Eclipse – Table Viewer – One Trace
Use Full Trace (No Slicing)

Supporting Applications : [LDAP Connections]

```
import org.eclipse.jface.text.reconciler.DirtyRegion;
import org.eclipse.swt.graphics.FontData;
import org.eclipse.jface.text.Region;
import org.eclipse.jface.action.Separator;
import org.eclipse.jface.action.IMenuListener;
import org.eclipse.ui.IActionBars;
import org.eclipse.jface.text.DefaultIndentLineAutoEditStrategy;
import org.eclipse.jface.text.source.SourceViewerConfiguration;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.jface.resource.JFaceResources;
import org.eclipse.ui.IWorkbench;
import org.eclipse.jface.resource.FontRegistry;
import org.eclipse.jface.text.rules.FastPartitioner;
import org.eclipse.ui.editors.text.templates.ContributionTemplateStore;
import org.eclipse.jface.text.ITypedRegion;
import org.eclipse.jface.text.rules.RuleBasedPartitionScanner;
import org.eclipse.ui.IViewPart;
import org.eclipse.ui.editors.text.ILocationProvider;
import org.eclipse.ui.IWorkbenchPartSite;
import org.eclipse.jface.preference.PreferenceConverter;
import org.eclipse.jface.text.source.IAnnotationHover;
import org.eclipse.swt.graphics.RGB;
import org.eclipse.core.runtime.Plugin;
import org.eclipse.ui.editors.text.TextFileDocumentProvider;
import org.eclipse.ui.IDecoratorManager;
import org.eclipse.jface.text.IDocumentListener;
import org.eclipse.jface.viewers.SelectionChangedEvent;
import org.eclipse.jface.text.ITextDoubleClickStrategy;
import org.eclipse.jface.text.ITextHover;
import org.eclipse.jface.text.presentation.IPresentationReconciler;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.ui.ISelectionListener;
import org.eclipse.jface.viewers.DecoratingLabelProvider;
import org.eclipse.jface.text.rules.IPredicateRule;
import org.eclipse.jface.util.IPropertyChangeListener;
import org.eclipse.ui.IWorkbenchPage;
import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.jface.viewers.IColorProvider;
import org.eclipse.jface.viewers.StructuredSelection;
import org.eclipse.jface.resource.ImageRegistry;
import org.eclipse.ui.IWorkbenchPartReference;
import org.eclipse.jface.viewers.TableViewer;
import org.eclipse.jface.text.source.SourceViewer;
import org.eclipse.jface.text.DocumentEvent;
import org.eclipse.swt.widgets.Tree;
import org.eclipse.jface.viewers.LabelProvider;
import org.eclipse.swt.graphics.Device;
import org.eclipse.ui.IEditorInput;
import org.eclipse.jface.text.templates.GlobalTemplateVariables$Cursor;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.dnd.DragSourceListener;
import org.eclipse.jface.text.source.IAnnotationModel;
import org.eclipse.swt.dnd.ByteArrayTransfer;
import org.eclipse.ui.IWorkbenchPart;
import org.eclipse.jface.resource.ColorRegistry;
import org.eclipse.jface.text.reconciler.IReconciler;
import org.eclipse.swt.widgets.Menu;
import org.eclipse.ui.IViewSite;
import org.eclipse.jface.viewers.IFontProvider;
import org.eclipse.jface.viewers.ITableViewerListener;
import org.eclipse.jface.text.DefaultTextDoubleClickStrategy;
import org.eclipse.jface.text.TextEvent;
import org.eclipse.jface.text.ITextListener;
import org.eclipse.jface.text.IAutoEditStrategy;
import org.eclipse.jface.text.reconciler.MonoReconciler;
import org.eclipse.jface.action.Action;
import org.eclipse.ui.ISharedImages;
import org.eclipse.jface.text.templates.TemplateContextType;
import org.eclipse.jface.text.rules.Token;
import org.eclipse.core.runtime.NullProgressMonitor;
import org.eclipse.core.filebuffers.ITextFileBuffer;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import org.eclipse.jface.action.IMenuManager;
import org.eclipse.swt.graphics.Font;
import org.eclipse.jface.text.presentation.IPresentationDamager;
import org.eclipse.core.filebuffers.IDocumentSetupParticipant;
import org.eclipse.jface.text.presentation.PresentationReconciler;
import org.eclipse.core.commands.IHandler && actionHandler;
import org.eclipse.swt.graphics.Image;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.swt.dnd.Clipboard;
import org.eclipse.core.runtime.Preferences;
```

```
import org.eclipse.ui.help.IWorkbenchHelpSystem;
import org.eclipse.jface.preference.IPreferenceStore;
import java.util.Iterator;
import org.eclipse.jface.viewers.IDoubleClickListener;
import org.eclipse.jface.action.IToolBarManager;
import org.eclipse.jface.action.MenuManager;
import java.io.File;
import org.eclipse.core.runtime.IPath;
import org.eclipse.jface.commands.ActionHandler;
import org.eclipse.swt.widgets.Table;
import org.eclipse.core.runtime.Preferences$IPropertyChangeListener;
import org.eclipse.ui.IPartListener2;
import org.eclipse.jface.viewers.TableViewer;
import org.eclipse.jface.text.IDocumentPartitioner;
import org.eclipse.ui.editors.text.templates.ContributionContextTypeRegistry;
import org.eclipse.swt.widgets.Display;
import org.eclipse.jface.text.reconciler.IReconcilingStrategyExtension;
import org.eclipse.ui.actions.ActionFactory;
import org.eclipse.core.runtime.Path;
import org.eclipse.core.commands.Command;
import org.eclipse.swt.graphics.Color;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.ISelectionService;
import org.eclipse.swt.custom.StyledText;
import org.eclipse.ui.PlatformUI;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.ILabelDecorator;
import org.eclipse.jface.text.presentation.IPresentationRepairer;
import org.eclipse.jface.text.IDocument;
import org.eclipse.swt.dnd.DropTargetListener;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.dnd.Transfer;
import org.eclipse.jface.text.IRegion;
import org.eclipse.ui.plugin.AbstractUIPlugin;
import org.eclipse.jface.viewers.ViewerSorter;
import org.eclipse.jface.viewers.ISelectionChangedListener;
import org.eclipse.jface.viewers.ITreeContentProvider;
import org.eclipse.ui.editors.text.TextFileDocumentProvider$FileInfo;
import org.eclipse.jface.text.contentassist.IContentAssistant;
import org.eclipse.jface.text.reconciler.IReconcilingStrategy;

public class AppLocationProvider
implements ILocationProvider, IWorkbenchPart, IEditorInput  {

      public IPath getPath(appLocationProvider && object) {
            IPath ipath = Plugin.getStateLocation();
            ipath.append(String):(IPath)||(IPath):(IPath);
      }

      public void createPartControl(Composite) {
            AppTextFileDocumentProvider appTextFileDocumentProvider = new AppTextFileDocumentProvider();
            MenuManager menuManager = new MenuManager(String)||(String,String)||();
            ActionHandler actionHandler = new ActionHandler(appAction); // REPEATED!
            GridData gridData = new GridData(int)||(int,int,boolean,boolean)||(int,int,boolean,boolean,int,int)||(int,int)||();     // REPEATED!
            Appreferences$IPropertyChangeListener appreferences$IPropertyChangeListener = new Appreferences$IPropertyChangeListener();
            AppLocationProvider appLocationProvider = new AppLocationProvider();
            appTextFileDocumentProvider.connect(appLocationProvider);
            IDocument idocument3 = appTextFileDocumentProvider.getDocument(appLocationProvider && object);
            idocument3.set();
            PlatformUI.getWorkbench();    // REPEATED!
            String string6 = appAction.getActionDefinitionId();   // REPEATED!
            AppStructuredContentProvider appStructuredContentProvider = new AppStructuredContentProvider(appViewerSorter);
            AppTextListener appTextListener = new AppTextListener();
            AppPropertyChangeListener appPropertyChangeListener = new AppPropertyChangeListener(appTextListener && appLocationProvider);
            AppLabelProvider appLabelProvider = new AppLabelProvider(appreferences$IPropertyChangeListener && appPropertyChangeListener);
            AppViewerSorter appViewerSorter = new AppViewerSorter(appreferences$IPropertyChangeListener && appPropertyChangeListener);
            AppSourceViewerConfiguration appSourceViewerConfiguration = new AppSourceViewerConfiguration(appTextListener && appLocationProvider);
            JFaceResources.getFont();
            String string = ActionFactory.getId();    // REPEATED!
            IWorkbench iworkbench = AbstractUIPlugin.getWorkbench();
            IPath ipath = Plugin.getStateLocation();
            Separator separator = new Separator(String)||();      // REPEATED!
            Composite composite = new Composite(Composite,int)||();     // REPEATED!
            SourceViewer sourceViewer = new SourceViewer(composite);
            TableViewer tableViewer = new TableViewer(composite);
            tableViewer.setUseHashlookup(boolean);
            tableViewer.setSorter(appViewerSorter);
            tableViewer.setContentProvider(appStructuredContentProvider);
            GridLayout gridLayout = new GridLayout(int,boolean)||();    // REPEATED!
            /*                      Cyclic Statements                   */
            IDecoratorManager idecoratorManager = iworkbench.getDecoratorManager();
            Iterator iterator = structuredSelection.iterator();   // REPEATED!
            ISelection iselection1 = tableViewer.getSelection();  // REPEATED!
            AppPartListener2 appPartListener2 = new AppPartListener2();
            iworkbenchPage.showView(String):(IViewPart)||(String,String,int):(IViewPart); // REPEATED!
            boolean app_boolean3 = control.setFocus();
            itoolBarManager.update(boolean);    // REPEATED!
            composite.setLayout(gridLayout);    // REPEATED!
            iactionBars.setGlobalActionHandler(string && appAction);    // REPEATED!
            iworkbenchHelpSystem.setHelp(composite);  // REPEATED!
```

```
		control.setMenu(menu);
		tableViewer.setLabelProvider(appLabelProvider && decoratingLabelProvider);
		SelectionChangedEvent selectionChangedEvent = new SelectionChangedEvent(sourceViewer && tableViewer
                                                                                && structuredSelection);      // REPEATED!
		IMenuManager imenuManager = iactionBars.getMenuManager();
		tableViewer.setInput(iworkbenchPartSite && iviewSite && appLocationProvider); // REPEATED!
		IToolBarManager itoolBarManager = iactionBars.getToolBarManager();      // REPEATED!
		Object object1 = iworkbench.getAdapter(Class);  // REPEATED!
		DecoratingLabelProvider decoratingLabelProvider = new DecoratingLabelProvider(appLabelProvider && idecoratorManager);
		sourceViewer.setEditable(boolean);  // REPEATED!
		IViewPart iviewPart = iworkbenchPage.findView(String);      // REPEATED!
		StructuredSelection structuredSelection = new StructuredSelection(appLocationProvider);   // REPEATED!
		IWorkbenchPage iworkbenchPage = iworkbenchWindow.getActivePage(); // REPEATED!
		boolean app_boolean2 = command.setHandler(IHandler && actionHandler);   // REPEATED!
		idecoratorManager.getLabelDecorator(String):(ILabelDecorator)||():(ILabelDecorator);
		IViewSite iviewSite = IWorkbenchPart.getViewSite();   // REPEATED!
		IWorkbenchWindow iworkbenchWindow = iworkbench.getActiveWorkbenchWindow();    // REPEATED!
		menuManager.addMenuListener(appMenuListener);
		iworkbenchPartSite.setSelectionProvider(tableViewer);
		Menu menu = menuManager.createContextMenu(table && control);
		iactionBars.updateActionBars();
		appAction.selectionChanged(selectionChangedEvent);    // REPEATED!
		menuManager.setRemoveAllWhenShown(boolean);
		control.setLayoutData(gridData);    // REPEATED!
		sourceViewer.setDocument(idocument1 && idocument2 && idocument3); // REPEATED!
		IActionBars iactionBars = iworkbenchPartSite.getActionBars();     // REPEATED!
		itoolBarManager.add(separator && appAction);    // REPEATED!
		AppMenuListener appMenuListener = new AppMenuListener();
		IWorkbenchPartSite iworkbenchPartSite = IWorkbenchPart.getSite(); // REPEATED!
		sourceViewer.addTextListener(appTextListener && appLocationProvider);   // REPEATED!
		Command command = object1.getCommand(string6);  // REPEATED!
		ipath.append(String):(IPath)||(IPath):(IPath);  // REPEATED!
		Control control1 = sourceViewer.getControl();   // REPEATED!
		sourceViewer.configure(appSourceViewerConfiguration);
		StyledText styledText = sourceViewer.getTextWidget(); // REPEATED!
		imenuManager.update(String):()||():();
		Object object2 = tableViewer.getInput();  // REPEATED!
		Control control = tableViewer.getControl();     // REPEATED!
		IWorkbenchHelpSystem iworkbenchHelpSystem = iworkbench.getHelpSystem(); // REPEATED!
		control.setFont();
	}

	public Object getAdapter(Class) {
	}

	public  dispose() {
		appLabelProvider.dispose();
		menuManager.dispose();
		appStructuredContentProvider.dispose();
		PlatformUI.getWorkbench();
		IWorkbenchWindow iworkbenchWindow = IWorkbenchPart.getActiveWorkbenchWindow();
		ISelectionService iselectionService = iworkbenchWindow.getSelectionService();
		iselectionService.removeSelectionListener(appPartListener2);
		control.dispose();
		IWorkbenchPartSite iworkbenchPartSite = IWorkbenchPart.getSite();
		iworkbenchPartSite.setSelectionProvider(tableViewer);
		tableViewer.removeDoubleClickListener(appPartListener2);
	}

	public  setFocus() {
		Control control = tableViewer.getControl();
		boolean app_boolean3 = control.setFocus();
	}
}

public class AppAction
implements ISelectionChangedListener
extends Action {

	public  AppAction() {
		PlatformUI.getWorkbench();
		ISharedImages isharedImages = getSharedImages();
		ImageDescriptor imageDescriptor = isharedImages.getImageDescriptor(String);
		String string5 = appAction.getText();
		Path path = new Path(String,String[],int)||()||(String)||(String,String);
		tableViewer.addSelectionChangedListener(appAction);
		/*                      Cyclic Statements                    */
		IPreferenceStore ipreferenceStore = AbstractUIPlugin.getPreferenceStore();
		appAction.setText(String);
		appAction.setToolTipText(String);
		ImageDescriptor.createFromURL();
		boolean app_boolean = ipreferenceStore.getBoolean(String);
		appAction.setEnabled(boolean);
		Plugin.find(path);
	}

	public void selectionChanged(SelectionChangedEvent) {
		IPreferenceStore ipreferenceStore = AbstractUIPlugin.getPreferenceStore();    // REPEATED!
		boolean app_boolean = ipreferenceStore.getBoolean(String);  // REPEATED!
		ISelection iselection = selectionChangedEvent.getSelection();     // REPEATED!
```

```
            Iterator iterator = iselection.iterator();      // REPEATED!
            Display.getCurrent();   // REPEATED!
            Clipboard clipboard = new Clipboard(display);   // REPEATED!
            clipboard.getContents(appByteArrayTransfer);    // REPEATED!
            clipboard.dispose();    // REPEATED!
            /*                      Cyclic Statements                   */
            appAction.setToolTipText(String);   // REPEATED!
            appAction.setEnabled(boolean);      // REPEATED!
            appAction.setText(String);    // REPEATED!
      }
}

public class AppPredicateRule
implements IPredicateRule  {

      public IToken evaluate(appRuleBasedPartitionScanner) {
            IToken itoken1 = appPredicateRule.evaluate(appRuleBasedPartitionScanner);     // REPEATED!
            display.asyncExec(Runnable);
            /*                      Cyclic Statements                   */
            int app_int1 = appRuleBasedPartitionScanner.getColumn();    // REPEATED!
            int app_int3 = appRuleBasedPartitionScanner.read();   // REPEATED!
      }
}

public class AppPartListener2
implements IPartListener2, ISelectionListener, ITableViewerListener, IDoubleClickListener  {

      public  AppPartListener2() {
            tableViewer.addTreeListener(appPartListener2);
            tableViewer.addDoubleClickListener(appPartListener2);
            PlatformUI.getWorkbench();
            ISelectionService iselectionService = iworkbenchWindow.getSelectionService();
            iselectionService.addSelectionListener(appPartListener2);
            IWorkbenchPage iworkbenchPage1 = iworkbenchPartSite.getPage();
            iworkbenchPage1.addPartListener(appPartListener2);
      }

      public void partBroughtToTop(IWorkbenchPartReference) {
      }

      public void partDeactivated(IWorkbenchPartReference) {
            IWorkbenchPart iworkbenchPart = IWorkbenchPartReference.getPart(boolean);     // REPEATED!
      }

      public void partVisible(IWorkbenchPartReference) {
      }

      public void partHidden(IWorkbenchPartReference) {
      }

      public void partActivated(IWorkbenchPartReference) {
            IWorkbenchPart iworkbenchPart = IWorkbenchPartReference.getPart(boolean);     // REPEATED!
      }

      public void selectionChanged(iworkbenchPart && structuredSelection) {
            PlatformUI.getWorkbench();
            IWorkbenchPage iworkbenchPage = iworkbenchWindow.getActivePage();
            IDocument idocument = sourceViewer.getDocument();     // REPEATED!
            idocument.set();  // REPEATED!
            StructuredSelection structuredSelection = new StructuredSelection(appLocationProvider);   // REPEATED!
            SelectionChangedEvent selectionChangedEvent = new SelectionChangedEvent(sourceViewer && tableViewer &&
                                                                                    structuredSelection);   // REPEATED!
            appAction.selectionChanged(selectionChangedEvent);    // REPEATED!
            Iterator iterator = structuredSelection.iterator();
            Display.getDefault();
      }

      public void partClosed(IWorkbenchPartReference) {
      }

      public void partOpened(IWorkbenchPartReference) {
      }
}

public class AppViewerSorter
extends ViewerSorter {

      public void sort(tableViewer) {
            IPreferenceStore ipreferenceStore = AbstractUIPlugin.getPreferenceStore();    // REPEATED!
            int app_int = ipreferenceStore.getInt(String);  // REPEATED!
      }

      public int compare(tableViewer) {
      }
}

public class AppStructuredContentProvider
implements IStructuredContentProvider {

      public void dispose() {
```

```
	}

	public Object[] getElements(iworkbenchPartSite && iviewSite && appLocationProvider) {
		IPreferenceStore ipreferenceStore = AbstractUIPlugin.getPreferenceStore();	// REPEATED!
		boolean app_boolean = ipreferenceStore.getBoolean(String);  // REPEATED!
	}

	public void inputChanged(tableViewer && iworkbenchPartSite && iviewSite && object2 && appLocationProvider) {
	}

}



public class AppLabelProvider
implements IColorProvider, IFontProvider
extends LabelProvider {

	public Image getImage(appLocationProvider) {
		Path path = new Path(String,String[],int)||()||(String)||(String,String);	// REPEATED!
		/*			Cyclic Statements			*/
		Image image1 = imageRegistry.get(String); // REPEATED!
		ImageDescriptor.createFromURL();	// REPEATED!
		imageRegistry.put(image);	// REPEATED!
		ImageRegistry imageRegistry = AbstractUIPlugin.getImageRegistry();	// REPEATED!
		Plugin.find(path);	// REPEATED!
	}

	public Color getBackground(Object) {
	}

	public String getText(appLocationProvider) {
	}

	public Color getForeground(Object) {
	}

	public Font getFont(Object) {
	}
}

public class AppRuleBasedPartitionScanner
extends RuleBasedPartitionScanner {

	public  AppRuleBasedPartitionScanner() {
		appRuleBasedPartitionScanner.setPredicateRules(IPredicateRule[]);
		Token token = new Token(int)||(Object);
		AppPredicateRule appPredicateRule = new AppPredicateRule(token);
	}

	public int read() {
	}
}

public class AppByteArrayTransfer
extends ByteArrayTransfer {

	public int[] getTypeIds() {
	}
}

public class AppTextFileDocumentProvider
implements IDocumentListener
extends TextFileDocumentProvider {

	public  AppTextFileDocumentProvider() {
		AppDocumentSetupParticipant appDocumentSetupParticipant = new AppDocumentSetupParticipant();
	}

	public void documentAboutToBeChanged(DocumentEvent) {
	}

	public void documentChanged(DocumentEvent) {
		int app_int6 = DocumentEvent.getOffset(); // REPEATED!
		int app_int5 = DocumentEvent.getLength(); // REPEATED!
		String string1 = DocumentEvent.getText(); // REPEATED!
		IDocument idocument1 = DocumentEvent.getDocument();	// REPEATED!
		idocument1.get(int,int):(String)||():(String);  // REPEATED!
		Region region = new Region(int,int);	// REPEATED!
	}

	public TextFileDocumentProvider$FileInfo createFileInfo(appLocationProvider) {
		IDocument idocument2 = ITextFileBuffer.getDocument();
		IAnnotationModel iannotationModel = ITextFileBuffer.getAnnotationModel();
		appDocumentSetupParticipant.setup(idocument2);
		/*			Cyclic Statements			*/
		idocument2.get(int,int):(String)||():(String);
		idocument2.addDocumentListener(appTextFileDocumentProvider);
	}
}
```

```
public class AppSourceViewerConfiguration
extends SourceViewerConfiguration {

      public IReconciler getReconciler(sourceViewer) {
            AppReconcilingStrategyExtension appReconcilingStrategyExtension = new AppReconcilingStrategyExtension(appTextListener &&
                                                                                                                    appLocationProvider);
            MonoReconciler monoReconciler = new MonoReconciler(appReconcilingStrategyExtension);
            NullProgressMonitor nullProgressMonitor = new NullProgressMonitor();
            monoReconciler.setProgressMonitor(nullProgressMonitor);
            monoReconciler.setDelay(int);
      }

      public IContentAssistant getContentAssistant(sourceViewer) {
      }

      public IPresentationReconciler getPresentationReconciler(sourceViewer) {
            AppPresentationDamager appPresentationDamager = new AppPresentationDamager(appTextListener && appLocationProvider);
            String string8 = appSourceViewerConfiguration.getConfiguredDocumentPartitioning(sourceViewer);
            PresentationReconciler presentationReconciler = new PresentationReconciler();
            presentationReconciler.setDocumentPartitioning(string7 && string8);
            /*                    Cyclic Statements                    */
            presentationReconciler.setDamager(appPresentationDamager);  // REPEATED!
            presentationReconciler.setRepairer(appPresentationDamager); // REPEATED!
      }

      public void getTextHover(sourceViewer) {
            AppTextHover appTextHover = new AppTextHover(appTextListener && appLocationProvider);
      }

      public ITextDoubleClickStrategy getDoubleClickStrategy(sourceViewer) {
            AppTextDoubleClickStrategy appTextDoubleClickStrategy = new AppTextDoubleClickStrategy();
      }

      public IAnnotationHover getAnnotationHover(sourceViewer) {
            AppAnnotationHover appAnnotationHover = new AppAnnotationHover(appTextListener && appLocationProvider);
      }

      public IAutoEditStrategy[] getAutoEditStrategies(sourceViewer) {
            DefaultIndentLineAutoEditStrategy defaultIndentLineAutoEditStrategy = new DefaultIndentLineAutoEditStrategy();
            AppAutoEditStrategy appAutoEditStrategy = new AppAutoEditStrategy(appTextListener && appLocationProvider);
      }

      public String[] getConfiguredContentTypes(sourceViewer) {
      }

      public String getConfiguredDocumentPartitioning(sourceViewer) {
      }
}

public class AppPresentationDamager
implements IPresentationDamager, IPresentationRepairer  {

      public IRegion getDamageRegion(ITypedRegion,DocumentEvent,boolean) {
      }

      public void setDocument(idocument1 && idocument2 && idocument3) {
      }

      public void setDocument(idocument1 && idocument2 && idocument3) {
      }
}

public class AppTextListener
implements ITextListener  {

      public void textChanged(TextEvent) {
      }
}

public class AppWorkbenchPage
implements IWorkbenchPage  {

      public IViewPart findView(String) {
      }
}

public class AppReconcilingStrategyExtension
implements IReconcilingStrategyExtension, IReconcilingStrategy  {

      public  AppReconcilingStrategyExtension() {
            AppPropertyChangeListener appPropertyChangeListener = new AppPropertyChangeListener(appTextListener && appLocationProvider);
      }

      public void setDocument(idocument1 && idocument2 && idocument3) {
      }

      public void initialReconcile() {
            AppLabelProvider appLabelProvider = new AppLabelProvider(appreferences$IPropertyChangeListener && appPropertyChangeListener);
            String string = ActionFactory.getId();    // REPEATED!
```

```
            Separator separator = new Separator(String)||();
            Composite composite = new Composite(Composite,int)||();
            Table table = new Table(composite);
            TableViewer tableViewer = new TableViewer(table);
            MenuManager menuManager = new MenuManager(String)||(String,String)||();
            AppMenuListener appMenuListener = new AppMenuListener();
            display.asyncExec(Runnable);
            GridData gridData = new GridData(int)||(int,int,boolean,boolean)||(int,int,boolean,boolean,int,int)||(int,int)||();     // REPEATED!
            AppPartListener2 appPartListener2 = new AppPartListener2();
            PlatformUI.getWorkbench();    // REPEATED!
            AppStructuredContentProvider appStructuredContentProvider = new AppStructuredContentProvider();
            tableViewer.setContentProvider(appStructuredContentProvider);
            tableViewer.setLabelProvider(appLabelProvider && decoratingLabelProvider);
            StructuredSelection structuredSelection = new StructuredSelection(appLocationProvider);
            GridLayout gridLayout = new GridLayout(int,boolean)||();
            /*                  Cyclic Statements                   */
            itoolBarManager.update(boolean);
            composite.setLayout(gridLayout);
            Menu menu = menuManager.createContextMenu(table && control);
            iactionBars.setGlobalActionHandler(string && appAction);    // REPEATED!
            iworkbenchHelpSystem.setHelp(composite);
            iactionBars.updateActionBars();
            control.setMenu(menu);
            menuManager.setRemoveAllWhenShown(boolean);
            control.setLayoutData(gridData);    // REPEATED!
            IActionBars iactionBars = iviewSite.getActionBars();  // REPEATED!
            itoolBarManager.add(separator && appAction);    // REPEATED!
            IMenuManager imenuManager = iactionBars.getMenuManager();
            IWorkbenchPartSite iworkbenchPartSite = IWorkbenchPart.getSite(); // REPEATED!
            tableViewer.setInput(iworkbenchPartSite && iviewSite && appLocationProvider);
            IWorkbenchPart.createPartControl(Composite);
            Object object1 = IWorkbenchPart.getAdapter(Class);
            IToolBarManager itoolBarManager = iactionBars.getToolBarManager();
            tableViewer.setSelection(structuredSelection);  // REPEATED!
            IViewSite iviewSite = IWorkbenchPart.getViewSite();   // REPEATED!
            Control control = tableViewer.getControl();     // REPEATED!
            menuManager.addMenuListener(appMenuListener);
            iviewSite.setSelectionProvider(tableViewer && tableViewer);
      }

      public void reconcile((DirtyRegion,IRegion):()||(IRegion):()) {
            Display.getDefault();
      }

      public void setProgressMonitor(nullProgressMonitor) {
      }
}

public class AppDocumentEvent
extends DocumentEvent {

      public String getText() {
            int app_int6 = DocumentEvent.getOffset();
            int app_int5 = DocumentEvent.getLength();
            String string1 = DocumentEvent.getText();
            appReconcilingStrategyExtension.reconcile(DirtyRegion,IRegion):()||(IRegion):();
      }
}

public class AppWorkbenchWindowActionDelegate
implements IWorkbenchWindowActionDelegate  {
}

public class AppMenuListener
implements IMenuListener  {

      public menuAboutToShow(ImenuManager menuManager) {
            AppragSourceListener appragSourceListener = new AppragSourceListener();
            tableViewer.addDragSupport(appragSourceListener);
            AppByteArrayTransfer appByteArrayTransfer = new AppByteArrayTransfer();
            Transfer.registerType();
            IWorkbenchPartSite iworkbenchPartSite = IWorkbenchPart.getSite();
            IWorkbenchWindow iworkbenchWindow1 = iworkbenchPartSite.getWorkbenchWindow();
            AppropTargetListener appropTargetListener = new AppropTargetListener();
            tableViewer.addDropSupport(appropTargetListener);
            /*                  Cyclic Statements                   */
            AppWorkbenchWindowActionDelegate appWorkbenchWindowActionDelegate = new AppWorkbenchWindowActionDelegate(tableViewer &&
                                                                                                                      appAction && iworkbenchWindow);
            AppAction appAction = new AppAction(sourceViewer && tableViewer && appPropertyChangeListener);
            menuManager.add(appAction && separator); // REPEATED!
      }
}

public class AppPropertyChangeListener
implements IPropertyChangeListener  {

      public  AppPropertyChangeListener() {
            Preferences preferences = Plugin.getPluginPreferences();
            preferences.addPropertyChangeListener(appreferences$IPropertyChangeListener && appPropertyChangeListener);
            IPreferenceStore ipreferenceStore = AbstractUIPlugin.getPreferenceStore();
```

```
            ipreferenceStore.addPropertyChangeListener(appreferences$IPropertyChangeListener && appPropertyChangeListener);
      }
}

public class AppAnnotationHover
implements IAnnotationHover  {
}

public class AppAutoEditStrategy
implements IAutoEditStrategy  {
}

public class AppTextDoubleClickStrategy
implements ITextDoubleClickStrategy  {

      public  AppTextDoubleClickStrategy() {
            DefaultTextDoubleClickStrategy defaultTextDoubleClickStrategy = new DefaultTextDoubleClickStrategy();
      }
}

public class AppTextHover
implements ITextHover  {
}

public class AppDocumentSetupParticipant
implements IDocumentSetupParticipant  {

      public void someMethod() {
            IDocumentPartitioner idocumentPartitioner = idocument2.getDocumentPartitioner(string7 && string8);
            AppRuleBasedPartitionScanner appRuleBasedPartitionScanner = new AppRuleBasedPartitionScanner();
            FastPartitioner fastPartitioner = new FastPartitioner(appRuleBasedPartitionScanner);
            fastPartitioner.connect(idocument2);
            idocument2.setDocumentPartitioner(fastPartitioner && string7 && string8);
      }
}

public class Appreferences$IPropertyChangeListener
implements Preferences$IPropertyChangeListener  {

      public  Appreferences$IPropertyChangeListener() {
            Preferences preferences = Plugin.getPluginPreferences();
            preferences.addPropertyChangeListener(appreferences$IPropertyChangeListener && appPropertyChangeListener);
            IPreferenceStore ipreferenceStore = AbstractUIPlugin.getPreferenceStore();
            ipreferenceStore.addPropertyChangeListener(appreferences$IPropertyChangeListener && appPropertyChangeListener);
      }
}

public class AppragSourceListener
implements DragSourceListener  {
}

public class AppropTargetListener
implements DropTargetListener  {
}

public class SomeClass {

      public void someMethod() {
            GlobalTemplateVariables$Cursor globalTemplateVariables$Cursor = new GlobalTemplateVariables$Cursor(); // REPEATED!
            ContributionContextTypeRegistry contributionContextTypeRegistry = new ContributionContextTypeRegistry();    // REPEATED!
            AppLocationProvider appLocationProvider = new AppLocationProvider();
            RGB rGB = new RGB(float,float,float)||(int,int,int);
            appTextFileDocumentProvider.documentChanged(DocumentEvent);
            /*                    Cyclic Statements                    */
            TemplateContextType templateContextType = contributionContextTypeRegistry.getContextType(String);     // REPEATED!
            FontRegistry fontRegistry = new FontRegistry(display);
            FontData[] fontDataArray = font1.getFontData();
            RGB rGB1 = color.getRGB();
            IWorkbench iworkbench = AbstractUIPlugin.getWorkbench();    // REPEATED!
            Color color = display.getSystemColor(int);
            ColorRegistry colorRegistry = new ColorRegistry(display);
            Object object1 = iworkbench.getAdapter(Class);  // REPEATED!
            PreferenceConverter.setDefault(rGB && ipreferenceStore && fontData && rGB1);
            contributionContextTypeRegistry.addContextType(TemplateContextType);    // REPEATED!
            IPreferenceStore ipreferenceStore = AbstractUIPlugin.getPreferenceStore();    // REPEATED!
            ipath.append(String):(IPath)||(IPath):(IPath);  // REPEATED!
            IPath ipath = Plugin.getStateLocation();  // REPEATED!
            FontData fontData = new FontData(string3);
            String string3 = FontData.getName();
            ContributionTemplateStore contributionTemplateStore = new ContributionTemplateStore(contributionContextTypeRegistry &&
                                                                                                 ipreferenceStore);      // REPEATED!
            ipreferenceStore.setDefault(String,String):()||(String,boolean):()||(String,float):()||(String,double):()||
                                                                                 (String,long):()||(String,int):();
            Display display = iworkbench.getDisplay();      // REPEATED!
            Font font1 = display.getSystemFont();
            templateContextType.addResolver(globalTemplateVariables$Cursor);  // REPEATED!
            int app_int2 = FontData.getHeight();
            contributionTemplateStore.load();   // REPEATED!
      }
}
```

Description of False Negatives:

The class AppLocationProvider should extend ViewPart. So, we have one false negtive (i.e., **extends** ViewPart).