above compoments (in this ppt, all the codes are in simplified version).

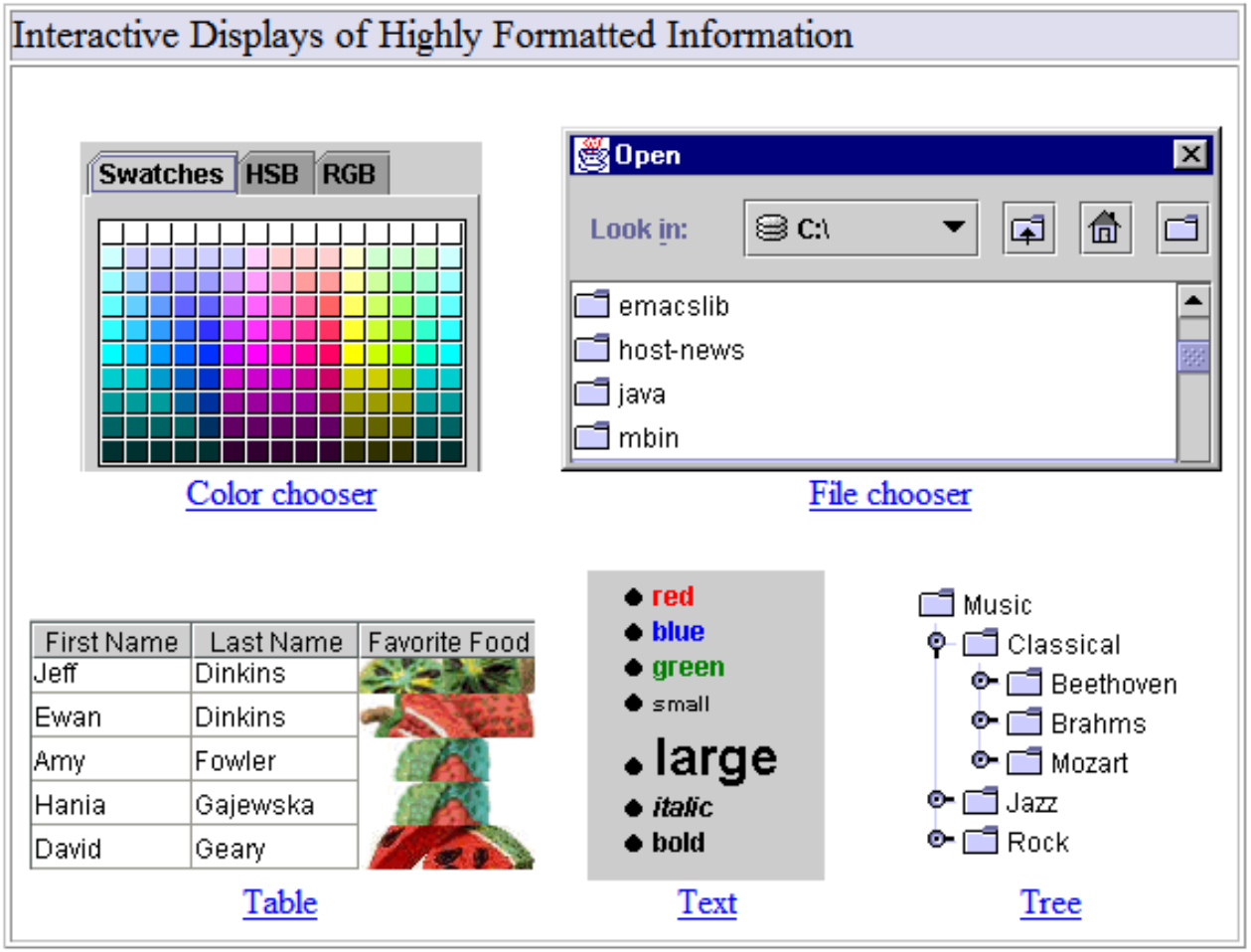
Example of Basic Controls



Example of Uneditable Information Displays



Example of Interactive Displays



Layout Managers

- *BorderLayout* – has five areas: “North”, “East”, “South”, “West”, “Center”
- *FlowLayout* – allocates controls from left to right, starting new lines when necessary. Controls are variable in size
- *GridLayout* – you define the number of columns
  - Controls are allocated left to right, in as many rows as needed
  - All controls are equal in size, e.g., for a Calculator

JApplet Component

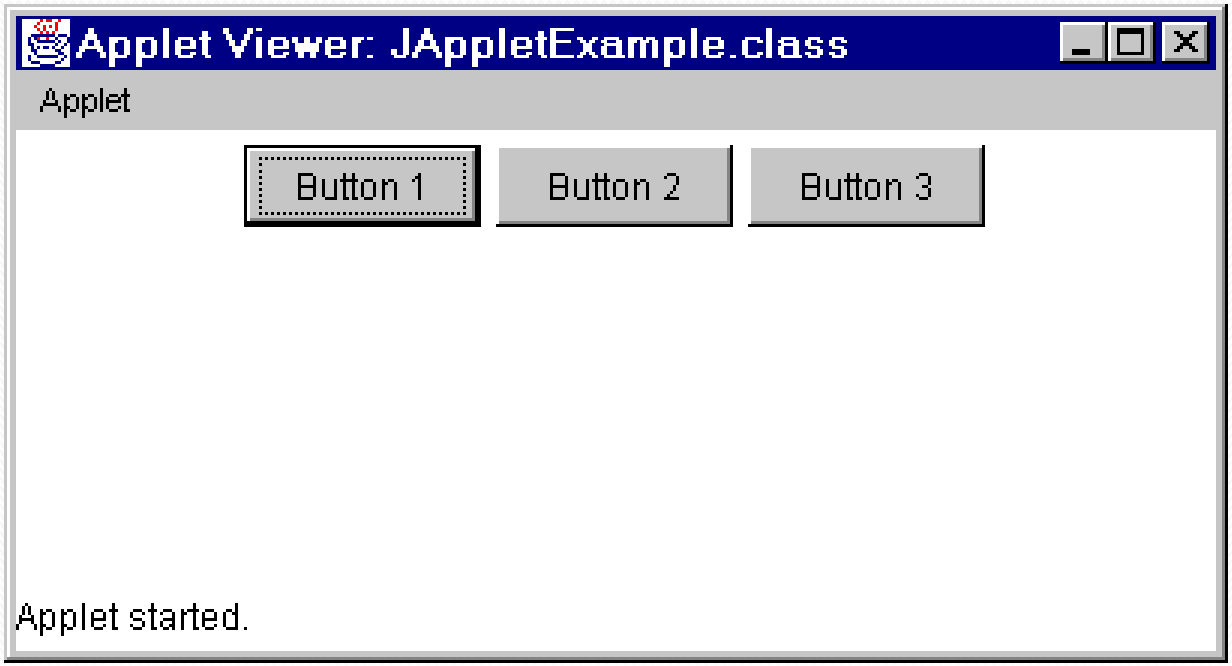
- Content Pane
  - A JApplet contains a content pane in which you can add components
  - You can access the content pane using getContentPane()
- Layout Manager
  - FlowLayout is the default layout manager for JApplet
  - BorderLayout is the default layout manager for content pane

JApplet: Example Code

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

public class JAppletExample extends JApplet {
  public void init() {
    Container content = getContentPane();
    content.setBackground(Color.white);
    content.setLayout(new FlowLayout());
    content.add(new JButton("Button 1"));
    content.add(new JButton("Button 2"));
    content.add(new JButton("Button 3"));
  }
}
```

JApplet: Example Output



JFrame Component

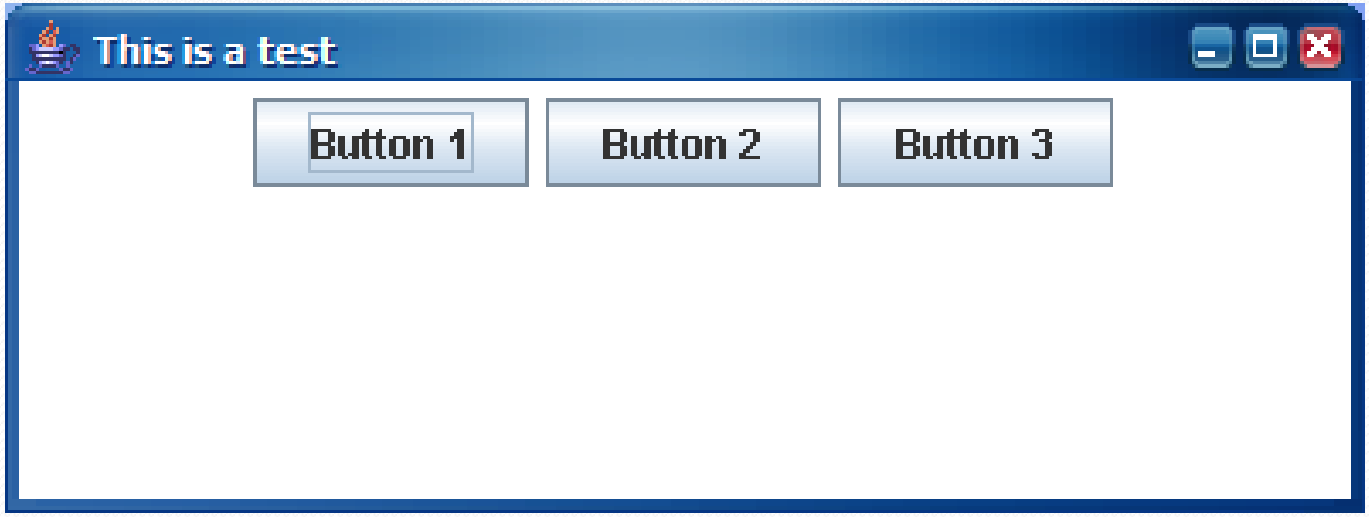
- Content Pane
  - *JFrame* uses content pane just as *JApplet* does
- Auto-Close Behavior
  - *JFrame* close automatically when you click on Close button, but closing the last *JFrame* does not cause your program to exit the Java application (just hides the window)
    - The "main" *JFrame* needs a *WindowListener* to call *System.exit()* (preferred)
    - Or it can call *setDefaultCloseOperation(EXIT_ON_CLOSE)*

JFrame: Example Code

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

public class JFrameExample {
    public static void main(String[] args) {
            JFrame f = new JFrame("This is a test");
            f.setSize(400,150);
            Container content = f.getContentPane();
            content.setBackground(Color.white);
            content.setLayout(new FlowLayout());
            content.add(new JButton("Button 1"));
            content.add(new JButton("Button 2"));
            content.add(new JButton("Button 3"));
            f.addWindowListener(new WindowAdapter {
                public void windowClosing(WindowEvent event) {
                    System.exit(0);
                }
            });
            f.setVisible(true);
    }
}
```

JFrame: Example Output



Other Swing Control Components…

- Please refer to the handout(Swing handout 1.2) for actual codes and try them out to see how it is displayed on your panel…

Event Handlers

- When a user moves mouse or hits key, an event is generated.
- One or more listener objects can register to be notified about events of a particular kind from a particular event source (e.g., mouse or keyboard)
- The class (*MyClass*) must declare that it implements the ActionListener interface. Example:

  public class MyClass implements ActionListener { …
- The event handler (*MyClass*) must register as a listener typically during initialization of the instance of the class. Example:

  somecomponent.addActionListener(instanceOfMyClass);
- Class (*MyClass*) must provide an implementation for the methods of the *ActionListener* interface. Example:

```
public void actionPerformed(ActionEvent e) {
        … // code that reacts to the action
}
```

Beeper Action Listener Example

```
public class Beeper ... implements Action Listener {
    // Declare that beepers are action event listeners
    ...
    JButton button = new JButton("Button");
    bottomPanel.add(button);
    button.addActionListener(this);
    // This beeper registers as a listener on the button class
    // Beeper listens for the user to click the button
    ...
    public void actionPerformed(ActionEvent e) {
        // When the user clicks the button,
        // the beeper is invoked by the button
        // to perform the action, i.e., make a beep sound
    }
}
```

Action Listeners

- In general, there will be more than one action event (button click) for which an instance of an object (this beeper) is listening
- The *getSource()* method for the *ActionEvent* returns the object (button) that generated the event
- On the previous slide, if the beeper was listening on more than one button

```
public void actionPerformed(ActionEvent e) {
        if (e.getSource() == button) // then make a beep sound
}
```

- You can also set a string on an action event (button click) *button.setActionCommand("Beep Button")* and subsequently retrieve it *if (e.getActionCommand() == "Beep Button") …*

MouseListener Interface

- The *MouseListener* interface contains five methods:
  - *mousePressed*
  - *mouseReleased*
  - *mouseEntered*
  - *mouseExited*
  - *mouseClicked*
- Even if you care only about mouse clicks, if your class directly implements *MouseListener*, you must implement all five *MouseListener* methods
- Methods for events you don't care about can have empty bodies
  - To avoid cluttering the code with empty methods, Swing provides adapter classes that implement empty versions of all of the interface's methods

```
MyClass extends MouseAdapter {
    someObject.addMouseListener(this);
    public void mouseClicked(MouseEvent e) {
      // Event handler implementation goes here
    }
}
```

Inner Classes

- Java does not support multiple inheritance
  - *MyClass* can't extend both *Applet* and *MouseAdapter* classes
- The solution is to define an inner class (a class within a class) that extends the *MouseAdapter* class

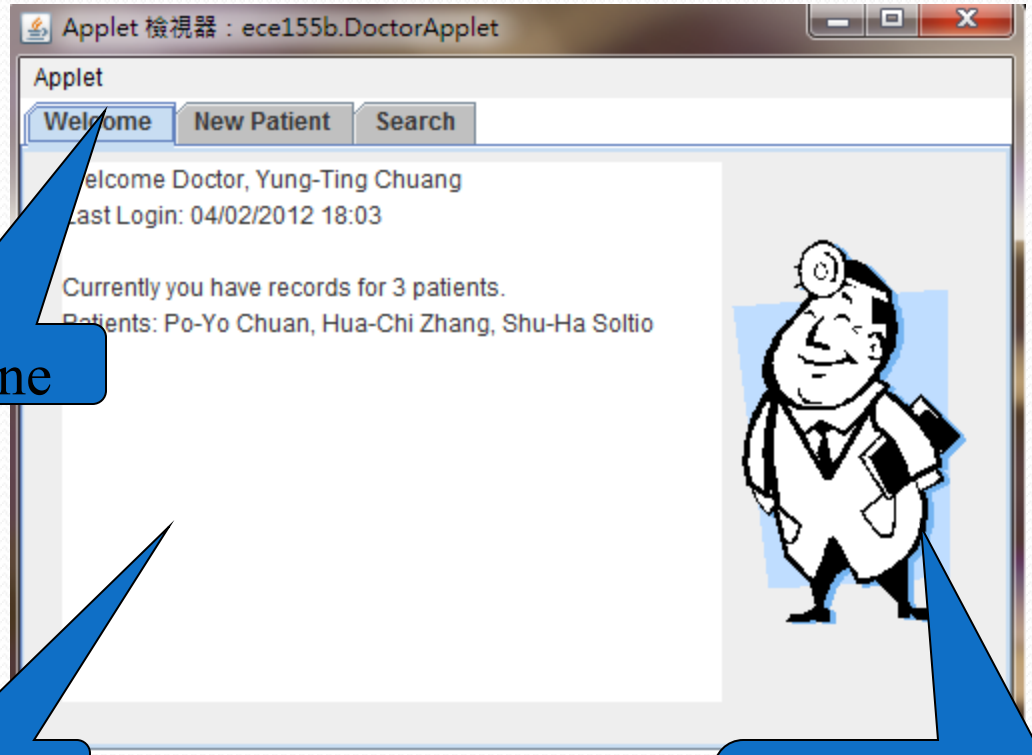
Inner Classes (continued)

- Example:

```
MyClass extends Applet {
    someObject.addMouseListener(new MyAdapter());
    …
    class MyAdapter extends MouseAdapter {
        public void mouseClicked(MouseEvent e) {
                    // Event handler implementation goes here
        }
    }
}
```

- To make a local variable available to an inner class, save a copy of the variable as a *final* local variable

Possible GUI for Project 1

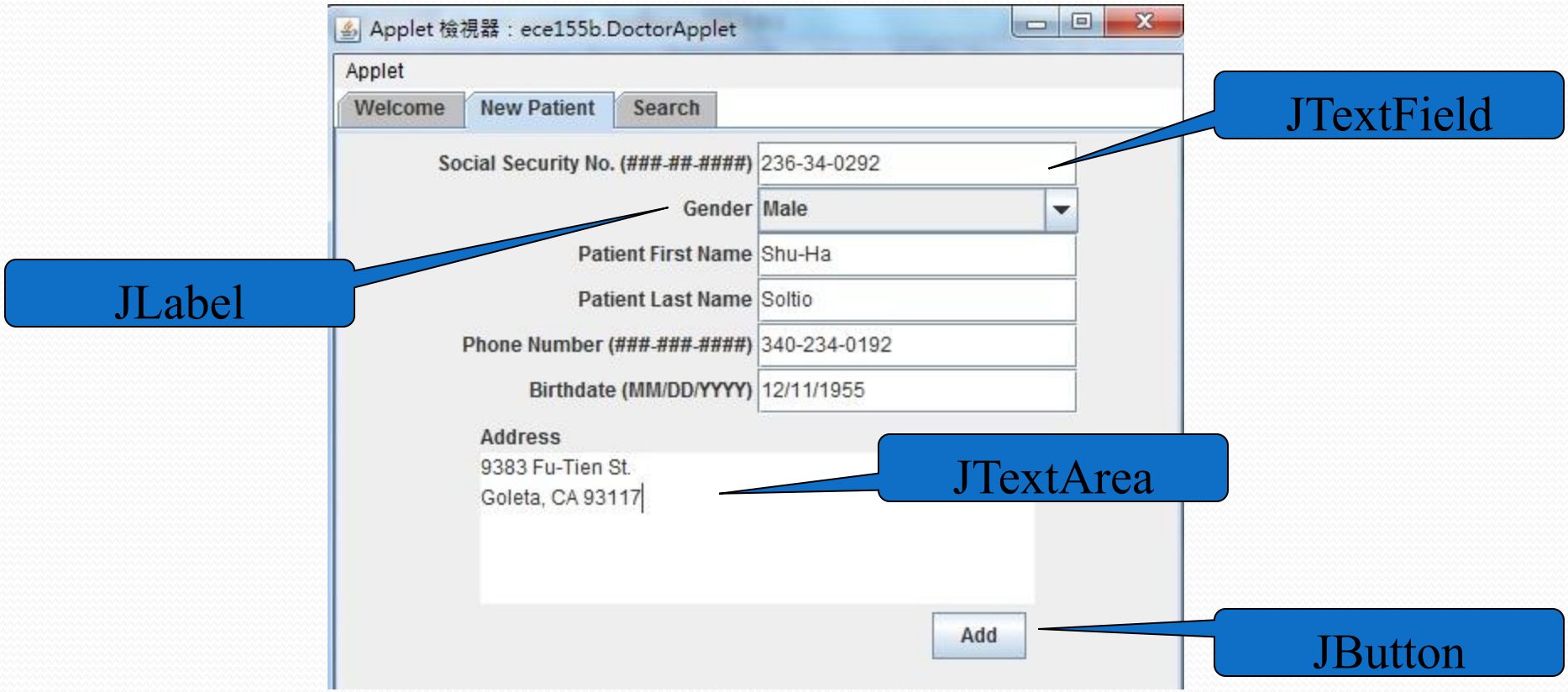


JTabbedPane

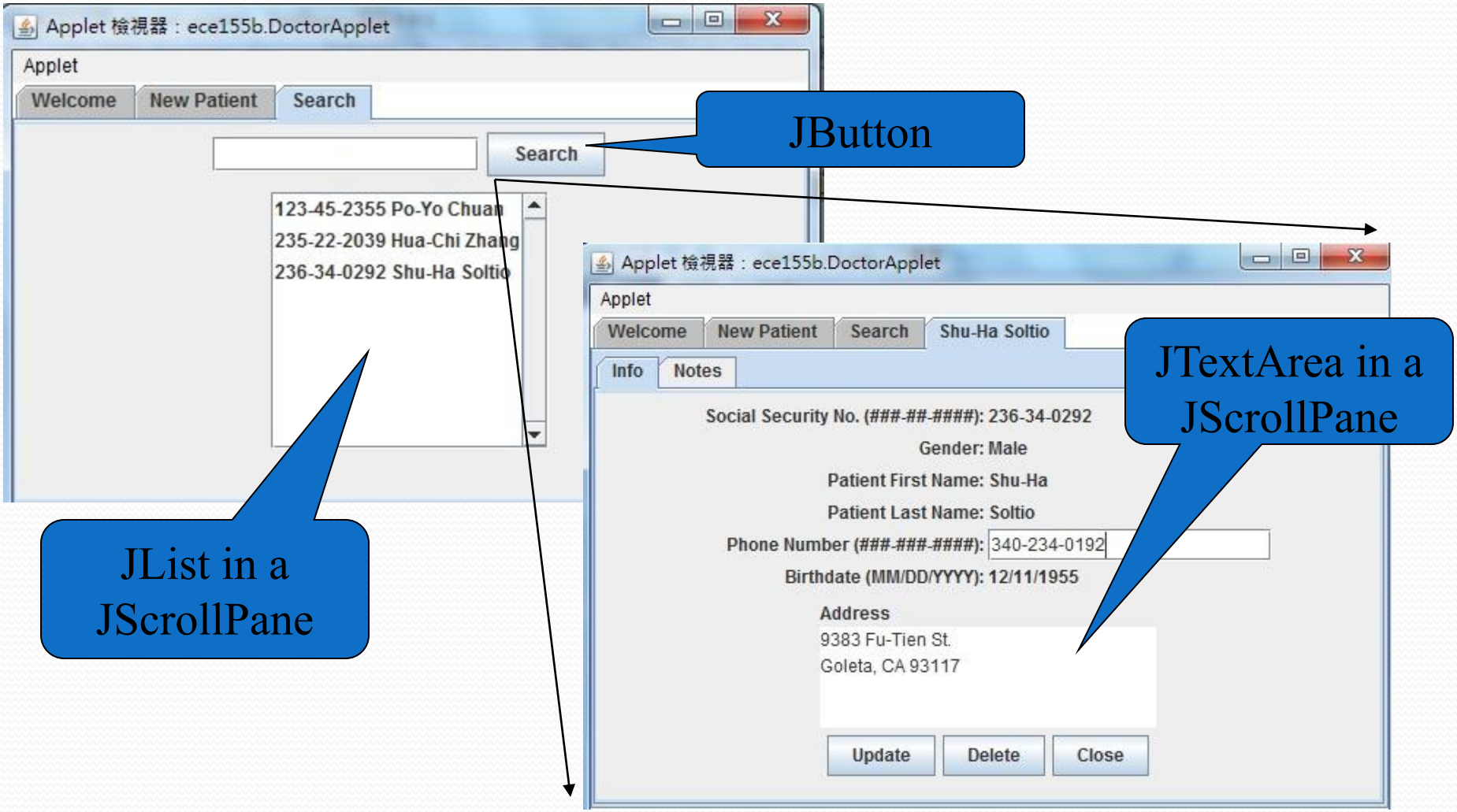
Text area

JLabel with ImageIcon

Possible GUI for Project 1(cont)



Possible GUI for Project 1 (cont)



Network Computing
Lecture 1.3: Applet

Instructor: *Louise Moser*

TA: *Yung-Ting Chuang*

Quarter: Spring 2012

Applet

- **Applets Tutorial**
  http://download.oracle.com/javase/tutorial/deployment/applet/index.html

- **Java Applet Code**
  - Your class should extend JApplet instead of JFrame (which you would use if you were creating a Java application instead of a Java Applet)
  - Use init(), instead of a constructor, to initialize applet
  - Use start(), instead of main, to start the execution of the applet
  - Use stop() to stop the execution of the applet
  - Use destroy() to removes the applet from the context of the appletviewer or browser
  - You do not need to create an instance of your class in the main() function, because an instance of your applet class is automatically created when an applet is loaded in browser
  - Use ***appletviewer*** tool to test your applet code. Example command: appletviewer -J-Djava.security.policy=dist/java.policy dist/Doctor.html

Sample HTML File (which includes applet in it)

```
<HTML>
    <BODY>
        <APPLET code ="ece155b.DoctorApplet" width=500
          height=500></APPLET>
    </BODY>
<HTML>
```

Java Policy File (Important!)

- Use ***policytool*** to add policies in your ***.java.policy*** file
- http://download.oracle.com/javase/6/docs/technotes/guides/security/PolicyFiles.html Has details about applet security restrictions and setting policy
- For our project, we put java policy file on the same directory as your jar & html files

Sample Java policy file

```
grant {
   permission java.io.FilePermission “<<ALL FILES>>”,
   “read, write, delete, execute”;
};

grant {
   permission java.security.AllPermission;
};
```

Network Computing
Lecture 1.4: Project IDEs

Instructor: *Louise Moser*

TA: *Yung-Ting Chuang*

Quarter: Spring 2012

Project IDEs

Ant

http://ant.apache.org/

Same as make file, more advanced and better handling. We will use to compile, deploy, execute and backup our projects.

Netbeans

http://www.netbeans.org/

Free Java development environment, which is/will be installed on ECI computers. Requires little effort to warm up with the IDE, will ease the programming task for you.

Cygwin (optional. Recommend if you're using windows)

http://www.cygwin.com/

I would advise all windows users to install cygwin. Ant tool has linux and windows versions, it is not necessary to install cygwin.

Ant

- We will use *ant* to compile, organize, backup, and deploy class projects. A template for class projects will be provided and students will continue developing the actual project.

- Instead of writing shell commands, the configuration files are XML-based, calling out a target tree where various tasks get executed.

- Following is example ant script that has several targets:
- ***init***: Create necessary directories
- ***clean***: Clean unnecessary files (.class)
- ***compile***: Compile java files
- ***jar***: Create executable jar file
- ***run***: Run the project

- Usage:
- > ant targetname (Ex. ant run, ant compile)

Example of ant scripts in build.xml

build.xml	
<project name="Applet project" default="jar" basedir=".">	
<description>Builds, tests, and runs the project</description> <property name="src" location="src" /> <property name="dist" location="dist" /> <property name="build" location="build" />	*Descriptions*
<target name="clean"> <delete dir="${build}"/> </target>	*Clean target, delete directories of class files or old files*
<target name="init"> <mkdir dir="${dist}" /> <mkdir dir="${build}" /> </target>	*Create necessary directories for compilation and deployment*
<target name="compile" depends="init"> <javac srcdir="${src}" destdir="${build}"/> </target>	*Compile java source files under src directory defined, then put class files into destdir directory*
<target name="jar" depends="compile"> <jar destfile="${dist}/Project.jar" basedir="${build}"> <fileset dir="${build}"> </jar> </target>	*Compress all class files into jar file, which is standalone executable java format.*
<target name="run" depends="jar"> <java jar="${dist}/Project.jar" fork="true"/> </target>	*Run the project*
</project>	

Ant (continued)

- Dependency provides recursive execution of targets.
  For example, you do not need to type:

  > ant init

  > ant compile

  > ant jar

  > ant run

- Simply type: “ant run” will run previous commands if necessary since we have defined dependency to jar which has dependency on compile, etc.

- From class website, you will be given a build.xml file which will help you to start up with the project. I will demonstrate how to execute and run the project1 template later the discussion.

NetBeans

- NetBeans IDE is a free, open-source Integrated Development Environment for software developers.
- The IDE runs on many platforms including Windows, Linux, Solaris, and the MacOS.
- It is easy to install and use straight out of the box.
- The NetBeans IDE provides developers with all the tools they need to create professional cross-platform desktop, enterprise, web and mobile applications.
- Existing project wizards will help you to start with the new project.