UML Tool Actions - Diagram Elements Actions 1.0 Component Specification

1. Design

The Diagram Elements Actions component provides the Actions related to the DiagramElements declared in Diagram Interchange component. Supported actions are remove, copy, paste and cut DiagramElement into/from Diagram, which provided by the Application.

1. Design Patterns

Command Pattern : One instance of XXXDiagramElementAction represents one action.

2. Industry Standards

None

3. Required Algorithms

No specific algorithm required. The way this component works is quite straightforward.

For Copy : wraps given DiagramElement in Transferable, then put it inside Clipboard.

For Paste : extract DiagramElement from given Transferable, then insert it into the specified Diagram.

For Remove : removes the given DiagramElement from the specified Diagram.

For Cut : do copy and then remove.

For undo and redo, they works intuitively.

4. Component Class Overview

Following are overview of each class, for details please refer to Documentation tab on zuml file (Poseidon).

RemoveDiagramElementAction implements UndoableAction :

RemoveDiagramElementAction will provide action of removing diagram element given by the Application. This DiagramElement and a reference to its Diagram will be kept for undo purpose. It also provides undo and redo the action.

CopyDiagramElementAction implements TransientAction, ClipboardOwner:

CopyDiagramElementAction provides action of copying DiagramElement into Clipboard. CopyDiagramElementAction instantiated and configured by DiagramElement and Clipboard that passed by the Application. This Action will the copy the DiagramElement into the Clipboard.
This class will wrap the DiagramElement into Transferable object with DataFlavor gotten from DiagramElementDataFlavorManager.
This Transferable object will be transferred into the Clipboard

CopiedElement :

Implements Transferable, this class wraps DiagramElement to transfer it into Clipboard. It has a DataFlavor instance showing the element type of the DiagramElement.

.

PasteDiagramElementAction implements UndoableAction :

PasteDiagramElementAction will perform action of pasting the DiagramElement inside given Transferable object into the specified Diagram. It also can undo and redo the pasting action.

CutDiagramElementAction extends CompoundUndoableAction :

A compound class consists of CopyDiagramElementAction and RemoveDiagramElementAction. CutDiagramElementAction will perform cut i.e. copy and remove on the specified DiagramElement at the specified Diagram into the specified Clipboard.

It has CopyDiagramElementAction and RemoveDiagramElementAction one instance each.

Its execute method executes CopyDiagramElementAction first and then RemoveDiagramElementAction. Its undo simply undo RemoveDiagramElementAction. And its redo simply re-execute CopyDiagramElementAction and RemoveDiagramElementAction.

DiagramElementDataFlavorManager :

DiagramElementDataFlavorManager will keep all suitable DataFlavors for DiagramElement. Application can refer to DiagramElementDataFlavorManager to check whether a certain Transferable object (gotten from clipboard) or a certain DataFlavor is a DiagramElement or not.

If necessary, additional DataFlavor can be added on runtime.

5. Component Exception Definitions

ActionExecutionException

Thrown by execute() method when any of RuntimeException below occurs during execution. This Exception class will wrap the RuntimeException. As for redo() and undo(), javax.swing.undo.CannotUndoException and javax.swing.undo.CannotRedoException is used instead.

Following are reusable RuntimeExceptions from Java that are being used, for details please refer to Poseidon Documentation tabs.

java.lang.IllegalArgumentException

For constructors and some other method if null is being passed as argument.

java.lang.UnsupportedOperationException

Thrown when unsupported method are being called.

java.util.NoSuchElementException

Thrown when trying to remove inexisting DiagramElement from a Diagram. Might happen in RemoveDiagramElementAction.execute() or in PasteDiagramElementAction.undo() method.

javax.swing.undo.CannotUndoException
Thrown when undo operation failed for some reason.

javax.swing.undo.CannotRedoException
Thrown when redo operation failed for some reason.

6. Thread Safety

Although thread safeness is not really required for this component (which we will explain later), this component is indeed thread safe since most of its attributes are made final. Application will create and execute the Actions, Actions here acts as atomic operations in point of view of the Application, so thread safety will be the Application's responsibility. After all, this component is not to be accessed by multiple threads since each thread (if more than one) using this component will have its own instance. However, it is possible that multiple actions access same class simultaneously e.g. Clipboard, the Application should assumes this responsibility.

2. Environment Requirements

2.1 Environment

- Java1.5 is required for compilation and executing test cases.

2.2 TopCoder Software Components

- ActionManager 1.0
    - Used as manager of this component actions.
- DiagramInterchage 1.0
    - Used as manager of DiagramElement and Diagram which are object of this component actions.

7.3 Third Party Components

None

3. Installation and Configuration

3.1 Package Name

This component is contained in **com.topcoder.uml.actions.diagram.elements**

8.2 Configuration Parameters

Parameter	Description	Values

3.3 Dependencies Configuration

None

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration if any.
- Execute 'ant test' within the directory that the distribution was extracted to.

9.2 Required steps to use the component

When Application wants to do actions on DiagramElement, it has to creates and instance of the Action and then call the execute() method at the action instance. These instances can be stored in some list or so for performing undo/redo further on. Refer to documentation tabs in Poseidon for details.

4.3 Demo

Typical usage of this component will be :

Application will use this component to perform actions for it. To use this component, Application creates an instance of the XXXDiagramElementAction for each Action it wants to execute. He then can keep these instances as list of actions taken for further use e.g. Undo, redo.

- For RemoveDiagramElementAction :
  - Construction and execution :

```
//Application has DiagramElement diagramElement to be removed from
Diagram targetDiagram.

removeDiagramElement = new
RemoveDiagramElementAction(diagramElement, targetDiagram);

removeDiagramElement.execute(); // executes removal
```

  - Undo :

```
removeDiagramElement.undo();
```

  - Redo :

```
removeDiagramElement.redo();
```

- For CopyDiagramElementAction :
  - Construction and execution :

```
//Application has DiagramElement diagramElement to be copied into
Clipboard clipboard

copyDiagramElement = new CopyDiagramElementAction(diagramElement,
clipboard);

copyDiagramElement.execute();
```

- For PasteDiagramElementAction :
  - Construction and execution :

```
//Application has Transferable transferable which contains
DiagramElement to be pasted into Diagram diagram. It can check
```

```
whether it contains DiagramElement by consulting
DiagramElementDataFlavorManager.

DiagramElementDataFlavorManager = new
DiagramElementDataFlavorManager();

if(diagramElementDataFlavorManager.isSuitableForDiagramElement(trans
ferable) {

pasteDiagramElement = new PasteDiagramElementAction(transferable,
diagram);

pasteDiagramElement.execute();

}
```

- Undo :

```
pasteDiagramElement.undo();
```

- Redo :

```
pasteDiagramElement.redo();
```

- For CutDiagramElementAction :
  - Construction and execution :

```
//Application has DiagramElement diagramElement to be cut from
Diagram diagram into Clipboard clipboard.

cutDiagramElement = new CutDiagramElementAction(element, diagram,
clipboard);

cutDiagramElement.execute();
```

- Undo :

```
cutDiagramElement.undo();
```

- Redo :

```
cutDiagramElement.redo();
```

5. Future Enhancements

Copied and pasted diagram element (i.e. the Transferable object) can supports multiple types of DataFlavor at once, this can be used for future enhancement.