AOT LAB

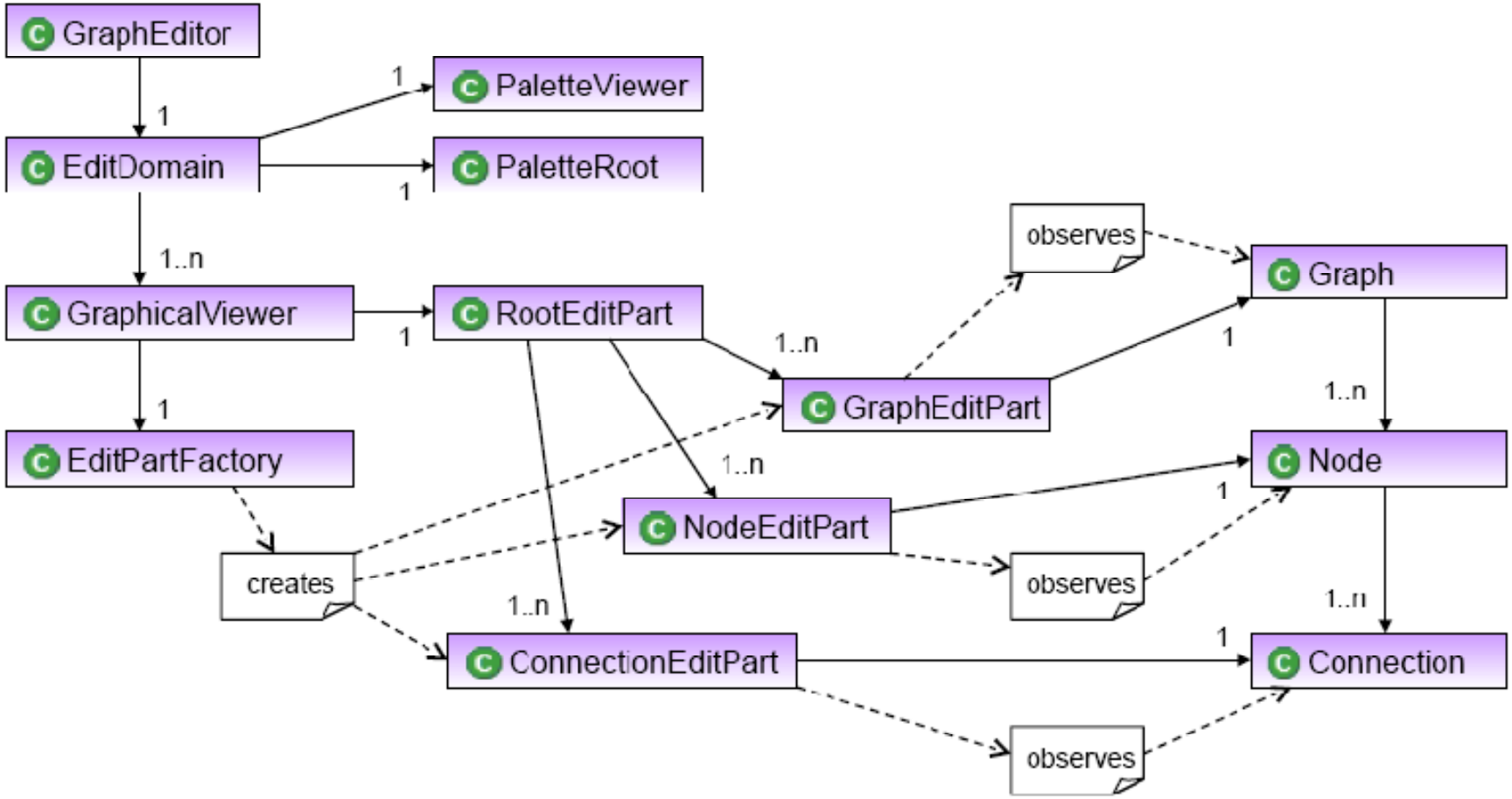
Agent and **O**bject **T**echnology **Lab**
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

GEF Tutorial

Alessandro Negri
negri@ce.unipr.it
http://www.ce.unipr.it/people/negri/

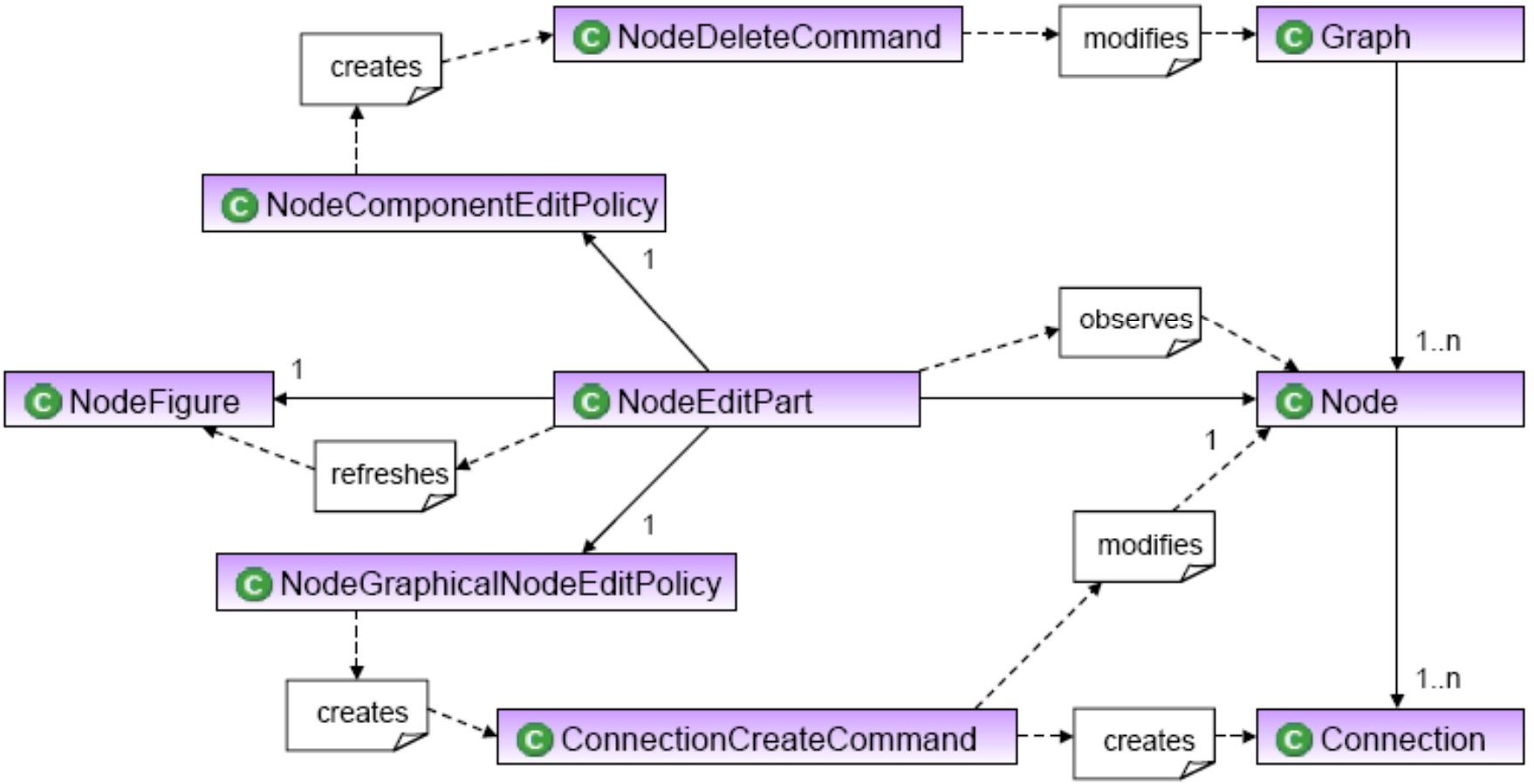
AOT LAB

GEF Editor Structure



AOT LAB

GEF MVC Interactions



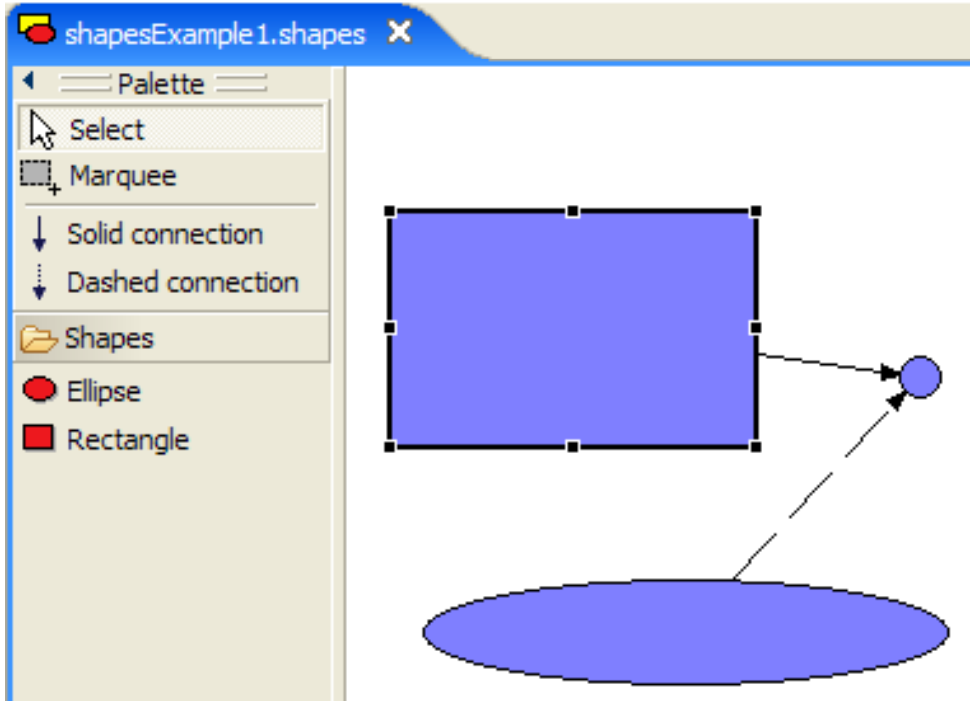
AOT LAB

A Minimal Shapes GEF Editor

- Creation of Eclipse plug-in with Editor
- Add the GraphicalViewer: GraphicalViewer
  - Special kind of EditPartViewer
  - EditPartViewers are adapters for SWT controls that manage the EditParts
  - They are populated by setting their contents
- Add the RootEditPart: ScalableFreeformRootEditPart
  - Brigdes gap between EditPartViewer and its contents
  - Can provide services such as zooming and freeform figures
- Define the model to be used: ShapesDiagram
- Create the view and the controller: ShapeEditPart
- Define the ShapeEditPartFactory

Today's Activity

- Create your own plug-in
  - Based on the simple Shapes example contributed by Elias Volanakis (included into GEF official examples)
  - Skeleton provided for the plug-in

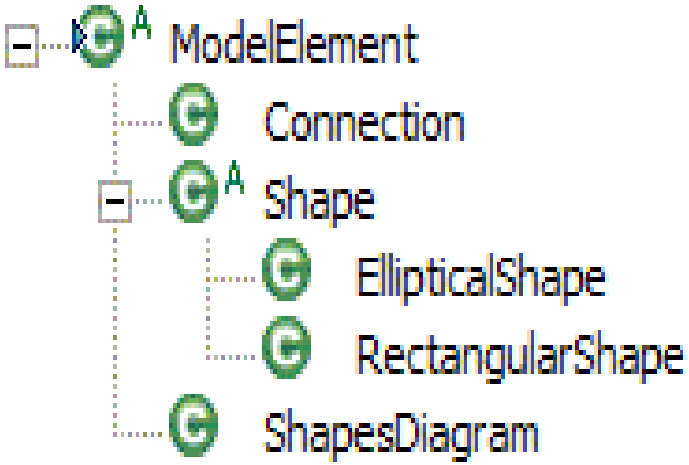


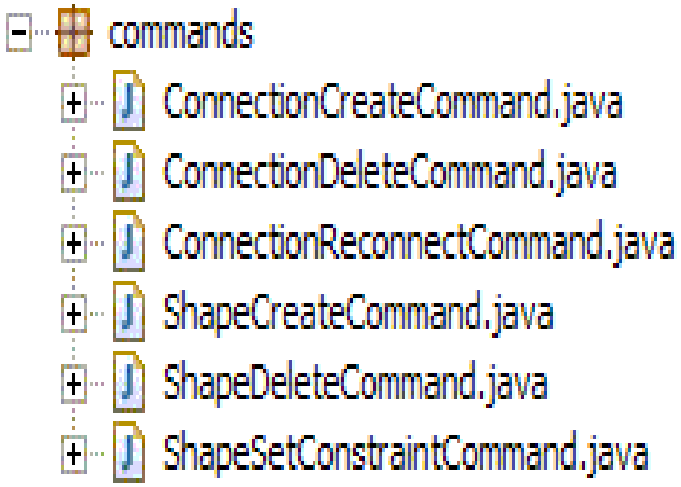
Set-up Sample Project

- Unzip shapes_example.zip before launching Eclipse
- Open Eclipse
- Start with a clean workspace
- Then from within Eclipse
  - Import -> Existing Project Into Workspace
  - Select shapes_example.zip from Archived Files Option or from the previous directory
- Switch to the Java™ perspective

Model

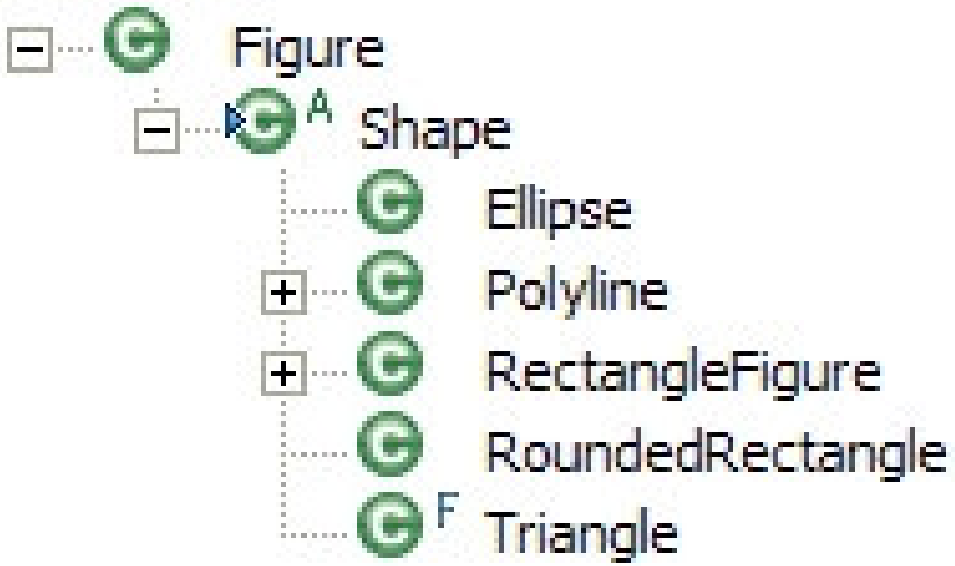
- Provided!
- So are the Commands
- ModelElement implements IPropertySource and Serializable





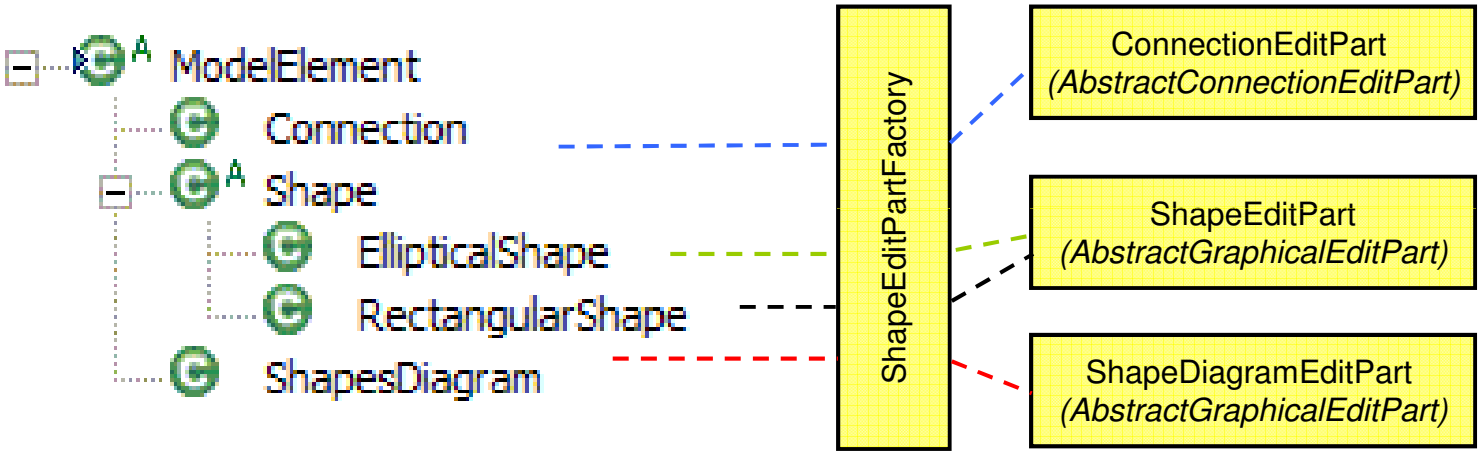
View

- Available in Draw2d
- Basic Shapes: RectangleFigure and Ellipse



AOT LAB

Controller



ShapeDiagramEditPart

- Listen to the model only when the EditPart is active
  - activate()
  - deactivate()
  - propertyChange()
- createFigure()
  - FreeformLayer with FreeformLayout if you want to scroll into negative co-ordinates; Figure with XYLayout otherwise
- getModelChildren()
- createEditPolicies()
  - COMPONENT_ROLE: RootComponentEditPolicy (provided by GEF) to prevent deletion of the content EditPart
  - LAYOUT_ROLE: Subclass XYLayoutEditPolicy
    - For creating, moving and resizing children
    - createChildEditPolicy() : default will do

ShapeEditPart

- Listen to the model for changes
- Create RectangleFigure or Ellipse based on the model instance
- Implement NodeEditPart to support connections
- ChopboxAnchor or EllipseAnchor (return the same anchor in all methods)
- Edit Policies
  - COMPONENT_ROLE: Sub-class ComponentEditPolicy to provide delete support
  - GRAPHICAL_NODE_ROLE: Sub-class GraphicalNodeEditPolicy
- getModelSourceConnections() and getModelTargetConnections()
- refreshVisuals()
  - ((GraphicalEditPart)getParent()).setLayoutConstraint(…)

ConnectionEditPart

- Listen to model for changes
- Create figure
  - PolylineConnection with a target decoration and proper lineStyle
- Edit Policies
  - CONNECTION_ENDPOINTS_ROLE: ConnectionEndpointEditPolicy (provided by GEF) to select the connection's endpoints
  - CONNECTION_ROLE: Sub-class ConnectionEditPolicy to support delete

AOT LAB

Bring It All Together - ShapesEditor

- Functionality not related to GEF is provided
  - We've added TODO:Tutorial to mark missing functionality
- Constructor: Provide a new DefaultEditDomain
- configureGraphicalViewer()
  - RootEditPart
  - EditPartFactory
- initializeGraphicalViewer()
  - Set the viewer's contents
- Palette
  - getPaletteRoot() and getPalettePreferences()
  - Delegate to ShapesEditorPaletteFactory

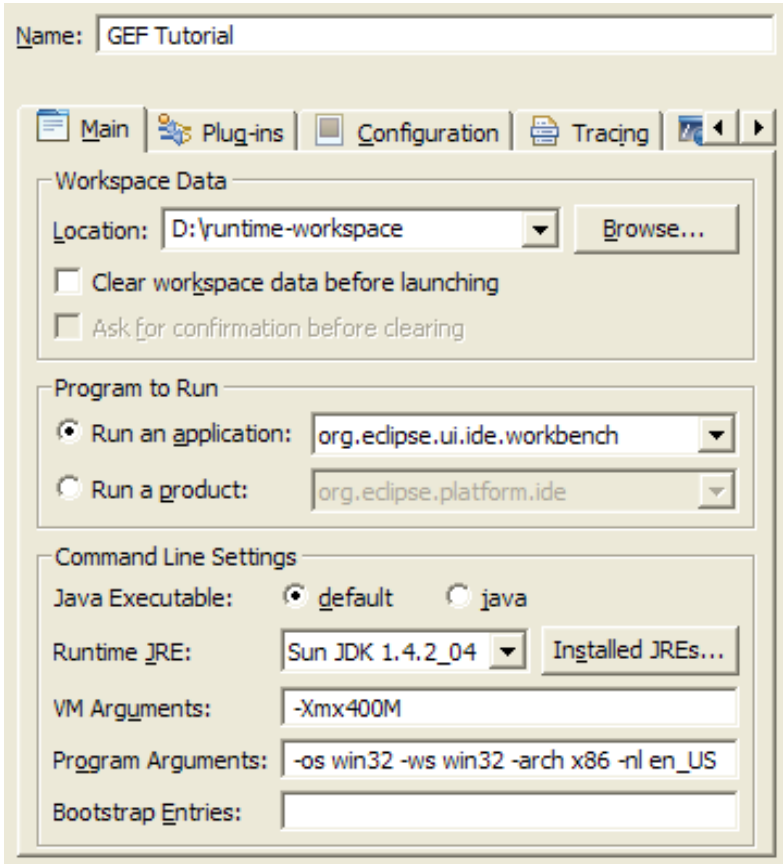
Bring It All Together - ShapesEditorPaletteFactory

- createPalettePreferences()
  - ShapesPlugin#getDefault()#getPreferenceStore()
- createPaletteRoot()
  - Tools Group has PanningSelectionToolEntry, MarqueeToolEntry and ConnectionCreationToolEntry
  - Shapes Drawer has CombinedTemplateCreationEntry for the two shapes
    - The templates can be null for now (used for DND)
  - Tip: Do not forget to provide a default tool (usually Selection)

AOT LAB

Test Drive

- Launch the Runtime Workbench
  - Run menu -> Debug… -> Click on Entire Application -> Click New
- Test functionality
  - Create a Simple Project and a new Shapes example using the wizard
  - Click and drop from palette (notice drag and drop doesn't work yet)
  - Select, move, resize, connect parts
  - Properties View
  - Select a shape and hit '.' (the period key); then try '/' (the slash key)
  - Bring up the Palette View
  - Right-click on the palette
  - Drag the shapes into negative region
- From here, we'll incrementally add desired features



AOT LAB

(Native) Drag-n-Drop from Palette

- Add DragSourceListener to the Palette and DropTargetListener to the graphical viewer
- ShapesEditor
  - createPaletteViewerProvider() - TemplateTransferDragSourceListener
  - initializeGraphicalViewer() – TemplateTransferDropTargetListener
- ShapesEditorPaletteFactory
  - Change CombinedTemplateCreationEntries' to return the model classes as templates
  - Sub-class TemplateTransferDropTargetListener to provide the CreationFactory: use SimpleFactory

Context Menu

- ShapesEditorContextMenuProvider#buildContextMenu()
  - Add Undo, Redo, Delete and Save actions from the Editor's ActionRegistry
- ShapesEditor#configureGraphicalViewer()
  - Create and hook the context menu

Actions

- ShapesEditorActionBarContributor
  - All ShapesEditor instances share this Contributor
  - buildActions() – Add RetargetActions for undo, redo, delete
  - contributeToToolBar() – Add undo and redo retarget actions to the toolbar
    - Do not create new actions here!
- Register the contributor in plugin.xml where you register your editor
  - contributorClass= "org.eclipse.gef.examples.shapes.ShapesEditorActionBarContributor"

Connection Routing

- ShapesEditor#initializeGraphicalViewer()
  - Get the connection layer from the RootEditPart
  - Create a new ShortestPathConnectionRouter and add it to the connection layer
  - Add the router's layoutListener to the content EditPart's content pane (i.e., the figure that is going to have the connections)
  - Voila!

Outline

- ShapesOutlinePage extends ContentOutlinePage
  - Set-up as inner class in ShapesEditor since it shares a few things with it
  - Use the GEF TreeViewer
  - Since we're creating a new viewer, we'll need to define a new EditPartFactory and new EditParts
- EditPart
  - Extend AbstractTreeEditPart
  - Much simpler than earlier ones
    - No Edit Policies needed since we're only supporting selection
  - Listen to the model for changes
  - We do not show connections in the outline, so no need to worry about those
  - Override getImage() and getText()

ShapesOutlinePage

- Override createControl()
  - Outline viewer should share EditDomain/CommandStack with the Editor's viewer
  - Provide the EditPartFactory
  - Register the context menu for this viewer
  - Register viewer with the SelectionSynchronizer
    - Tip: Don't forget to remove the viewer from the SelectionSynchronizer when the outline page is disposed – override dispose()
  - Set the contents
- Override init()
  - Register Editor's actions as GlobalActionHandlers (undo, redo, delete)
  - pageSite.getActionBars().setGlobalActionHandler(…)
- Hook with the Editor
  - ShapesEditor#getAdapter()

Other Features

- Customizer for the Palette
- Zooming (already supported, but no means to manipulate it)
- Grid
- Snap To Geometry
- Rulers & Guides

AOT LAB

Agent and **O**bject **T**echnology **Lab**
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

GEF TUTORIAL

Alessandro Negri
negri@ce.unipr.it
http://www.ce.unipr.it/people/negri/