GEF – Figure – One Trace
Use Concept Trace Slicing

Supporting Applications : [Flow (Activity)]

```
import org.eclipse.gef.editpolicies.GraphicalNodeEditPolicy;
import org.eclipse.draw2d.IFigure;
import org.eclipse.jface.action.IAction;
import org.eclipse.gef.EditPartFactory;
import org.eclipse.gef.editpolicies.DirectEditPolicy;
import org.eclipse.gef.EditPolicy;
import org.eclipse.gef.requests.CreateRequest;
import org.eclipse.gef.editpolicies.NonResizableEditPolicy;
import org.eclipse.gef.palette.PaletteRoot;
import org.eclipse.gef.NodeEditPart;
import org.eclipse.gef.requests.GroupRequest;
import org.eclipse.gef.KeyHandler;
import org.eclipse.gef.ui.palette.PaletteViewer;
import org.eclipse.gef.EditPartViewer;
import org.eclipse.gef.commands.CommandStack;
import org.eclipse.gef.ui.parts.GraphicalViewerKeyHandler;
import org.eclipse.gef.dnd.TemplateTransferDropTargetListener;
import org.eclipse.gef.editparts.AbstractGraphicalEditPart;
import org.eclipse.gef.ui.actions.ActionRegistry;
import java.util.EventObject;
import java.util.List;
import org.eclipse.gef.EditDomain;
import org.eclipse.gef.ui.actions.DirectEditAction;
import org.eclipse.gef.ContextMenuProvider;
import org.eclipse.ui.IEditorInput;
import org.eclipse.gef.palette.SelectionToolEntry;
import org.eclipse.ui.IWorkbenchPartSite;
import org.eclipse.ui.IEditorPart;
import org.eclipse.gef.editpolicies.ComponentEditPolicy;
import org.eclipse.gef.editparts.ScalableRootEditPart;
import org.eclipse.gef.KeyStroke;
import org.eclipse.gef.editpolicies.ContainerEditPolicy;
import org.eclipse.gef.commands.CommandStackListener && appGraphicalEditorWithPalette;
import org.eclipse.gef.dnd.TemplateTransferDragSourceListener;
import org.eclipse.gef.editpolicies.RootComponentEditPolicy;
import org.eclipse.ui.IWorkbenchPart;
import org.eclipse.gef.DefaultEditDomain;
import org.eclipse.ui.part.WorkbenchPart;
import org.eclipse.gef.ui.parts.GraphicalEditor;
import org.eclipse.gef.editpolicies.LayoutEditPolicy;
import org.eclipse.gef.commands.Command;
import org.eclipse.gef.EditPart;
import org.eclipse.gef.ui.parts.GraphicalEditorWithPalette;
import org.eclipse.gef.GraphicalViewer;

public class AppLayoutEditPolicy
extends LayoutEditPolicy {

	public Command getCreateCommand(CreateRequest) {
		AppCommand appCommand = new AppCommand();
		Object object1 = CreateRequest.getNewObject();  // REPEATED!
		EditPart editPart = editPolicy.getHost(); // REPEATED!
		Object object = editPart.getModel();      // REPEATED!
	}

	public EditPolicy createChildEditPolicy(appAbstractGraphicalEditPart && editPart1) {
		AppNonResizableEditPolicy appNonResizableEditPolicy = new AppNonResizableEditPolicy();
	}
}

public class AppContainerEditPolicy
extends ContainerEditPolicy {

	public Command getCreateCommand(CreateRequest) {
	}
}

public class AppAbstractGraphicalEditPart
implements NodeEditPart
extends AbstractGraphicalEditPart {

	public  AppAbstractGraphicalEditPart() {
		AppGraphicalEditorWithPalette appGraphicalEditorWithPalette = new AppGraphicalEditorWithPalette(appAbstractGraphicalEditPart && editPart1);
	}

	public List getModelTargetConnections() {
		Object object = editPart3.getModel();     // REPEATED!
	}

	public List getModelChildren() {
		Object object = editPart3.getModel();     // REPEATED!
	}

	public void deactivate() {
		/*                        Cyclic Statements                  */
		EditDomain editDomain = editPartViewer.getEditDomain();
		commandStack.removeCommandStackListener(CommandStackListener && appGraphicalEditorWithPalette);
		Object object = editPart3.getModel();     // REPEATED!
		CommandStack commandStack = editDomain.getCommandStack();
		EditPartViewer editPartViewer = editPart3.getViewer();      // REPEATED!
	}

	public void setFigure(ifigure2) {
	}

	public void refreshVisuals() {
		/*                        Cyclic Statements                  */
		IFigure ifigure = editPart2.getFigure();  // REPEATED!
		Object object = editPart3.getModel();     // REPEATED!
	}

	public void activate() {
		/*                        Cyclic Statements                  */
		EditDomain editDomain = editPartViewer.getEditDomain();
		Object object = editPart3.getModel();     // REPEATED!
		commandStack.addCommandStackListener(CommandStackListener && appGraphicalEditorWithPalette);
		CommandStack commandStack = editDomain.getCommandStack();
		EditPartViewer editPartViewer = editPart3.getViewer();      // REPEATED!
	}
```

```
	public List getModelSourceConnections() {
		Object object = editPart3.getModel();     // REPEATED!
	}

	public boolean isSelectable() {
	}

	public IFigure getContentPane() {
		IFigure ifigure = editPart2.getFigure();  // REPEATED!
	}

	public void createEditPolicies() {
		AppComponentEditPolicy appComponentEditPolicy = new AppComponentEditPolicy();
		RootComponentEditPolicy rootComponentEditPolicy = new RootComponentEditPolicy();
		AppGraphicalNodeEditPolicy appGraphicalNodeEditPolicy = new AppGraphicalNodeEditPolicy();
		AppLayoutEditPolicy appLayoutEditPolicy = new AppLayoutEditPolicy();
		AppDirectEditPolicy appDirectEditPolicy = new AppDirectEditPolicy();
		AppContainerEditPolicy appContainerEditPolicy = new AppContainerEditPolicy();
		appAbstractGraphicalEditPart.installEditPolicy(appDirectEditPolicy && appContainerEditPolicy && rootComponentEditPolicy &&
				appLayoutEditPolicy && appGraphicalNodeEditPolicy && appComponentEditPolicy); // REPEATED!
	}

	public IFigure createFigure() {
	}
}

public class AppNonResizableEditPolicy
extends NonResizableEditPolicy {

	public void hideFocus() {
		EditPart editPart = editPolicy.getHost();
		IFigure ifigure = editPart.getFigure();   // REPEATED!
	}

	public void showPrimarySelection() {
		EditPart editPart = editPolicy.getHost(); // REPEATED!
		IFigure ifigure = editPart.getFigure();   // REPEATED!
	}

	public void hideSelection() {
		EditPart editPart = editPolicy.getHost(); // REPEATED!
		IFigure ifigure = editPart.getFigure();   // REPEATED!
	}
}

public class AppditPartFactory
implements EditPartFactory  {

	public EditPart createEditPart(object1 && appAbstractGraphicalEditPart && editPart) {
		AppAbstractGraphicalEditPart appAbstractGraphicalEditPart = new AppAbstractGraphicalEditPart();
		appAbstractGraphicalEditPart.setModel(object1); // REPEATED!
	}
}

public class AppGraphicalEditorWithPalette
extends GraphicalEditorWithPalette {

	public void setInput(IEditorInput) {
	}

	public void commandStackChanged(EventObject) {
		WorkbenchPart.firePropertyChange(int);
		/*                    Cyclic Statements                    */
		List list1 = editPart2.getSourceConnections();  // REPEATED!
		IFigure ifigure = editPart2.getFigure();  // REPEATED!
		List list = editPart2.getChildren();      // REPEATED!
		IFigure ifigure1 = editPart2.getContentPane();  // REPEATED!
		Object object = editPart3.getModel();     // REPEATED!
	}

	public PaletteRoot getPaletteRoot() {
		SelectionToolEntry selectionToolEntry = new SelectionToolEntry(String)||(String,String)||();
		PaletteRoot paletteRoot = new PaletteRoot();
		paletteRoot.setDefaultEntry(selectionToolEntry);
		paletteRoot.addAll(List);
	}

	public void initializeGraphicalViewer() {
		GraphicalViewer graphicalViewer = GraphicalEditor.getGraphicalViewer(); // REPEATED!
		graphicalViewer.setContents(EditPart):()||(Object):();
		TemplateTransferDropTargetListener templateTransferDropTargetListener = new TemplateTransferDropTargetListener(editPartViewer && graphicalViewer);
		graphicalViewer.addDropTargetListener(templateTransferDropTargetListener);
	}

	public void configureGraphicalViewer() {
		KeyStroke.getPressed(); // REPEATED!
		ScalableRootEditPart scalableRootEditPart = new ScalableRootEditPart();
		KeyHandler keyHandler = new KeyHandler();
		AppditPartFactory appditPartFactory = new AppditPartFactory();
		/*                    Cyclic Statements                    */
		GraphicalViewerKeyHandler graphicalViewerKeyHandler = new GraphicalViewerKeyHandler(graphicalViewer);
		ActionRegistry actionRegistry = GraphicalEditor.getActionRegistry();    // REPEATED!
		graphicalViewer.setEditPartFactory(appditPartFactory);
		IAction iaction1 = actionRegistry.getAction(Object);  // REPEATED!
		graphicalViewer.setRootEditPart(scalableRootEditPart);
		keyHandler.put(directEditAction && iaction1);   // REPEATED!
		KeyHandler keyHandler1 = graphicalViewerKeyHandler.setParent(keyHandler);
		graphicalViewer.setKeyHandler(graphicalViewerKeyHandler);
		IWorkbenchPartSite iworkbenchPartSite = IWorkbenchPart.getSite(); // REPEATED!
		graphicalViewer.setContextMenu(appContextMenuProvider);
		GraphicalViewer graphicalViewer = GraphicalEditor.getGraphicalViewer(); // REPEATED!
		AppContextMenuProvider appContextMenuProvider = new AppContextMenuProvider(graphicalViewer && actionRegistry);
	}

	public void createActions() {
		DirectEditAction directEditAction = new DirectEditAction(IWorkbenchPart)||(IEditorPart);
		/*                    Cyclic Statements                    */
		List list5 = GraphicalEditor.getSelectionActions();
		ActionRegistry actionRegistry = GraphicalEditor.getActionRegistry();
		actionRegistry.registerAction(directEditAction);
	}

	public void initializePaletteViewer() {
		PaletteViewer paletteViewer = GraphicalEditorWithPalette.getPaletteViewer();  // REPEATED!
		TemplateTransferDragSourceListener templateTransferDragSourceListener = new TemplateTransferDragSourceListener(paletteViewer);
```

```
            paletteViewer.addDragSourceListener(templateTransferDragSourceListener);      // REPEATED!
      }
}

public class AppComponentEditPolicy
extends ComponentEditPolicy {

      public Command createDeleteCommand(GroupRequest) {
            AppCommand appCommand = new AppCommand();
            EditPart editPart = editPolicy.getHost(); // REPEATED!
            /*                      Cyclic Statements                   */
            EditPart editPart3 = editPart.getParent();
            Object object = editPart3.getModel();     // REPEATED!
      }
}

public class AppCommand
extends Command {

      public void execute() {
            /*                      Cyclic Statements                   */
            IFigure ifigure = editPart2.getFigure();
            EditPart editPart2 = editPartViewer.getContents();
            appAbstractGraphicalEditPart.refreshChildren();
            EditPartViewer editPartViewer = editPart2.getViewer();      // REPEATED!
      }
}

public class AppDirectEditPolicy
extends DirectEditPolicy {
}

public class AppContextMenuProvider
extends ContextMenuProvider {
}

public class AppGraphicalNodeEditPolicy
extends GraphicalNodeEditPolicy {
}

public class SomeClass {

      public void someMethod() {
            DefaultEditDomain defaultEditDomain = new DefaultEditDomain(IEditorPart);
            GraphicalEditor.setEditDomain(defaultEditDomain);
            /*                      Cyclic Statements                   */
            List list1 = editPart2.getSourceConnections();  // REPEATED!
            IFigure ifigure = editPart2.getFigure();  // REPEATED!
            List list = editPart2.getChildren();      // REPEATED!
            IFigure ifigure1 = editPart2.getContentPane();  // REPEATED!
            Object object = editPart3.getModel();     // REPEATED!
      }
}
```