Change request log

Team & Role

Swetha Varadarajan: Made the concept location, did the change request, implemented validation code and also verified manually. Wrote the document.

Kartikay Sharma: Concept location, change request, validation.

(We both worked on it independently and came up with the same solution)

Change Request

The Alternate Mix and Merge modules of PDFsam provide buttons to move a document up or down in their respective list of files. However, these buttons are not ideal when the list of documents is long, and the user wants to move one of them to the top or the bottom. You are requested to add two new buttons in these modules, to allow the user to move a selected document to the top and bottom of the list.

Concept Location

Step #	Description	Rationale
1	*We used Eclipse IDE to run the application. We made certain changes to the pom file to make it run. These changes are described in the README file in the github repository.*	
2	*Swetha's concept location step1: Entered "move up button" in instasearch tool of eclipse IDE.*	*Three files were listed. SelectionTableToolbarWithoutMoveTest.java, SelectionTableToolbar.java, SelectionTableToolbarTest.java. Since two of these files were test files, I picked the one without the test abbreviated to it.*
3	*Swetha's concept location step2: Inspected SelectionTableToolbar.java and saw the "MoveType.TOP" keyword. When searching for this keyword using Instasearch, came across the file "MoveTypeTest.java". This file contained two other methods: MoveType.UP and MoveType.BOTTOM. This gave an insight to re-use the methods that use MoveType.TOP and MoveType.DOWN for our change request. Seeing all these dependencies, I marked the class "SelectionTableToolbar.java" as located and the first class to be changed.*	
4	*Kartikay's concept location step1: used "add document" keyword using Instasearch.*	*Inspected the MergeSelectionPaneTest.java and AlternateMixSelectionPaneTest.java. It didn't give much insight.*
5	*Kartikay's concept location step2: used "move up" keyword using Instasearch.*	*Inspected SelectionTable.java and saw various move events.*
6	*Kartikay's concept location step3: used keyword "move selected event".*	*This did not give much luck*
7	*Kartikay's concept location step4: used keyword "move type".*	*Inspected the list of files and maked "SelectionTableToolbar.java" as the place where the concept is located.*

Time spent (in minutes): Swetha : 10 minutes. Kartikay: 15 mins

[redacted]Impact Analysis

Step #	Description	Rationale
1	*We inspected the SelectionTableToolbar.java and saw how it is used in AlternateMix and Merge Module.*	*To track the classes that could be impacted by the change. We realized that if we make a change in this class, both the modules will be able to see the change as this class is been used in both the modules. We were able to analyze this in UML creation phase as well when viewing the call hierarchy.*
2	*We inspected the two test classes that creates instance of this class:* *SelectionTableToolbarWithoutMoveTest.java, SelectionTableToolbarTest.java.*	*We realized that SelectionTableToolbarTest.java can fail if we don't include the appropriate test cases after including the necessary features.*
3	*The class SelectionTableToolbarWithoutMoveTest.java was discarded from the list of classes to change*	*Because it did not include test cases for the selection toolbar with move tests.*

Time spent (in minutes): 25

[redacted]Prefactoring (optional)

We realized that there is no need for prefactoring for this change.

Time spent (in minutes): 5

[redacted]Actualization

Step #	Description	Rationale
1	*We added the following two items to the item list of the SelectionPane in order to create instances of the two buttons.* *new MoveTopButton(ownerModule), new MoveBottomButton(ownerModule)*	
2	*We created two classes MoveTopButton extends BaseMoveSelectedButton and MoveBottomButton extends BaseMoveSelectedButton*	*We copied the moveUp and moveDown implementations and replaced the movetype to top and bottom respectively.*
3	*We created unit tests for the functionality in SelectionTableToolbarTest.java and also performed functional testing. We also ran the existing test cases.*	*To make sure everything works.*

Time spent (in minutes): 30

[redacted]Post factoring (optional)

We realized that there is no post-factoring that can be done for this change request.

Time spent (in minutes): 10

[redacted]Validation

Step #	Description	Rationale
1	*Test case defined:* *enableByFiringSingleSelectionChange* *Inputs: Is the particular node (button). Here nodes are moveTOP and moveBOTTOM* *Expected output: True if the node is enabled/disabled according to the code.*	*This is the regular expected behavior.* *The test passed.*
2	*Manual test case:* *Input: Adding 5 files to the list, Select the third file and move it to the top of list. Select the same file and move it to the bottom of the list.* *Output: The third file should be at the top of the list and then at the bottom of the list.*	*This is the regular expected behavior.* *The test passed.*

Time spent (in minutes): 20

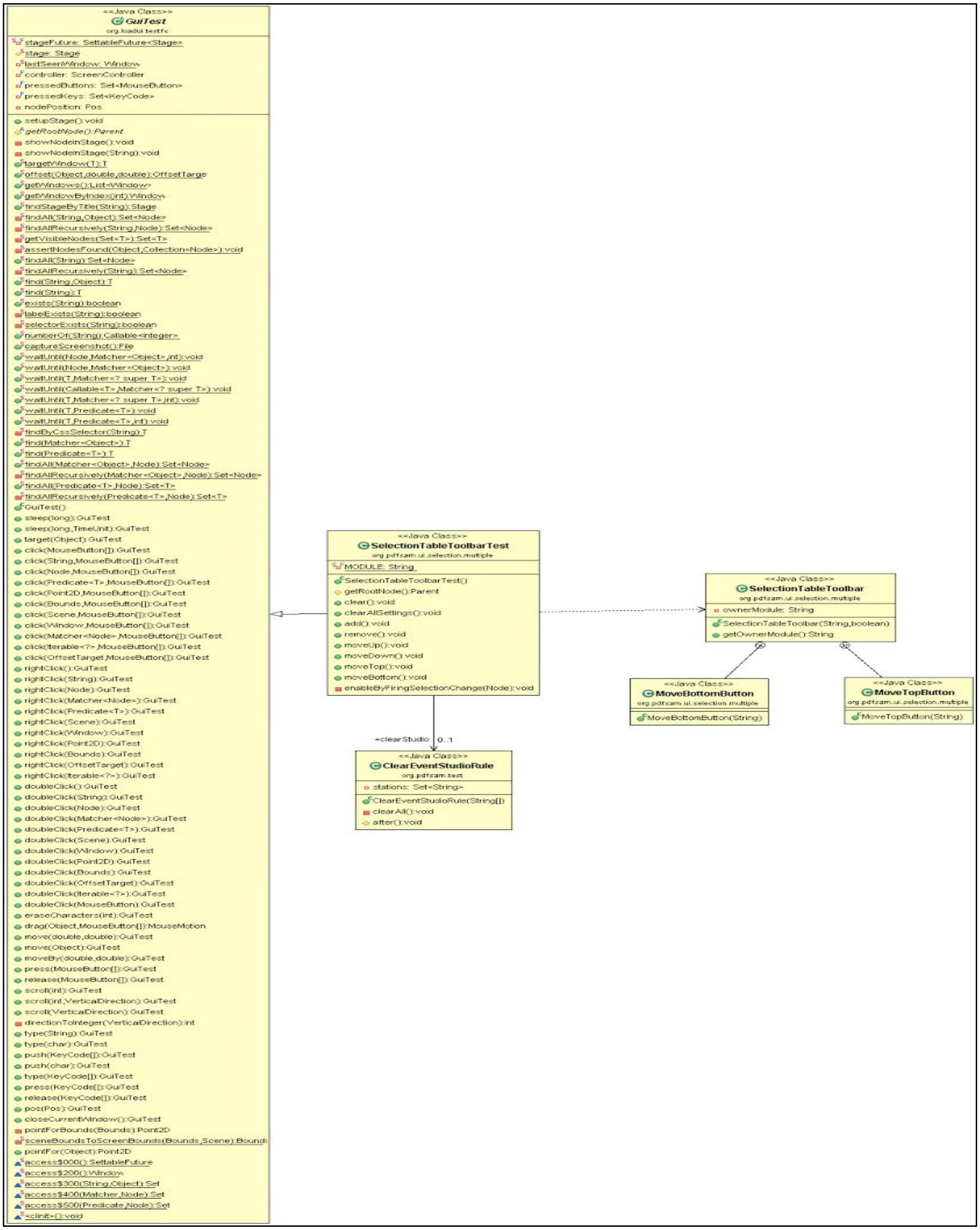
[redacted]Timing

Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	15
Impact Analysis	25
Prefactoring	5
Actualization	30
Post Factoring	10
Verification	20
Total	105

[redacted]Reverse engineering

We couldn’t get the license for UML sequence diagram. But, we were able to create a partial UML diagram for the affected class and its dependencies. The first diagram is very huge. So, we removed the GUI Test class and shown the reduced UML diagram in the following diagram.



<<Java Class>>
GuiTest
org.loadui.testfx
stageFuture: SettableFuture<Stage>
stage: Stage
lastSeenWindow: Window
controller: ScreenController
pressedButtons: Set<MouseButton>
pressedKeys: Set<KeyCode>
nodePosition: Pos
setupStage():void
getRootNode():Parent
showNodeInStage():void
showNodeInStage(String):void
targetWindow(T):T
offset(Object,double,double):OffsetTarge
getWindows():List<Window>
getWindowByIndex(int):Window
findStageByTitle(String):Stage
findAll(String,Object):Set<Node>
findAllRecursively(String,Node):Set<Node>
getVisibleNodes(Set<T>):Set<T>
assertNodesFound(Object,Collection<Node>):void
findAll(String):Set<Node>
findAllRecursively(String):Set<Node>
find(String,Object):T
find(String):T
exists(String):boolean
labelExists(String):boolean
selectorExists(String):boolean
numberOf(String):Callable<Integer>
captureScreenshot():File
waitUntil(Node,Matcher<Object>,int):void
waitUntil(Node,Matcher<Object>):void
waitUntil(T,Matcher<? super T>):void
waitUntil(Callable<T>,Matcher<? super T>):void
waitUntil(T,Matcher<? super T>,int):void
waitUntil(T,Predicate<T>):void
waitUntil(T,Predicate<T>,int):void
findByCssSelector(String):T
find(Matcher<Object>):T
find(Predicate<T>):T
findAll(Matcher<Object>,Node):Set<Node>
findAllRecursively(Matcher<Object>,Node):Set<Node>
findAll(Predicate<T>,Node):Set<T>
findAllRecursively(Predicate<T>,Node):Set<T>
GuiTest()
sleep(long):GuiTest
sleep(long,TimeUnit):GuiTest
target(Object):GuiTest
click(MouseButton[]):GuiTest
click(String,MouseButton[]):GuiTest
click(Node,MouseButton[]):GuiTest
click(Predicate<T>,MouseButton[]):GuiTest
click(Point2D,MouseButton[]):GuiTest
click(Bounds,MouseButton[]):GuiTest
click(Scene,MouseButton[]):GuiTest
click(Window,MouseButton[]):GuiTest
click(Matcher<Node>,MouseButton[]):GuiTest
click(Iterable<?>,MouseButton[]):GuiTest
click(OffsetTarget,MouseButton[]):GuiTest
rightClick():GuiTest
rightClick(String):GuiTest
rightClick(Node):GuiTest
rightClick(Matcher<Node>):GuiTest
rightClick(Predicate<T>):GuiTest
rightClick(Scene):GuiTest
rightClick(Window):GuiTest
rightClick(Point2D):GuiTest
rightClick(Bounds):GuiTest
rightClick(OffsetTarget):GuiTest
rightClick(Iterable<?>):GuiTest
doubleClick():GuiTest
doubleClick(String):GuiTest
doubleClick(Node):GuiTest
doubleClick(Matcher<Node>):GuiTest
doubleClick(Predicate<T>):GuiTest
doubleClick(Scene):GuiTest
doubleClick(Window):GuiTest
doubleClick(Point2D):GuiTest
doubleClick(Bounds):GuiTest
doubleClick(OffsetTarget):GuiTest
doubleClick(Iterable<?>):GuiTest
doubleClick(MouseButton):GuiTest
eraseCharacters(int):GuiTest
drag(Object,MouseButton[]):MouseMotion
move(double,double):GuiTest
move(Object):GuiTest
moveBy(double,double):GuiTest
press(MouseButton[]):GuiTest
release(MouseButton[]):GuiTest
scroll(int):GuiTest
scroll(int,VerticalDirection):GuiTest
scroll(VerticalDirection):GuiTest
directionToInteger(VerticalDirection):int
type(String):GuiTest
type(char):GuiTest
push(KeyCode[]):GuiTest
push(char):GuiTest
type(KeyCode[]):GuiTest
press(KeyCode[]):GuiTest
release(KeyCode[]):GuiTest
pos(Pos):GuiTest
closeCurrentWindow():GuiTest
pointForBounds(Bounds):Point2D
sceneBoundsToScreenBounds(Bounds,Scene):Bound
pointFor(Object):Point2D
access$000():SettableFuture
access$200():Window
access$300(String,Object):Set
access$400(Matcher,Node):Set
access$500(Predicate,Node):Set
<clinit>():void
<<Java Class>>
SelectionTableToolbarTest
org.pdfsam.ui.selection.multiple
MODULE: String
SelectionTableToolbarTest()
getRootNode():Parent
clear():void
clearAllSettings():void
add():void
remove():void
moveUp():void
moveDown():void
moveTop():void
moveBottom():void
enableByFiringSelectionChange(Node):void
+clearStudio
0..1
<<Java Class>>
ClearEventStudioRule
org.pdfsam.test
stations: Set<String>
ClearEventStudioRule(String[])
clearAll():void
after():void
<<Java Class>>
SelectionTableToolbar
org.pdfsam.ui.selection.multiple
ownerModule: String
SelectionTableToolbar(String,boolean)
getOwnerModule():String
<<Java Class>>
MoveBottomButton
org.pdfsam.ui.selection.multiple
MoveBottomButton(String)
<<Java Class>>
MoveTopButton
org.pdfsam.ui.selection.multiple
MoveTopButton(String)

Figure 1: UML showing all the dependencies for Change Request #1 of PDFSam

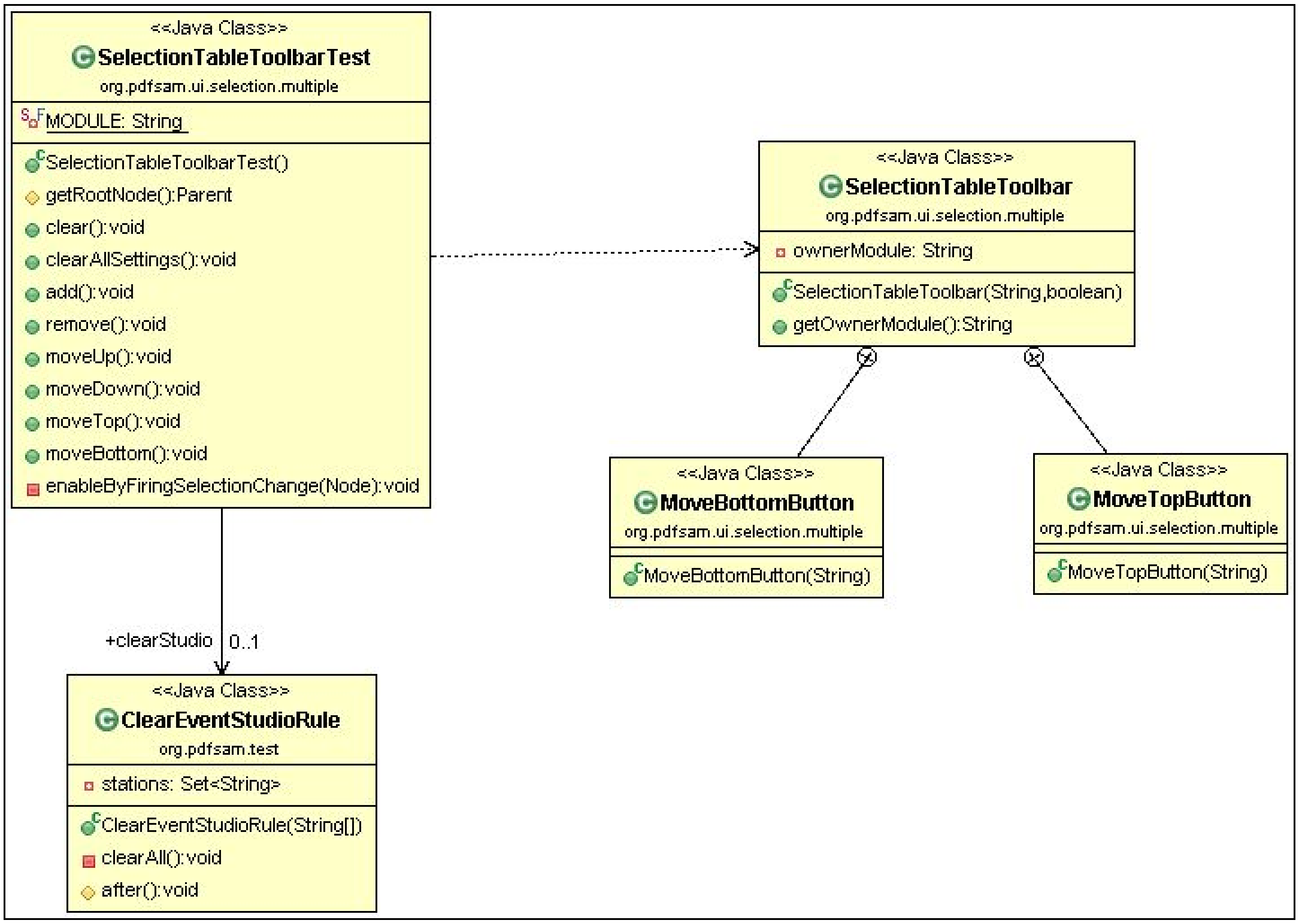


Figure 2: UML showing all reduced dependencies for Change Request #1 of PDFSam

[redacted] Conclusions

For this change, concept location was easy. But, we were playing with other change requests before diving into this one. So, we had a small experience in using the appropriate keyword to find the location. The impact analysis was a bit challenging because of the complex dependencies. Analyzing through UML diagram was easy to comprehend. This also made us realize that having a good knowledge of various tools and plugins is essential for quicker software development. Actualization and Validation were also pretty straightforward to implement.

Classes and method changed:

- SelectionTableToolbarTest.java
- SelectionTableToolbar.java