Change request log

Team

Kartikay Sharma: Implementation, verification, document writing

Swetha Varadarajan: Concept location, Document writing.

Change Request

The Merge Module throws an exception upon attempting to merge page ranges that intersect. You are requested to fix this issue by allowing intersection of ranges during the merging operation.

Concept Location

Step #	Description	Rationale
1	*We started where we left off with change, where we managed to bypass the exception. But it had the undesirable effect of not allowing duplicates in the overlapping region in the intersections. We decided to proceed along the same approach and find the solution.*	*We felt that the required change had been implemented in parts, and so we only needed to implement the remaining requirement.*
2	*We ran the system and executed a merge task and looked at the generated logs. We found that* ***org.sejda.impl.sambox.MergeTask*** *class is responsible for handling the merging operation. So we analysed it.*	*We wanted to find if there were any clues about which classes are responsible for executing the merge task.*
3	*We found that, when it performs the merge operation, it reads the inputs in the MergeParameters object and calls the getPages() method overridden from org.sejda.model.pdf.page.PageSelection interface, to get the list of all pages for those objects over its page ranges. We decided that we needed to find another approach for the problem because the call returns a Set, and not a List, of pages. This meant that the duplicated pages would be considered only once in the set.*	*All the strategies, to proceed along this approach, led to dead ends.*
4	*Now we knew that if the PdfMergeInput object had multiple ranges, there wasn't a solution where we could satisfy all the requirements of the change request. So we decided to check if there was a way where we could create multiple PdfMergeInput objects for an input file when multiple page ranges are provided by the user.*	*This approach theoretically solves both issues. The page ranges would now be in different objects, so there won't be any intersection problems and all the duplicates would also be preserved.*
5	*On analyzing the merge module, we found that MergeSelectionPane class is responsible for reading the inputs from the selection table and creates the PdfMergeInput objects.*	*We marked this class as "Located"*

Time spent (in minutes): 150

Impact Analysis

Step #	Description	Rationale
1	*We marked that MergeSelectionPane **needed to be changed** as it where the concept is located.*	*To track the classes that could be impacted by the change.*
2	*We realized that the statement creates PdfMergeInput in apply() method (in MergeSelectionPane Class) makes a call to SelectionTableRowData.toPageRangeSet(). We decided to add another method .toPageRangeList() to this class, so this **needed to be changed** as well*	*For page ranges [1-3, 2-4, 1, 1, 1, 1, 1], this call returns a set of [1-3, 2-4, 1]. To get the correct values we need a list instead of a set.*
3	*SelectionTableRowData.toPageRangeSet() delegated the work to ConversionUtils.toPageRangeSet(), so we decided to add toPageRangeList() in this class too. So ConversionUtils class too **needed to be changed.***	*For page ranges [1-3, 2-4, 1, 1, 1, 1, 1], this call returns a set of [1-3, 2-4, 1]. To get the correct values we need a list instead of a set.*
4	*We made a list of classes that use MergeSelectionPane. There were two classes- MergeSelectionPaneTest.java and MergeModule.java* *We inspected the class MergeModule.java*	*We realized this class **need not be changed** because it makes an instance of MergeParameterBuilder.*
5	*We inspected MergeSelectionPaneTest.java and marked that it **needed to be changed.***	*The notEmptyPageSelection() test case was designed to work with previous implementation. It expects that MergeSelectionPane class will create one PdfMergeInput object for one file. But our new implementation creates as many objects as there are page ranges.*

Time spent (in minutes): 45

Prefactoring **(optional)**

We didn't have any pre-factoring to do.

Time spent (in minutes): 5

Actualization

Step #	Description	Rationale
1	*In the MergeSelectionPane.java, we modified the apply() method. We updated lambda expression in map() call so that it returned a List of PdfMergeInput objects for each page range. We also updated the lambda expression in the following forEach() statement, so that it calls the addInput() method for each object in the List returned from the map() call.*	*This satisfies the requirements of the change request.*

2	*In the MergeSelectionPane.java, we also replaced the call i.toPageRangeSet() to i.toPageRangeList()*	*To get all the required page ranges including duplicates.*
3	*In the SelectionTableRowData.java, we added toPageRangeList() method which delegates the work to ConversionUtils.toPageRangeList() method*	*The method returns a list of page ranges as a List, rather than as a Set. This preserves duplicates.*
4	*In the ConversionUtils.java, we added toPageRangeList() method which works similar to toPageRangeSet() except that it returns the result as a List*	*The method returns a list of page ranges as a List, rather than as a Set. This preserves duplicates.*
5	*In MergeSelectionPaneTest.java, we changed the notEmptyPageSelection() test case.*	*We saw that the behavior of notEmptyPageSelection() Test is now different because the number of calls it makes to addInput() method of MergeParametersBuilder now depends on the the page ranges (There will be as many calls to addInput() as there are page ranges because, now, there would be as many PdfMergeInput objects). Also, it previously expected the selection range size to remain constant for any number of page ranges. But in the new implementation the PdfMergeInput object will only contain one page range, so the assertion statement, that checks the selection size, had to be changed as well. It now expects that the size would be 1.*
6	*We created unit tests for the SelectionTableRowDataTest.java and MergeSelectionPaneTest.java for the new methods in the respective classes.*	*To make sure the newly added methods behave as expected.*

Time spent (in minutes): 180

Post factoring **(optional)**

We didn't have any pre-factoring to do

Time spent (in minutes): 5

Validation

Step #	Description	Rationale
1	***pdfsam-fx/src/test/java/org/pdfsam/ui/selection/multiple/SelectionTableRowDataTest.java*** *TestCase: emptyListTest()* *Inputs: void* *Expected output: Validation should not fail*	*This test case performs the same functionality test as emptyListSet() but does so by calling toPageRangeList() instead of toPageRangeSet().* *The test passed.*
2	***pdfsam-core/src/test/java/org/pdfsam/support/params/ConversionUtilsTest.java*** *TestCase: public void invalid2()* *Inputs: void* *Expected output: Validation should not fail*	*This is the regular expected behavior.* *This test case performs the same functionality test as invalid1() but does so by calling ConversionUtils.toPageRangeList() instead of ConversionUtils.toPageRangeSet().* *The test passed.*

3	***pdfsam-core/src/test/java/org/pdfsam/support/params/ConversionUtilsTest.java*** *TestCase: public void invalidRange2()* *Inputs: void* *Expected output: Validation should not fail*	*This test case performs the same functionality test as invalidRange1() but does so by calling ConversionUtils.toPageRangeList() instead of ConversionUtils.toPageRangeSet().* *The test passed.*
4	***pdfsam-core/src/test/java/org/pdfsam/support/params/ConversionUtilsTest.java*** *TestCase: public void endLower2()* *Inputs: void* *Expected output: Validation should not fail*	*This test case performs the same functionality test as endLower1() but does so by calling ConversionUtils.toPageRangeList() instead of ConversionUtils.toPageRangeSet().* *The test passed.*
5	***pdfsam-core/src/test/java/org/pdfsam/support/params/ConversionUtilsTest.java*** *TestCase: public void singlePage2()* *Inputs: void* *Expected output: Validation should not fail*	*This test case performs the same functionality test as singlePage1() but does so by calling ConversionUtils.toPageRangeList() instead of ConversionUtils.toPageRangeSet().* *The test passed.*
6	***pdfsam-core/src/test/java/org/pdfsam/support/params/ConversionUtilsTest.java*** *TestCase: public void rangePage2()* *Inputs: void* *Expected output: Validation should not fail*	*This test case performs the same functionality test as rangePage1() but does so by calling ConversionUtils.toPageRangeList() instead of ConversionUtils.toPageRangeSet().* *The test passed.*
7	***pdfsam-core/src/test/java/org/pdfsam/support/params/ConversionUtilsTest.java*** *TestCase: public void endPage2()* *Inputs: void* *Expected output: Validation should not fail*	*This test case performs the same functionality test as endPage1() but does so by calling ConversionUtils.toPageRangeList() instead of ConversionUtils.toPageRangeSet().* *The test passed.*
8	***pdfsam-core/src/test/java/org/pdfsam/support/params/ConversionUtilsTest.java*** *TestCase: public void startPage2()* *Inputs: void* *Expected output: Validation should return true*	*This test case performs the same functionality test as startPage1() but does so by calling ConversionUtils.toPageRangeList() instead of ConversionUtils.toPageRangeSet().* *The test passed.*
9	***pdfsam-core/src/test/java/org/pdfsam/support/params/ConversionUtilsTest.java*** *TestCase: public void multiple2()* *Inputs: void* *Expected output: Validation should not fail*	*This test case performs the same functionality test as multiple1() but does so by calling ConversionUtils.toPageRangeList() instead of ConversionUtils.toPageRangeSet().* *The test passed.*
10	***Regression Testing***	*Ran the updated test suite, all test cases passed*
11	***Manual Testing with the following inputs*** ***Test 1:*** 1. *Inputs: Blank Page Range* 2. *Expected output: Output file should contain all pages* ***Test 2:*** 3. *Inputs: [1-3,2-4]* 4. *Expected output: Output file should contain 1,2,3,2,3,4 pages in order* ***Test 3:*** 5. *Inputs: [1-3,2-4,1,2,1,2,1]* 6. *Expected output: Output file should contain 1,2,3,2,3,4,1,2,1,2,1 pages in order*	*All Tests passed*

Time spent (in minutes): 40

Timing

Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	150
Impact Analysis	45
Prefactoring	5
Actualization	180
Post-factoring	5
Verification	40
Total	325

Reverse engineering

Figure 1 and figure 2 shows the partial UML diagram. Figure 1 doesn't show the attributes because of lack of space. It also doesn't show the classes from previous change and its effect on the second change. The second UML diagram has no attributes and operations. It shows the related classes from both the changes. It can be seen that there is no relation between the set of classes used for the first and the the set used for the second change.

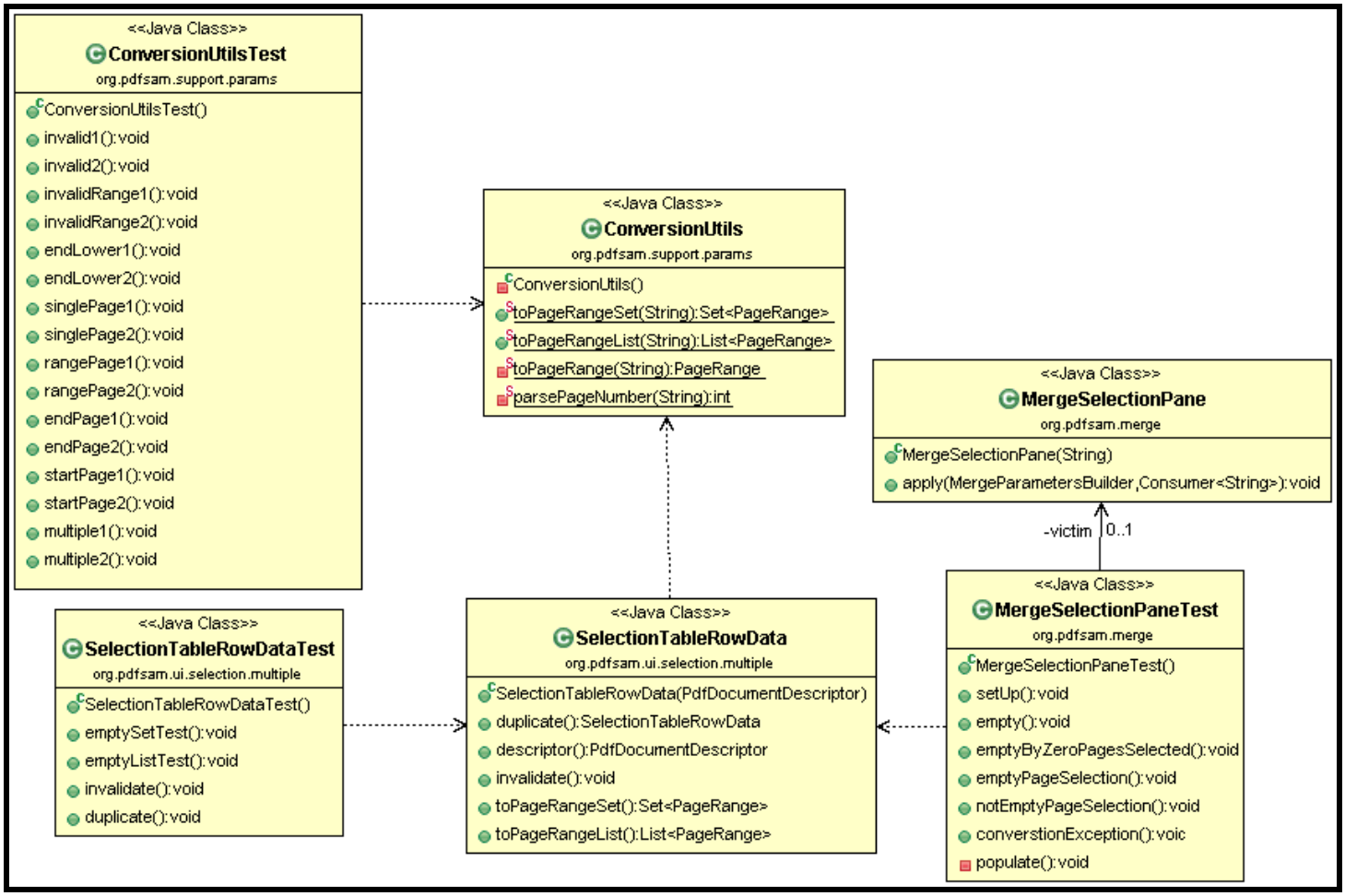

Figure 1: UML diagram showing the relationship between classes used for Change Request #3

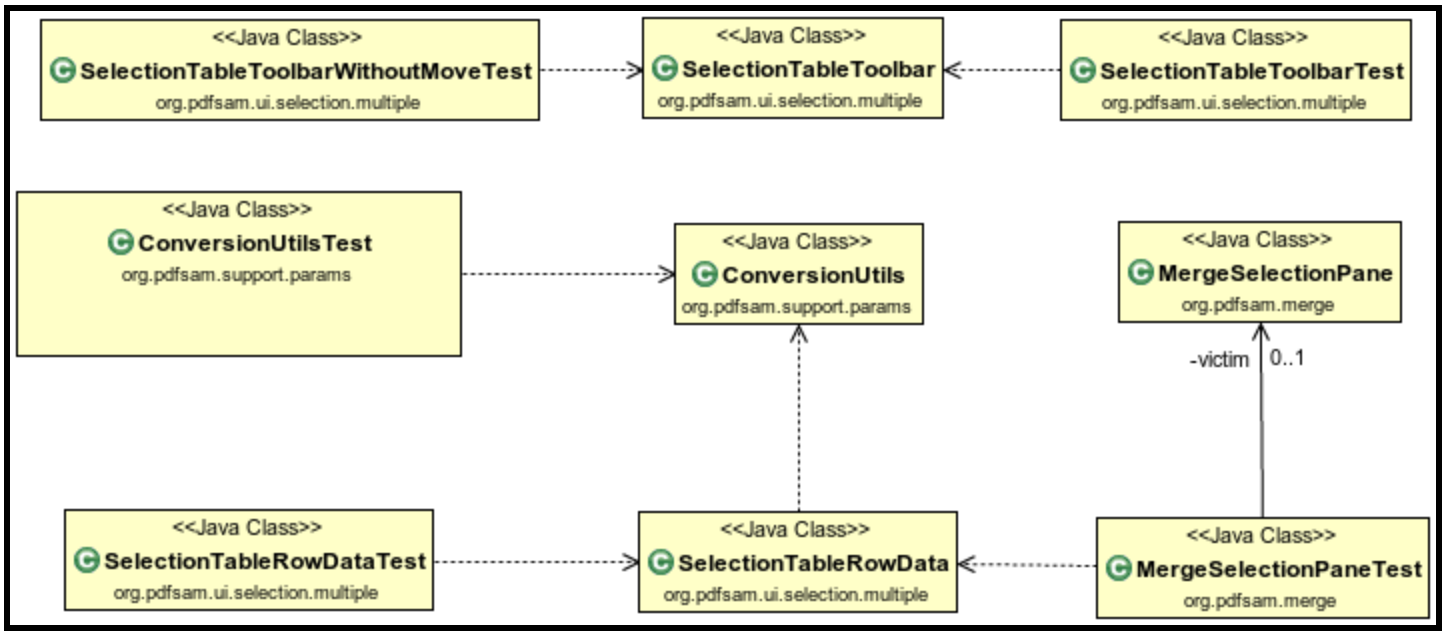

Figure 2: UML diagram showing the relationship between classes used for Change Request #1 and #3

Conclusions

For this change, we spent most of the time in concept location. This is because we tried to implement the change using our previous approach and we spent a lot of time chasing dead ends. Once we found the concept location, the next hurdle was actualization, where we had to modify the lambda expressions in such a way that it worked with the previously defined test cases. Other stages were straightforward.

Classes and methods changed:

- *pdfsam-fx/src/test/java/org/pdfsam/ui/selection/multiple/SelectionTableRowDataTest.java*
  - *public void emptySetTest() throws ConversionException()* (RENAMED)
  - *public void emptyListTest() throws ConversionException()*
- *pdfsam-fx/src/main/java/org/pdfsam/ui/selection/multiple/SelectionTableRowData.java*
  - *public List<PageRange> toPageRangeList() throws ConversionException()*
- *pdfsam-core/src/main/java/org/pdfsam/support/params/ConversionUtils.java*
  - *public static List<PageRange> toPageRangeList(String selection) throws ConversionException()*
- *pdfsam-core/src/test/java/org/pdfsam/support/params/ConversionUtilsTest.java*
  - *public void invalid1()* (RENAMED)
  - *public void invalid2()*
  - *public void invalidRange1()* (RENAMED)
  - *public void invalidRange2()*
  - *public void endLower1()* (RENAMED)
  - *public void endLower2()*
  - *public void singlePage1()* (RENAMED)
  - *public void singlePage2()*
  - *public void rangePage1()* (RENAMED)
  - *public void rangePage2()*
  - *public void endPage1()* (RENAMED)
  - *public void endPage2()*
  - *public void startPage1()* (RENAMED)
  - *public void startPage2()*
  - *public void multiple1()* (RENAMED)
  - *public void multiple2()*
- *pdfsam-merge/src/main/java/org/pdfsam/merge/MergeSelectionPane.java*
  - *public void apply(MergeParametersBuilder builder, Consumer<String> onError)*
- *pdfsam-merge/src/test/java/org/pdfsam/merge/MergeSelectionPaneTest.java*
  - *public void notEmptyPageSelection() throws Exception*

NOTE: The github commit log shows more changes. But these changes were essentially the deletions that we made from our previous version of the same change request.