# 17-June-09 Announcements

- A2 Demos (cont)
- Code for today's cut 'n paste/drag 'n drop demo is on the web site. ClipboardDemo.java is an old demo that doesn't make use of recent Java approaches. TransferDemo.zip does (and is fuller-featured, to boot).

University of Waterloo

# CS 349
# Cut ‘n Paste; Drag ‘n Drop

Byron Weber Becker
Spring 2009

Slides mostly by Michael Terry

Demo reworked/expanded/replaced by Byron Weber Becker

University of Waterloo

# Transferring Data

- Cut and paste via the clipboard and drag and drop allows for (relatively) easy data transfer within and between applications
- Expected behavior of any application
- Demo

# The Clipboard

- Ubiquitous data transfer method
  - Copy information (or pointer to information) to clipboard
  - Other applications can read data clipboard
- *Any* application can read this information
  - A potential security risk
  - Clipboard not accessible to Java applets running in web browser
- Requires common data formats to work seamlessly
  - Text is no problem
  - What about other formats?

University of Waterloo

# The Clipboard: Formats

- Consider graphics
- How do we deal with:
  - Drawings in vector-based drawing programs?
  - Bitmap images?
  - Images from different file formats (JPEG, TIFF, GIF…)
  - 3D graphics?
  - PostScript drawings?
  - Charts?
  - Proprietary graphics formats?

University of Waterloo

# The Clipboard

- When data is placed on clipboard, application indicates the formats in which it can provide the data
  - Example: “I can provide it as a vector image, bitmap image, or as text”
- Data is not always copied to clipboard immediately
  - Why not?
  - What are implications?

# Placing Data on Clipboard

- Data may be available in many formats
  - Wasteful to put all formats on clipboard at once
- Data may never be pasted
  - Again, wasteful to commit memory to a copy unless it is needed
- If data is not immediately placed on clipboard:
  - Must create a copy if user changes data locally
  - Must put it on clipboard if application exits
- Clipboard a function of the underlying windowing system, toolkit
  - Java will do it differently from Cocoa from Windows…

# Java Clipboards

- Relevant packages:
  - java.awt.datatransfer (Clipboard, Drag and Drop)
  - java.awt.dnd (Drag and Drop support)
- Relevant classes
  - Clipboard
  - DataFlavor
  - Transferable
  - Toolkit

University of Waterloo

# Java Clipboards

- Local and system clipboards
- Local clipboards are named clipboards holding data only accessible by the application
  - new Clipboard(“My clipboard”);
- System clipboard is operating-system-wide clipboard
  - Toolkit.getDefaultToolkit().getSystemClipboard()
- System clipboard not available to applets

University of Waterloo

# Copying Data to Clipboard

Basic steps:

1. Get clipboard
2. To copy, create a Transferable object
   - Defines methods for responding to queries about what data formats (DataFlavors) are available
   - Defines method for getting data of specified type
3. Set clipboard contents to the new Transferable object

- Transferable object encapsulates all the data to handle the copy operation later
  - Similarities to what other object?

# Transferable

- Encapsulates all data to copy object
- Similar in spirit to UndoableEdit
- Methods:
  - DataFlavor[ ] getTransferDataFlavors( )
  - boolean isDataFlavorSupported(DataFlavor flavor)
  - Object getTransferData(DataFlavor flavor)

# Pasting Data from Clipboard

- Basic steps:
  1. Get clipboard
  2. See if it supports the desired data format (DataFlavor)
  3. Get the data, casting it to the proper Java object

# Code Review: Cut ‘n Paste

- DTPicture
  - first half: setting, painting image, focus highlighting
- TransferDemo
  - doCopyOrCut
  - Related
    - selectedPic
    - PicFocusListener
  - doPaste
- PictureTransferable

University of Waterloo

# TransferHandler

- The TransferHandler class will be used for drag ‘n drop. It can also be used for supporting cut ‘n paste.
- The cut ‘n paste support:
  - providing Action objects (actionListeners) for cut/copy/paste
  - exportToClipboard
- See Java Tutorial for more info
  - The current version is oriented towards Java 6; at least one example didn’t work on the VM nor on my Mac.

University of Waterloo

# Drag and Drop

- Uses same Transferable, DataFlavor objects to pass information around
- Need to specify drag and drop sources

University of Waterloo

# Supporting Drag

- "Dragging" refers to copying something *from* your control
- To support dragging:
  - Create a DragSource object
  - Create a default drag gesture recognizer on object (createDefaultDragGestureRecognizer)
  - Pass in a DragGestureListener
  - DragGestureListener starts the actual drag and specifies the data represented in the drag operation
- Some of this is now handled by the TransferHandler class.

University of Waterloo

# Supporting Drop

- Drop support allows stuff to be dropped on component
- To support dropping:
  - Create DropTarget object
  - Provide support for dragEnter, drop, dragExit, dragOver, dropActionChanged events
  - Many of these involve querying DataFlavors
  - To accept drop, will need to transfer data from Transferable, just like in Pasting
  - Set component's drop target to the DropTarget object just created
- Much of this is now handled by TransferHandler.

University of Waterloo

# TransferHandler

- Methods:
  - boolean importData(JComponent c, Transferable t)
  - int getSourceActions(JComponent c)
    - returns one of COPY, MOVE, or COPY_OR_MOVE
  - Transferable createTransferable(JComponent c)
  - void exportAsDrag(JComponent c, InputEvent e, int action)
    - action is one of COPY, MOVE, or COPY_OR_MOVE
  - void exportDone(JComponent source, Transferable data, int action)

University of Waterloo

# Code Review: Drag ‘n Drop

- TransferDemo
  - Set handler in constructor
- DTPicture
  - DragGesture inner class
- PictureTransferHandler
  - dropping (canImport, importData)
  - dragging (getSourceActions, createTransferable, exportDone)

University of Waterloo

# Politics of Data Formats

- Data formats can be instruments of control
  - The value of an application resides in its ability to create, manipulate, manage, and reference data
  - The more that an individual or company's worth is tied up in data, the more reliant they become on the tools that allow them to access and manipulate that data
  - This creates a market *disincentive* to create open data formats
  - Why?