# Clipboard and Drag-and-Drop

## Data Transfer

- Methods to enable “user-interface level data transfer” within an application and between applications
  – clipboard (copy, cut, paste)
  – drag-and-drop (drag data from one view/application to another)

## Clipboard Transfer

- Data transfer method using a (system-level) **generic data buffer**
  - Copy/Cut data from document to clipboard
  - Paste data from clipboard to document
- Like undo, clipboard ”copy and paste” is *expected in a GUI*
- Clipboard design and access
  - access to clipboard contents a potential security risk?
  - how to handle different data, and different formats?
    (copy text, image, data table, HTML, SVG, …)

## Clipboard Supported Data Formats

- When data is placed on clipboard, application indicates formats
  - data can be provided as vector image, bitmap image, text, …
- Need way to deal with:
  - Formatted text like HTML, RTF, MS Office, …
  - Vector-based drawing? (SVG, Illustrator, …)
  - Images in different file formats (JPG, PNG, TIF, …)
  - PostScript/PDF drawings?
  - Tables? Charts? Grouped objects? Filters?
  - Proprietary graphics formats? (Photoshop layers)
  - 3D meshes? Video?
- MacOS Human Interface Guidelines specify all application must:
  - at least support plaintext or image on clipboard
  - at least accept plaintext or image from clipboard

## Placing Data on Clipboard

- Data is (usually) not put into clipboard immediately
  - multiple data formats could take space (e.g. image)
  - clipboard may never be pasted (use “cut” like delete)
- Application manages clipboard data until “paste” occurs
  - If application exits, it may put all data into clipboard regardless

## Java Clipboard API

- Clipboard and drag and drop package:
  `java.awt.datatransfer`
- Key classes:
  `Clipboard`
  `DataFlavor`
  `Transferable`
- Supports 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()`

## Steps to Copy Data to Clipboard

1. Get clipboard

```
Clipboard cb = Toolkit.getDefaultToolkit()
    .getSystemClipboard();
```

2. Create a Transferable object
   - methods to list/query supported data formats
   - method to get data in specified format

```
Transferable transferObject = new Transferable() { … }
```

1. Set clipboard contents to Transferable object

```
  cb.setContents(transferObject, this);
```

## Transferable Object

- Encapsulates all data to copy
  - Command pattern, similar in spirit to UndoableEdit
- Data formats are called “flavors”
- Data format methods:

```
DataFlavor[ ] getTransferDataFlavors()
boolean isDataFlavorSupported(DataFlavor flavor)
Object getTransferData(DataFlavor flavor)
```

## Creating a Transferable

```
Transferable transferObject = new Transferable() {

    // text data to put on clipboard
    private String text = textArea.getSelectedText();

    public Object getTransferData(DataFlavor flavor) {
        // could convert data to whatever you want here
        if (flavor.equals(DataFlavor.stringFlavor)) {
            return text;
        }
        throw new UnsupportedFlavorException(flavor);
    }

    public DataFlavor[] getTransferDataFlavors() {
        return new DataFlavor[] { DataFlavor.stringFlavor };
    }

    public boolean isDataFlavorSupported(DataFlavor flavor) {
        return flavor.equals(DataFlavor.stringFlavor);
    }
};
```

## Steps to Paste Data from Clipboard

1. Get clipboard

```
Clipboard cb = Toolkit.getDefaultToolkit()
    .getSystemClipboard();
```

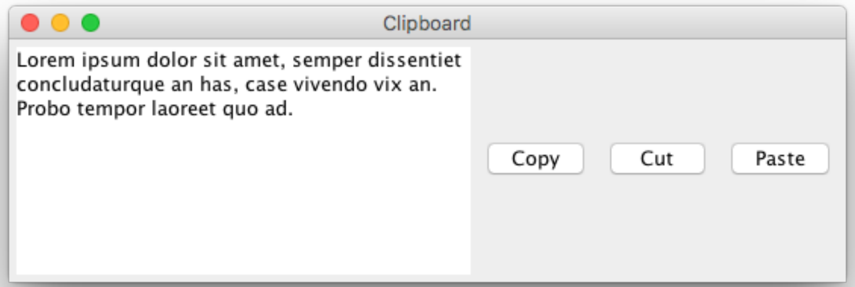
2. See if clipboard supports desired data format (DataFlavor)

```
if (cb.isDataFlavorAvailable(DataFlavor.stringFlavor)) {
```

3. Get the data, casting it to the proper Java object

```
String t = (String)cb.getData(DataFlavor.stringFlavor);
textArea.replaceSelection(t);
```

## Code Demo: Cut-and-Paste

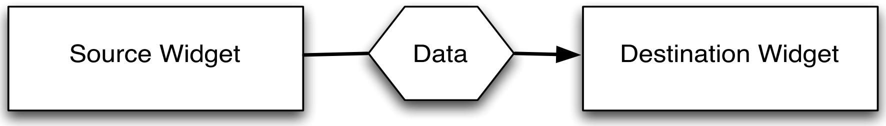


Lorem ipsum dolor sit amet, semper dissentiet concludaturque an has, case vivendo vix an. Probo tempor laoreet quo ad.

```
COPY: `Lorem`
COPY: set system clipboard to Transferable
  Transferable.getTransferDataFlavors
  Transferable.getTransferDataFlavors
  Transferable.getTransferData as java.awt.datatransfer.DataFlavor[mimetype=applicatio
  Transferable.getTransferData as java.awt.datatransfer.DataFlavor[mimetype=applicatio
  Transferable.getTransferData as java.awt.datatransfer.DataFlavor[mimetype=applicatio
PASTE
  Transferable.getTransferDataFlavors
PASTE: 1 available flavours ...
  Transferable.getTransferDataFlavors
  Unicode String  java.awt.datatransfer.DataFlavor[mimetype=application/x-java-seriali
  Transferable.isDataFlavorSupported: java.awt.datatransfer.DataFlavor[mimetype=applic
PASTE: DataFlavor.stringFlavor available
  Transferable.getTransferData as java.awt.datatransfer.DataFlavor[mimetype=applicatio
PASTE: 'Lorem'
```

## Drag-and-Drop

- Also uses Transferable, DataFlavor objects
- Attach a TransferHandler to each widget to manage data transfer
- Need to detect the start-of-drag gesture

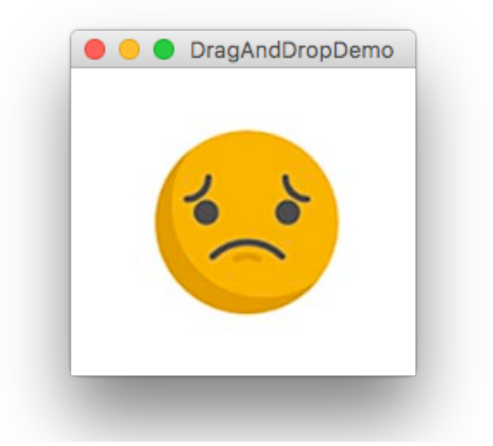


- has a TransferHandler that implements createTransferable and exportDone
- has a mouseListener to detect the start-of-drag gesture

- has a TransferHandler that implements importData

## **Supporting Drop** (in Drag-and-Drop)

- "Drop" refers to pasting data **into** your widget at end of drag
- To support dropping:
  1. Create a TransferHandler
  2. Override TransferHandler.importData method
  3. Set the TransferHandler on the widget



## **DragAndDropDemo.java**

```
// create image transfer handler
private class ImageTransferHandler extends TransferHandler {
   ...
   public boolean importData(JComponent c, Transferable t) {
        if (t.isDataFlavorSupported(DataFlavor.imageFlavor)) {
            image = (Image)t.getTransferData(DataFlavor.imageFlavor);
            label.setIcon(new ImageIcon(image));
            return true;
        }
        return false;
    }
   ...
}


// In view setup
imageLabel = new JLabel();
imageLabel.setOpaque(true);
imageLabel.setTransferHandler(new ImageHandler());
```

## **Supporting Drag** (in Drag-and-Drop)

- "Drag" refers to copying data **out of** your widget at start of drag
- Steps to add "Drag" support:
  1. Create a TransferHandler
  2. Override createTransferable with data Transferable
  3. Set the TransferHandler on the widget
  4. Define a mouse listener that knows when a drag **starts**
  5. On drag, get widget's transfer handler and call exportAsDrag

## DragAndDropDemo.java

```java
// create an image transfer handler
private class ImageTransferHandler extends TransferHandler {

    // create Transferable for handler
    protected Transferable createTransferable(JComponent c) {
        return new Transferable() {
            private Image img =
                    ((ImageIcon)imageLabel.getIcon()).getImage();

            public Object getTransferData(DataFlavor flavor) {
                if (flavor.equals(DataFlavor.imageFlavor)) {
                    return this.img;
                }
                throw new UnsupportedFlavorException(flavor);
            }
          ...
          };
     }
```

## DragAndDropDemo.java

```java
// A simple recognizer for the drag gesture
private class DragGesture extends MouseInputAdapter {

    private boolean armed = true;

    public void mouseDragged(MouseEvent e) {
        // Initiate drag and drop at start of drag only
        if (armed) {
            JComponent c = (JComponent)e.getSource();
            TransferHandler handler = c.getTransferHandler();
            handler.exportAsDrag(c, e, TransferHandler.COPY);
            armed = false;
        }
    }

    public void mouseReleased(MouseEvent e) {
        armed = true;
    }
}
```

## TransferHandler Methods

```
Transferable createTransferable(JComponent c)
```

```
boolean importData(JComponent c, Transferable t)
```

```
int getSourceActions(JComponent c)
```

- returns one of COPY, MOVE, or COPY_OR_MOVE

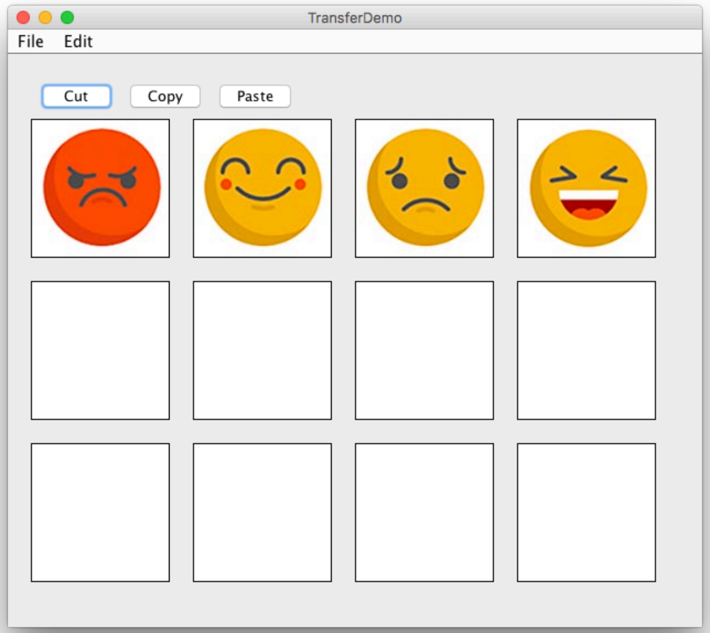
```
void exportAsDrag(JComponent c, InputEvent e,  int action)
```

- action is one of COPY, MOVE, or COPY_OR_MOVE

```
void exportDone(JComponent source, Transferable data,
         int action)
```

## TransferDemo.java

- Clipboard and Drag-and-Drop combined


TransferDemo
File
Edit
Cut
Copy
Paste