J.73

THE I/O PACKAGE

Java I/O is defined in terms of streams.

Streams are ordered sequences of data that have a source and a destination.

I/O is defined in terms of classes and methods.

The most basic classes are:

- InputStream
- OutputStream
- RandomAccessFile

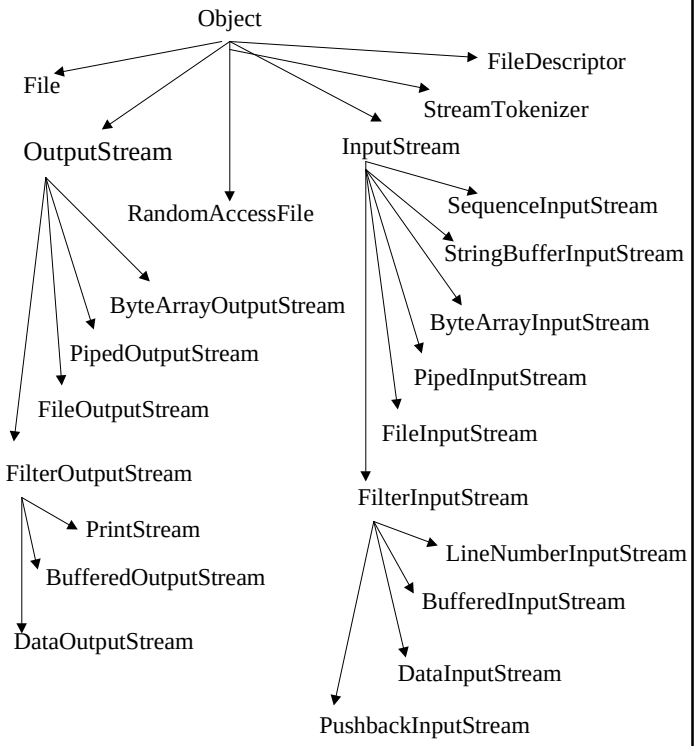
J.74

```
import java.io.*;

class Translate {
  public static void main(String[] args) {
  InputStream in = System.in;
  OutputStream out = System.out;

 if (args.length != 2) error("must provide from/to arguments");

  String from = args[0], to = args[1];    int ch, i;

  if (from.length() != to.length())
    error("from and to must be the same length");

    try {
       while ((ch = in.read()) != -1) {
         if   ((i = from.indexOf(ch)) != -1)
            out.write(to.charAt(i));
         else  out.write(ch);
         }
    }  catch(IOException e) {
        error("I/O Exception: " + e); }  }

public static void error(String err) {
      System.err.print("Translate: " + err);
      System.exit(1);  // Error return  }   }   }
```

```
Java  Translate  a  z
I am a dog.
I zm z dog.
```

J.75

I/O Class Hierarchy



J.76

EVENTS

If an applet is interactive, it must be able to receive and respond to user input:

- mouse clicks (down, up, click)
- mouse movements (position, drags)
- key presses (press, release, type)
- user interface events (buttons, menus, etc.)
- window events (open, close, exit)

Events are the devices provided by Java to handle these things.

J.77

Java Event Models

- Unfortunately, there are now two event models.

- The Java 1.0 model is simple and well-suited to writing basic applets, but does not scale well to complicated interfaces.

- The Java 1.1 model solves many of the shortcomings of the 1.0 model, but is not yet supported by many browsers.

- We suggest you stick to 1.0 at this time, but those who are adventurous may want to try 1.1.

J.78

The Java 1.0 Event Model

- All events are represented by the Event class, which has instance variables that describe the event:

  - id: the type of event
  - Event: possible values of id
  - target: object generating the Event

  - other fields for specific types of Events

- Java 1.0 events are dispatched first to the handleEvent() method of the Component object on which they occur.

- The event methods return boolean values: true if the event has been handled, else false.

1.0 EXAMPLE

J.79

```
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Event;
import java.awt.Point;

public class Lines extends java.applet.Applet {

final int MAXLINES = 20;
Point starts[] = new Point[MAXLINES];
Point ends[] = new Point[MAXLINES];
Point anchor;
Point currentpoint;
int currline = 0;

public void init() {
  setBackground(Color.white);
}

public boolean mouseDown(Event evt, int x, int y) {

  if (currline < MAXLINES) {
      anchor = new Point(x,y);
      return true;
      }
  else {
      System.out.println("Too many lines.");
      return false;
      }
}
```

J.80

```
public boolean mouseUp(Event evt, int x, int y) {

 if (currline < MAXLINES) {
       addline(x,y);
       return true;
       }
       else return false;
 }

 public boolean mouseDrag(Event evt, int x, int y) {

   if (currline < MAXLINES) {
       currentpoint = new Point(x,y);
       repaint();
       return true;
       }
       else return false;
 }

 void addline(int x, int y) {
   starts[currline] = anchor;
   ends[currline] = new Point(x,y);
   currline++;
   currentpoint = null;
   anchor = null;
   repaint();
 }
```

J.81

```
public void paint(Graphics g) {

  /* Draw existing lines */

  for (int i = 0; i < currline; i++) {
    g.drawLine(starts[i].x, starts[i].y, ends[i].x, ends[i].y);
  }

  /* Draw the current line */

  g.setColor(Color.blue);

  if (currentpoint != null)
    g.drawLine(anchor.x, anchor.y, currentpoint.x, currentpoint.y);
  }

}
```

J.82

The Java 1.1 Event Model

- The Java 1.1 event model is used by both AWT and Java Beans.
- Different classes of events are represented by different Java classes.
- Every event is a subclass of java.util.EventObject.
- AWT events are subclasses of java.awt.AWTEvent.
- Every event has a source object, which can be obtained with getSource().
- Every AWT event has a type value, which can be obtained with getID() and which distinguishes the types of events in one class.

J.83

Event Listener

- An object interested in receiving events is an event listener.
- An object that generates events is an event source.
- An event source maintains a list of listeners who want to be notified when the event occurs.
- When a user input event occurs on the event source, it notifies all the listeners.

J.84

1.1 Example

```
import java.awt.Graphics;
import java.awt.Color;
import java.awt.event.*;
import java.awt.Point;

public class LinesNew extends java.applet.Applet
  implements MouseListener,MouseMotionListener {

final int MAXLINES = 20;
Point starts[] = new Point[MAXLINES];
Point ends[] = new Point[MAXLINES];
Point anchor;
Point currentpoint;
int currline = 0;

public void init() {
  setBackground(Color.white);

/* Register event Listeners */

addMouseListener(this);
addMouseMotionListener(this);
}
```

J.85

```
/* Signatures Needed for Listener Interfaces */

  public void mouseMoved(MouseEvent e) {}
  public void mouseClicked(MouseEvent e) {}
  public void mouseEntered(MouseEvent e) {}
  public void mouseExited(MouseEvent e) {}

  /* Replaces mouseDown */

  public void mousePressed(MouseEvent e) {

    if (currline < MAXLINES)
        anchor = new Point(e.getX(),e.getY());
    else
        System.out.println("Too many lines.");
  }

  /* Replaces mouseUp */

  public void mouseReleased(MouseEvent e) {

  if (currline < MAXLINES)
        addline(e.getX(),e.getY());
  }
```

J.86

```
/* Replaces mouseDrag */

 public void mouseDragged(MouseEvent e) {

    if (currline < MAXLINES) {
        currentpoint = new Point(e.getX(),e.getY());
        repaint();
        }
 }

 void addline(int x, int y) {
    starts[currline] = anchor;
    ends[currline] = new Point(x,y);
    currline++;
    currentpoint = null;
    anchor = null;
    repaint();
 }
```

J.87

```
public void paint(Graphics g) {

  /* Draw existing lines */

  for (int i = 0; i < currline; i++) {
    g.drawLine(starts[i].x, starts[i].y, ends[i].x, ends[i].y);
  }

  /* Draw the current line */

  g.setColor(Color.blue);

  if (currentpoint != null)
    g.drawLine(anchor.x, anchor.y, currentpoint.x,
               currentpoint.y);
  }

}
```

J.88