Change request log

Team
gitrams

Change Request

Change Request ID: #1 (PDFsam)

Description of the change request: In the Alternate Mix and Merge modules of PDFsam add two new buttons "Move Top" and "Move Bottom" to allow the user to move a selected document to the top and bottom of the list.

Concept Location

- IDE Features used: **InstaSearch** for searching the relevant keywords such as 'Move Up' that had similar functionality that we wanted to implement.
- Queries used when searching: 'Move Up' and 'Move Down'
- Interactions with the system: Split and Merge page of PDFsam
- Classes visited: `SelectionTableToolbar, MultipleSelectionAndFocus, MoveType`
- The first class found to be changed (this is when concept location ends): `SelectionTableToolbar`

Step #	Description	Rationale
1	*We compiled and ran PDFsam v3.3.5*	
2	*We interacted with split and merge function of PDFsam to understand basic move up and down features.*	*To get familiar with the existing features such as Move Up and Move Down, that had similar functionality to what we wanted to implement We identified the elements we had to change and add to implement the change request.*
3	*We searched for "Move Up" using the InstaSearch plugin installed in Eclipse Oxygen2*	*Because we wanted to add Move Top button beside the Move Up button.*
4	*From over 10 results, outputted we clicked on following classes SelectionTableToolbar, MultipleSelectionAndFocus, MoveType. The class was inspected using the Eclipse IDE*	*We reasoned that we can add buttons called 'Move Top' and 'Move Bottom' in the same class.*
5	*We went through the class `MoveType` that was a dependency of `SelectionTableToolbar`. We noticed that the Move Top and Move Bottom was already implemented but was not enabled in v3.3.5*	*We noticed that the public `enum MoveType {}` implemented Move Up functionality.*
6	*We decided to add two more classes `MoveTopButton and MoveBottomButton` to the `SelectionTableToolbar` to enable the already written Move Top and Bottom functionality*	*Instead of implementing those buttons from scratch we decided to use the already written piece of code from public `enum MoveType {}`.*
7	*The methods will require us to add two more arguments to `getItems().addAll()` inside SelectionTableToolbar corresponding to the newly added buttons.*	*We confirmed that the class `SelectionTableToolbar` had to be modified.*

Time spent (in minutes): 70

Impact Analysis

Use the table below to describe each step you follow when performing impact analysis for this change request. Include as many details as possible, including why classes are visited or why they are discarded from the estimated impact set.

Do not take the impact analysis of your changes lightly. Remember that any small change in the code could lead to large changes in the behavior of the system. Follow the impact analysis process covered in the class. Describe in details how you followed this process in the change request log. Provide details on how and why you finished the impact analysis process.

Step #	Description	Rationale
1	*We inspected the class SelectionTableToolbar*	*To track the classes that could be impacted by the change.*
2	*We took the help of the UML plugin to locate affected classes*	*UML tool can locate related classes easily by visiting all the classes in the project.*

Time spent (in minutes): 20

Actualization

Use the table below to describe each step you followed when changing the code. Include as many details as possible, including why classes/methods were modified, added, removed, renamed, etc.

Step #	Description	Rationale
1	*We decided to add two more classes MoveTopButton and MoveBottomButton to the SelectionTableToolbar and enable the already written Move Top and Bottom functionality*	*Instead of implementing those buttons from scratch we decided to use the already written piece of code.*
2	*We added two more arguments to* `getItems().addAll()` *inside SelectionTableToolbar corresponding to the newly added buttons.*	*We committed the incremental changes to github just to make sure we can have a log that might help us go back if needed.*

Time spent (in minutes): 40

Postfactoring (optional)

For this change request we did not require to postfactor any part of the code base except for adding a few comments. This is because the Move Top and Move Bottom functionalities were similar to the Move Up and Down buttons' functionalities that were already present in the code base.

Step #	Description	Rationale
1	*Added comments regarding the newly implemented buttons.*	*We wanted to keep a record/documentation in the form of comments in the source code.*

Time spent (in minutes): 10

[redacted]Validation

Use the table below to describe any validation activity (e.g., testing, code inspections, etc.) you performed for this change request. Include the description of each test case, the result (pass/fail) and its rationale.

Step #	Description	Rationale
1	*We performed manual testing.*	*To validate the functionality of the new features.*
2	*We started the AlternateMix module of PDFsam and added a lot of documents, making sure more than 2 documents are added.*	*At least more than two documents should be loaded to verify correct functionality*
3	*We clicked on any random document in the list of loaded documents and selected "Move to Top"*	*The document moved to the top of the list, demonstrating correct functionality.*
4	*We clicked on any random document in the list of loaded documents and selected "Move to Bottom"*	*The document moved to the bottom of the list, demonstrating correct functionality.*
5	*We selected a document which was already at the top of the list and clicked "Move to Top"*	*The document stayed at its place, ensuring correct behavior for corner cases too.*
6	*We selected a document which was already at the bottom of the list and clicked "Move to Bottom"*	*The document stayed at its place, ensuring correct behavior for corner cases too.*

Time spent (in minutes): 20

[redacted]Timing

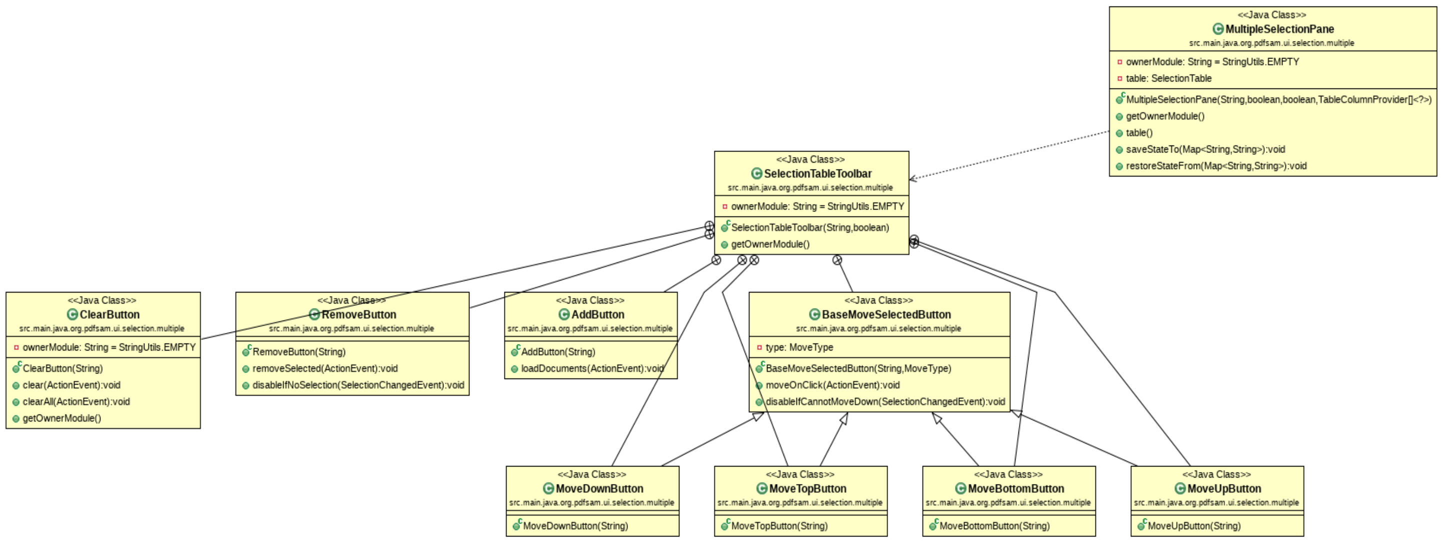
Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	70
Impact Analysis	20
Prefactoring	0
Actualization	40
Postfactoring	10
Verification	20
Total	160

[redacted]Reverse engineering

Create a UML sequence diagram (or more if needed) corresponding to the main object interactions affected by your change.

Create a partial UML class diagram of the classes visited while navigating through the code. Include the associations between classes (e.g., inheritance, aggregations, compositions, etc.), as well as the important fields and methods of each class that you learn about. The diagram may have disconnected components. Use the UML tool of your preference. When a significant fact about a class or method is learned, indicate it via annotations on the diagram. **For each change request, start with the diagram produced in the previous change request. For the first, you will start from scratch.**



<<Java Class>>
MultipleSelectionPane
src.main.java.org.pdfsam.ui.selection.multiple
ownerModule: String = StringUtils.EMPTY
table: SelectionTable
MultipleSelectionPane(String,boolean,boolean,TableColumnProvider[]<?>)
getOwnerModule()
table()
saveStateTo(Map<String,String>):void
restoreStateFrom(Map<String,String>):void
<<Java Class>>
SelectionTableToolbar
src.main.java.org.pdfsam.ui.selection.multiple
ownerModule: String = StringUtils.EMPTY
SelectionTableToolbar(String,boolean)
getOwnerModule()
<<Java Class>>
ClearButton
src.main.java.org.pdfsam.ui.selection.multiple
ownerModule: String = StringUtils.EMPTY
ClearButton(String)
clear(ActionEvent):void
clearAll(ActionEvent):void
getOwnerModule()
<<Java Class>>
RemoveButton
src.main.java.org.pdfsam.ui.selection.multiple
RemoveButton(String)
removeSelected(ActionEvent):void
disableIfNoSelection(SelectionChangedEvent):void
<<Java Class>>
AddButton
src.main.java.org.pdfsam.ui.selection.multiple
AddButton(String)
loadDocuments(ActionEvent):void
<<Java Class>>
BaseMoveSelectedButton
src.main.java.org.pdfsam.ui.selection.multiple
type: MoveType
BaseMoveSelectedButton(String,MoveType)
moveOnClick(ActionEvent):void
disableIfCannotMoveDown(SelectionChangedEvent):void
<<Java Class>>
MoveDownButton
src.main.java.org.pdfsam.ui.selection.multiple
MoveDownButton(String)
<<Java Class>>
MoveTopButton
src.main.java.org.pdfsam.ui.selection.multiple
MoveTopButton(String)
<<Java Class>>
MoveBottomButton
src.main.java.org.pdfsam.ui.selection.multiple
MoveBottomButton(String)
<<Java Class>>
MoveUpButton
src.main.java.org.pdfsam.ui.selection.multiple
MoveUpButton(String)

Conclusions

For this change, the hardest part was building the software. We faced several difficulties but were finally able to build it. Concept location was not very hard as we could locate the target class easily. Actualization was also not tough as most of the code was already there, we just had to enable it.

Classes and methods changed:

- pdfsam-fx/src/main/java/org/pdfsam/ui/selection/multiple/SelectionTableToolbar.java
  - SelectionTableToolbar(String ownerModule, boolean canMove)
  - MoveTopButton (new class added)
  - MoveBottomButton (new class added)