GEF – Connection – One Trace
Use Concept Trace Slicing

Supporting Applications : [Flow]

```
import org.eclipse.gef.requests.CreateConnectionRequest;
import org.eclipse.gef.editpolicies.GraphicalNodeEditPolicy;
import org.eclipse.gef.editpolicies.ConnectionEditPolicy;
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.editpolicies.NonResizableEditPolicy;
import org.eclipse.gef.palette.PaletteRoot;
import org.eclipse.gef.NodeEditPart;
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.editparts.AbstractConnectionEditPart;
import org.eclipse.gef.dnd.TemplateTransferDropTargetListener;
import org.eclipse.gef.ui.actions.ActionRegistry;
import java.util.EventObject;
import org.eclipse.gef.EditDomain;
import java.util.List;
import org.eclipse.gef.ui.actions.DirectEditAction;
import org.eclipse.gef.ContextMenuProvider;
import org.eclipse.ui.IEditorInput;
import org.eclipse.draw2d.Connection;
import org.eclipse.ui.IWorkbenchPartSite;
import org.eclipse.gef.palette.SelectionToolEntry;
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.dnd.TemplateTransferDragSourceListener;
import org.eclipse.gef.commands.CommandStackListener && appGraphicalEditorWithPalette;
import org.eclipse.gef.editpolicies.ContainerEditPolicy;
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.editpolicies.ConnectionEndpointEditPolicy;
import org.eclipse.gef.commands.Command && appCommand && command2;
import org.eclipse.gef.GraphicalViewer;

public class AppCommand
extends Command {

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

      public boolean canExecute() {
      }
}

public class AppAbstractConnectionEditPart
implements NodeEditPart
extends AbstractConnectionEditPart {

      public  AppAbstractConnectionEditPart() {
            AppGraphicalEditorWithPalette appGraphicalEditorWithPalette = new AppGraphicalEditorWithPalette(appAbstractConnectionEditPart && editPart1);
      }

      public void deactivate() {
            /*                    Cyclic Statements                  */
            EditDomain editDomain = editPartViewer.getEditDomain();
            commandStack.removeCommandStackListener(CommandStackListener && appGraphicalEditorWithPalette);
            Object object = editPart4.getModel();     // REPEATED!
            CommandStack commandStack = editDomain.getCommandStack();
            EditPartViewer editPartViewer = editPart2.getViewer();      // REPEATED!
      }

      public IFigure createFigure() {
      }

      public void getSourceConnectionAnchor(editPart1 && appAbstractConnectionEditPart) {
            IFigure ifigure = editPart4.getFigure();  // REPEATED!
      }

      public List getModelTargetConnections() {
            Object object = editPart4.getModel();     // REPEATED!
      }

      public void refreshVisuals() {
            /*                    Cyclic Statements                  */
            IFigure ifigure = editPart4.getFigure();  // REPEATED!
            Object object = editPart4.getModel();     // REPEATED!
      }

      public List getModelSourceConnections() {
            Object object = editPart4.getModel();     // REPEATED!
      }

      public void getTargetConnectionAnchor(editPart1 && appAbstractConnectionEditPart) {
            IFigure ifigure = editPart4.getFigure();  // REPEATED!
      }

      public List getModelChildren() {
            Object object = editPart4.getModel();     // REPEATED!
      }

      public void createEditPolicies() {
```

```
            AppComponentEditPolicy appComponentEditPolicy = new AppComponentEditPolicy();
            RootComponentEditPolicy rootComponentEditPolicy = new RootComponentEditPolicy();
            AppGraphicalNodeEditPolicy appGraphicalNodeEditPolicy = new AppGraphicalNodeEditPolicy();
            AppLayoutEditPolicy appLayoutEditPolicy = new AppLayoutEditPolicy();
            AppDirectEditPolicy appDirectEditPolicy = new AppDirectEditPolicy();
            ConnectionEndpointEditPolicy connectionEndpointEditPolicy = new ConnectionEndpointEditPolicy();
            AppContainerEditPolicy appContainerEditPolicy = new AppContainerEditPolicy();
            AppConnectionEditPolicy appConnectionEditPolicy = new AppConnectionEditPolicy();
            appAbstractConnectionEditPart.installEditPolicy(appContainerEditPolicy && appGraphicalNodeEditPolicy && appComponentEditPolicy && appDirectEditPolicy &&
                                        connectionEndpointEditPolicy && appConnectionEditPolicy && rootComponentEditPolicy && appLayoutEditPolicy); // REPEATED!
      }

      public IFigure getContentPane() {
            IFigure ifigure = editPart4.getFigure();  // REPEATED!
      }

      public void setFigure(ifigure2) {
      }

      public void activate() {
            /*                      Cyclic Statements                 */
            EditDomain editDomain = editPartViewer.getEditDomain();
            Object object = editPart4.getModel();     // REPEATED!
            commandStack.addCommandStackListener(CommandStackListener && appGraphicalEditorWithPalette);
            CommandStack commandStack = editDomain.getCommandStack();
            EditPartViewer editPartViewer = editPart2.getViewer();      // REPEATED!
      }
}

public class AppditPartFactory
implements EditPartFactory  {

      public EditPart createEditPart(appAbstractConnectionEditPart && editPart) {
            AppAbstractConnectionEditPart appAbstractConnectionEditPart = new AppAbstractConnectionEditPart();
            appAbstractConnectionEditPart.setModel(Object); // REPEATED!
      }
}

public class AppGraphicalNodeEditPolicy
extends GraphicalNodeEditPolicy {

      public Command getConnectionCreateCommand(CreateConnectionRequest) {
            AppCommand appCommand = new AppCommand();
            CreateConnectionRequest.setStartCommand(Command && appCommand && command2);   // REPEATED!
            EditPart editPart = editPolicy.getHost(); // REPEATED!
            Object object = editPart.getModel();      // REPEATED!
      }

      public Command getConnectionCompleteCommand(CreateConnectionRequest) {
            Command command = CreateConnectionRequest.getStartCommand();      // REPEATED!
            EditPart editPart = editPolicy.getHost(); // REPEATED!
            Object object = editPart.getModel();      // REPEATED!
      }
}

public class AppLayoutEditPolicy
extends LayoutEditPolicy {

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

public class AppNonResizableEditPolicy
extends NonResizableEditPolicy {

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

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

public class AppGraphicalEditorWithPalette
extends GraphicalEditorWithPalette {

      public void setInput(IEditorInput) {
      }

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

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

      public PaletteRoot getPaletteRoot() {
            SelectionToolEntry selectionToolEntry = new SelectionToolEntry(String)||(String,String)||();
```

```
            PaletteRoot paletteRoot1 = new PaletteRoot();
            paletteRoot1.setDefaultEntry(selectionToolEntry);
            paletteRoot1.addAll(List);
      }

      public void commandStackChanged(EventObject) {
            WorkbenchPart.firePropertyChange(int);
            /*                      Cyclic Statements                   */
            EditPart editPart4 = appAbstractConnectionEditPart.getTarget();
            List list1 = editPart3.getSourceConnections();  // REPEATED!
            IFigure ifigure = editPart3.getFigure();  // REPEATED!
            EditPart editPart3 = appAbstractConnectionEditPart.getSource();
            List list = editPart3.getChildren();      // REPEATED!
            IFigure ifigure1 = editPart2.getContentPane();  // REPEATED!
            Object object = editPart3.getModel();     // REPEATED!
            Connection connection = appAbstractConnectionEditPart.getConnectionFigure();  // REPEATED!
      }

      public void createActions() {
            DirectEditAction directEditAction = new DirectEditAction(IWorkbenchPart)||(IEditorPart);
            /*                      Cyclic Statements                   */
            List list4 = GraphicalEditor.getSelectionActions();
            ActionRegistry actionRegistry = GraphicalEditor.getActionRegistry();
            actionRegistry.registerAction(directEditAction);
      }
}

public class AppContainerEditPolicy
extends ContainerEditPolicy {
}

public class AppComponentEditPolicy
extends ComponentEditPolicy {
}

public class AppDirectEditPolicy
extends DirectEditPolicy {
}

public class AppConnectionEditPolicy
extends ConnectionEditPolicy {
}

public class AppContextMenuProvider
extends ContextMenuProvider {
}

public class SomeClass {

      public void someMethod() {
            DefaultEditDomain defaultEditDomain = new DefaultEditDomain(IEditorPart);
            GraphicalEditor.setEditDomain(defaultEditDomain);
            /*                      Cyclic Statements                   */
            EditPart editPart4 = appAbstractConnectionEditPart.getTarget();
            List list1 = editPart3.getSourceConnections();  // REPEATED!
            IFigure ifigure = editPart3.getFigure();  // REPEATED!
            EditPart editPart3 = appAbstractConnectionEditPart.getSource();
            List list = editPart3.getChildren();      // REPEATED!
            IFigure ifigure1 = editPart2.getContentPane();  // REPEATED!
            Object object = editPart3.getModel();     // REPEATED!
            Connection connection = appAbstractConnectionEditPart.getConnectionFigure();  // REPEATED!
      }
}
```