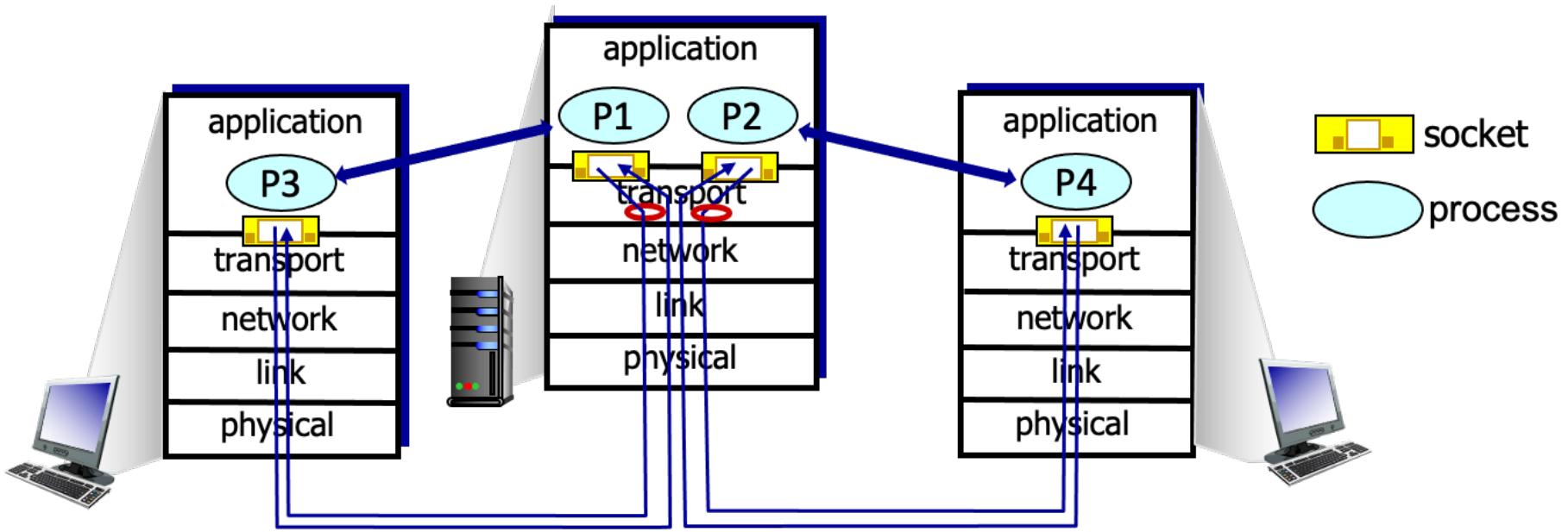
Sockets for Clients

Hyang-Won Lee
Department of CSE, Konkuk University
leehw@konkuk.ac.kr
https://sites.google.com/view/leehwko/

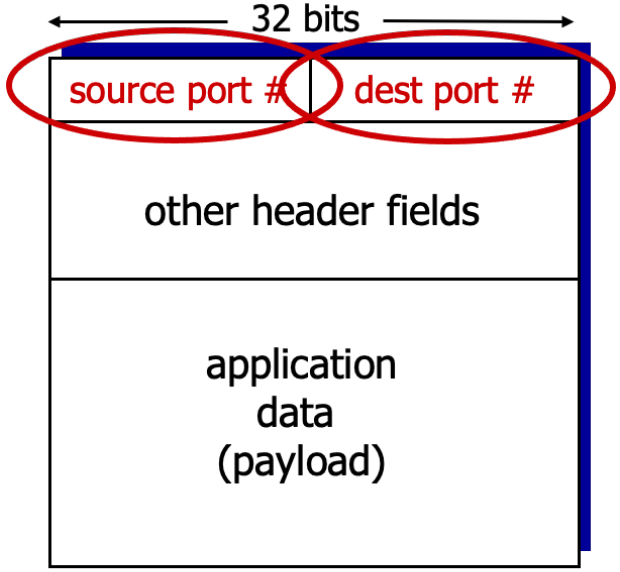
Multiplexing/demultiplexing (transport layer)

- Multiplexing at sender
  - Handle data from multiple sockets, add transport header
- Demultiplexing at receiver
  - Use header info to deliver received segments to correct socket



How demultiplexing works

- Host receives IP datagrams
  - Each datagram has src/dst IP addr
  - Each datagram carries one segment
  - Each segment has src/dst port number
- Host uses IP addresses & port numbers to direct segment to appropriate socket

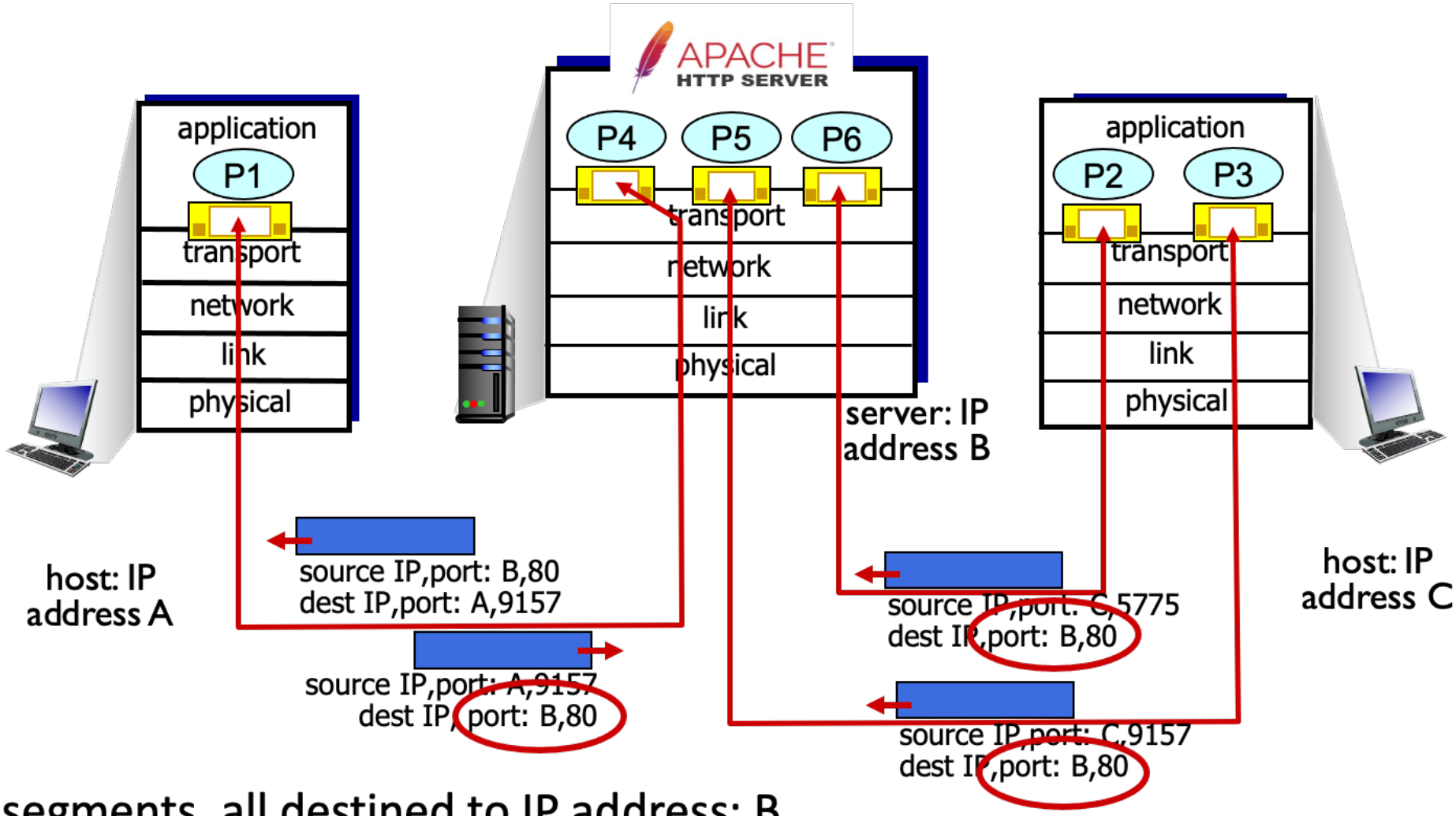


TCP/UDP segment format

Connection-oriented demultiplexing

- TCP socket identified by 4-tuple
  - Source IP address
  - Source port number
  - Destination IP address
  - Destination port number
- Server may support many simultaneous TCP sockets
  - Each socket identified by its own 4-tuple
  - Each socket associated with a different connecting client

Example



Three segments, all destined to IP address: B,
dest port: 80 are demultiplexed to *different* sockets

Port numbers

- Range (16bits; 0~65535)
  - 0~1023: reserved for privileged services (well-known ports) that should not be used without IANA registration
    - 21: FTP
    - 22: SSH / SFTP
    - 80: Web server
  - 2014~49151: registered ports that should not be used without IANA registration
    - Non-well-known ports that are used by S/W vendors for their own server applications
  - 49152~65535: dynamic/private ports

Socket Class

- java.net.Socket
  - fundamental class for performing client-side TCP operations
  - used by URL, URLConnection, Applet and JEditorPane

*used by both client and server

- Basic constructors
  - ```
    public Socket(String host, int port) throws UnknownHostException, IOException
    public Socket(InetAddress host, int port) throws IOException
    ```
  - these constructors connect the socket

```
try {
  Socket toOReilly = new Socket("www.oreilly.com", 80);
  // send and receive data...
} catch (UnknownHostException ex) {
  System.err.println(ex);
} catch (IOException ex) {
  System.err.println(ex);
}
```

exception if connection can't be opened
```

Example

```
import java.net.*;
import java.io.*;

public class LowPortScanner {

  public static void main(String[] args) {

    String host = args.length > 0 ? args[0] : "localhost";

    for (int i = 1; i < 1024; i++) {
      try {
        Socket s = new Socket(host, i);
        System.out.println("There is a server on port " + i + " of "
         + host);
        s.close();
      } catch (UnknownHostException ex) {
        System.err.println(ex);
        break;
      } catch (IOException ex) {
        // must not be a server on this port
      }
    }
  }
}
```

Other Constructors

- Picking local interface to connect from
  - ```
    public Socket(String host, int port, InetAddress interface, int localPort)
          throws IOException, UnknownHostException
    public Socket(InetAddress host, int port, InetAddress interface, int localPort)
          throws IOException
    ```
    remote local

- Without connection

  ```
  public Socket()
  ```

  - can connect later using connect(,) method

    ```
    public void connect(SocketAddress endpoint, int timeout) throws IOException
    ```
    ```

SocketAddress Class

- Represent a connection endpoint
- Empty abstract class with no methods except a constructor
- Provide a convenient store for socket connection info (IP addr, port)
- Getter
  - `public SocketAddress getRemoteSocketAddress()`
  - `public SocketAddress getLocalSocketAddress()`

```
Socket socket = new Socket("www.yahoo.com", 80);
SocketAddress yahoo = socket.getRemoteSocketAddress();
```

-
```
Socket socket2 = new Socket();
socket2.connect(yahoo);
```
can also do this

InetSocketAddress Class

- Subclass of SocketAddress
- Constructors
  - `public InetSocketAddress(InetAddress address, int port)`
  - `public InetSocketAddress(String host, int port)`
  - `public InetSocketAddress(int port)`
- Example

```
try {
  Socket socket = new Socket();
  // fill in socket options
  SocketAddress address = new InetSocketAddress("time.nist.gov", 13);
  socket.connect(address);
  // work with the sockets...
} catch (IOException ex) {
  System.err.println(ex);
}
```

Procedure

- Connect to a remote machine
- Send data
- Receive data
- Close a connection

only needed by server

- Bind to a port
- Listen for incoming data
- Accept connections from remote machines on the bound port

Example

```java
import java.net.*;
import java.io.*;

public class DaytimeClient {

  public static void main(String[] args) {
    String hostname = args.length > 0 ? args[0] : "time.nist.gov";
    Socket socket = null;
    try {
      socket = new Socket(hostname, 13);
      socket.setSoTimeout(15000);
      InputStream in = socket.getInputStream();
      StringBuilder time = new StringBuilder();
      InputStreamReader reader = new InputStreamReader(in, "ASCII");
      for (int c = reader.read(); c != -1; c = reader.read()) {
        time.append((char) c);
      }
      System.out.println(time);
    } catch (IOException ex) {
      System.err.println(ex);
    } finally {
      if (socket != null) {
        try {
          socket.close();
        } catch (IOException ex) {
          // ignore
        }
      }
    }
  }
}
```

Writing to Server

- Get output stream and write
  - `OutputStream out = socket.getOutputStream();`
  - the rest is the same as usual output stream

```java
import java.net.*;
import java.io.*;

public class DictClient3 {
    public static void main(String[]args) {
        String host = "dict.org";
        try {
            Socket soc = new Socket(host,2628);
            OutputStream out = soc.getOutputStream();
            Writer writer = new OutputStreamWriter(out, "UTF-8");
            writer = new BufferedWriter(writer);
            InputStream in = soc.getInputStream();
            BufferedReader reader = new BufferedReader(
                    new InputStreamReader(in, "UTF-8"));
            String request = "DEFINE fd-eng-lat gold";//syntax: DEFINE DB query
            writer.write(request);
            writer.flush();
            soc.shutdownOutput();


            for (String line = reader.readLine(); line != null; line = reader.readLine()) {
                System.out.println(line);
            }

            soc.close();
        } catch (UnknownHostException e) {
            System.out.println("Cannot found the host at "+host);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- modify the code so that multiple query strings can be processed

Getting Info about Socket

- Socket attributes
  - remote address
  - remote port
  - local address
  - local port
- Methods
  - ```
    public InetAddress getInetAddress()
    public int getPort()
    public InetAddress getLocalAddress()
    public int getLocalPort()
    ```
    ```

Example

```
import java.net.*;
import java.io.*;

public class SocketInfo {

  public static void main(String[] args) {

    for (String host : args) {
      try {
        Socket theSocket = new Socket(host, 80);
        System.out.println("Connected to " + theSocket.getInetAddress()
            + " on port "  + theSocket.getPort() + " from port "
            + theSocket.getLocalPort() + " of "
            + theSocket.getLocalAddress());
      }  catch (UnknownHostException ex) {
        System.err.println("I can't find " + host);
      } catch (SocketException ex) {
        System.err.println("Could not connect to " + host);
      } catch (IOException ex) {
        System.err.println(ex);
      }
    }
  }
}
```

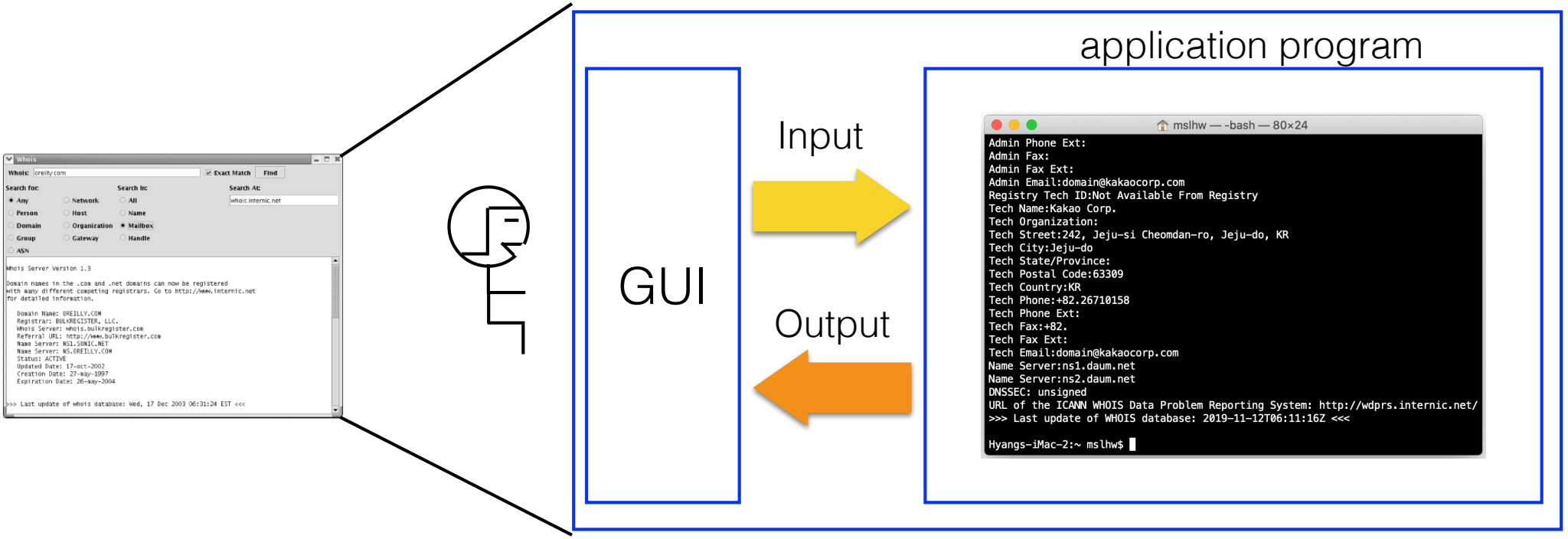
Closed or Connected

- Methods
  - isClosed()
    - false if connection is open, or connection has never been opened
  - isConnected()
    - true if socket has ever been connected to a remote host
- Quiz
  - what is the right way to check if socket connection is currently open?
    - socket.isConnected() && !socket.isClosed()

GUI in Java

GUI vs CLI

- Command line interface (CLI)
  - keyboard based operations: LINUX, UNIX
- Graphic user interface (GUI)
  - mouse, keyboard



General Procedure

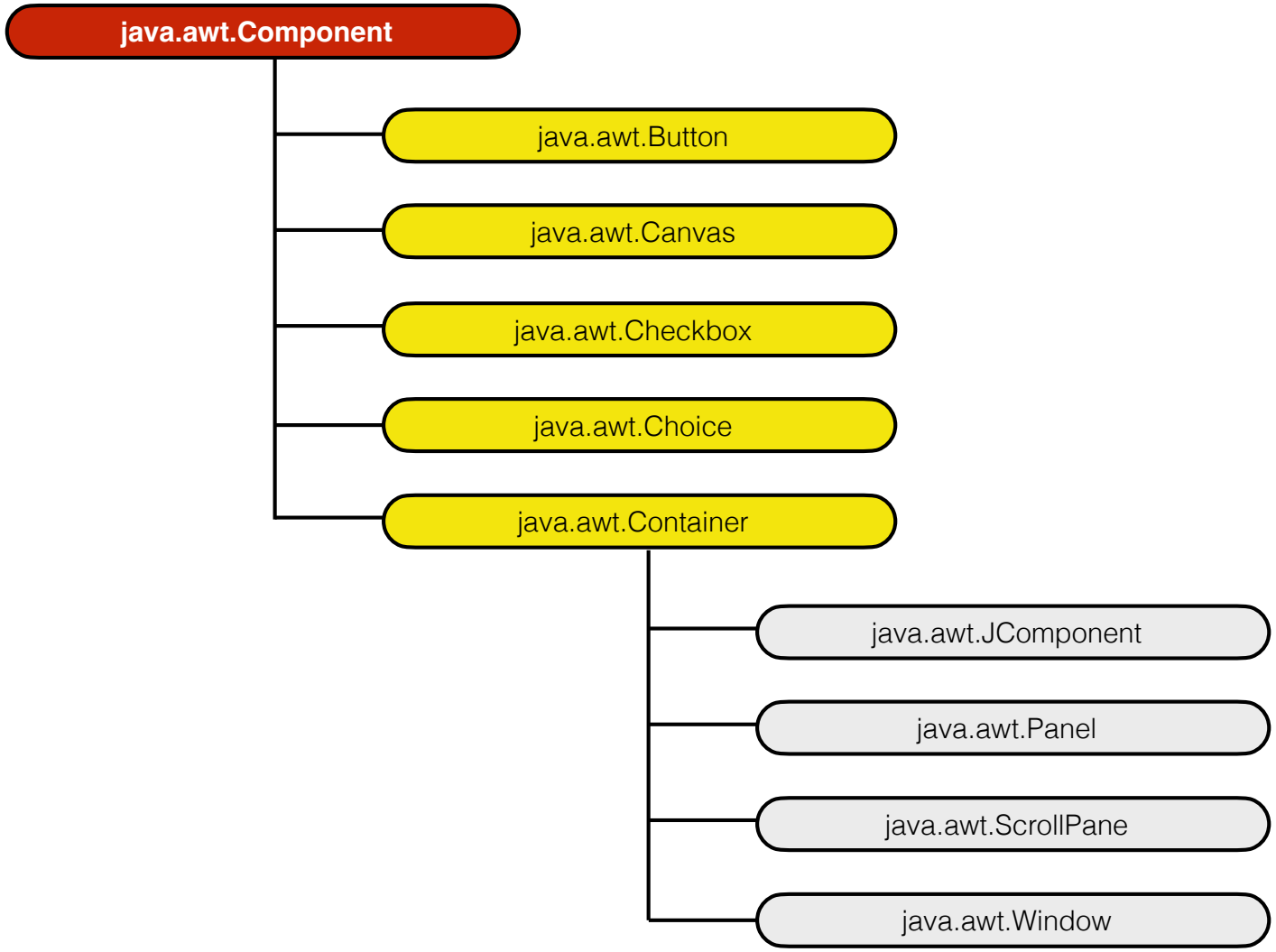
- Construct frame (or window) object
- Define layout (using Layout object)
- Construct components (button, scroll bar, icon, toolbar,…) and place onto the defined layout
- Register event handler that respond to user input (click, keyboard input, mouse movement)
- Implement code executed upon event(s)

GUI in Java

- Abstract window toolkit (AWT)
  - minimum common toolkit for GUI development
- Swing
  - support for more complex components (table, tree, progress bar,…)
  - better OS compatibility
  - prefix "J": e.g., JLabel, JFrame,…
- Simple widget toolkit (SWT)
  - GUI toolkit in eclipse

AWT

- Components



AWT Example

```
import java.awt.*;

public class Gui1AwtBasic {
    public static void main(String[] args) {
        Frame f = new Frame("Hello World");
        f.setLayout(new FlowLayout());
        Label label = new Label("Welcome to AWT");
        f.add(label);
        f.setSize(200, 200);
        f.setVisible(true);
    }
}
```

create separate window (frame)
construct Layout object
create label
add label to window
set dimension
show

Swing Example

ui1AwtBasic.ja | Gui2SwingBasic. | Gui4SwingLayout

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

public class Gui2SwingBasic extends JFrame {
    public Gui2SwingBasic() {
        super("Hello World");
        getContentPane().setLayout(new FlowLayout());
        JLabel label = new JLabel("Welcome to Swing");
        getContentPane().add(label);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(200,200);
        setVisible(true);
    }

    public static void main(String[] args) {
        Gui2SwingBasic app = new Gui2SwingBasic();
    }
}
```

close enabled

```java
import java.awt.*;

public class Gui3AwtLogin extends Frame{

    public Gui3AwtLogin() {
        super("AWT 기본예제");
        setLayout(new BorderLayout());
        Label t1 = new Label("ID");
        Label t2 = new Label("Passwd");
        TextField id = new TextField(10);
        TextField pwd = new TextField(10);
        Button btn = new Button("Login");

        Panel p1 = new Panel();
        p1.add(t1);
        p1.add(id);

        Panel p2 = new Panel();
        p2.add(t2);
        p2.add(pwd);
        p2.add(btn);

        add(p1, BorderLayout.NORTH);
        add(p2, BorderLayout.SOUTH);

        setSize(250,100);
        setVisible(true);
    }

    public static void main(String[] args) {
        Gui3AwtLogin app = new Gui3AwtLogin();
    }

}
```

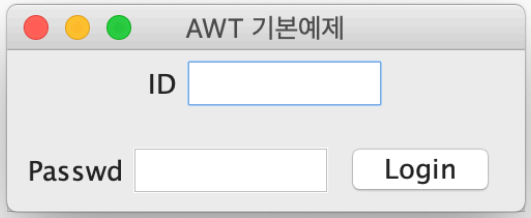
create labels and textfields

create button

add label and textfield to panel

add label and textfield to panel

add panels to north and south



AWT vs. Swing

Feature	AWT	Swing
text output	Label	JLabel
multi-line text output	Multiple Labels	Multiple JLabels or JLabel and HTML content
multi-line format text output	Multiple Labels, fonts	JLabel and HTML content
single-line text input	TextField	JTextField
multi-line text input	TextArea	JTextArea
image output	N/A	JLabel
text and image output	N/A	JLabel
ToolTip popup help	N/A	setToolTip on component, subclass JToolTip
text input style	N/A	JEditorPane
item list selection	List	JList
push button and text	Button	JButton
push button, text/image	N/A	JButton
drawing area	Canvas	JPanel
on/off checkbox	CheckBox	JCheckBox

AWT vs. Swing

Feature	AWT	Swing
radio selection	CheckBoxGroup	ButtonGroup and Menu
dropdown list selection	Choice	JComboBox
text input or dropdown list	N/A	JComboBox
scroll	ScrollPane	JScrollPane
window (top)	Dialog, Frame, Window	JDialog, JFrame, JWindow
window (basic)	Window	JWindow
window (frame)	Frame	JFrame
window (dialog)	Dialog	JDialog
menu	Menu	JMenu
menu items	MenuItem	JMenuItem
menu shortcut	key stroke	key stroke
popup menu	PopupMenu	JPopupMenu
menu bar	MenuBar	JMenuBar
inserted symbol output	N/A	Caret

Container and Layout

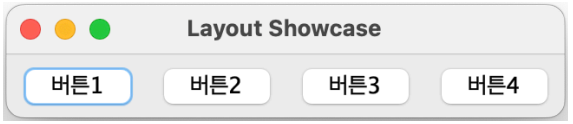
- Container classes
  - Panel, Frame, Window, Dialog, ScrollPane
- Common layout classes
  - FlowLayout, GridLayout, BorderLayout, CardLayout, GridBagLayer
- Applying layout

```
Frame f = new Frame();
f.setLayout(new BorderLayout());
```

- Adding components

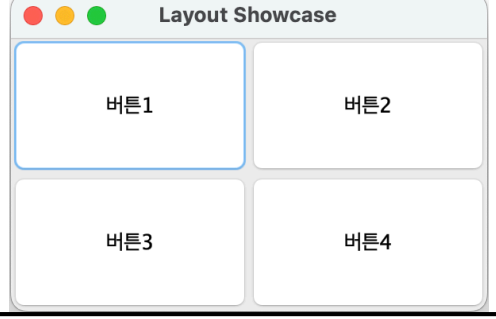
```
Button b = new Button("Button 1");
f.add(b, BorderLayout.WEST);
```

FlowLayout



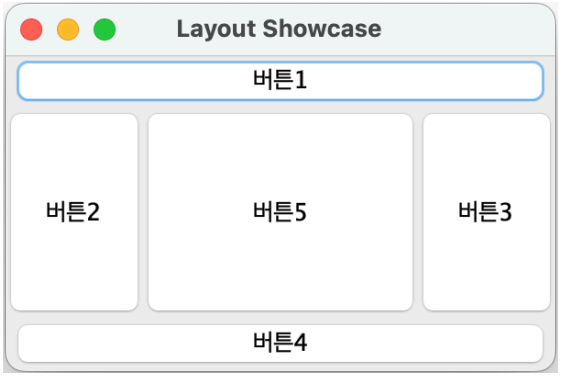
Fields	static int CENTER	각 행의 컴포넌트가 가운데 위치하게 된다
	static int LEANDING	각 행의 컴포넌트가 컨테이너의 리딩 엣지(컨테이너의 왼쪽 단)에 정렬되게 한다
	static int LEFT	각 행의 컴포넌트를 왼쪽으로 정렬한다
	static int RIGHT	각 행의 컴포넌트를 오른쪽으로 정렬한다
	static int TRAILING	각 행의 컴포넌트를 컨테이너 트레일링 엣지(컨테이너의 오른쪽 단)에 정렬되게 한다
Constructors	FlowLayout()	왼쪽/오른쪽으로 5단위 간격만큼 가운데에 위치시켜 FlowLayout을 생성한다
	FlowLayout(int align)	align(정렬 속성)값에 따라 왼쪽/오른쪽으로 5단위 간격이 있는 FlowLayout을 생성한다
	FlowLayout(int align, int hgap, int vgap)	align(정렬 속성), hgap(수평 간격), vgap(수직 간격)이 있는 FlowLayout을 생성한다
Methods	int getAlignment()	정렬 상태값을 반환한다
	int getHgap()	수평 간격 설정값을 반환한다
	int getVgap()	수직 간격 설정값을 반환한다
	void setAlignment(int align)	정렬 상태를 설정한다
	void setHgap(int gap)	수평 간격을 설정한다
	void setVgap(int vgap)	수직 간격을 설정한다

GridLayout



constructors	GridLayout()	1행 1열의 GridLayout을 생성한다
	GridLayout(int rows, int cols)	rows만큼 행과 cols만큼 열이 있는 GridLayout을 생성한다
	GridLayout(int rows, int cols, int hgap, int vgap)	rows만큼 행과 cols만큼 열, hgap만큼 수평 간격, vgap만큼 수직 간격이 있는 GridLayout을 생성한다이 있는 GridLayout을 생성한다
methods	void addLayoutComponent(String name, Component comp)	name 문자열이 있는 comp 컴포넌트 추가한다
	int getColumns()	전체 열 개수를 반환한다
	int getRows()	전체 행 개수를 반환한다
	int getHgap()	수평간격 설정값을 반환한다
	void layoutContainer(Container parent)	parent 컨테이너를 설치한다
	void setColumns(int cols)	전체 열 개수를 설정한다
	void setRows(int rows)	전체 행 개수를 설정한다
	void setHgap(int hgap)	수평 간격을 설정한다
	void setVgap(int hgap)	수직 간격을 설정한다

BorderLayout



Fields	static String CENTER	컨테이너의 가운데로 제약한다
	static String EAST	컨테이너의 오른쪽으로 제약한다
	static String LINE_END	행 방향 마지막에 위치한다
	static String LINE_START	행 방향 맨 앞쪽에 위치한다
	static String NORTH	컨테이너의 위쪽으로 제약한다
	static String PAGE_END	맨 마지막 줄의 끝에 위치한다
	static String PAGE_START	왼쪽 위 맨 앞쪽 행에 위치한다
	static String SOUTH	컨테이너의 아래쪽으로 제약한다
	static String WEST	컨테이너의 왼쪽으로 제약한다
Constructors	BorderLayout()	컨포넌트 간에 간격 없이 배치되는 BorderLayout을 생성한다
	BorderLayout(int hgap, int vgap)	컨포넌트 간에 hgap칸큼 수평간격, vgap만캄 수직간격이 있도록 BorderLayout을 생성한다
Methods	int getHgap()	수평간격 설정값을 반환한다
	int getVgap()	수직간격 설정값을 반환한다

CardLayout

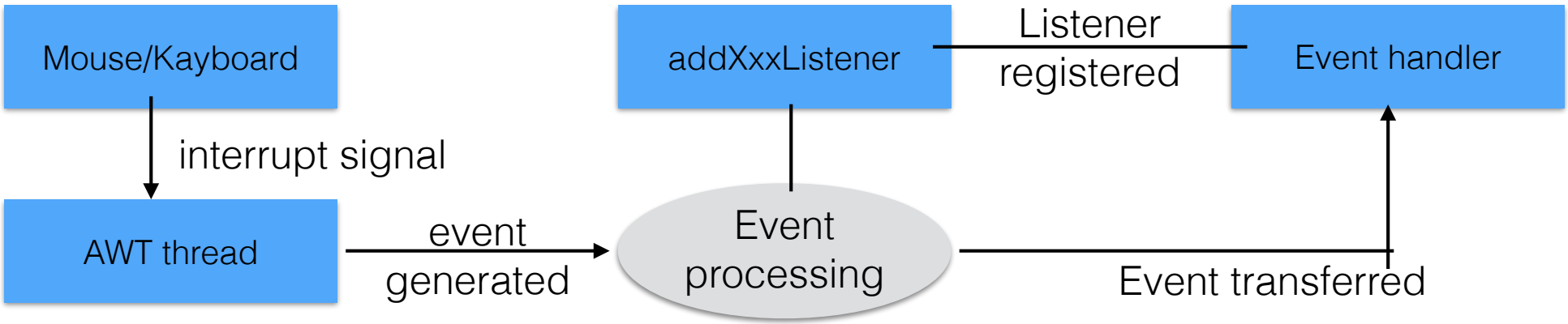
Constructors	CardLayout()	간격이 없는 CardLayout을 생성한다
	CardLayout(int hgap, int vgap)	hgap만큼 수평간격, vgap만큼 수직간격이 있는 CardLayout을 생성한다
Methods	int first(Container parent)	컨테이너의 처음 카드로 전환한다
	int getHgap()	수평간격 설정값을 반환한다
	int getVgap()	수직간격 설정값을 반환한다
	void last(Container parent)	컨테이너의 마지막 카드로 전환한다
	void next(Container parent)	컨테이너의 현재 카드에서 다음 카드로 전환한다
	void previous(Container parent)	컨테이너의 현재 카드에서 이전 카드로 전환한다
	void removeLayoutComponent(Component comp)	comp 컴포넌트를 레이아웃에서 삭제한다
	void setHgap(int hgap)	컴포넌트 간의 수평 간격을 설정한다
	void setVgap(int vgap)	컴포넌트 간의 수직 간격을 설정한다
	void show(Container parent, String name)	name 문자열이 있는 컴포넌트로 전환한다

Example

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

public class Gui4SwingLayout extends JFrame{
    JButton button1 = new JButton("버튼1");
    JButton button2 = new JButton("버튼2");
    JButton button3 = new JButton("버튼3");
    JButton button4 = new JButton("버튼4");
    JButton button5 = new JButton("버튼5");

    Panel p1 = new Panel();
    Panel p2 = new Panel();

    public void flowLayout() {
        p1.setLayout(new FlowLayout());
        p1.add(button1);
        p1.add(button2);
        p1.add(button3);
        p1.add(button4);
    }
    public void gridLayout() {
        p1.setLayout(new GridLayout(2,2));
        p1.add(button1);
        p1.add(button2);
        p1.add(button3);
        p1.add(button4);
    }
    public void borderLayout() {
        p1.setLayout(new BorderLayout());
        p1.add(button1, BorderLayout.NORTH);
        p1.add(button2, BorderLayout.WEST);
        p1.add(button3, BorderLayout.EAST);
        p1.add(button4, BorderLayout.SOUTH);
        p1.add(button5, BorderLayout.CENTER);
    }
    public void cardLayout() {
        final CardLayout card = new CardLayout();
        setLayout(card);
        p1.add(button1);
        p1.add(button2);
        p1.add(button3);
        //p2.add(button3);
        p2.add(button4);
        p2.add(button5);
        add("p1",p1);
        add("p2",p2);

        button3.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent arg0) {
                card.show(getContentPane(), "p2");
            }

        });
    }
    public Gui4SwingLayout() {
        super("Layout Showcase");
        getContentPane().add(p1);
        //flowLayout();
        //gridLayout();
        //borderLayout();
        cardLayout();
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(300,200);
        setVisible(true);
    }

    public static void main(String[] args) {
        Gui4SwingLayout app = new Gui4SwingLayout();
    }
}
```

Event Handler

- Procedure

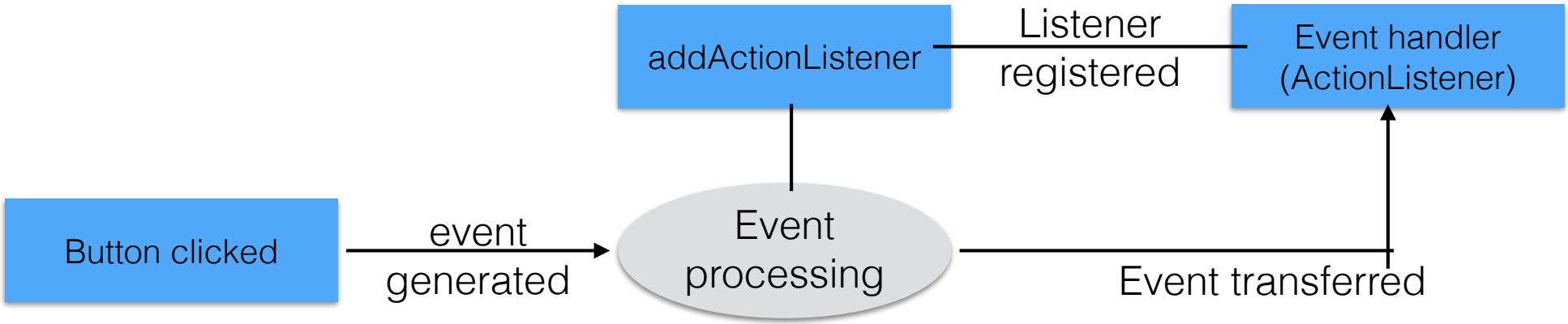


- Events

Events	Description
ActionEvent	컴포넌트에 정의된 행위가 발생했을 때 나타나는 이벤트이다. addActionListener 메서드로 등록하며, ActionListener에 이벤트를 넘긴다
MouseEvent	컴포넌트에 마우스 액션이 발생할 때 나타나는 이벤트이다. addMouseListener 메서드로 등록하며, MouseListener, MouseAdapterListener에 이벤트를 전달한다
MouseWheelEvent	컴포넌트 안에서 마우스 휠을 회전했을 때 나타나는 이벤트이다. addMouseWheelListener 메서드로 등록하며, MouseWheelListener에 이벤트를 전달한다
KeyEvent	컴포넌트 안에서 키 입력으로 나타나는 이벤트이다. addKeyListener 메서드로 등록하며, KeyListener에 이벤트를 전달한다
ComponentEvent	컴포넌트의 이동, 크기 변경 등 변화가 있을 대 나타나는 이벤트이다. addComponentListener 메서드로 등록하며, ComponentListener에 이벤트를 전달한다
ContainerEvent	컴포넌트를 추가하거나 삭제할 때 컨테이너에 변화가 발생하면 나타나는 이벤트이다. addContainerListener 메서드로 등록하며, ContainerListener에 이벤트를 전달한다
ItemEvent	항목을 선택하거나 해제할 때 나타나는 이벤트이다. addItemListener 메서드로 등록하며, ItemListener에 이벤트를 전달한다

Example

- Procedure
  - define handler to process an event
  - add component to event handler
  - override method required and implement function to execute upon the occurrence of event
- Button event example



Example

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

public class Gui5EventHandler extends JFrame implements ActionListener{
    int index = 0;
    String[] msgs = {"첫 번째 문장","두 번째 문장","세 번째 문장"};
    JButton button1 = new JButton("<<");
    JButton button2 = new JButton(">>");
    JButton button3 = new JButton(msgs[0]);

    public Gui5EventHandler() {
        BorderLayout layout = new BorderLayout();
        setLayout(layout);
        button1.addActionListener(this);
        button2.addActionListener(this);
        button3.setEnabled(false);
        add(button1, BorderLayout.WEST);
        add(button2, BorderLayout.EAST);
        add(button3, BorderLayout.CENTER);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(300,100);
        setVisible(true);
    }
```

Example

```
    @Override
    public void actionPerformed(ActionEvent e) {
        Object obj = e.getSource();
        if(obj == button1) {
            index--;
        }
        else if(obj == button2) {
            index++;
        }
        if(index > 2)
            index = 0;
        else if(index < 0) {
            index = 2;
        }
        button3.setText(msgs[index]);
    }

    public static void main(String[] args) {
        Gui5EventHandler app = new Gui5EventHandler();
    }
}
```

GUI Example: Whois

Whois (RFC 954)

- Internet domain name and network number directory service
  - look up databases maintained by network information centers (e.g., IANA)
  - give details about domain name and related info
- Procedure
  - (client) open TCP socket to port 43 on server
  - (client) send search string terminated by \r\n
    - search string: name, list of names, domain name,…
  - (server) send unspecified amount of human-readable info
  - (client) display received info to user

Prefixes

- Should be placed before search string

Prefix	Meaning
Domain	Find only domain records.
Gateway	Find only gateway records.
Group	Find only group records.
Host	Find only host records.
Network	Find only network records.
Organization	Find only organization records.
Person	Find only person records.
ASN	Find only autonomous system number records.
Handle or !	Search only for matching handles.
Mailbox or @	Search only for matching email addresses.
Name or :	Search only for matching names.
Expand or *	Search only for group records and show all individuals in that group.
Full or =	Show complete record for each match.
Partial or suffix	Match records that start with the given string.
Summary or $	Show just the summary, even if there's only one match.
SUBdisplay or %	Show the users of the specified host, the hosts on the specified network, etc.

example (type at the terminal)

```
% whois Person Partial Elliot
```

Whois Class

```
import java.net.*;
import java.io.*;

public class Whois {

  public final static int DEFAULT_PORT = 43;
  public final static String DEFAULT_HOST = "whois.internic.net";

  private int port = DEFAULT_PORT;
  private InetAddress host;

  public Whois(InetAddress host, int port) {
    this.host = host;
    this.port = port;
  }

  public Whois(InetAddress host) {
    this(host, DEFAULT_PORT);
  }

  public Whois(String hostname, int port)
   throws UnknownHostException {
    this(InetAddress.getByName(hostname), port);
  }
```

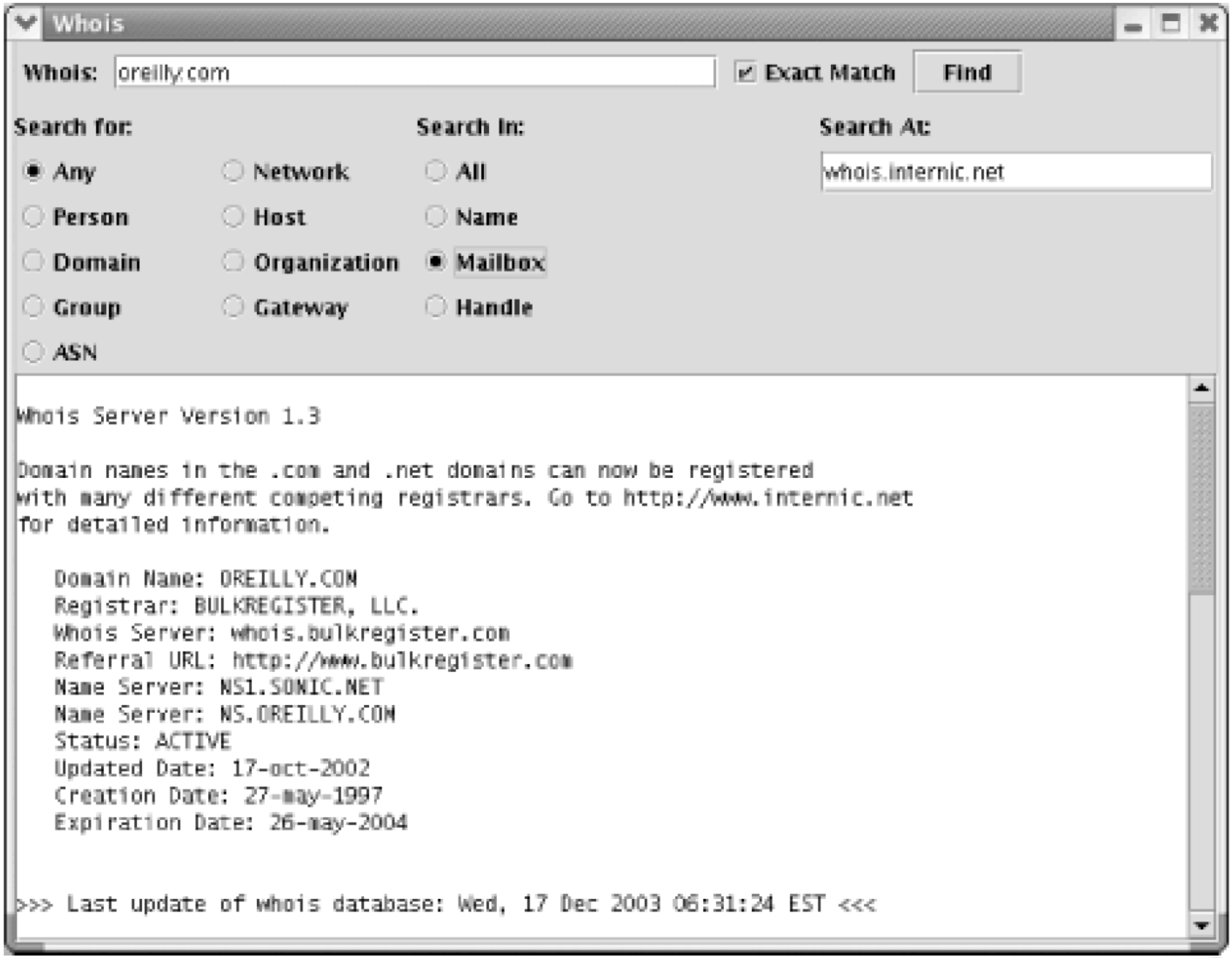
```
public Whois(String hostname) throws UnknownHostException {
  this(InetAddress.getByName(hostname), DEFAULT_PORT);
}

public Whois() throws UnknownHostException {
  this(DEFAULT_HOST, DEFAULT_PORT);
}

// Items to search for
public enum SearchFor {
  ANY("Any"), NETWORK("Network"), PERSON("Person"), HOST("Host"),
  DOMAIN("Domain"), ORGANIZATION("Organization"), GROUP("Group"),
  GATEWAY("Gateway"), ASN("ASN");
  private String label;

  private SearchFor(String label) {
    this.label = label;
  }
}

// Categories to search in
public enum SearchIn {
  ALL(""), NAME("Name"), MAILBOX("Mailbox"), HANDLE("!");

  private String label;

  private SearchIn(String label) {
    this.label = label;
  }
}
```

td.)

```
public String lookUpNames(String target, SearchFor category,
    SearchIn group, boolean exactMatch) throws IOException {

  String suffix = "";
  if (!exactMatch) suffix = ".";

  String prefix = category.label + " " + group.label;
  String query = prefix + target + suffix;

  Socket socket = new Socket();
  try {
    SocketAddress address = new InetSocketAddress(host, port);
    socket.connect(address);
    Writer out
        = new OutputStreamWriter(socket.getOutputStream(), "ASCII");
    BufferedReader in = new BufferedReader(new
        InputStreamReader(socket.getInputStream(), "ASCII"));
    out.write(query + "\r\n");
    out.flush();

    StringBuilder response = new StringBuilder();
    String theLine = null;
    while ((theLine = in.readLine()) != null) {
      response.append(theLine);
      response.append("\r\n");
    }
    return response.toString();
  } finally {
    socket.close();
  }
}
```

```
  public InetAddress getHost() {

    return this.host;
  }

  public void setHost(String host)
      throws UnknownHostException {
    this.host = InetAddress.getByName(host);
  }
}
```

GUI


Whois
Whois: oreilly.com
Exact Match
Find
Search for:
Any
Person
Domain
Group
ASN
Network
Host
Organization
Gateway
Search In:
All
Name
Mailbox
Handle
Search At:
whois.internic.net
Whois Server Version 1.3
Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.
Domain Name: OREILLY.COM
Registrar: BULKREGISTER, LLC.
Whois Server: whois.bulkregister.com
Referral URL: http://www.bulkregister.com
Name Server: NS1.SONIC.NET
Name Server: NS.OREILLY.COM
Status: ACTIVE
Updated Date: 17-oct-2002
Creation Date: 27-may-1997
Expiration Date: 26-may-2004
>>> Last update of whois database: Wed, 17 Dec 2003 06:31:24 EST <<<

Graphical Whois Client Interface

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import javax.swing.*;
public class WhoisGUI extends JFrame {

  private JTextField searchString = new JTextField(30);
  private JTextArea names = new JTextArea(15, 80);
  private JButton findButton = new JButton("Find");;
  private ButtonGroup searchIn = new ButtonGroup();
  private ButtonGroup searchFor = new ButtonGroup();
  private JCheckBox exactMatch = new JCheckBox("Exact Match", true);
  private JTextField chosenServer = new JTextField();
  private Whois server;

  public WhoisGUI(Whois whois) {
    super("Whois");
    this.server = whois;
    Container pane = this.getContentPane();

    Font f = new Font("Monospaced", Font.PLAIN, 12);
    names.setFont(f);
    names.setEditable(false);
```

```
JPanel centerPanel = new JPanel();
centerPanel.setLayout(new GridLayout(1, 1, 10, 10));
JScrollPane jsp = new JScrollPane(names);
centerPanel.add(jsp);
pane.add("Center", centerPanel);

// You don't want the buttons in the south and north
// to fill the entire sections so add Panels there
// and use FlowLayouts in the Panel
JPanel northPanel = new JPanel();
JPanel northPanelTop = new JPanel();
northPanelTop.setLayout(new FlowLayout(FlowLayout.LEFT));
northPanelTop.add(new JLabel("Whois: "));
northPanelTop.add("North", searchString);
northPanelTop.add(exactMatch);
northPanelTop.add(findButton);
northPanel.setLayout(new BorderLayout(2,1));
northPanel.add("North", northPanelTop);
JPanel northPanelBottom = new JPanel();
northPanelBottom.setLayout(new GridLayout(1,3,5,5));
northPanelBottom.add(initRecordType());
northPanelBottom.add(initSearchFields());
northPanelBottom.add(initServerChoice());
northPanel.add("Center", northPanelBottom);

pane.add("North", northPanel);
```

```
    pane.add("North", northPanel);

    ActionListener al = new LookupNames();
    findButton.addActionListener(al);
    searchString.addActionListener(al);
  }
  private JPanel initRecordType() {
    JPanel p = new JPanel();
    p.setLayout(new GridLayout(6, 2, 5, 2));
    p.add(new JLabel("Search for:"));
    p.add(new JLabel(""));

    JRadioButton any = new JRadioButton("Any", true);
    any.setActionCommand("Any");
    searchFor.add(any);
    p.add(any);

    p.add(this.makeRadioButton("Network"));
    p.add(this.makeRadioButton("Person"));
    p.add(this.makeRadioButton("Host"));
    p.add(this.makeRadioButton("Domain"));
    p.add(this.makeRadioButton("Organization"));
    p.add(this.makeRadioButton("Group"));
    p.add(this.makeRadioButton("Gateway"));
    p.add(this.makeRadioButton("ASN"));

    return p;
  }
```

```
private JRadioButton makeRadioButton(String label) {
  JRadioButton button = new JRadioButton(label, false);
  button.setActionCommand(label);
  searchFor.add(button);
  return button;
}

private JRadioButton makeSearchInRadioButton(String label) {
  JRadioButton button = new JRadioButton(label, false);
  button.setActionCommand(label);
  searchIn.add(button);
  return button;
}

private JPanel initSearchFields() {
  JPanel p = new JPanel();
  p.setLayout(new GridLayout(6, 1, 5, 2));
  p.add(new JLabel("Search In: "));

  JRadioButton all = new JRadioButton("All", true);
  all.setActionCommand("All");
  searchIn.add(all);
  p.add(all);

  p.add(this.makeSearchInRadioButton("Name"));
  p.add(this.makeSearchInRadioButton("Mailbox"));
  p.add(this.makeSearchInRadioButton("Handle"));
```

```
    return p;
}

private JPanel initServerChoice() {
    final JPanel p = new JPanel();
    p.setLayout(new GridLayout(6, 1, 5, 2));
    p.add(new JLabel("Search At: "));

    chosenServer.setText(server.getHost().getHostName());
    p.add(chosenServer);
    chosenServer.addActionListener( new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent event) {
            try {
                server = new Whois(chosenServer.getText());
            } catch (UnknownHostException ex) {
                 JOptionPane.showMessageDialog(p,
                    ex.getMessage(), "Alert", JOptionPane.ERROR_MESSAGE);
            }
        }
    } );

    return p;
}
```

```
private class LookupNames implements ActionListener {

  @Override
  public void actionPerformed(ActionEvent event) {
    names.setText("");
    SwingWorker<String, Object> worker = new Lookup();
    worker.execute();
  }
}

private class Lookup extends SwingWorker<String, Object> {

  @Override
  protected String doInBackground() throws Exception {
    Whois.SearchIn group = Whois.SearchIn.ALL;
    Whois.SearchFor category = Whois.SearchFor.ANY;

    String searchForLabel = searchFor.getSelection().getActionCommand();
    String searchInLabel = searchIn.getSelection().getActionCommand();

    if (searchInLabel.equals("Name")) group = Whois.SearchIn.NAME;
    else if (searchInLabel.equals("Mailbox")) {
      group = Whois.SearchIn.MAILBOX;
    } else if (searchInLabel.equals("Handle")) {
      group = Whois.SearchIn.HANDLE;
    }
```

```
  if (searchForLabel.equals("Network")) {
    category = Whois.SearchFor.NETWORK;
  } else if (searchForLabel.equals("Person")) {
    category = Whois.SearchFor.PERSON;
  } else if (searchForLabel.equals("Host")) {
    category = Whois.SearchFor.HOST;
  } else if (searchForLabel.equals("Domain")) {
    category = Whois.SearchFor.DOMAIN;
  } else if (searchForLabel.equals("Organization")) {
    category = Whois.SearchFor.ORGANIZATION;
  } else if (searchForLabel.equals("Group")) {
    category = Whois.SearchFor.GROUP;
  } else if (searchForLabel.equals("Gateway")) {
    category = Whois.SearchFor.GATEWAY;
  } else if (searchForLabel.equals("ASN")) {
    category = Whois.SearchFor.ASN;
  }

  server.setHost(chosenServer.getText());
  return server.lookUpNames(searchString.getText(),
     category, group, exactMatch.isSelected());
}
```

```
    @Override
    protected void done() {
      try {
        names.setText(get());
      } catch (Exception ex) {
        JOptionPane.showMessageDialog(WhoisGUI.this,
            ex.getMessage(), "Lookup Failed", JOptionPane.ERROR_MESSAGE);
      }
    }
  }

  public static void main(String[] args) {
    try {
      Whois server = new Whois();
      WhoisGUI a = new WhoisGUI(server);
      a.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
      a.pack();
      EventQueue.invokeLater(new FrameShower(a));
    } catch (UnknownHostException ex) {
      JOptionPane.showMessageDialog(null, "Could not locate default host "
          + Whois.DEFAULT_HOST, "Error", JOptionPane.ERROR_MESSAGE);
    }
  }
```

```
  private static class FrameShower implements Runnable {

    private final Frame frame;

    FrameShower(Frame frame) {
      this.frame = frame;
    }

    @Override
    public void run() {
     frame.setVisible(true);
    }
  }
}
```