Change request log

Team

Kartikay Sharma: Implementation and test cases

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 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	*We ran the application and created the exception by trying to specify the pages with intersection.*	*By examining the exception, we saw that it is coming from the org.sejda.model.input.PdfMergeInput file.*
3	*By analyzing the sejda/sejda-model/src/main/java/org/sejda/model/input/PdfMergeInput.java file, we realized that the exception is thrown because of no intersection concept*	*We marked this class as "located" and also figured out the file where this validation is taking place: NoIntersectionsValidator.java in the sejda package.*

Time spent (in minutes): 40

Impact Analysis

Step #	Description	Rationale
1	*We made a list of classes that use PdfMergeInput. There were two test classes and 2 java classes. MergeSelectionPane.java and MergeParametersBuilder.java*	*To track the classes that could be impacted by the change.*
2	*We marked that MergeParameterBuilder need to be changed as it builds the parameter needed for the Merge module.*	
3	*We inspected the class MergeSelectionPane.java*	*We realized this class need not be changed because it makes an instance of MergeParameterBuilder.*
3	*It took us a long while to figure out how to implement no intersection concept using reflections API*	*We believe that this is not a part of impact analysis because we planned to include a new class and at this stage,realized that this is not going to affect the existing classes behavior.*

Time spent (in minutes): 3600 (not continuous. but in many intervals)

[redacted] Prefactoring (optional)

Step #	Description	Rationale
1	*Similar to NoIntersectionValidator.java, we created a file called IntersectionValidator.java. We didn't make any new changes at this stage except for removing the "No" from the original filename.*	*We included this file in a similar path as that of sejda package. model/validators/IntersectionValidator.java with pdfsam-core as the parent folder.*
2	*We build the system to run with these changes*	*We tested everything was working as before, after the refactoring.*

Time spent (in minutes): 40

[redacted] Actualization

Step #	Description	Rationale
1	*In the intersectionValidator.java, we removed the validation of intersection of pages. Now, the function returns true when pages intersect.*	
2	*We created a file called Util.java in a folder called util under pdfsam-core/main files.*	*In order to utilize the new validation implemented in intersectionValidator.java in the merge module. This inclusion is common to all modules that use page numbers as input. So, we decided to place this file under pdfsam-core directory.*
3	*We created unit tests for the new class and also performed functional testing. We also ran the existing test cases.*	*To make sure everything works.*

Time spent (in minutes): 180

[redacted] Post factoring (optional)

We didn't have any post-factoring to do

Time spent (in minutes): 5

[redacted] Validation

We created two test cases

Step #	Description	Rationale
1	*Test case defined in pdfsam-core /src/test/java/org/pdfsam/util/UtilTest.java* *Inputs: NoIntersectionValidator class* *Expected output: Validation should return true*	*This is an expected behavior (when pages intersect and is validated using NoIntersectionValidatorClass)* *The test passed.*
2	*Test case defined in pdfsam-core /src/test/java/org/pdfsam/util/UtilTest.java* *Inputs:IntersectionValidator class* *Expected output: Validation should return true*	*This is an expected behavior (when pages interesct and is validated using IntersectionValidatorClass)* *The test passed.*

Time spent (in minutes): 40

[redacted]Timing

Summarize the time spent on each phase.

Phase Name	**Time (in minutes)**
Concept location	40
Impact Analysis	3600
Prefactoring	40
Actualization	180
Postfactoring	5
Verification	40
Total	3905

[redacted]Reverse engineering

Partial UML. It doesn't show the attributes of PdfMergeInput and MergeParametersBuilder classes because of lack of space. The first diagram doesn't show the classes from previous change and its effect on this 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.

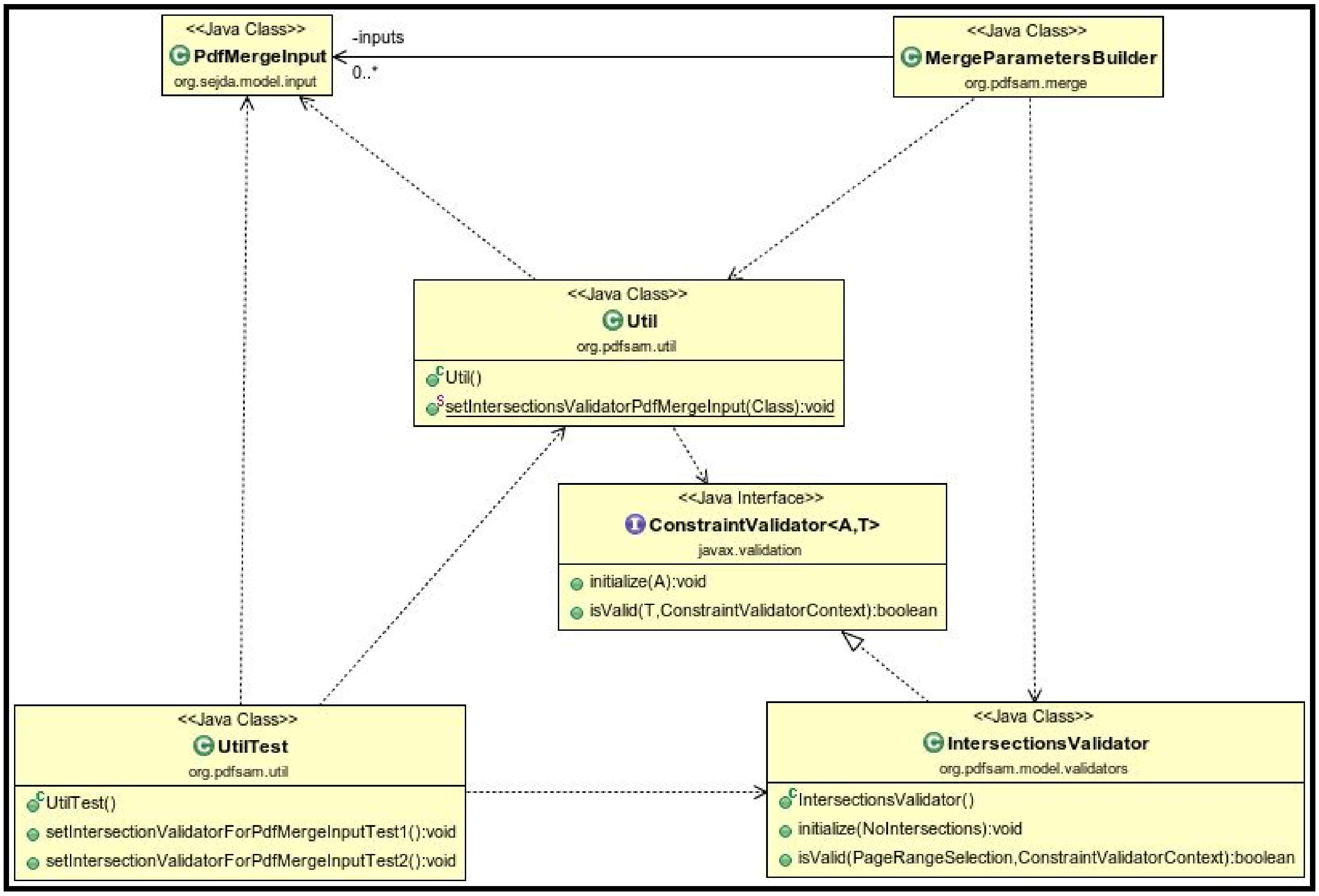

Figure 1: UML diagram showing the relationship between classes used for Change Request #2

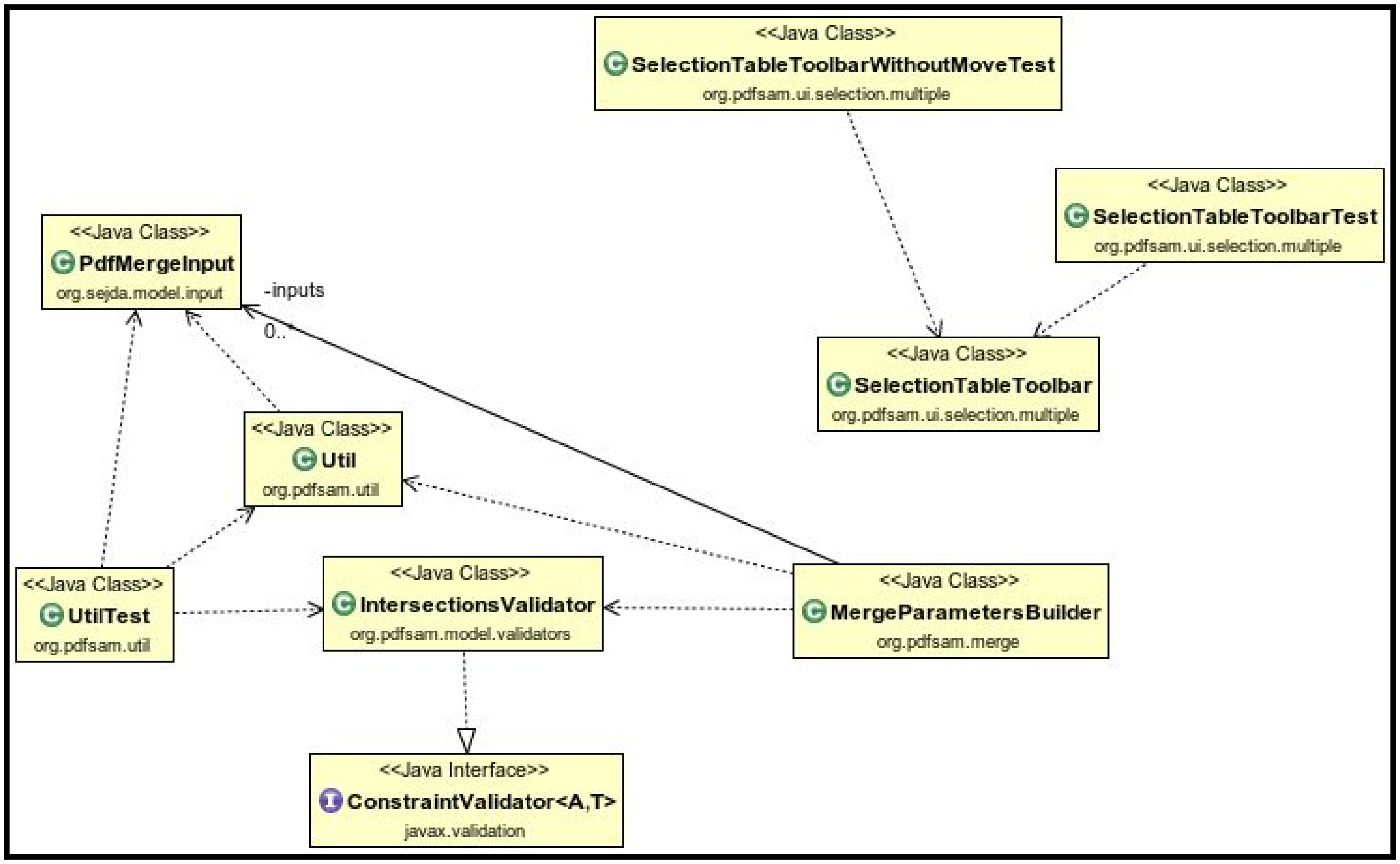

Figure 2: UML diagram showing the relationship between classes used for Change Request #1 and #2

[redacted]Conclusions

For this change, except for validation part, every other step was difficult to analyze. It took us several hours to figure out the appropriate implementation. We spent most of the time thinking how to implement Intersection Validation using Reflections API concept.

Classes and methods changed:

- *pdfsam-core/src/main/java/org/pdfsam/model/validators/IntersectionsValidator.java*
- *pdfsam-core/src/main/java/org/pdfsam/util/Util.java*
- *pdfsam-core/src/test/java/org/pdfsam/util/UtilTest.java*
- *pdfsam-merge/src/main/java/org/pdfsam/merge/MergeParametersBuilder.java*