AWT Enhancements

- A (much!) better AWT
  - new and better event model
  - lightweight components
  - printing
  - data transfer
    - cut & paste, drag & drop
  - popup menus
  - menu shortcuts
  - focus traversal
  - better:
    - color, fonts, cursors, scrolling, images, clipping

AWT Enhancements

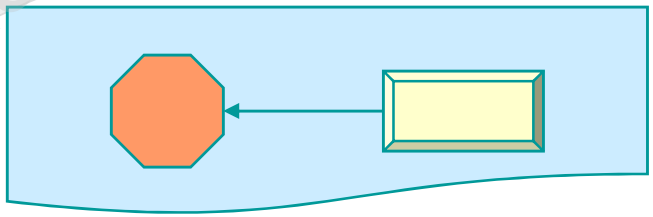
- The Java 1.1 Event Model
  - original 1.0 method didn't cope with complex UIs well
  - design problem manifested as frequently requiring switch statements
    - not good OO 'style'

```
public boolean handleEvent (Event e)
  {
  if (e.target == scrollBar)
    ...
  else if (e.target == button)
    ...
  else if (e.target == menu)
    ...
```

  - (important!) couldn't easily tell what events a component generates
    - big problem for JavaBeans

AWT Enhancements

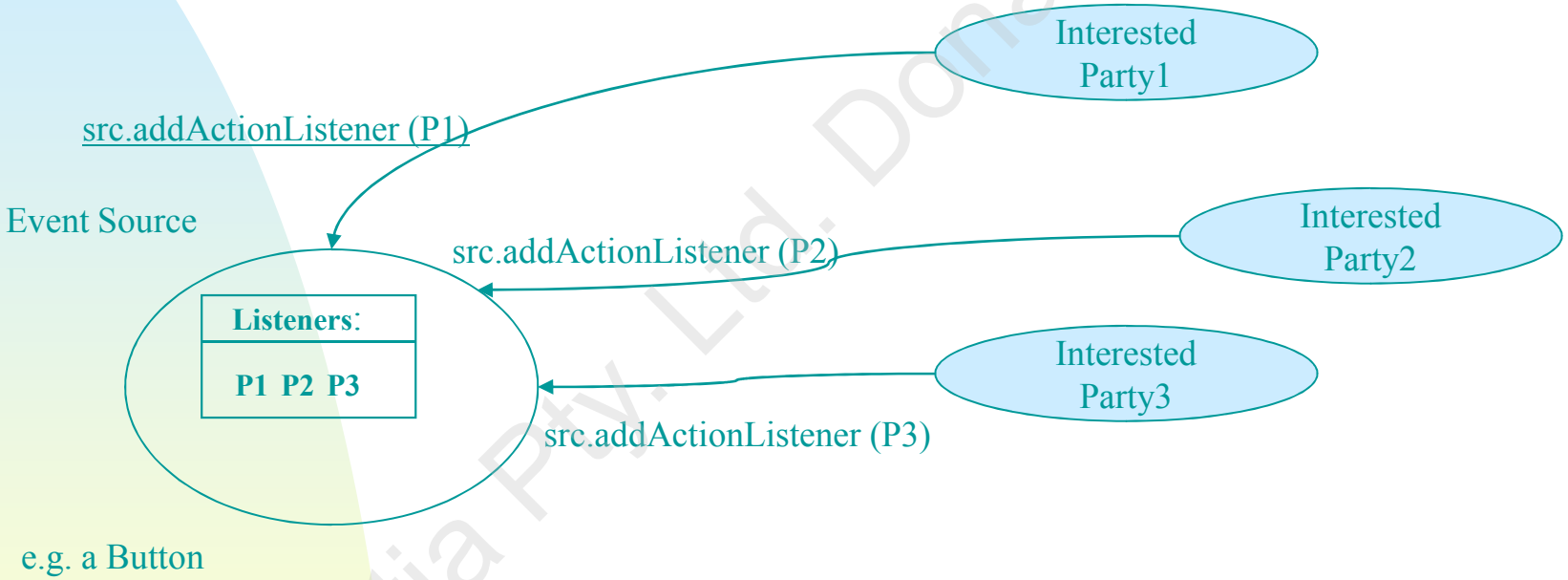
- delegation to the rescue!
  - different classes of event get different Java classes to deal with/define them
    - object creates a component
    - registers (assigns) a listener for the component
      - often implemented by an inner class
    - when component is activated, it fires its (specific class of) event at the registered listener
    - listener performs the requisite action delegated to it by (on behalf of) the creator



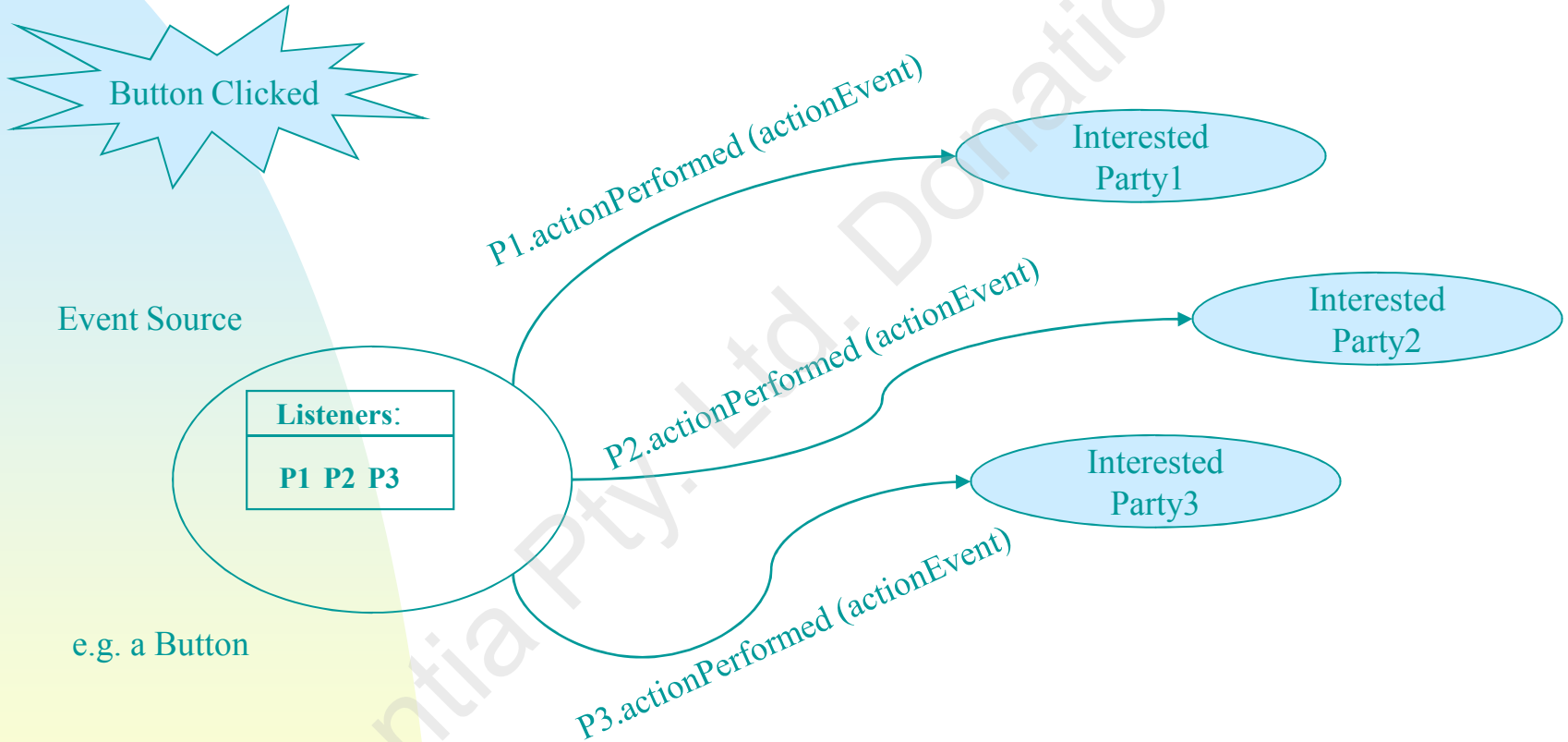
AWT Enhancements

- java.awt.event package
  - 10 event classes
    - ActionEvent, MouseEvent, WindowEvent, etc.
      - event carries around all relevant information regarding its origin, etc.
      - getSource (), getX (), getKeyChar (), etc.
  - 11 listener interfaces
    - for objects interested in receiving events
  - 7 adapter classes
    - implement interfaces specifying multiple methods
    - all methods supplied as ‘no-ops’
    - lets programmer subclass the adapter and implement only immediately relevant functionality
      - facilitates convenient use of inner classes

AWT Enhancements



AWT Enhancements



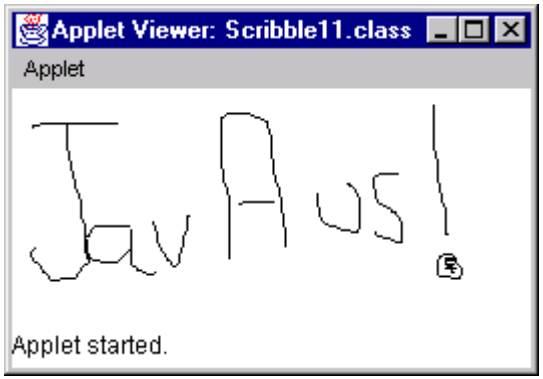
AWT Enhancements

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

public class Scribble11 extends java.applet.Applet
  {
  int last_x = 0, last_y = 0;

  public void init ()
    {
    addMouseListener
      (
      new MouseAdapter()
        {
        public void mousePressed (MouseEvent e)
          { last_x = e.getX (); last_y = e.getY (); }
        }
      );

    addMouseMotionListener
      (
      new MouseMotionAdapter ()
        {
        public void mouseDragged (MouseEvent e)
          {
          Graphics g = getGraphics ();
          int x = e.getX (), y = e.getY ();
          g.setColor (Color.black);
          g.drawLine (last_x, last_y, x, y);
          last_x = x; last_y = y;
          }
        }
      );
    }
  }
```



AWT Enhancements

Listener Interface	Adapter Class	Methods
ActionListener	none	actionPerformed(ActionEvent)
AdjustmentListener	**none**	**adjustmentValueChanged(AdjustmentEvent)**
ComponentListener	ComponentAdapter	componentHidden(ComponentEvent)
		componentMoved(ComponentEvent)
		componentResized(ComponentEvent)
		componentShown(ComponentEvent)
ContainerListener	**ContainerAdapter**	**componentAdded(ContainerEvent)**
		componentRemoved(ContainerEvent)
FocusListener	FocusAdapter	focusGained(FocusEvent)
		focusLost(FocusEvent)
ItemListener	**none**	**itemStateChanged(ItemEvent)**
KeyListener	KeyAdapter	keyPressed(KeyEvent)
		keyReleased(KeyEvent)
		keyTyped(KeyEvent)
MouseListener	**MouseAdapter**	**mouseClicked(MouseEvent)**
		mouseEntered(MouseEvent)
		mouseExited(MouseEvent)
		mousePressed(MouseEvent)
		mouseReleased(MouseEvent)
MouseMotionListener	MouseMotionAdapter	mouseDragged(MouseEvent)
		mouseMoved(MouseEvent)
TextListener	**none**	**textValueChanged(TextEvent)**
WindowListener	WindowAdapter	windowActivated(WindowEvent)
		windowClosed(WindowEvent)
		windowClosing(WindowEvent)
		windowDeactivated(WindowEvent)
		windowDeiconified(WindowEvent)
		windowIconified(WindowEvent)
		windowOpened(WindowEvent)

AWT Enhancements

- delegation style may give rise to hundreds of tiny, special-purpose classes
  - awkward
  - polluting
  - time consuming
  - error prone
- this is what inner classes were introduced for…

```
public void init ()
    {
    addMouseListener
      (
      new MouseAdapter()
        {
        public void mousePressed (MouseEvent e)
          { last_x = e.getX (); last_y = e.getY (); }
        }
      );
    }
```

AWT Enhancements

- there also exists a low-level interface to AWT 1.1 events
  - similar to 1.0
  - more traditional
    - enable events of interest
    - switch on incoming event's ID to handle it
  - not covered here...

AWT Enhancements

- Lightweight components
  - a class that directly extends one of
    - Component
    - Container
  - uses:
    - to create a component that isn't rectangular
    - to create a custom component that's partly (or entirely) transparent
    - when it is necessary to have multiple components that can draw in each others' areas
    - when a program contains many components and would be too inefficient otherwise

AWT Enhancements

- Printing
  - in AWT 1.0, applets couldn't be printed
    - browser just left white space
  - new in 1.1:
    - java.awt.PrintJob class
    - print () and printAll () applet methods
      - for applets, default method just calls paint () with graphics context supplied by context (i.e. Browser)
      - printAll () traverses entire hierarchies of components and calls print () for everything
  - pagination, etc. left up to applet/application

AWT Enhancements

```
public void actionPerformed (ActionEvent e)
  {
  PrintJob pjob =
      getToolkit ().getPrintJob (this, "Printing Test", prefs);
  if (pjob != null)
    {
    Graphics pg = pjob.getGraphics();
    if (pg != null)
      {
      canvas.printAll(pg);
      pg.dispose(); // flush page
      }
    pjob.end();
    }
  }
```

user's printing preferences *(Property)*

AWT Enhancements

- PrintGraphics interface used to detect whether rendering to screen or to a print device
  - wouldn't want to highlight current selection in hard-copy version, for example

```
public void paint(Graphics g)
  {
  if (g instanceof PrintGraphics)
    {
    // printing is occurring
    …
```

- applets can't initiate printing, only respond to request from context
  - otherwise you'd get printing virii...

AWT Enhancements

- JDK 1.2 introduced java.awt.print package
  - PrinterJob class
    - printDialog, getPrinterJob, defaultPage, validatePage
  - Paper class
    - setSize, setImageableArea
  - PageFormat class
    - setPaper, getHeight, etc.
  - Book class
    - *“represents a document in which pages may have different page page formats and page painters”*
    - append, getNumberOfPages
  - Printable interface
    - defines print method

AWT Enhancements

```
import java.awt.*;
import java.awt.event.*;
import java.awt.print.*;
import javax.swing.*;
public class PrintDemo extends JFrame
  {
  public static void main (String [] s)
    {
    PrintDemo p = new PrintDemo ();
    p.addWindowListener
      (
      new WindowAdapter ()
        {
        public void windowClosing (WindowEvent e)
          {System.exit (0);}
        }
      );
    p.setSize (new Dimension (200, 80));
    p.setVisible (true);
    }
  public PrintDemo ()
    {
    setTitle (getClass ().getName ());
    PrintingJButton b = new PrintingJButton ("Hello Printing World");
    b.addActionListener
      (
      new ActionListener ()
        {
        public void actionPerformed (ActionEvent e)
          {
          PrinterJob printJob = PrinterJob.getPrinterJob ();
          printJob.setPrintable ((PrintingJButton) e.getSource ());
          PageFormat def = printJob.defaultPage (),
                     pf = printJob.pageDialog (def);
          if (! pf.equals (def))  // if pf.equals (def), dialog was cancelled by user
            if (printJob.printDialog ())
              try { printJob.print (); }
              catch (Exception pE) { pE.printStackTrace (System.err); }
          }
        }
      );
    getContentPane ().add (b);
    }
```

```
class PrintingJButton extends JButton implements Printable
  {
  public PrintingJButton (String s) { super (s); }
  public int print (Graphics g, PageFormat pf, int pi)
    throws PrinterException
    {
    if (pi >= 1)
      return Printable.NO_SUCH_PAGE;

    Graphics2D g2 = (Graphics2D) g;
    g2.translate(pf.getImageableX(), pf.getImageableY());
    Font f = Font.getFont ("SansSerif");
    g2.setFont (f);
    paint (g2);
    return Printable.PAGE_EXISTS;
    }
  }
}
```

AWT Enhancements

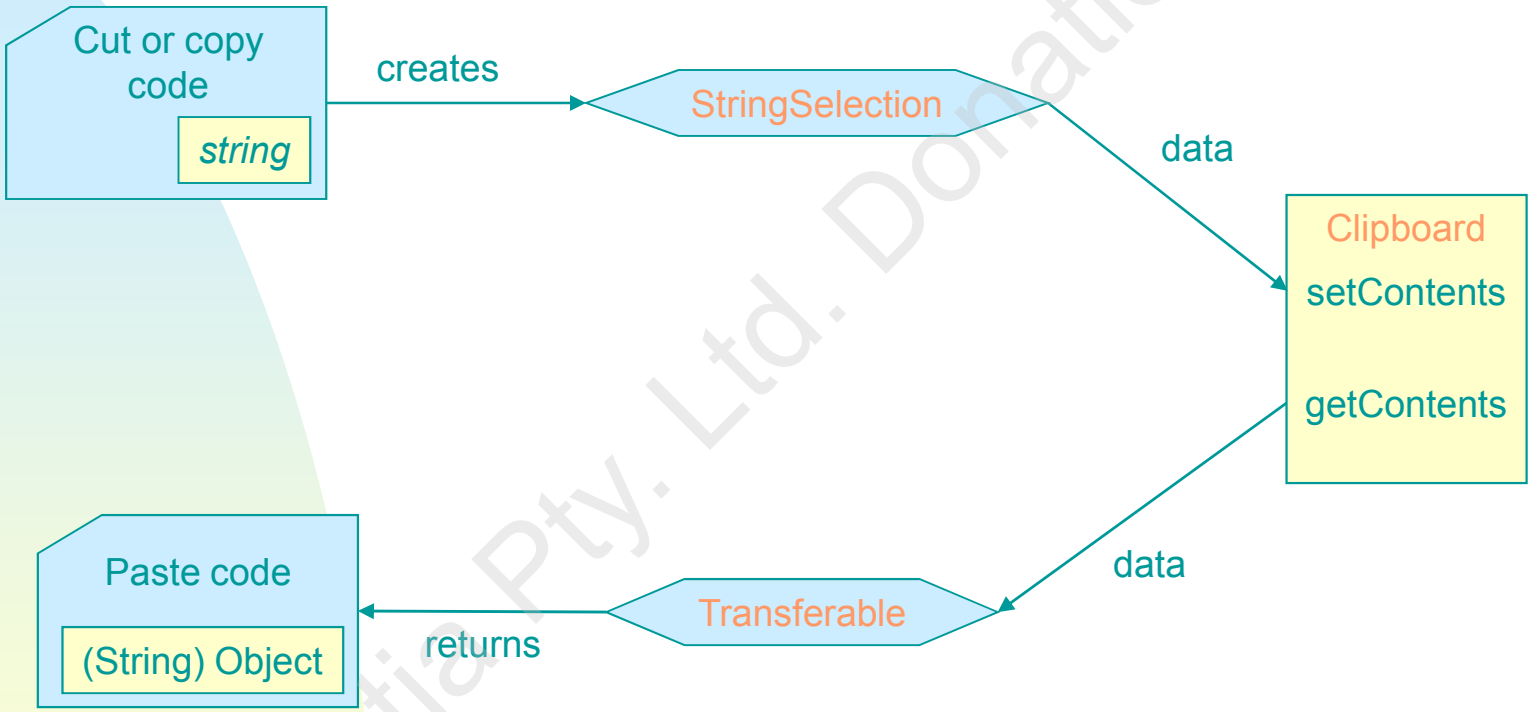
- Data transfer
  - java.awt.datatransfer package
  - cut & paste
  - drag & drop to come…
  - several aspects:
    - Clipboard class
    - StringSelection class
    - DataFlavor class
    - Transferable interface
    - ClipboardOwner interface
    - UnsupportedFlavorException

AWT Enhancements

- DataFlavor
  - human-readable representation of transmission format
    - [name, data-type specification]

- Transferable interface
  - must be implemented by any object that wants to make data available for transfer. Includes:
    - getTransferDataFlavors ()
    - isDataFlavorSupported ()
    - getTransferData ()

AWT Enhancements



AWT Enhancements

```
public void copy () {
  Clipboard c = this.getToolkit ().getSystemClipboard ();
  StringSelection s = new StringSelection (..., DataFlavor.stringFlavor);
  c.setContents (s, s);
  }

public void cut () { copy (); clear ();  }

public void paste () {
  Clipboard c = this.getToolkit ().getSystemClipboard ();
  Transferable t = c.getContents (this);
  if (t == null) {
    this.getToolkit ().beep ();
    return;
  }
  try {
    String s = (String) t.getTransferData (DataFlavor.stringFlavor);
    }
  catch (final UnsupportedFlavorException e) {
    this.getToolkit ().beep ();
    }
  catch (final Exception e) {
    this.getToolkit ().beep ();
    }
  }
```

s is notified when the content of the clipboard is set to something else

AWT Enhancements

- Applets can't access the system clipboard
  - security risk
- system clipboard limited to String/textual flavors